A cell-based approach for generating solid objects from orthographic projections

Hiroshi Masuda and Masayuki Numao.

IBM Research, Tokyo Research Laboratory
1623-19, Shimo-tsuruma, Yamato-shi, Kanagawa 242, Japan
tel: +81-462-73-4889
fax: +81-462-73-7413

Abstract

This paper describes an efficient method for converting orthographic projections into solid models based on non-manifold topology and the assumption-based truth maintenance system (ATMS), and then describes an error recovery method for incorrect orthographic projections. A combination of non-manifold modeling and ATMS achieves excellent performance for conversion problems. In our method, all solid candidates are maintained by a cellular model using non-manifold topology. Since a combination of cells in a cellular model determines the shape of a solid, solid models that match orthographic projections can be derived by solving constraints between cells and orthographic projections. A sufficient group of constraints can be expressed as a set of Boolean equations, which can be solved efficiently by using ATMS. Our method can even be applied to incorrect draftings. In actual design, many draftings contain human errors, but conventional methods cannot handle such incorrect draftings. We discuss methods for detecting inconsistent lines that should not have been included, or missing lines that should have been included.

keywords: orthographic projections, solid model, non-manifold, ATMS

INTRODUCTION

Solid models are useful for supporting design and manufacturing. In actual design, however, 2D draftings are still commonly used, because they are helpful for specifying functional mechanical parts, and many engineering data are stored as 2D draftings. 2D-to-3D conversion allows existing draftings to be used for simulations such as FEM analysis, and will accelerate the shift of design environments to 3D CAD systems.

Much work has already been done on converting orthographic projections into solid models [1]-[9]. However, it is difficult to develop a practical conversion system for the following reasons:

1. Conventional methods are time-consuming, because systems have to choose solutions from a very large number of potential solid candidates. This process is basically NP-complete. In addition, since different data structures are used for representing wireframe, surface, and solid representations, which appear during the conversion process, it is complicated to maintain relationships between each representation and orthographic projections.

2. Conventional algorithms are constructed on the premise that draftings are perfect. However, it is impractical for most designers to make perfect draftings. An investigation of hundreds of orthographic projections in several companies showed that most draftings had inconsistent lines or missing lines.

Idesawa [1] and Wesley [2] each proposed a well-known method for solving conversion problems that involves no errors in orthographic projections. In both methods, first, 3D vertices and edges are calculated by means of orthographic projections, and then faces are detected by traversing wireframe models. In the next step of Idesawa's method, solid models are searched for as combinations of faces after unnecessary vertices, edges, and faces have been eliminated by using knowledge about solid shapes and orthographic projections. However, this method is very time-consuming, because a tremendous number of solid candidates can be generated from combinations of faces. Kim [3] proposed heuristic rules for reducing the number of combinations, but it is still time-consuming. In Wesley's method, on the other hand, solid models are searched for as combinations of primitive solids. After faces have been detected, regions surrounded by faces are searched for and represented by primitive solids [4]. Candidate solid models are generated by calculating the union of primitives, and they are compared with orthographic projections. Sakurai [5], Gu [6], and Lequette [7] extended this method to deal

with curved surfaces. Yan [8] described a detailed algorithm that improves efficiency by removing pathological cases in geometric calculation. However, in these methods based on Wesley's approach, it is not practical to examine all combinations of primitive solids, because 2^n solid candidates can be generated from n primitive solids, and it is very time-consuming to calculate and examine the unified shapes of all combinations of primitive solids. Although efficient reasoning methods are indispensable, none have been proposed so far. In addition, none of the proposed methods can handle incorrect draftings. Aldefeld [9] proposed a solid generation method based on pattern recognition. A solid model is constructed as a combination of uniform-thickness partial shapes, which are generated by searching for specific patterns in 2D representation. However, in this approach, it is difficult to cover the full semantics of 2D representation.

In this paper, we propose an efficient conversion method based on non-manifold topology and the assumption-based truth maintenance system (ATMS) [10], and then propose an error recovery method for incorrect orthographic projections. A key idea of our method is to use cellular models as intermediate models. Cellular models are very powerful for 2D-to-3D conversion, because they can maintain all possible solid candidates in a single model, and are useful for deriving constraints between solids and orthographic projections. A sufficient group of constraints are expressed as a set of Boolean equations, and they are quickly solved by ATMS. Our conversion algorithm is so simple and flexible that it can easily be extended to cope with incorrect draftings.

PROCESS OF CELL-BASED CONVERSION

In this paper, it is supposed that orthographic projections consist of straight lines, circles, or ellipses, and that solid models consist of planes, cylinders, cones, spheres, and tori, in which the directions of cylinders, cones, and tori are parallel to those of the X, Y, or Z axes.

Figure 1 shows the process of our conversion method. In the first two steps, a wireframe model is generated from corresponding 2D lines in orthographic projections, and a surface model is generated by searching loops in the wireframe model. These steps are same as in Idesawa's and Wesley's methods [1, 2]. A key feature of our method is that a cellular model is gen-

erated as an intermediate representation. An example of a cellular model is geometric model A in Figure 2. Since cellular models include edges shared by more than two faces, they cannot be represented by data structures for 2-manifold solid models. In our system, a non-manifold geometric modeling system is used for managing all the geometric models inside the dotted line in Figure 1. The data structure of our modeling system is similar to the radial-edge structure proposed by Weiler [11]. Detailed discussions of our non-manifold geometric modeler may be found in previous publications [12, 13].

In Figure 2, cellular model A has three cells, which are referred to as c_1, c_2, and c_3. When one or more of these cells are selected, the boundary elements of solid shapes can be quickly extracted from a cellular model by traversing topological elements in the data structure [12]. Figure 2 shows solid shapes extracted from cellular model A. Solid models B and C are obtained by selecting $\{c_1, c_2, c_3\}$ and $\{c_1, c_3\}$, respectively, and extracting their boundary elements.

After a cellular model has been generated, combinations of cells that match the given orthographic projections must be searched for. Such combinations are calculated by solving constraints between a cellular model and orthographic projections. Constraints are derived by topological relationships between edges and cells, and by correspondences between edges and lines in projections. For example, in cellular model A in Figure 2, e_1 appears in a solid shape if c_1 and c_3 are selected. In this case, e_1 is related to $\{c_1, c_3\}$. If e_1 is projected onto line d in a 2D view, e_1 is also related to d, and therefore d is related to $\{c_1, c_3\}$. In this way, orthographic projections are constrained by combinations of cells. Solid shapes that satisfy the given orthographic projections are obtained by solving these constraints. All constraints can be described as Boolean equations, which can be solved by several reasoning methods. In our system, ATMS is used as a solver, because it is efficient and flexible for conversion problems.

GENERATION OF CELLULAR MODELS

This section describes geometric modeling for 2D-to-3D conversion. As shown in Figure 1, wireframe, surface, and cellular models are gradually generated from orthographic projections. They can be uniformly managed

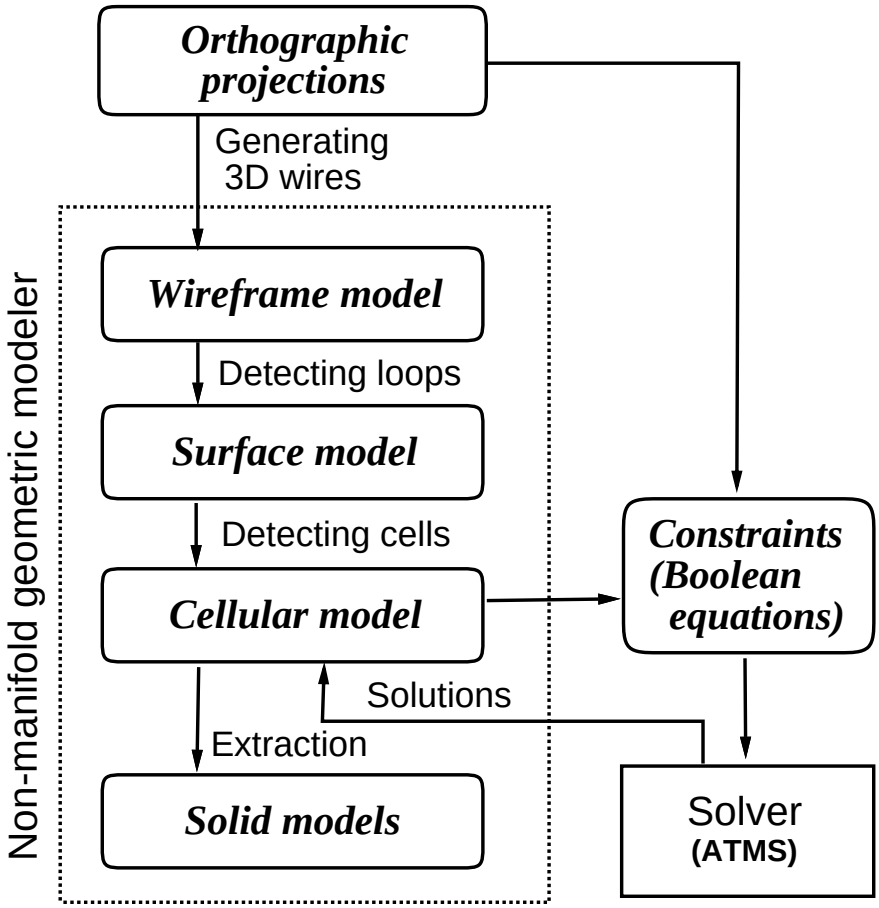

Figure 1: Process for converting orthographic projections into solid models

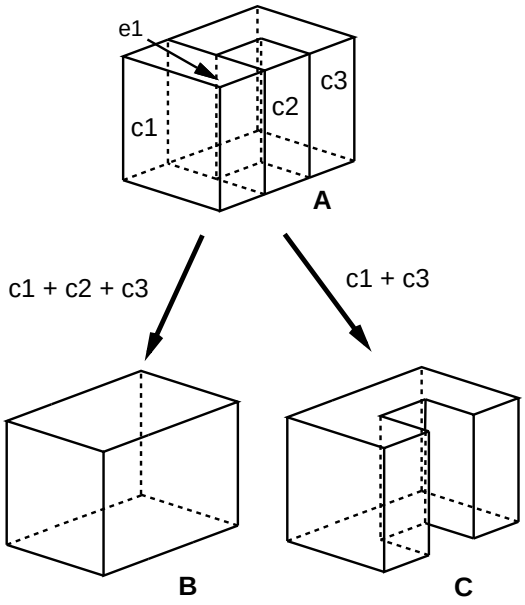

Figure 2: A cellular model and extracted solid models

by means of cell-complex-based Euler operators proposed by the present authors [12]. The whole modeling process is shown in Figure 3. Each geometric model is generated as follows.

Pre-process for curved surfaces

When 3D edges are calculated from corresponding lines in three 2D views, projections of silhouette edges and tangency edges may not appear in each orthographic projection. Sakurai solved this problem by classifying types of vertices and edges [5], but we solve it more simply by drawing additional lines for quadric surfaces and tori. This method is useful for avoiding exceptional processes during 3D geometric modeling.

Additional lines to be drawn are illustrated in Figure 4. This figure shows projections of a solid shape that consists of a cylinder, a torus, and planes, where solid lines represent the original lines, and dotted lines represent candidate projections of silhouette or tangency edges. Unnecessary additional lines may be drawn, but they do not have any side effects in our algorithm. Additional lines for the cylinder are marked by circles and ones of the torus by squares.

Additional lines for cylinders or cones are drawn by extending straight lines from the extrema of circles or ellipses to other projections. In this figure, dotted lines with circle markers are drawn by extending lines as indicated by the arrows. The range of each dotted line is determined compared with parallel solid lines in the same view. In the cases of spheres and tori, a pair of circles, each marked by a black triangle, are detected in each of the top and side views, and the corresponding solid circle line is detected in the front view. Since pairs of circles and their corresponding circles determine candidate surfaces, additional straight lines and circle lines can be drawn as shown by the dotted lines with square markers.

Wireframe model

3D vertices are calculated from triplets of 2D points. When the coordinate values of 2D points in the front, top, and side views are $P_f(x_f, y_f)$, $P_t(x_t, z_t)$, $P_s(y_s, z_s)$, respectively, and they satisfy

$$x_f = x_t,\ y_f = y_s,\ z_t = z_s,$$

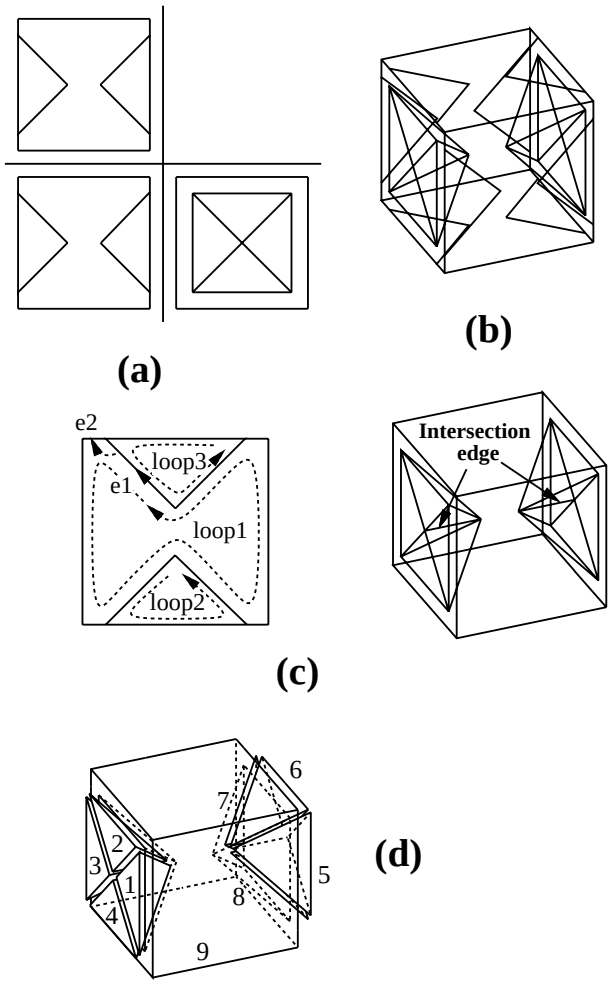

Figure 3: Modeling process

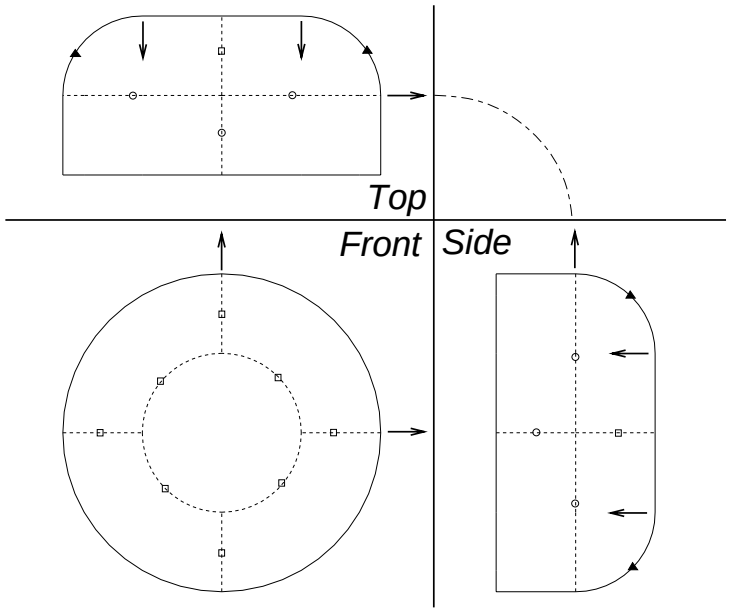

Figure 4: Additional lines for curved surfaces

a 3D vertex is generated at (x_f, y_f, z_t). By examining all the possible triplets, all 3D vertices are obtained.

A 3D edge is generated by examining the connectivity of two 3D vertices. By examining all pairs of 3D vertices, a wireframe model is obtained. Here, a pair of 3D vertices are referred to as v_1 and v_2, and their corresponding 2D points are referred to as P_{f1} and P_{f2} in the front view, and P_{t1} and P_{t2} in the top view. When the three pairs $< P_{f1}, P_{f2} >$, $< P_{t1}, P_{t2} >$ and $< P_{s1}, P_{s2} >$ are equivalent or connected by a line in the projection, a 3D edge is defined and its equation is calculated. 3D edges maintain pointers to corresponding 2D lines in each projection. The relationships are maintained during the whole modeling process.

Surface model

In the next step, a wireframe model is converted into a surface model by defining face elements. Faces are detected by traversing loops in a wireframe model. Since the types of surfaces are limited to planes, cylinders, cones, spheres, and tori, equations of faces can be determined by a pair of connected edges that are not on the same curve. The equations of planes are determined by pairs of two straight edges, the equations of cylinders and cones are determined by pairs of straight and circular edges, and the equations of spheres and tori are determined by pairs of circular edges. When the equation of a surface is determined, loops are searched for on the surface. Figure 3 (c) shows loops on a plane. In this example, when edge e_1 is selected as the starting edge, edge e_2, which is the first clockwise edge of e_1, is selected as the next one, and finally *loop*1 is traversed counter-clockwise.

When loops are detected, faces are defined and embedded in a wireframe model. If two faces intersect, they are subdivided by generating intersection edges, as shown in Figure 3 (c). If a face includes another face, it becomes a face with holes.

Cellular model

A cellular model is generated by detecting regions surrounded by faces and maintaining them as *cells* in the data structure. In the radial-edge structure [11], cells are very quickly searched for by traversing adjacent relationships

around edges. When a cell includes another cell, it becomes a cell with cavities.

In Figure 3, nine cells are detected. They are shown in Figure 3(d).

Extraction of boundary elements

When a set of cells is selected, topological elements on the boundary are selected as follows:

1. When only one adjacent cell of a face is a selected one, the face is a boundary element.
2. Edges and vertices on boundary faces are also boundary elements.
3. When a boundary edge is shared by two faces on the same surface, it is eliminated.

The solid shapes B and C in Figure 2 can be extracted by following these steps. The solid shape becomes a solution, when projected views of the selected topological elements coincide with the given orthographic projections.

REASONING BY ATMS

The next problem is to search for appropriate solid shapes in 2^n candidates, which can be extracted from a cellular model with n cells. In this section, a new reasoning method is proposed for obtaining solid shapes that match the given orthographic projections. A sufficient group of constraints is described as a set of Boolean equations, which are solved by using ATMS.

ATMS

ATMS is normally used for dependency management of other inference systems, such as rule-based systems [10], but it has its own inference mechanism based on propositional logic, and we use it as a Boolean equation solver. ATMS basically accepts Horn clause rules of the form

$$x_1, ..., x_k \Rightarrow n.$$

which means "if all of $x_1, ..., x_k$ are true then n should be true" (material implication), and

$$x_1, ..., x_k \Rightarrow False.$$

which means "at least one of $x_1, ..., x_k$ is false" (nogood conditions). Each symbol $(x_1, ..., x_n, n)$ in the rule is either an assumption or a variable: a variable is derived by a rule, if it appears in the right-hand side of the rule. Therefore, the set of rules defines the chain of derivation from the assumptions to the variables; that is, some variables are derived from other variables, each of which is derived from other variables, and so on, until the variables are derived from the set of assumptions. The nogood condition, on the other hand, defines the sets of variables or assumptions that cause inconsistency. The basic function of ATMS is to calculate the sets of assumptions which support the variables and the sets of assumptions that cause contradiction. Its goal is to assign consistent sets of assumptions to each variable.

Basic terminology of ATMS

The following terms and descriptions are used in this section:

- An *assumption* is a propositional symbol that is presumed to be true and that cannot be deduced. It is denoted as an uppercase letter: $C_1, C_2, ...$
- A *variable* is a propositional symbol that is defined by a justification. It is denoted by a lowercase letter: $e_1, e_2, ...$
- A *justification* is a Hone clause that defines a derivation. It is described as a rule: $x_1, ..., x_k \Rightarrow n$.
- An *environment* is defined as a conjunction of assumptions that support the variable. It is denoted by C_1C_2. An environment should be minimal, that is, it should not contain any other environments as subsets.
- A *label* is defined as a set of environments that support the variable. For example, when the variable x is true in environment C_1C_2 or C_1C_3, the label of x is $\{C_1C_2, C_1C_3\}$.
- A *nogood* environment is defined as a conjunction of assumptions that cause inconsistency. The nogood environment is derived from the nogood rule: $x_1...x_k \Rightarrow nogood$.

- A *maximum consistent environment* is defined as an environment that cannot maintain consistency when more assumptions are added.

Justifications for conversion problems

If ATMS is used as a constraint solver, there is a problem as to which components of cellular models and orthographic projections should be represented as the ATMS assumptions, variables, and justifications. We will define the problem as follows:

1. C_i expresses an assumption that cell i is selected, and $\overline{C_i}$ expresses an assumption that cell i is not selected. When assumptions are defined as cells in a cellular model, every justification can be deduced to the conditions of cells.

2. e_j expresses a proposition that edge j in a cellular model is used as an edge of a solid shape.

3. d_k expresses a proposition that line k in orthographic projections appears in projected lines of a solid shape.

Solid shapes that match the given orthographic projections are derived as sets of cells by means of the following justifications:

Justification 1: *Solid models are related to orthographic projections.*

All lines in orthographic projections must appear in projected lines of a solid shape. When the original orthographic projections consist of lines $1, 2, ..., n$, the following justification represents the condition that projections of solid shapes must include all lines in orthographic projections. Additional lines for silhouette and tangency edges are not included in this justification.

$$d_1 d_2 ... d_n \Rightarrow solid.$$

Justification 2: *Orthographic projections are related to 3D edges.*

When 3D edges $1, 2, ..., n$ are projected onto line i in a 2D view, at least one of the n edges must appear in a solid shape. This condition is described by the following justifications:

$$e_1 \Rightarrow d_i.$$

$$e_2 \Rightarrow d_i.$$

$$...$$

$$e_n \Rightarrow d_i.$$

Justification 3: *3D edges are related to cells.*

Edges in a cellular model appear in a solid shape according to neighboring cells. In Figure 5, there are four cells around edge 1. Edge 1 appears in solid shapes that have the following combinations of cells:

$$\begin{array}{cccc} C_1\overline{C_2C_3C_4}, & \overline{C_1}C_2\overline{C_3C_4}, & \overline{C_1C_2}C_3\overline{C_4}, & \overline{C_1C_2C_3}C_4, \\ C_1C_2C_3\overline{C_4}, & C_1C_2\overline{C_3}C_4, & C_1\overline{C_2}C_3C_4, & \overline{C_1}C_2C_3C_4. \end{array}$$

Each environment is described as a justification such as

$$C_1\overline{C_2C_3C_4} \Rightarrow e_1.$$

Nogood Condition 1: *Cells.*

C_1 and $\overline{C_1}$ cannot both be true, because of the definition. Therefore,

$$C_1\overline{C_1} \Rightarrow nogood.$$

Nogood Condition 2: *Intersecting edges.*

Intersecting edges may be generated when faces intersect as shown in Figure 3(c). Such edges must not appear in a solid shape, because they are

not drawn in the original orthographic projections. Intersecting edge e_1 does not appear by adding

$$e_1 \Rightarrow nogood.$$

Nogood Condition 3: *Non-manifold conditions.*

If non-manifold objects are not desired, environments for non-manifold conditions should be added as *nogood*. For example, if a non-manifold shape is generated by $C_1, \overline{C_2}, C_3$, and $\overline{C_4}$, the following condition is added:

$$C_1\overline{C_2}C_3\overline{C_4} \Rightarrow nogood.$$

Nogood Condition 4: *Additional lines.*

Additional lines may be drawn in orthographic projections for managing silhouette edges and tangency edges. If edge 1 is generated by using additional lines, it must be a silhouette or tangency edge. Therefore, when a combination of cells around edge 1 generates neither a tangency edge nor a silhouette edge, the combination must be specified as *nogood*.

Computation of the solution

From the above justifications, ATMS computes the environment of each variable by means of the label update propagation algorithm [14]. Every time a justification is added, ATMS renews the labels of e_i, d_i, and *solid*, taking account of the nogood environments. The label of *solid*, which is a set of combinations of cells, expresses candidate solutions.

The environment represents necessary conditions to hold the variable: the environment may not contain all the assumptions necessary to solve the problem. Therefore, we have to expand the environment by adding assumptions until no other assumptions can be added without the combination becoming nogood. This expanded environment is called the maximum consistent environment. For example, if a cellular model consists of two cells, then

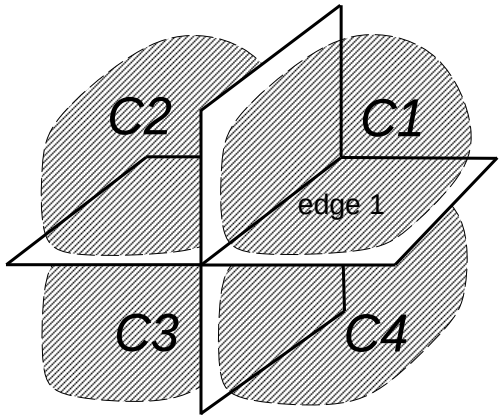

Figure 5: Cells around an edge

we want to know whether each of the cells is selected or not. But sometimes the environment consists of only one assumption, corresponding to one cell. In this case, we have to expand the environment to find the solutions. Figure 6 shows how to obtain the maximum consistent environments. For simplicity, let us consider a case in which only cells 1 and 2 exist in a cellular model, and suppose that the nogood environments are $\{C_1C_2, C_1\overline{C_1}, C_2\overline{C_2}\}$, and that the label of *solid* is calculated as $\{C_1\}$. Since environments that include $C_1C_2, C_1\overline{C_1}$, or $C_2\overline{C_2}$ are inconsistent, environments above the dotted line in this lattice are nogood. Environments in the label of *solid*, which include C_1, are shown by bold lines. The maximum consistent environments are connected by bold lines and exist below the dotted lines. In this example, the solution is $\{C_1\overline{C_2}\}$.

Since the maximum consistent environments are searched for only in the superset of the label of *solid*, solutions can be effectively found.

Results

Figure 7(a) shows an example to which this method was applied. Since the orthographic projections in (a) are ambiguous, multiple solutions exist. A cellular model is shown in Figure 7(b), which consists of 10 cells. In this example, 241 justifications and 42 nogood conditions are added to ATMS. 16

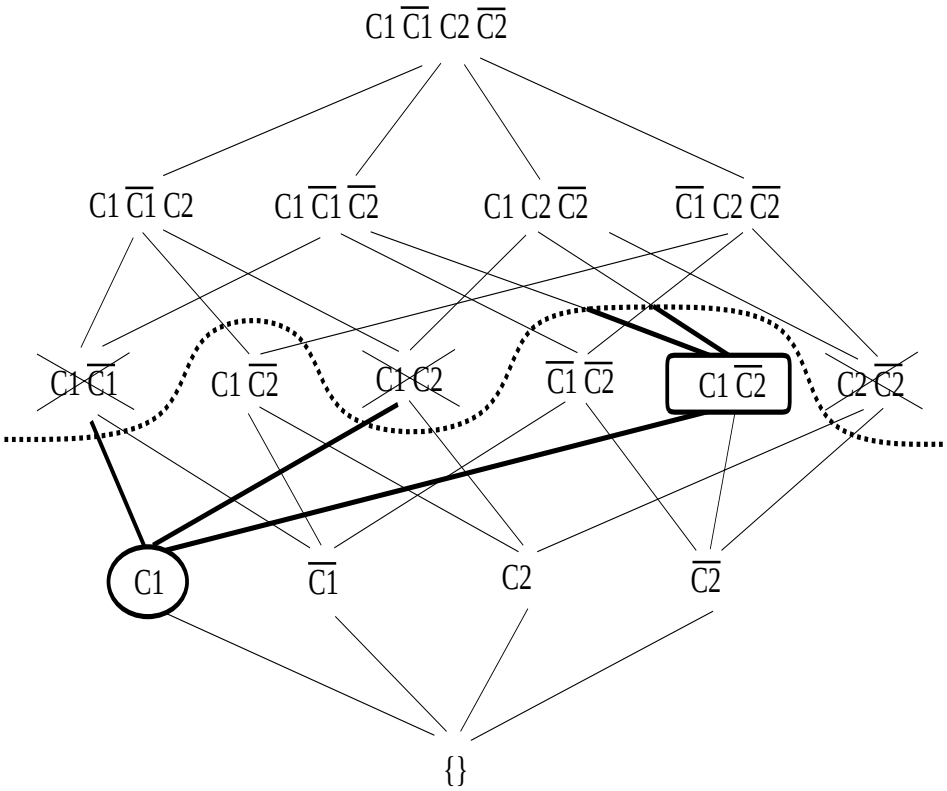

Figure 6: Lattice of potential environments

solutions are obtained by searching for the maximum consistent environments if hidden lines and solid lines are not distinguished. Four of the 16 solutions are shown in Figure 7(c). The elapsed times for modeling and reasoning in Figure 7 were 0.20 seconds and 0.21 seconds , respectively, and the total was 0.41 seconds.

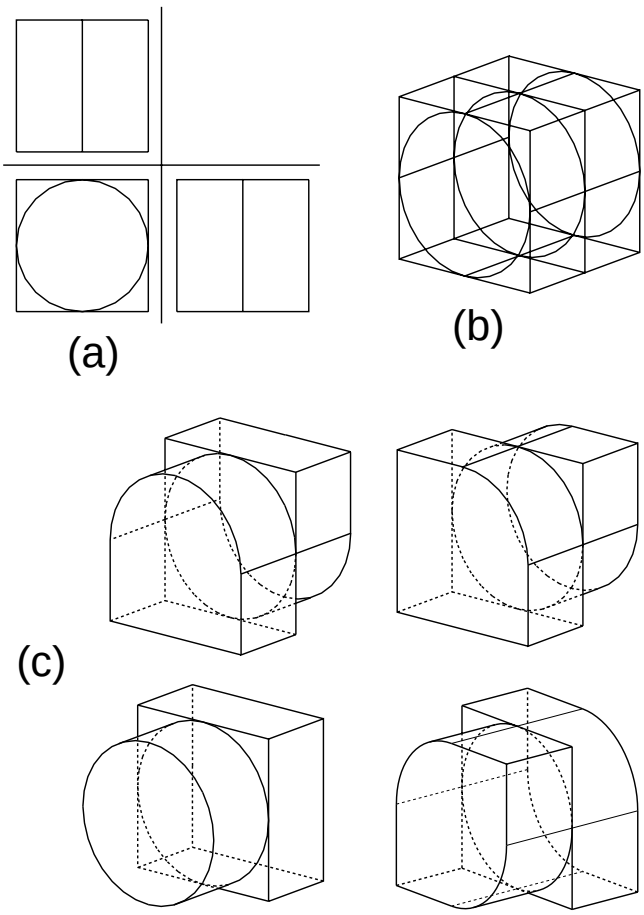

Figure 7: Ambiguous orthographic projections

Table 1 shows the elapsed time for conversion of several examples on an IBM RS/6000-980. In the example shown in Figure 3, 13 solid shapes are obtained as solutions, and the total elapsed time was 0.30 seconds. Fig-

ure 8 shows more complicated orthographic projections. Solutions of these orthographic projections are uniquely determined. The elapsed times for converting them are shown in Table 1. The results show that our method's performance is excellent in comparison with those of existing methods.

Example	Number of cells	Number of solutions	Elapsed time (seconds)		
			Modeling	Reasoning	Total
Fig. 3	9	13	0.14 s	0.16 s	0.30 s
Fig. 7	10	16	0.20 s	0.21 s	0.41 s
Fig. 8a	7	1	0.82 s	0.30 s	1.12 s
Fig. 8b	17	1	3.22 s	0.47 s	3.69 s
Fig. 8c	24	1	3.31 s	6.52 s	9.83 s

Table 1: Elapsed times for conversion of orthographic projections (IBM RS/6000-980)

INCORRECT DRAFTINGS

In actual design, many draftings contain human errors, but conventional methods cannot handle such incorrect draftings. In this section, two methods are proposed for generating solids from incorrect orthographic projections. Errors are classified as inconsistent lines, which should not have been included, or missing lines, which should have been included. The two cases are discussed separately.

Inconsistent lines

First, let us suppose that draftings have inconsistent lines but no missing lines. Figure 9 shows orthographic projections that have inconsistent lines and match no solid shapes. It is very important to detect and recover errors, because it is difficult for most designers to find trivial errors such as this case.

When a search for solids that match three projections fails, it is reasonable to search for solid shapes that match two or one projections. By using

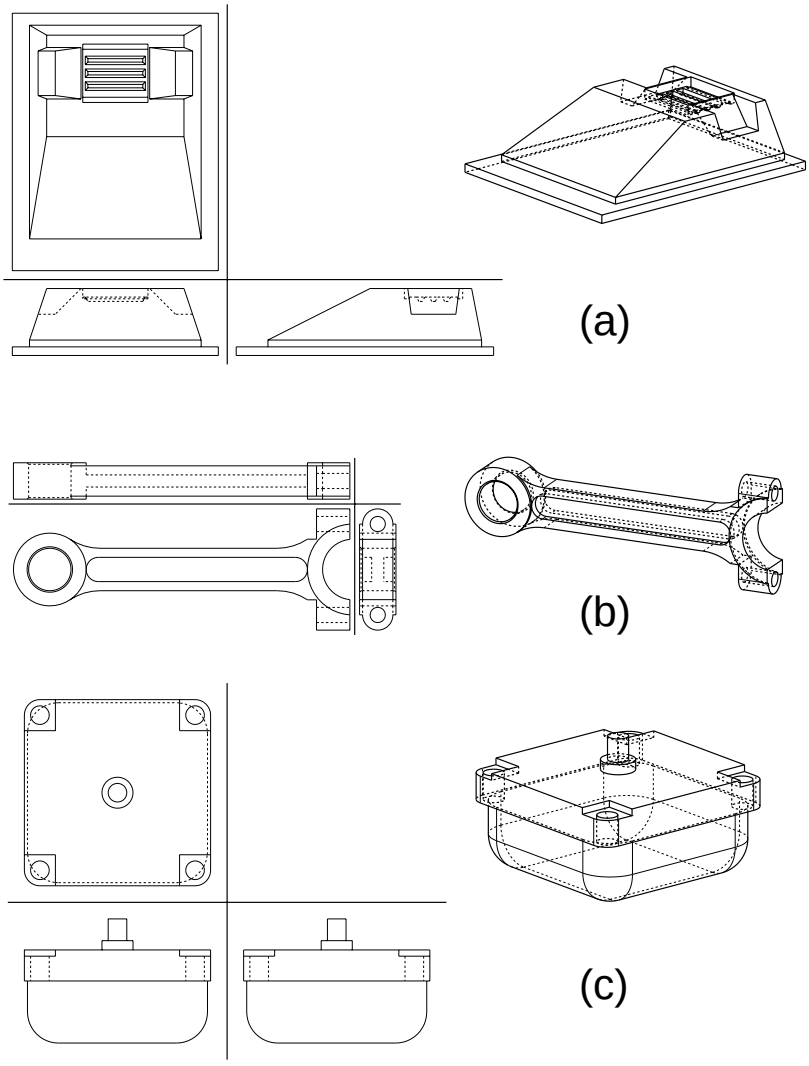

Figure 8: Orthographic projections and generated solid models

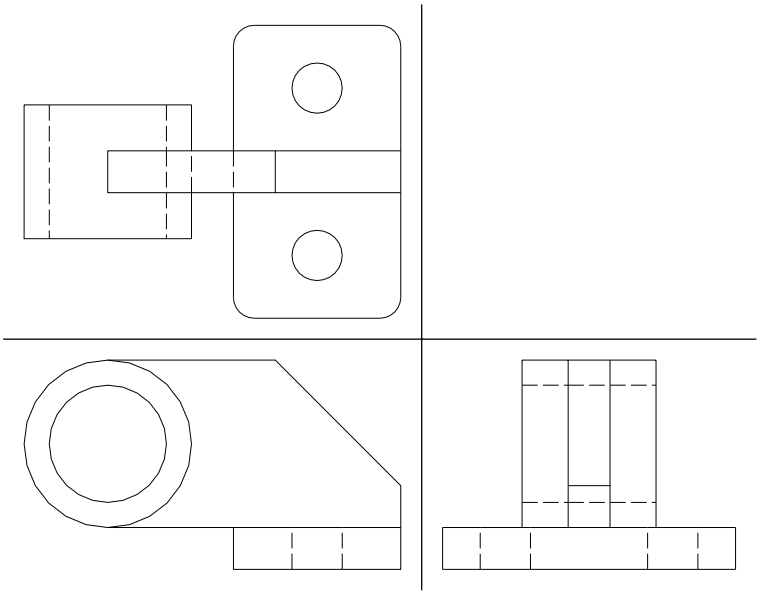

Figure 9: Orthographic projections that have inconsistent lines

this criterion, if at least one of three projections is correct, proper solid candidates can be calculated. Such solid shapes are very easily searched for by modifying Justification 1 in the above. Here, the front view, the top view, and the side view are referred to as v_F, v_T, and v_S, respectively, and variables that represent lines in these 2D views are referred to as $\{d_{F1}, d_{F2}, ..., d_{Fp}\}$, $\{d_{T1}, d_{T2}, ..., d_{Tq}\}$, and $\{d_{S1}, d_{S2}, ..., d_{Sr}\}$. The justifications for these views are as follows:

Justification 1.1: *Each projection is related to lines.*

$$d_{F1} d_{F2} ... d_{Fp} \Rightarrow v_F$$

$$d_{T1} d_{T2} ... d_{Tq} \Rightarrow v_T$$

$$d_{S1} d_{S2} ... d_{Sr} \Rightarrow v_S$$

The following are justifications for solid shapes that satisfy two or three projections:

Justification 1.2: *Solid models are related to two projections.*

$$v_F \, v_T \Rightarrow solid.$$

$$v_F \, v_S \Rightarrow solid.$$

$$v_T \, v_S \Rightarrow solid.$$

When solutions cannot be obtained by Justification 1.2, solid shapes that satisfy at least one projection are calculated by using the following justifications.

Justification 1.2': *Solid models are related to one projection.*

$$v_F \Rightarrow solid.$$

$$v_T \Rightarrow solid.$$

$$v_S \Rightarrow solid.$$

When multiple solid shapes are calculated, an appropriate solution is selected by the designers. Inconsistent lines are detected by comparing orthographic projections and projected lines of a generated solid shape. Figure 10(a) shows a solid shape that satisfy two projections in Figure 9. Figure 10(b) shows a detected inconsistent line, which must be eliminated to generate solid shape(a).

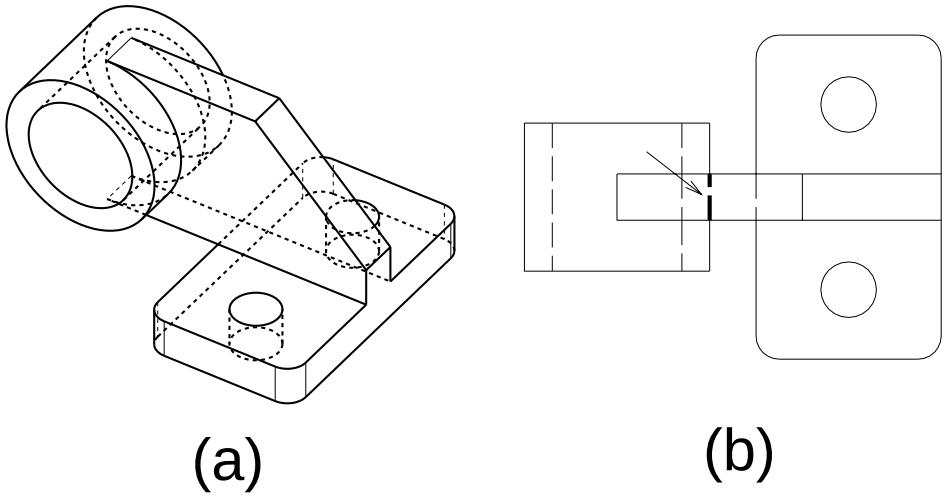

Figure 10: Detection of inconsistent lines. (a) A generated solid model. (b) The detected inconsistent line.

Missing lines

Actual draftings may contain missing lines. In particular, simple hidden lines are often omitted. If some lines are not drawn in orthographic projections, sufficient 3D edges cannot be generated, and therefore proper cellular models cannot be obtained. In such cases, it is necessary to compensate for missing lines. Since it is impossible to compensate for arbitrary missing lines without knowledge about products, the missing lines are assumed to be vertical or horizontal straight lines.

Figure 11 shows orthographic projections containing a missing line and a generated wireframe model. When a wireframe model is generated from these projections, sufficient edges are not generated. Comparison of the wireframe model and the orthographic projections reveals that the bold lines in Figure 11 do not correspond to the wireframe model. These bold lines obviously relate to missing lines in the top or side views. Here, so that 3D edges can be generated by using bold lines, supplementary horizontal and vertical lines are added to the top and side views. Supplementary lines are generated by extending from the end points of bold lines, as shown in Figure 12(a). Since

the supplementary lines are merely candidate missing lines, they must not be included in Justification 1. The constraints to be solved are the same as those in draftings with no errors. The solution for this example is obtained as shown in Figure 12(b). The solid shape is then compared with the original orthographic projections in Figure 11, and missing lines are determined as illustrated by the bold line in Figure 12(c).

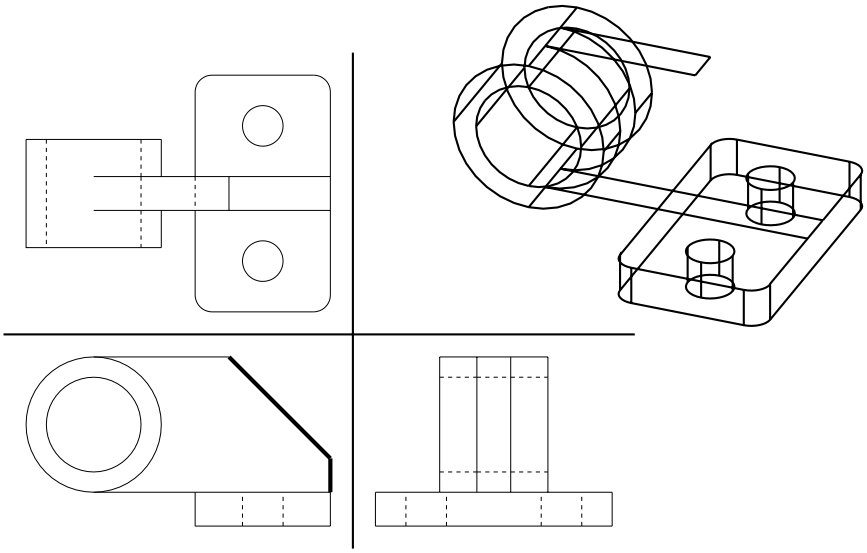

Figure 11: Orthographic projections that have missing lines and a generated wireframe model

In general, missing lines are compensated for as follows:

1. If missing lines are suspected after comparison of a wireframe model and orthographic projections, horizontal and vertical lines are added so that every 2D line in the original orthographic projections corresponds to at least one 3D edge. The wireframe model is then calculated again, and a surface model is generated.

2. Missing lines are searched for by comparing a surface model and orthographic projections. If missing lines are suspected, horizontal or

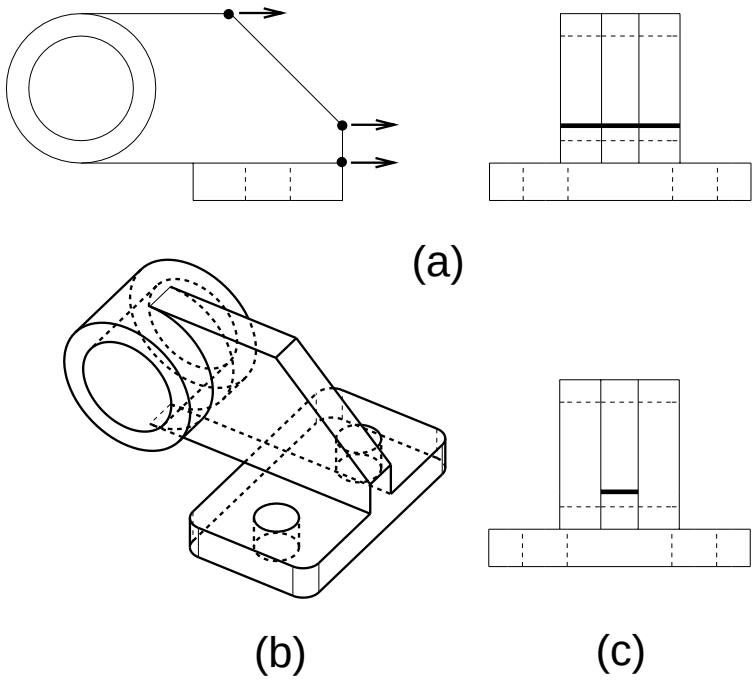

Figure 12: Recovery of missing lines. (a) Candidate missing lines that are compensated for. (b) The generated solid shape. (c) The detected missing line.

vertical lines are added, and the surface model is recalculated from the beginning. A cellular model is then generated.

3. Similarly, by comparing with the cellular model, missing lines are searched for, and if needed, the cellular model is calculated again.

The method for recovering missing lines is used in combination with the one for detecting inconsistent lines. When proper solutions are not obtained after missing lines have been recovered, solids that satisfy two or one projections are searched for.

Results

The elapsed times for Figures 9 and 11 were measured. When these draftings contain no errors, the elapsed time was 0.72 seconds on an IBM RS/6000-980. For Figure 9, the elapsed time for generating a solid and detecting inconsistent lines was 1.03 seconds in total. For Figure 11, the total elapsed time for generating a solid model and detecting missing lines was 1.09 seconds. The elapsed times when draftings contain errors is a little longer, because reasoning is applied twice for Figure 9, and candidate missing lines are compensated for in the case of Figure 11.

This method is very efficient and powerful for detecting human errors in orthographic projections.

CONCLUSION

This paper has described a new conversion method based on non-manifold topology, cellular representation, and ATMS.

1. Non-manifold topology is suitable for constructing 2D-to-3D conversion systems, because it uniformly manages wireframe models, surface models, cellular models, and solid models.

2. Cellular representation based on non-manifold topology is a very powerful way of representing a very large number of potential solid candidates. It allows various solid shapes to be extracted very quickly. Cellular models are also useful for deriving constraints between 3D shapes and orthographic projections.

3. ATMS is very efficient for solving Boolean equations of conversion problems, and it is so flexible that our conversion algorithm can be easily enhanced to handle inconsistent lines.

Our method was applied to complicated examples as shown in Figure 8, and the results indicate that it achieves excellent performance.

So far, very few ways of converting incorrect draftings have been proposed, although such draftings are very common in actual design. This paper has proposed two methods for handling incorrect draftings:

1. One method is for detecting inconsistent lines. When no solids can be searched for, solid shapes that match one or two projections are searched for. Inconsistent lines are detected by comparing the given orthographic projections and projected lines of a solid shape.

2. The other method is for recovering missing lines. If the proper cellular model cannot be obtained from orthographic projections, horizontal and vertical lines are added in the orthographic views to satisfy the condition that all 2D lines should correspond to at least one edge in a 3D model. Here, the missing lines are limited to vertical and horizontal lines, in order to reduce the number of candidate missing lines.

Method 1 does not work when all the projections are incorrect, but it can be easily extended to cover other conditions, in which, for example, only holes or only boundaries are correct. It is also possible for designers to specify such conditions interactively. Method 2 is intended to cover a large proportion of actual draftings, but it cannot convert draftings that lack essential information, which is usually supplemented by standards, conventions, knowledge, and experience. We intend to tackle this problem by following a rule-based approach.

Finally, our conversion system is written in C++ and runs on an IBM RS/6000. It was implemented on our own non-manifold geometric modeler and ATMS, which were developed for general purposes.

ACKNOWLEDGMENTS

The authors would like to thank S. Shimizu, who implemented the ATMS [15]. We also thank H. Matsuzawa, who developed an interface module with

a 2D CAD system, and K. Inoue and A. Okano, who helped develop our GUI.

References

[1] Idesawa, M 'A System to Generate a Solid Figure from a Three View' *Bulletin of the Japan Society of Mechanical Engineering* Vol.16 (1973) pp 216–225

[2] Wesley, M A and Markowsky, G 'Fleshing Out Projections' *IBM Journal of Research and Development* Vol. 25 (1981) pp 938–954

[3] C.Kim, M.Inoue, and S.Nishihara 'Heiristic Understanding of Three Orthographic Views,' *Journal of Information Processing* Vol.15 No.4 (1992) pp 510–518

[4] Markowsky, G and Wesley, M A 'Fleshing Out Wire Frames' *IBM Journal of Research and Development* Vol. 24 (1980) pp 582–596

[5] Sakurai, H and Gossard, D C 'Solid Model Input through Orthographic Views' *Proc of SIGGRAPH'83* Vol.17 No.3 (1983) pp 243–247

[6] K.Gu, Z.Tang, and J.Sun, 'Reconstruction of 3D Objects from Orthographic Projections,' *CG Forum,* Vol.5 (1985) pp 807–811

[7] R.Lequette, 'Automatic Construction of Curvilinear Solids from Wireframe Views,' *Computer Aided Design* Vol.20 No.4 (1988) pp 171-180

[8] Q.Yan, C.L.Philip, and Z.Tang 'Efficient Algorithm for the reconstruction of 3D objects from orthographic projections,' *Computer Aided Design* Vol.26 No.9 (1994) pp 699–717

[9] B.Aldefeld, 'On Automatic Recognition of 3D Structures from 2D Representations,' *Computer Aided Design,* Vol.15 No.2 (1983) pp 59–64

[10] de Kleer, J 'An Assumption-Based Truce Maintenance System' *Artificial Intelligence* Vol.28 No.2 (1986) pp 127–162

[11] Weiler, H 'Topological Structures for Geometric Modeling' *PhD.Thesis, Rensselaer Polytechnic Institute* (1986)

[12] Masuda, H, Shimada, K, Numao, M and Kawabe, S 'A Mathematical Theory and Applications of Non-Manifold Geometric Modeling' *Advanced Geometric Modeling for Engineering Applications, North-Holland* (1989) pp 89-103

[13] Masuda, H 'Topological Operators and Boolean Operations for Complex-Based Non-Manifold Geometric Models' *Computer Aided Design* Vol.25 No.2 (1993) pp 119-129

[14] deKleer, J 'A General Labeling Algorithm for Assumption-Based Truth Maintenance' *Proc. of AAAI* (1988) pp 188-192

[15] Shimizu S, Inoue K and Numao M 'An ATM-Based Geometric Constraint Solver for 3D CAD' *Proc. of Tools For Artificial Intelligence* (1991) pp 282–290