Proceedings of ICAD2004
The Third International Conference on Axiomatic Design
Seoul – June 21-24, 2004
ICAD-2004-48

FUNCTIONAL REQUIREMENTS TO SHAPE GENERATION IN CAD: DATABASE AND AUTOMATIC SHAPE ASSEMBLY

Jinpyung Chung
jinpyoung.jung@samsung.com
Samsung Corning Precision Glass Co., LTD
544, Myungam-Ri, Tangjung-Myun, Asan-City, ChungNam, Korea, 336-840

Kang-Soo Lee
kslee@hanbat.ac.kr
Hanbat National University
Dukmyungdong, Yuseonggu, Daejeon, 305-719, Korea

Nam P. Suh
npsuh@mit.edu
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA. 02139, USA

ABSTRACT

This paper presents details of databases and automatic assembled shape generation. The databases store links of FR-DP-FGF-INTERFACEs and their hierarchies generated based on the V-model. Each FGF is composed of links to a set of cells and relations of interfaces of the cells. The retrieval of proper FGFs from the database is performed by matching a query FR with stored FRs linked to the corresponding FGFs by a lexical search based on the frequency of words and the sequence of the words in the FR statements using a synonym checking system. The language-matching rate is calculated as a value of FR_metric between 0 and 1. A computer algorithm automatically combines and assembles the retrieved FGFs. Genetic algorithm (GA) searches for the best matching of interface types between FGFs and generates the corresponding assembly sequences based on the codes of the chromosomes. From the highest-valued chromosome, the computer algorithm operates automatic assembly of FGFs by coordinating, orienting, and positioning FGFs with reference to the given mating conditions. Geometric interface-ability between FGFs is calculated as a value of INTERFACE_metric between 0 and 1. The higher the values of FR_metric and INTERFACE_metric, the better the design solution for the given FRs that must be satisfied in the sense of language and geometric interface matching. The top-down decomposition and bottom-up integration in the V-model reduce the number of possible combinations of interfacing FGFs. The method presented in this paper has demonstrated that a "functional CAD" can aid designers in generating conceptual design solutions from functional descriptions, in reusing existing CAD models, and in creating new designs.

Keywords: FR-DP-FGF-INTERFACES, database, language matching, assembled shape generation, interface-ability, axiomatic design

1 INTRODUCTION

The organization of design information is one of the most important factors that determine performance and functionality of the CAD systems. Most CAD systems are primarily concerned with data structure and information flow for representing and constructing 3D shapes on computers. Those CAD systems enable us to generate complex shape and to utilize information on the shape in designing and manufacturing products. Feature-based representation or design is one of the well-known results of such an effort. The concept of *design by features* was firstly proposed for the feature-based design. (Pratt & Wilson 1985) It is generally agreed that one of the key benefits using features is that designers can put their deign intent into the geometric model.

Many techniques of using geometric features have been developed for such applications as machining automation, diagnosis of defects of geometric models, generation of finite element models, and so on. Nevertheless, there still exist two main bottlenecks for the feature-based design systems to be useful for design tasks. One bottleneck is the lack of the means of defining geometric features based on functions. The other is the lack of a method to reuse a lot of geometric features for design tasks. The bottlenecks have not been overcome yet, because most research on geometric features has been focused on their physical characteristics. For example, a feature-based design system, called *design with features*, has been proposed. (Dixon and Cunningham 1987) The geometric features used in the system have been classified by only geometric characteristics into types; primitives, intersections, add-ons, macros, and whole-forms. A feature library has been constructed based on the classified features. Another feature-based system has been proposed based on frame approach. (Sreevalsan and Shah 1992) Designers used templates to edit features by formal language. Taxonomy based on the templates has been useful to handle geometric features in the physical domain. However, those systems do not deal with information on functions of the features. Pre-defined features collected using those approaches may not be useful for conceptual design tasks.

There exist many CAD approaches to using database. Most of them classify CAD models by similarity of the shape, stored them in the database, and searched similar shape to an input geometric model. In these approaches, it is relatively easy to collect and classify CAD models and to find out similar shapes, but still this database cannot help designers to generate design solutions. It is usually used to find out similar parts for manufacturing purpose and to classify shapes.

In this paper, we have attempted to answer the following questions related to the feature-based design.

- How can we generate the geometric features starting from a given set of functional requirements?
- How can a computer reuse the geometric features to generate candidate shapes in the conceptual design phase?

To answer these two questions, the following four sub-topics have been considered and will be explained in this paper as a method to automatically generate assembled shapes from functional descriptions.

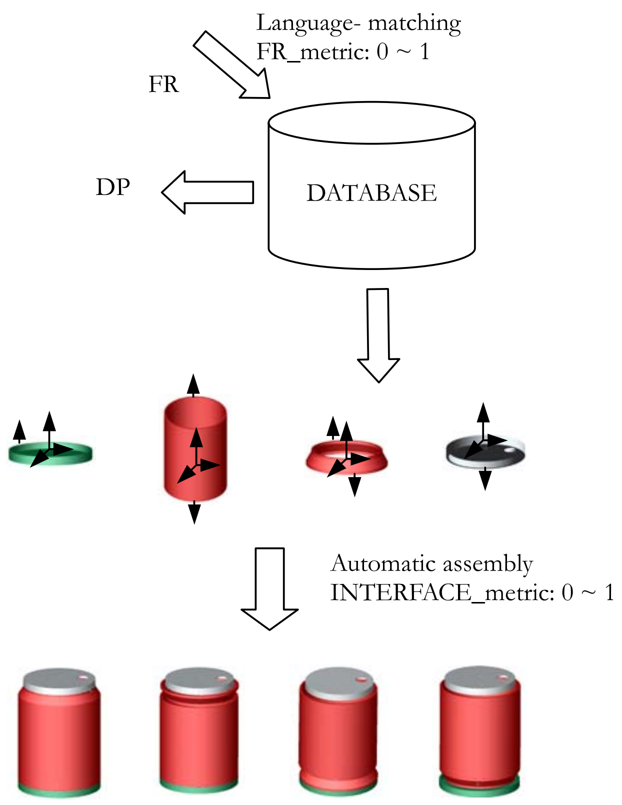
- Data structure
- Functional geometric feature
- Language matching
- Automatic assembly of the geometric features

Introductory example

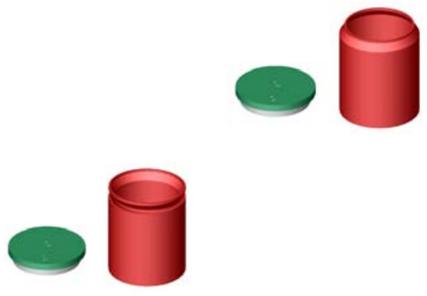
The following is one design scenario that serves as an introductory example for the proposed method. In this scenario, a designer starts a design problem by describing FRs and DPs in the V-model as shown in Figure 1. The designer describes a top-level FR and then a beverage can as the corresponding top-level DP. From the top-level DP, the designer can detail the design further by decomposing it into four lower-level FRs. Then, the corresponding underlined DPs (DP1 through DP4) and shapes can be found in the database by language matching between FR statements. The language-matching rate has been calculated to a value of FR_metric by a language-matching algorithm. Each FR has been linked to each 'Functional Geometric Feature (FGF),' which has its own interfaces (i.e. mating faces) in a local coordinate frame. Computer algorithms automatically assemble the FGFs into the candidate shapes using information on the interfaces of the FGFs. The algorithms rank each candidate shape by measuring its geometric interface-ability to a value of INTERFACE_metric and then discard lower-valued candidate shapes. From the candidate shapes with higher values of FR_metric and INTERFACE_metric, the designer can construct design matrices to finally decide the best shapes that satisfy all the FRs without any conflict. The shapes in Figure 1 (a) are determined as the best solutions among a lot of candidate shapes based on the INTERFACE_metric, and those in Figure 1 (b) must be discarded, because the corresponding values of INTERFACE_metric are lower that those in (a). Figure 1 (c) shows final results generated by scaling the shapes shown in (a) based on the boundary constraints.

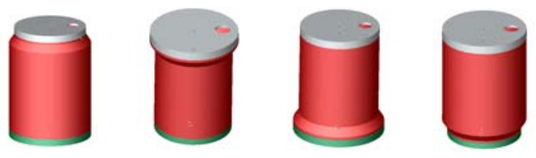
FR0 = contain and carry beverage
FR1 = enclose the bottom with resistance to an impact
FR2 = provide a volume to contain beverage
FR3 = reduce the material and increase stiffness of the body
FR4 = cover the body and provide pathway of beverage

DP0 = beverage can
DP1 = <u>bottom</u>
DP2 = <u>hollow cylinder</u>
DP3 = <u>conical section of the body</u>
DP4 = <u>top</u>



(a) Assembled shapes with INTERFACE_metric =1.0((2*3/6)*(3/3)*(3/3))





(b) Assembled shapes with INTERFACE_metric = 0.67((2*2/6)*(1/1)*(1+1)/2) or 0.583((2*3/6)*(1/1)*(1+(1+0.5)/2)/3)

(c) Shapes through scaling with INTERFACE_metric =1.0

Figure 1. Introductory example: beverage can

2 DATABASE

Figure 2 shows the system architecture and the flow chart. The flow chart shows how the proposed method can help designers in the V-model to generate assembled candidate shapes. The existing design knowledge should be collected and stored in the databases before the designers use the databases. The top two blocks on the flow chart show this knowledge engineering process. The flow chart has been modified from that of Thinking Design Machine. (Suh & Sekimoto 1990) The proposed CAD system must consists of knowledge engineering tool, databases, and a CAD engine. It is not easy to generalize design knowledge related to geometry. Thus, we collect design cases instead of generalizing all the existing design knowledge. (Aamodt & Plaza 1994) The design knowledge is collected as linked lists of FR-DP-FGF-INTERFACES and the corresponding hierarchies.

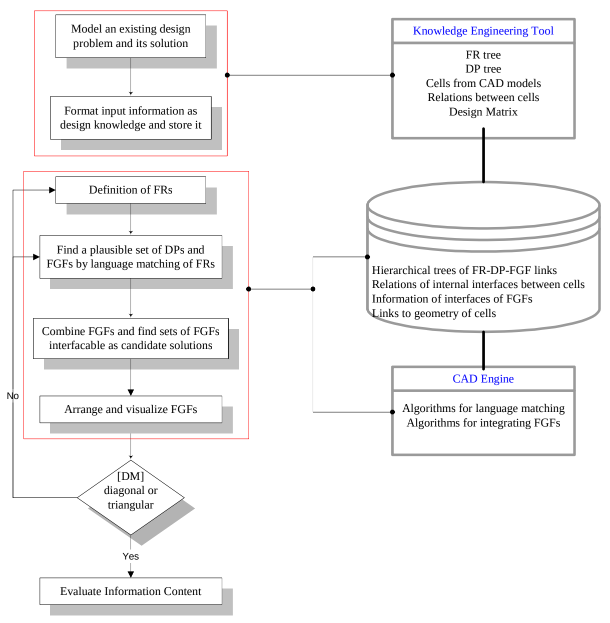

Model an existing design problem and its solution
Format input information as design knowledge and store it
Knowledge Engineering Tool
FR tree
DP tree
Cells from CAD models
Relations between cells
Design Matrix
Definition of FRs
Find a plausible set of DPs and FGFs by language matching of FRs
Combine FGFs and find sets of FGFs interfacable as candidate solutions
Arrange and visualize FGFs
Hierarchical trees of FR-DP-FGF links
Relations of internal interfaces between cells
Information of interfaces of FGFs
Links to geometry of cells
CAD Engine
Algorithms for language matching
Algorithms for integrating FGFs
No
[DM] diagonal or triangular
Yes
Evaluate Information Content

Figure 2. System architecture and flow chart

The V-model design process generates two types of design knowledge; one is knowledge about how FRs, DPs, and geometric entities (GEs) are related with each other, and the other is knowledge about how FRs, DPs, and GEs are decomposed in depth. A shape decomposition technique is used in the V-model to generate cells from the GEs in the CAD models, which decomposes the existing CAD models into solid cells based on FRs. These cells are stored and reused during the integration process. A set of cells mapped to an FR is called a functional geometric feature (FGF) to differentiate it from geometric features defined by only geometric characteristics. Figure 3 shows the design database, which store both types of design knowledge.

```
fr-decomposition("A", [FR0(FR1(FR1_1, FR1_2),
          FR2
          FR3(FR3_1(FR3_1_1, FR3_1_2), FR3_2)])

dp-decomposition("A", [DP0(DP1(DP1_1, DP1_2),
          DP2
          DP3(DP3_1(DP3_1_1, DP3_1_2), DP3_2)])

dp-geometry("DP1_A", "FGF1_A")
```

Figure 3. Fundamental structure of database

The database is used to search proper candidate DPs and FGFs for a given FR. The fr-decomposition database stores information of hierarchies of decomposed FRs and the dp-decomposition database stores information of hierarchies of decomposed DPs. A character, "A" is an index to the corresponding design case. The dp-geometry database stores information of FGFs mapped to the corresponding DPs. The fr-decomposition database and the dp-decomposition database can be constructed through the V-model design process.

Our hypothesis on FGFs is

"Functional geometric features (FGFs) can be geometrically separated from a whole solid body."[1]

This hypothesis must be true, if the V-model is used so that geometric entities can be mapped to each FR. However, there may exist more than one way to separate the geometric entities into solid cells in practice. This is because an FGF is not defined by geometric characteristics but by FRs, which is different from other techniques for feature recognition or cell decomposition. (Sakurai & Gossard 1990; Sakurai 1995; Wang & Kim 1998; Li & Liu 2002; Woo 2002) In this paper, different decompositions of a CAD model are considered as different design cases.

The guideline of separating correct FGFs satisfying a certain FR from the whole solid body for collecting a set of cells is as follows:

"To determine the correspondence between an FR and an FGF, the cells corresponding to the FGF can be removed from the whole solid body. If the resulting solid body can no longer satisfy the FR, the removed cells are the proper FGF for the FR."

[1] This is consistent with the Independence Axiom and the FR/DP mapping of axiomatic design.

In this paper, only *cut* operation has been used. The whole solid is decomposed into solid cells by the *cut* operation s shown in Figure 4.

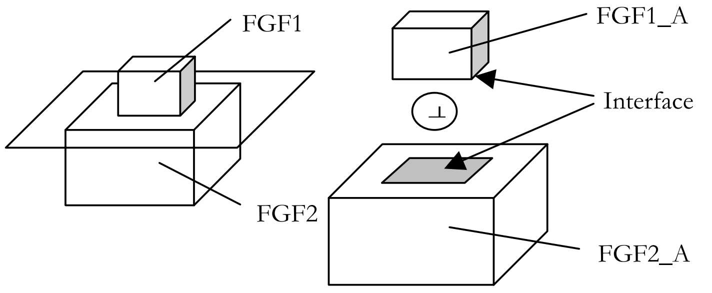


Figure 4. Cut operations

The cells of a lower level FGF is a subset or elements of its parent FGF, because FGFs are defined by zigzagging decomposition between functional domain and physical domain as explained in the part 1. Information on the geometry of the cell can be encapsulated, but only information on the interfaces of the cell is used for the computer algorithms to interface the cell. All the geometry of the cell is needed only for visualization of the assembled shape.

The cells of a lower level FGF is a subset or elements of its parent FGF, because FGFs are defined by zigzagging decomposition between functional domain and physical domain as explained in the part 1. Information on the geometry of the cell can be encapsulated, but only information on the interfaces of the cell is used for the computer algorithms to interface the cell. All the geometry of the cell is needed only for visualization of the assembled shape.

Interface types of mating faces can be classified as follows.

- Rigid attachment: two cells are merged into a solid. A face should be rigidly attached with another face.
- Assembly: two solid cells contact each other.
  - Static: two cells contact without relative motion between each other.
  - Kinematic: two cells contact with relative motion between each other. This interface can be classified further into "slide", "roll", and "rotate".

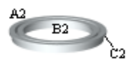
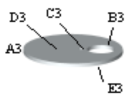
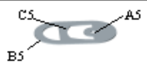
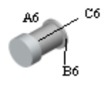
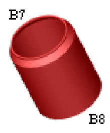
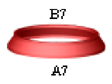
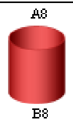
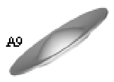
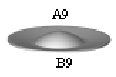
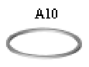
Two types of mating between components such as 'against,' and 'fits' have been considered in most mechanical assemblies. (Lee & Gossard 1985) [2] In this paper, the following types of interfaces are thought, but all kinds of rigid attachments have been grouped into only 'against:rigid_attachment.' Thus, four types of interfaces, 'against:rigid_attachment' 'against:assembly,' 'fits:assembly,' and 'tight-fits:assembly,' are mainly dealt with. A normal vector of the planar face and one point on the face are needed for 'against.' Two points on the centerline of each cylindrical face are needed for 'fits.' Below are the all kinds of interfaces must be considered.

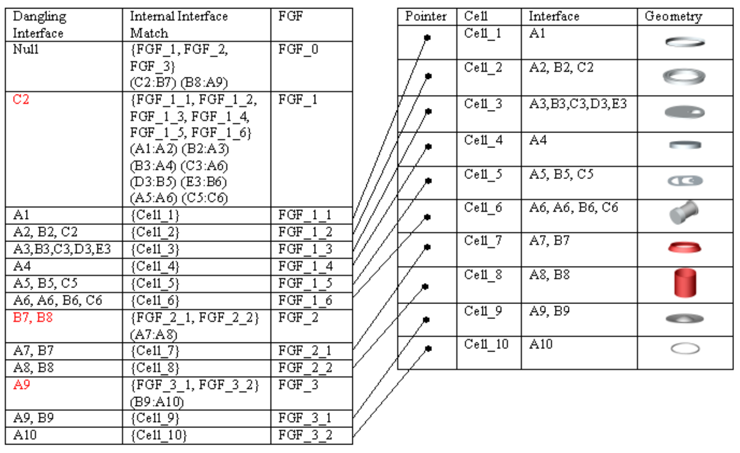
- Against:assembly: two planar faces are located on a plane with opposite normal vectors.
- Against_equal:rigid_attachment: all the boundaries of two planar faces are exactly matched together.
- Against_inside:rigid_attachment: a planar face is placed inside of the other planar face.
- Against_intersected:rigid_attachment: two planar faces are intersected on a plane.
- Fits:assembly: two centerlines of cylinder and hole are located on a line. This interface type is classified into 'cylinder,' or 'hole' for each mating face.
- Tight-fits: assembly: two centerlines of cylinder and hole are located on a line and two points on the cylindrical faces locates on a cylindrical face. This interface type is classified into 'cylinder,' or 'hole' for each mating face.

Table 1 shows a simple practice of how to generate the FGFs for a beverage can. The generated FGFs are decomposed by simple disassemblies or cut operations in the V-model. Figure 5 shows the detailed data structure between FGFs and cells for the beverage can example. Each leaf level FGF has been linked to each cell. Each cell keeps information on its interfaces, i.e. mating faces, and encapsulated geometry. A higher level FGF has been composed of a set of lower level FGFs. For example, FGF_0, FGF_1, FGF_2, or FGF_3 does not keep all the geometry of the corresponding shapes, but only a set of lower level FGFs, a list of internally matched interfaces between the lower level FGFs, and dangling interfaces that have not been used, but will be used to be interfaced. The shapes of the FGFs and their dangling interfaces are shown in the Table 1. As notably shown in the Table, two A6's are stored for FGF1_6 in the database, if one mating face (A6) may be used for mating two other faces (C3 and A5). This will give complete interface matching based on the existing knowledge on interface.

[2] 'Against' applies between planar faces and 'fits' applies between a solid cylinder and a hole.

Table 1. FRs, DPs and the corresponding FGFs of beverage can

FRs	DPs	FGFs	Dangling interfaces
FR0 = contain and carry beverage	DP0 = beverage can		{FGF_1, FGF_2, FGF_3}
FR1 = cover the body and provide pathway of beverage	DP1 = top		{FGF_1_1, FGF_1_2, FGF_1_3, FGF_1_4, FGF_1_5, FGF_1_6} C2(against:rigid_attachment)
FR1_1 = guide the flow of beverage	DP1_1 = cylindrical extrusion wall		{Cell_1} A1(against:rigid_attachment)
FR1_2 = store beverage	DP1_2 = circular pocket		{Cell_2} A2(against:rigid_attachment) B2(tight-fits:assembly:hole) C2(against:rigid_attachment)
FR1_3 = cover body of beverage can and flow beverage out	DP1_3 = circular cover with through hole		{Cell_3} A3(tight-fits:assembly:cylinder) B3(tight-fits:assembly:hole) C3(tight-fits:assembly:hole) D3(against:assembly) E3(against:assembly)
FR1_4 = enclose the hole before opening	DP1_4 = gate		{Cell_4} A4(tight-fits:assembly:cylinder)
FR1_5 = open the gate	DP1_5 = opening tab		{Cell_5} A5(tight-fits:assembly:hole) B5(against:assembly) C5(against:assembly)
FR1_6 = fasten opening tab on the cover	DP1_6 = fastener of opening tab to cover		{Cell_6} A6(tight-fits:assembly:cylinder) A6(tight-fits:assembly:cylinder) B6(against:assembly) C6(against:assembly)
FR2 = provide volume to contain beverage with a certain stiffness	DP2 = cylindrical body		{FGF_2_1, FGF_2_2} B7(against:rigid_attachment) B8(against:rigid_attachment)
FR2_1 = reduce the material and increase stiffness of the body	DP2_1 = conical section of the body		{Cell_7} A7(against:rigid_attachment) B7(against:rigid_attachment)
FR2_2 = provide a volume to contain beverage	FR2_2 = hollow cylinder		{Cell_8} A8(against:rigid_attachment) B8(against:rigid_attachment)
FR3 = enclose the bottom with resistance to an impact	DP3 = bottom		{FGF_3_1, FGF_3_2} A9(against:rigid_attachment)
FR3_1 = withstand a moderate impact when the can is dropped	DP3_1 = curvature of the bottom		{Cell_9} A9(against:rigid_attachment) B9(against:rigid_attachment)
FR3_2 = stand the beverage can and stack on top	DP3_2 = cylindrical extrusion		{Cell_10} A10(against:rigid_attachment)



Dangling Interface	Internal Interface Match	FGF
Null	{FGF_1, FGF_2, FGF_3} (C2:B7) (B8:A9)	FGF_0
C2	{FGF_1_1, FGF_1_2, FGF_1_3, FGF_1_4, FGF_1_5, FGF_1_6} (A1:A2) (B2:A3) (B3:A4) (C3:A6) (D3:B5) (E3:B6) (A5:A6) (C5:C6)	FGF_1
A1	{Cell_1}	FGF_1_1
A2, B2, C2	{Cell_2}	FGF_1_2
A3,B3,C3,D3,E3	{Cell_3}	FGF_1_3
A4	{Cell_4}	FGF_1_4
A5, B5, C5	{Cell_5}	FGF_1_5
A6, A6, B6, C6	{Cell_6}	FGF_1_6
B7, B8	{FGF_2_1, FGF_2_2} (A7:A8)	FGF_2
A7, B7	{Cell_7}	FGF_2_1
A8, B8	{Cell_8}	FGF_2_2
A9	{FGF_3_1, FGF_3_2} (B9:A10)	FGF_3
A9, B9	{Cell_9}	FGF_3_1
A10	{Cell_10}	FGF_3_2

Pointer	Cell	Interface	Geometry
●	Cell_1	A1	
●	Cell_2	A2, B2, C2	
●	Cell_3	A3,B3,C3,D3,E3	
●	Cell_4	A4	
●	Cell_5	A5, B5, C5	
●	Cell_6	A6, A6, B6, C6	
●	Cell_7	A7, B7	
●	Cell_8	A8, B8	
●	Cell_9	A9, B9	
●	Cell_10	A10	

Figure 5. Data structure between FGFs and cells for beverage can

3 GENERATION OF DESIGN SOLUTIONS

The first step of generation of design solutions is language matching between a query FR and saved FRs in the databases. Through this step, any candidate FGFs are found. In the second step, the candidate FGFs should be combined and assembled as candidate shapes. During the step, geometric interface-ability between the FGFs must be checked to rank the candidate shapes based on the geometric interface-ability.

3.1 LANGUAGE MATCHING OF FRs

In language matching of FRs, computer algorithm firstly searches all the saved FRs similar to a query FR in the fr-database. If the algorithm found an FR1 in a hierarchy "A" and an FR2 in a hierarchy "B" from a query FR (FR0) in the case base, the corresponding "DP1_A" and "DP2_B" can be thought of as different design solutions to satisfy the query FR. Many DPs can be found for a query FR as candidate solutions through this process. The matching a query FR (FR0) to DP1_A and DP2_B can be expressed as

FR0 \$ DP1_A, DP2_B.

Each DP has been also linked to each FGF in the database. Thus, the language matching finally gives

FR0 \$ FGF1_A, FGF2_B.

One difference in the structure of the database proposed in this paper from that of Thinking Design Machine (Suh and Sekimoto 1990) is a stored tree structure of FRs, DPs and FGFs. Hierarchical information flow using the tree structure is more compact and effective, because all the lower branches from a certain node in a tree can be extracted.

The language matching between a query FR and the saved FRs is performed by counting the frequency of appearing common words and the sequence of words. Both the query FR and a selected saved FR are decomposed into a set of words and pairs of words. For example, "engage the pawl to the engaged position," is decomposed into two sets as

Word set = {engage, pawl, to, engaged, position}
Word pair set = {(engage, pawl), (pawl, to), (to, engaged), (engaged, position)}.

Here, 'the,' 'a,' 'and,' 'or,' and pronouns are trimmed, because the word is not closely related to the meaning of the sentence. The word set is used to check the number of words common in the query FR and the saved FR. The word pair set is used to check the sequence of the matched words. When the words are compared, synonyms of each word are brought and matched by WordNet. (Miller et al. 1993) The metric is mathematically represented as

$$FR\_metric = \frac{2 \times \#\ of\ matched\ words\ +\ \#\ of\ matched\ word\ pairs\ of\ saved\ FR}{\#\ of\ total\ words\ +\ \#\ of\ matched\ word\ pairs\ of\ input\ FR} \quad \text{(a)}$$

If a query FR is exactly the same as a saved FR, this metric gives a value, 1.0, for the language matching between the FR statements. If not, it calculates the difference between the FR statements by the equation (a) defined as FR_metric. Synonym checking reduces errors of describing a word. A threshold value between 0 and 1 can be set to control the search space using the FR_metric. Higher threshold value means smaller search space of language matching.

3.2 ASSEMBLED SHAPE GENERATION

Integration of various FGFs to a complete shape is one of the key factors for geometry design with FRs. In the integration process, candidate FGFs are assembled, and geometric interface-ability between them are measured. The combination of FGFs is performed by genetic algorithm (GA) and geometric interface-ability is measured by an algorithm that automatically simulates human assembling operations to integrate them in the physical space. The following attributes are checked to quantify the geometric interface-ability. Firstly, interface types (IT) are checked by graph matching of mating faces of FGFs. Graph matching is known as NP-complete problem. (Gold and Rangarajan 1996) Secondly, relative angles between faces are measured. Thirdly, the success of positioning mating faces is checked.

- Interface type (IT): against:assembly, against:rigid_attachment, fits:assembly, tight-fits:assembly
- Angles between faces (AF)
- Positions (Pos)

3.2.1 COMBINATIONS OF FGFs

If three sets of candidate FGFs to satisfy FR1, FR2, and FR3 are supposed as

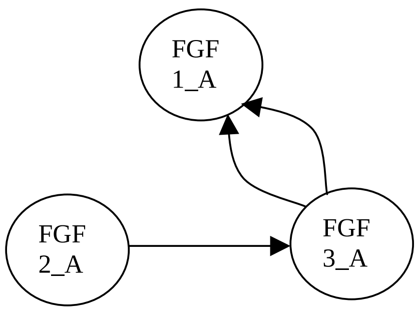
{
FR1 $ FGF1_A, FGF1_B
FR2 $ FGF2_A, FGF2_B, FGF2_C
FR3 $ FGF3_A, FGF3_C
}.

The possible combinations between the FGFs are 2*3*2 = 12. For each possible combination, there exist possible combinations of connectivity between FGFs.

Because there are a lot of possible combinations for interfacing FGFs and the combinations are performed by matching discrete numbers, we chose GA (Genetic Algorithm) to more efficiently check the combinations described above instead of using an algorithm for generating all the combinations. GA has been proved to solve combinatorial problems that have discrete numbers of design variables. GA chromosome can also present assembly sequences between FGFs including mappings each discrete number of gene to mating faces. GA can evolve with an acceptable speed to maximize the fitness. A sample chromosome used in GA has been defined as

FGFs :	1_A	2_A	2_A	3_A	3_A	1_A
Chromosome:	2	0	2	3	4	6
Interface pairs :	Pair1		Pair2		Pair3	

Each interface pair consists of two genes, which are mapped to the encoded mating faces of an FGF. The first FGF in a pair is a base component, which is fixed in the physical space for assembling operations. The second FGF is an interfacing component, which translates or rotates to be assembled to the base component. In Figure 6, a directional graph shows connectivity and also assembly sequences between three FGFs, based on what the chromosome described above.



Assembly sequence :
FGF2_A -> FGF3_A / FGF3_A -> FGF1_A

Figure 6. Connections and assembly sequences between FGFs

The number of pairs in a chromosome is ${}_{m}C_{2}$ = n, where m is the number of FGFs. Here, the number of pairs in the sample chromosome is ${}_{3}C_{2}$ = 3. Table 2 shows the codes for each gene, which are mapped to a set of mating faces. If each FGF has m mating faces, the number of codes for each FGF is ${}_{m}C_{1}+{}_{m}C_{2}+\ldots+{}_{m}C_{m}$. If m is 3, the number of codes is 6. Thus, each gene has a code number between 0 and 7. 0 means no connection between two FGFs in a pair.

Table 2. Encoding of interfaces of FGFs into genes

FGF	Code	Mating faces	FGF	Code	Mating faces	FGF	Code	Mating faces
	0	Null						
1_A	1	A	2_A	1	D	3_A	1	G
	2	B		2	E		2	H
	3	C		3	F		3	I
	4	A & B		4	D & E		4	G & H
	5	B & C		5	E & F		5	H & I
	6	C & A		6	F & G		6	I & G
	7	A & B & C		7	D & E & F		7	G & H & I

Table 3 shows the mapping between code numbers and the corresponding mating faces and the possible assembly sequences for the chromosome (2 0 2 3 4 6). The sequence of the genes represents assembly sequences of FGFs. There exists only possibility of interfaces between FGF_2 and FGF_3, because the code number, 2 or 3, represents one mating face, E or I. However, there are two possible combinations between mating faces of FGF_3 and FGF_1 as shown in Table 3. The assembly sequences of mating faces in the third pair can be G:C -> H:A or H:A-> G:C.

Table 3. Combinations of interfacing faces represented by a chromosome (2 0 2 3 4 6)

FGF 1	FGF_ 2	FGF_ 2	FGF_ 3	FGF_ 3	FGF_ 1
2	0	2	3	4	6
No interface		E:I		G:C & H:A	
				H:C & G:A	

GA ranks matching of interface types (IT) of mating faces. High fitness valued chromosome represents high possibility of matching of IT's of the mating faces. 1.0 is assigned for the fitness, if interface types of all the mating faces are completely matched. 0.0 is assigned, if there is no IT matching. The fitness has been represented as

$$fitness = IT\_metric = \frac{2 \times \#\ of\ matched\ ITs}{\#\ of\ mating\ faces} \quad \text{(b)}$$

Figure 7 shows the whole process for generating assembled shapes. Information of the chromosomes evolved in GA is passed to the assembling algorithm. The assembling algorithm automatically assembles FGFs, and measures INTERFACE_metric to satisfy pre-defined mating conditions for the automatic assembled shape generation. Gluing may include a detailed shape deformation process or a shape optimization technique. Thus, it is out of scope of this paper.

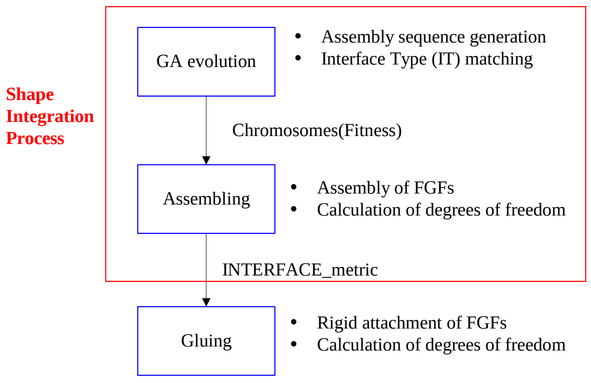

Shape Integration Process
GA evolution
• Assembly sequence generation
• Interface Type (IT) matching
Chromosomes(Fitness)
Assembling
• Assembly of FGFs
• Calculation of degrees of freedom
INTERFACE_metric
Gluing
• Rigid attachment of FGFs
• Calculation of degrees of freedom

Figure 7. Shape integration process of FGFs

3.2.2 AUTOMATIC ASSEMBLY AND INTERFACE-ABILITY OF FGFs

The goal of the assembling process is to automatically assemble the FGFs based on mating conditions and assembly sequences given from the chromosomes and to measure geometric interface-ability between the FGFs to a value of INTERFACE_metric. This assembling process can be implemented as a computer algorithm that emulates human assembling operations and solves assembly constraint equations. A flow chart in Figure 8 shows information flow based on which the computer algorithm performs the assembling operations. The algorithm performs coordinating, orienting, and positioning of the FGFs and calculates INTERFACE_metric. Here, coordinating is locating a local coordinate frame to a proper position. Orienting is only rotations of the FGFs to match directions of mating faces, and positioning is only translations of the FGFs to locate the mating faces to the given position.

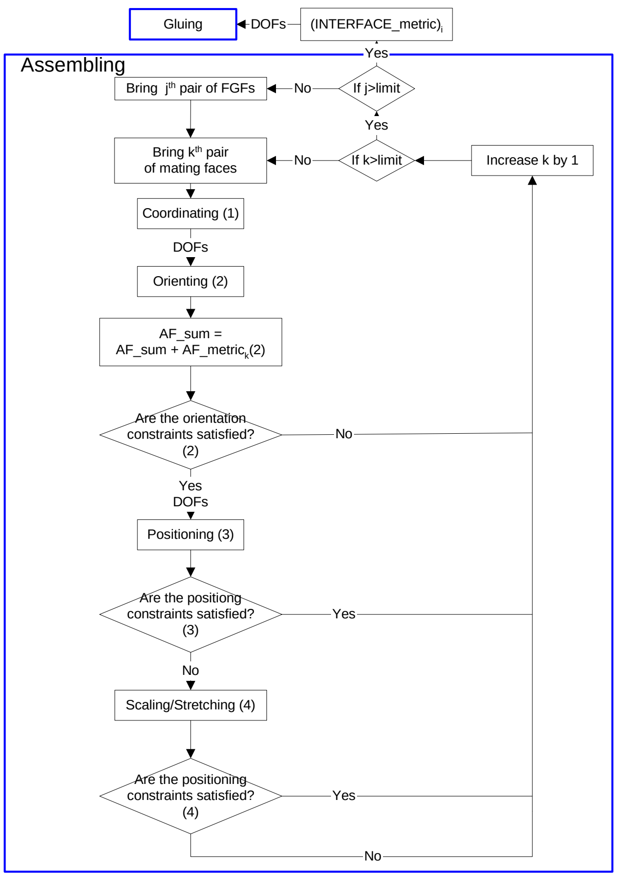

Gluing
DOFs
(INTERFACE_metric)$_i$
Assembling
Yes
Bring jth pair of FGFs
No
If j>limit
Yes
Bring kth pair of mating faces
No
If k>limit
Increase k by 1
Coordinating (1)
DOFs
Orienting (2)
AF_sum = AF_sum + AF_metric$_k$(2)
Are the orientation constraints satisfied? (2)
No
Yes
DOFs
Positioning (3)
Are the positiong constraints satisfied? (3)
Yes
No
Scaling/Stretching (4)
Are the positioning constraints satisfied? (4)
Yes
No

Figure 8. Flow chart for assembling process

The algorithm brings chromosomes from the highest value to the lower value. For a chromosome (2 0 4 6 4 6) according to the codes in Table 2, there are four possible combinations as shown in Table 4.

Table 4. Combinations of interfacing mating faces in a chromosome

Sequence	FGF1	FGF_2	FGF_2	FGF_3	FGF_3	FGF_1
	2	0	4	6	4	6
1	No interface		D:I -> E:G		G:C -> H:A	
2	No interface		D:G -> E:I		G:C -> H:A	
3	No interface		D:I -> E:G		H:C -> G:A	
4	No interface		D:G -> E:I		H:C -> G:A	

The algorithm recursively brings ith sequence (i = 1, 2, 3, 4) and jth interface pair of FGFs (j = 1, 2, 3) in Table 4, and kth pair of mating faces (k = 1, 2) in pair 2 and pair 3. In an interface pair, the first FGF is a base component and the second FGF is an interfacing component. Each FGF has its local coordinate frame. Thus, the first step of assembling operations is to locate the local coordinate frame with respect to the absolute coordinate frame. This is called 'coordinating.' Once the 'coordinating' has been done, the algorithm rotates the second FGF to match orientation of its mating face with that of the base FGF. This operation is called 'orienting.' Then, it translates the second FGF to locate its mating face to a proper position to satisfy mating conditions. This operation is called 'positioning.'

For example in Table 4, the algorithm recursively brings assembly sequences, and automatically assembles the three FGFs. In sequence 1, it brings 2nd interface pair of FGFs first, because there is no interface between FGF1 and FGF2. Then, it brings 1st pair of mating faces, D:I. It firstly locates the local coordinate frames of the FGF2 and the FGF3 coincidently with the absolute coordinate frame. Then, it rotates the FGF3 to a direction to match the orientation of I with that of D. In the orienting operation, unit normal vectors of two planar mating faces must be parallel and opposite for 'against' and two centerlines must be parallel for 'fits'. The angle differences between mating faces are calculated and updated for AF_metric during the operation. Then, it translates the FGF3 for positioning I to D. In the positioning operation, a point on the planar mating face must be on the other planar mating face for 'against' and two centerlines must be located on the same line for 'fits'. If scaling or stretching is needed, it is performed. If it fails positioning FGF3, it gives a defined value (= 0.5 in this paper) less than 1.0 for Pos_metric. The algorithm generates output position and orientation in a coordinate frame with allowable DOFs. It recursively checks the interface-ability of the second pair of mating faces, E:G, and then do the same steps for 3rd interface pair of FGFs (FGF3 and FGF1). Finally, three values of IT_metric determined by GA evolutions, AF_metric determined during orienting operation, and Pos_metric determined by positioning operation are multiplied into a value of INTERFACE_metric, which represents the geometric interface-ability for a given chromosome. The value of the INTERFACE_metric has been scaled between 0.0 and 1.0. 1.0 means complete interface-ability for the given FGFs and 0.0 means that none of pairs can be interfaced. Consequently, the proposed algorithm produces the assembled shapes with allowable DOFs, and ranks the assembled shapes based on the values of INTERFACE_metric.

All the operations are performed by transformation matrix (Paul 1984) and the corresponding mathematical equations represented as follows.

$$T=\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_R=\begin{bmatrix} n_x & o_x & a_x & 0 \\ n_y & o_y & a_y & 0 \\ n_z & o_z & a_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_T=\begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(c)}$$

where T_R denotes only rotation and T_T denotes only translation. The elements of T_R matrix always have the following relations represented as six mathematical equations.

$$
\begin{aligned}
&n_x^{\,2} + n_y^{\,2} + n_z^{\,2} = 1 \\
&o_x^{\,2} + o_y^{\,2} + o_z^{\,2} = 1 \\
&n_x o_x + n_y o_y + n_z o_z = 0 \\
&a_x = n_y o_z - n_z o_y \\
&a_y = o_x n_z - n_x o_z \\
&a_z = n_x o_y - n_y o_x
\end{aligned}
\qquad \text{(d)}
$$

DOFs are represented by $T_R(n)$ or $T_T(n)$. Here, n is a number less than or equal to three which represent degrees of freedom for rotation or translation. If an interfacing FGF has [$T_R(1)$, $T_T(2)$], it has one allowable DOF for rotation and two for translation.

Only scaling or stretching is performed for special cases in this paper to show its effects to generating complete assembled shapes. An FGF is originally created with its own purpose. Thus, the more the shape of the FGF deforms, the more its purpose is lost. In this sense, only the severe shape deformation is not allowed, but scaling or stretching is performed. Scaling of shapes means enlarging or reducing the size of the shapes in all directions in the physical space and stretching means doing it in one direction. The basic deformation matrix is a special form of the transformation matrix, which is

$$
D = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(e)}
$$

where S_x, S_y, and S_z are scaling factors. If all S_x, S_y, and S_z are equal, this matrix can be used for scaling. Otherwise, only one of S_x, S_y, and S_z is not zero, it is used for stretching.

4 INTEGRATION OF FGFS FOR BEVERAGE CAN DESIGN

This section explains the details on how the results of the introductory example shown in Figure 1 have been generated. Figure 9 shows the corresponding GA codes and one of GA chromosomes.

FGF	Gene	Mating faces	FGF	Gene	Mating faces	FGF	Gene	Mating faces	FGF	Gene	Mating faces
	0										
1	1	A1	2	1	A2	3	1	A3	4	1	A4
				2	B2		2	B3			
				3	A2&B2		3	A3&B3			

(a) GA codes of mating faces

FGF1	FGF2	FGF2	FGF3	FGF3	FGF4	FGF4	FGF2	FGF4	FGF1	FGF3	FGF1
1	1	2	1	2	1	0	0	0	0	0	0

(b) Example chromosome

Figure 9. GA codes of mating faces and an example chromosome

GA evolves to search the best matching of interface types to maximize fitness value defined as IT_metric. Then, the proposed assembly methods are applied to integrate the FGFs into the assembled shapes. In this example, GA evolves for 20 generations with 300 populations, and then against:assembly ($T_T(3)$, $T_R(3)$) and against:assembly ($T_T(2)$, $T_R(1)$) are used. Figure 1 shows the resultant candidate shapes generated by the assembly methods. Figure 1 (a) shows the assembled shapes after one more constraint, 'concentric,' has been applied to all the candidate assembled shapes. All the candidate shapes have INTERFACE_metric = 1.0. It means that all the candidate shapes are completely oriented and positioned satisfying all the pre-defined mating conditions between FGFs based on the proposed metric. Figure 1 (b) shows some other assembled shapes of INTERFACE_metric = 0.67 or 0.583. All these candidate shapes can be generated from three different chromosomes as follows.

FGF1	FGF2	FGF2	FGF3	FGF3	FGF4	FGF4	FGF2	FGF4	FGF1	FGF3	FGF1
0	0	1	1	0	0	0	0	1	1	0	0

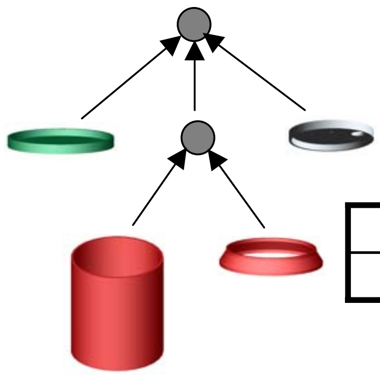
FGF1	FGF2	FGF2	FGF3	FGF3	FGF4	FGF4	FGF2	FGF4	FGF1	FGF3	FGF1
0	0	1	2	0	0	0	0	1	1	0	0

FGF1	FGF2	FGF2	FGF3	FGF3	FGF4	FGF4	FGF2	FGF4	FGF1	FGF3	FGF1
0	0	3	3	0	0	0	0	1	1	0	0

In the first chromosome, one pair of mating faces has been missed to be interfaced among total three pairs of mating faces. Thus, IT_metric is 0.67 (= 2*2/6). However, all the mating faces given by the chromosome can be correctly oriented and positioned: AF_metric = 1.0 and Pos_metric = 1.0. The first chromosome represents the first candidate shape in (b) and has INTERFACE_metric = 0.67 from IT_metric*AF_metric*Pos_metric. The same analysis can be applied to the second chromosome that represents the second candidate shape in (b). The third chromosome represents interfaces of all the pairs of mating faces. In this case, IT_metric is 1.0 (= 3*3/6), because all the interface types (against:assembly) can be matched correctly. Also, all the mating faces can be oriented correctly so that AF_metric = 1.0. However, one pair of mating faces cannot be positioned geometrically. This fact gives Pos_metric = 0.583 ((1+(1+0.5)/2)/3). INTERFACE_metric for the third chromosome is 0.583 from IT_metric*AF_metric*Pos_metric. There are two combinations for interfacing two pairs of mating faces; A2:A3 -> B2:B3 and A2:B3 -> B2:A3. These two combinations show two different candidate shapes in (b). The value of INTERFACE_metric in this case highly depends on a value, which is pre-set to 0.5 in Pos_metric. The pre-set value is used for degrading INTERFACE_metric if positioning is failed. If the value has been set to a larger number, the value of INTERFACE_metric will be increased for the same design problem.

Figure 10 shows the advantage of the V-model. If the body of the beverage can is assembled first, and then the bottom and the top are assembled to the assembled body, the chromosome for integration of the body has 2 genes, and that for the next assembly has 6 genes. Actual GA evolution of the two-step assembling is 3 generations with 200 populations plus 10 generations with 300 populations to get the same results as shown in Figure 1. Total calculations are 3600 chromosomes for this case. Thus, it is less than 6000 calculations, in which 12-gene chromosome has been used. This is rough estimation for GA calculations, but explains the advantage of top-down decompositions for bottom-up integrations in the V-model.

FGF1	FGF23	FGF23	FGF4	FGF4	FGF1
1	1	2	1	2	1



FGF2	FGF3
1	1

Figure 10. Advantage of top-down decomposition in the V-model

5 CONCLUSIONS

This paper presented a method that automatically generates assembled shapes from input functional descriptions. The method is generally applicable to all kinds of shape design problems, not ad-hoc approach for a specific design problem. It includes V-model design process, database, and computer algorithms for searching candidate FGFs and automatically assembling them. It generated interesting and reasonable shapes, which are satisfying input FRs, as shown in application examples. FR_metric (0.0 ~ 1.0) and INTERFACE_metric (0.0 ~ 1.0) have been made to rank the generated shapes and to support decision-making by computer algorithms and/or by human designers. FR_metric is a language-matching rate between an input FR statement and a saved FR statement in the database. INTERFACE_metric (0.0 ~ 1.0) is an interface-matching rate (interface type and geometric assemble-ability) between candidate FGFs. The computer algorithms can filter a lot of the unreasonable shapes based on the FR_metric first, and then the INTERFACE_metric. The candidate shapes with FR_metric = 1.0 and INTERFACE_metric = 1.0 can be considered as good design solutions among a lot of generated shapes. Design matrix shown in the part 1 must be constructed for each candidate shape to check satisfaction of all the FRs and finally to determine the best design solution.

This sort of shape generation using database is a combinatorial problem, which has a lot of combinations of FGFs and needs high computational power. This paper used several important techniques to handle the heavy computational load in a manageable level.

- All the detailed geometry has been encapsulated. The developed computer algorithms use only information on interfaces, i.e. mating faces. Thus, the number of combinations and computational complexity are relatively small comparing to other approaches using complex whole geometry.

- Several modules of the computer algorithms have been used. The language-matching module uses FR_metric and reduces a solution space into a certain level set by a threshold value for the FR_metric. The GA module reduces the solution space again by matching pre-defined interface-types between FGFs using IT_metric. Then, automatic assembling algorithm operates assembling of FGFs and calculates values of INTERFACE_metric. Because solution spaces are reduced consecutively through the steps, computational complexity of automatic assembling algorithm is in a manageable level.

Bottom-up integrations of FGFs in top-down hierarchical tree can eliminate many combinations and reduce search space for allowable combinations comparing to integrations only by bottom-up manner.

6 REFERENCES

Pratt MJ, Wilson PH (1985) Requirements of support of form features in a solid modeling system. CAM-I Report, R-85-ASPP-01, Arlington, Texas, USA

Dixon JR, Cunningham JJ (1987) Research in designing with features. Proceedings of the IFIP TC 5/WG 5.2 Workshop on Intelligent CAD, 137 – 148, Boston, MA. USA

Sreevalsan PC, Shah JJ (1992) Unification of form feature definition method. Proceedings of the IFIP WG 5.2 Working Conference on Intelligent Computer Aided Design, 83 – 99, Columbus, OH, USA

Suh NP, Sekimoto S (1990) Design of thinking design machine. CIRP Annals

Aamodt A, Plaza E (1994) Case-based reasoning: fundamental issues, methodological variations, and system approaches. AI Communications 7(1): 39 – 59

Sakurai H, Gossard DC (1990) Recognizing shape features in solid models. IEEE Computer Graphics and Applications 10(5): 22 – 32

Sakurai H (1995) Volume decomposition and feature recognition: part I – polyhedral objects. Computer Aided Design 27(11): 833 – 843

Wang E, Kim YS (1998) Form feature recognition using convex decomposition: results presented at the 1997 ASME CIE feature panel session. Computer Aided Design 30(13): 983 – 989

Li B, Liu J (2002) Detail feature recognition and decomposition in solid model. Computer Aided Design 34: 405 – 414

Woo Y (2002) Fast cell-based decomposition and applications to solid modeling. Computer Aided Design accepted

Lee K, Gossard DC (1985) A hierarchical data structure for representing assemblies: part 1. Computer Aided Design 17(1): 15 – 19

Miller GA, Beckwith R, Fellbaum C, Gross D, Miller K (1993) Introduction to WordNet : an on-line lexical database. http://www.cogsci.princeton.edu/~wn

Gold R, Rangarajan A (1996) A graduated assignment algorithm for graph matching. IEEE Trans Pattern Anal Mach Intell 18(4): 377 – 388

Paul RP (1984) Robot manipulators: mathematics, programming and control. The MIT Press

R. D. Coyne and J. S. Gero, "Knowledge-Based Design Systems," Addison-Wesley, 1990