Comparison of 2D and 3D objects for parts reuse

Jean-Pierre JUNG, Sandrine LEINEN

L.R.I.M.
Université de Metz
Ile du Saulcy
57045 Metz cedex
France
{jung,leinen}@lrim.sciences.univ-metz.fr

ABSTRACT

The design of new parts with a CAD software is a task which requires much reflection of the designer. We propose to assist him for this task by integrating the concept of re-use. Indeed, we will associate with the system of CAD a functionality allowing to detect automatically equalities or inclusions between the part in design and the library's data. The method employed in 2D, is based on a method of objects constraints graph normalization. Then, it adapts the graph formalism of the concept of CSP (Constraint Satisfaction Problem) for the comparison of normalized representations. In 3D, the method employed is based on a representation of the geometrical volumical model called DSG (Destructive Solid Geometry) and defined a rules system making possible the comparison of volumical representations.

Keywords : Computer Aided Design, constraints, modelization, Constrainst Satisfaction Problem, Destructive Solid Geometry

INTRODUCTION

When designing mechanical part, for example, the re-use is a fundamental concept to avoid a considerable loss of time. Indeed, it can exist libraries of parts previously built by possibly other users of the system and who contain parts which look like the part in design.

We propose an automatic detection of this likeness by the system and a proposal to the user of solutions which he can re-use with or without modifications. The advantage of this proposal is that the user of the system is really assisted, he does not have to know the contents of the libraries, it is the system which manages the information contained in these libraries. On the other hand, certain parameters must be fixed by the user so that this assistance of the system is not heavy or ineffective. We propose, either to leave choice to user to determine interactively the instant where he wish that the system search a part equivalent in the library, or that he fixed parameters like the number of elements or the number of constraints that the object in construction must contain before searching likeness with elements of libraries.

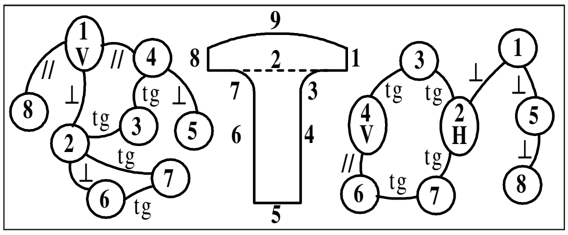
We propose to effectively manage the libraries at disposal, to classify the elements by family by respecting classes of equivalence: this first distribution accelerates search considerably. Moreover, it should be noted that the comparison of the elements between them is based on a comparison primarily of their graph of constraints. This graph of constraints contains all the components of an object and the existing relations between these various components. However, an object can be described by expressing its constraints in various manners. A same object can have, consequently, two equivalent graphs of constraints; then we must representing it only once in the library and finding any equivalence between the object in construction and its similar in the library.

Initially, we proposed in 2D a method of normalization of the constraints graph of the objects allowing to obtain a single representation for any object ([Leine97], [Leine98]). Then, we implemented two methods of comparison of the normalized constraints graphs; one algorithmic and the other based on the concept of CSP.

The principle of comparison of objects 2D representations is based on the concept of identical elements matching thanks to the processes of normalization implemented initially. In 3D, the comparison of representations of object will constitute to find matching between elements equivalent but not necessarily identical since we do not consider normalization preliminary

<u>Normalization and comparison of the objects representations in 2D</u>

An object is represented by its components and the relations inter-components. We chose to model this information in a constraints graph. In our study, the nodes of the graph represent segments or arcs and the relations labelling the arcs are 2D geometrical constraints like tangency, parallelism, perpendicularity,... The example of Fig.1. shows the no-unicity of this type of representation.



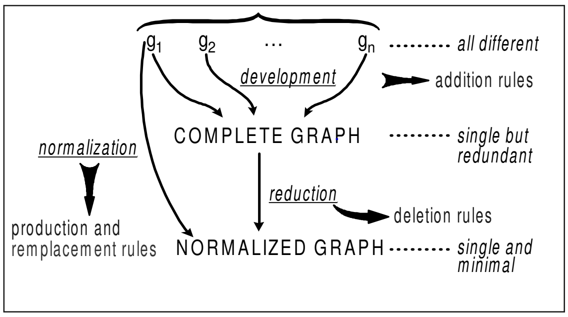
A rivet, two representations
Figure 1

This no-unicity is very nuisance, as we saw in introduction, to find in a reliable way the equality between two elements or to preserve in a single way a part in a library. This is why we propose a method of normalization of this representation.

<u>Normalized représentation</u>

A representation really simple to obtain is the representation of the whole of the constraints in a complete graph: absolutely all the relations between components are modelled. This graph contains a great number of redundancies and quickly becomes unusable because of its size which increases in an exponential way. We can then consider two solutions (Fig.2.):

– to simplify the complete graph by eliminating all the redundancies;
– to transform any initial graph into a normalized graph of minimal size.



Graph normalization
Figure 2

The second solution is more delicate to implement because of the termination of the process. On the other hand, it avoids building and handling even temporarily the complete graph. We thus established a set of rules allowing to obtain a restricted normalized graph.

In our example, the set of rules can be written easily by considering the properties of the Euclidean geometry. This system of rules is applied to the set of constraints until this set is stable. From the convergence of this series of constraints sets, we can deduce unicity of the solution. We demonstrate that this series converged towards a single solution because this series is monotonic and raised. This demonstration is based on the fact that we consider only rules of addition and replacement (no suppression) and that a order of application of the rules is defined taking account of the constraints priorities. A relation of order could be defined between the various sets obtained.

We distinguish two types of rules: rules of production and rules of replacement. The figure Fig.3. shows an example of each type of rule. For the first rule, among the three relations of parallelism one is redundant but we do not have any means of always removing the same relation. To avoid any ambiguity, the three relations of parallelism are thus preserved. So, we define the rule of production which adds a constraint of parallelism when there are already two. The rules of replacement replace a constraint by an other constraint with higher priority. The information of horizontality being more important than perpendicularity, so we replace D1⊥D2 by D2 Horizontal knowing that D1 is Vertical.

Poduction rules :

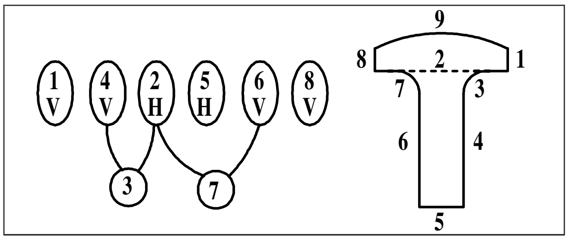
(D1 // D2) and (D1 // D3) ⇒D2 // D3

Replacement rules :

~~(D1⊥D2)~~ and D1 Vertical ⇒D2 Horizontal

Production and replacement rules
Figure 3

For the part previously considered, we obtain the following normalized graph where we are privileged the relations of horizontality and verticality then perpendicularity (Fig.4.).

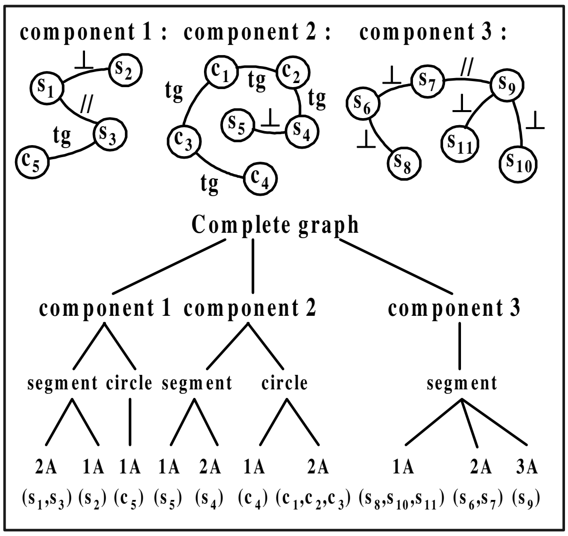


Normalized graph
Figure 4

Comparison of normalized representations

The comparison of parts, in particular to implement the concept of re-use can be carried out in a traditional algorithmic way or a way innovating by exploiting the concept of CSP.

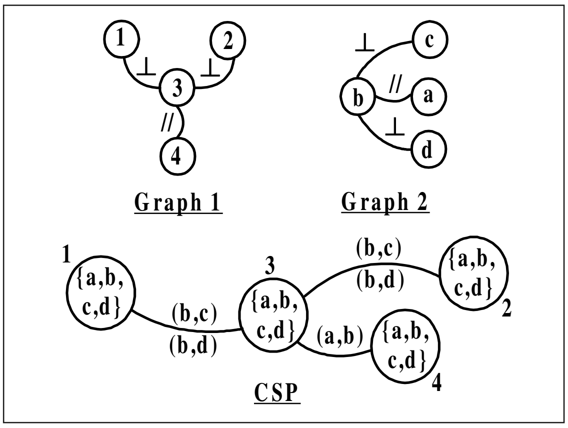
In an algorithmic way, the comparison of parts consists of a series of tests. So as to accelerate search, we are based on a distribution in classes of equivalence of the elements. This distribution in classes of equivalence is modelled in a tree structure. (Fig.5 example. (1A=1 arc, 2A=2 arcs, 3A=3 arcs)).



Distribution in classes of equivalence
Figure 5

This method is made powerful thanks to a set of pretreatments and tests adapted to the problem considered. The second method is more general, it is not related directly to the application like the preceding method. The concept of CSP is based on a modeling in the form of graph.

The problem of comparison of two normalized graphs is entirely represented by a graph. The nodes of one of the graphs are the nodes of the CSP, the nodes of the other graph constitute the possible fields of values, the arcs binding the various nodes are labelled by the constraints binding the various components (Fig.6 example.)



Representation using CSP
Figure 6

The advantage of this representation, in addition to the fact that the problem is clearly modelled is that there are robust algorithms of resolution. In the articles ([Pross93], [Pross95]), we can find a presentation and a comparison of various algorithms. We chose the algorithm of Forward-Checking (FC) ([Haral80]) for its good ratio cost/efficacity. The algorithm of search for consistency ahead or FC is a prospective method. When the process of search carries out an instanciation test of a variable, it checks the future variables and removes of the field of these variables all the values which are incompatible with the instanciation tested. The goal of this algorithm is to fail by detecting inconsistencies in the tree of search as early as possible thanks to the storage of the unsuccessful alternatives. The back return is chronological. This algorithm does more work by node than the other algorithms presented in the presented articles, but it tends to visit the least possible nodes.

Re-use of volumical objects

In the first part, we presented a possible implementation for the effective use of libraries in 2D. In this part, we present a modeling and a method allowing to obtain the same functionalities in 3D.

There are several volumical geometrical models in 3D but more used nowadays is CSG tree (Constructive Solid Geometry)([Nie89], [Requi80]). In this representation, a solid can be described as a binary tree where nodes are either regularized Boolean operations (union, intersection, difference) or geometrical transformations (rotation, translation, homothety or certain free deformations). Leaves contain either primitive volumes, or parameters of the geometrical transformations.

CSG model can be seen like a history of construction and one of the principal properties of CSG model is no-unicity. A same object can be described various ways. This characteristic is based on two fundamental characteristics of this representation:

- mathematically, for a same Boolean expression, it can exist one or more equivalent expressions:

$((A \cup B) \cap C = (A \cap C) \cup (B \cap C))$;

- geometrically, it is possible to create the same object with different sets of primitives.

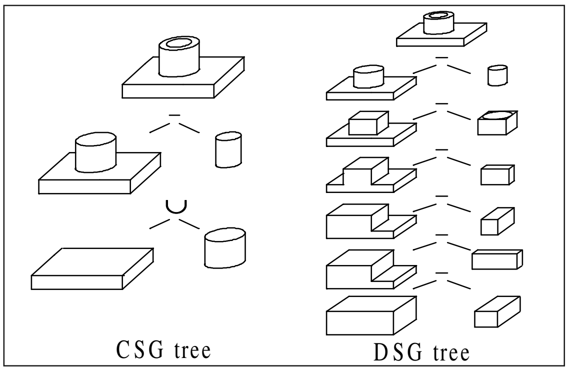
For these reasons but also because any primitive can be obtained by various combinations of Boolean operations, CSG representation is badly adapted to search of equality or to the comparison of objects. Indeed, there is an infinity of possibilities when we try to find all the primitives of an object by intersections or differences of the primitives of the other object.

Solutions generally suggested with the comparison of CSG trees are based on the concepts of SOD (Same Object Detection) and NOD (Null Object Detection). The difference of two CSG trees is evaluated, if the object obtained is null then the two compared objects are identical.

We propose to use an other volumical geometrical model: the DSG model [Perng90] (Destructive Solid Geometry) which makes possible the comparison of two objects representations. By supposing that the demonstration of the authors is correct, this manufacture model is deduced from CSG model and the representation obtained is equivalent to the initial representation. The principle of DSG tree consists in representing the object by a series of differences. A DSG tree form is: $P=S-E_1-...-E_m$ where S is the blank and E_i. an excess of matter belonging to the set primitives and such as $\forall$ $i \neq j$, $E_i \cap E_j \neq \emptyset$.

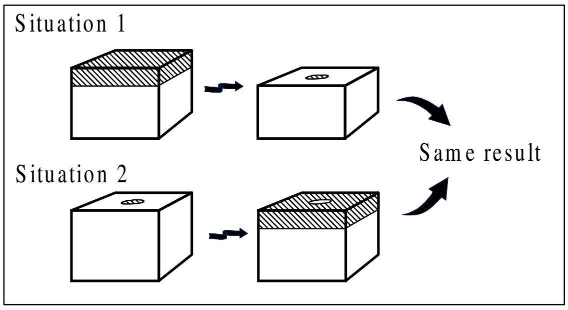
The principal stages of the transformation of a DSG tree into a CSG tree are the following ones. If we represent a CSG tree by a bracketed expression, then the first stage consists in removing the brackets. The result is a new expression of the form: $P=P_0 \pm P_1 \pm ... \pm P_n$ where $\pm$ means union or difference. Then each time that a primitive which is not of the block type appears in a union, it is replaced by a difference between its including block and its reverse primitive. Lastly, a union is replaced by an equivalent succession of differences, for example: $P_i+P_j=B_{i,j}-E_1-... -E_k$ where $B_{i,j}$ is the including block of P_i+P_j and E_k is a matter excess.

A DSG tree contains only the difference operator. Moreover, the basic solid is the merging of the final object thus for two same objects, even if construction is different, the basic solid is single. A solid will thus be represented by a set of withdrawals of elements from a basic solid, these elements not having a common intersection.



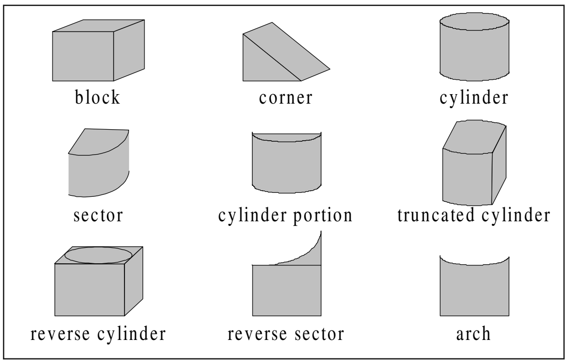
CSG and DSG tree
Figure 7

The problem of equality between two solids consists in finding the correspondences between the various elements belonging to the set of withdrawals. With this intention, we defined various sets of rules: rules of correspondence, decomposition and recombining. During the process of pairing of the withdrawals elements set, the correspondence rules allow to find equivalencies corresponding to different construction historical. For example, it must be possible to show the equality of situations 1 and 2 of Fig.8. First is obtained by the withdrawal of a paving stone of length and width identical to the initial paving stone then of a central cylinder. Second is obtained by the withdrawal of a central cylinder then of a reversed cylinder.



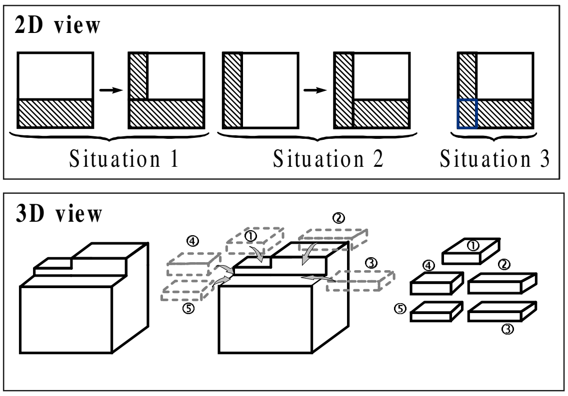
Matching rules
Figure 8

The set of primitives determines the number of correspondence rules to write, this number is countable. The set of primitives generally used is that of the figure Fig.9., it contains redundancies intended to facilitate the representation of an object in DSG model. We can notice that to each primitive corresponds its complementary.



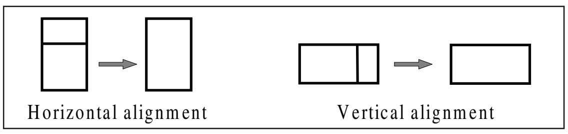
DSG model primitives
Figure 9

There is also a certain number of decomposition rules which allow to normalize some situations. For example, situations 1 and 2 of figure Fig.10. can be represented in a single way by situation 3.



Decomposition set
Figure 10

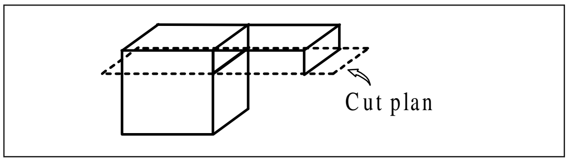
Finally, there are some recombining rules which allow to eliminate some useless cuttings: those respecting a same alignment (Fig.11.).



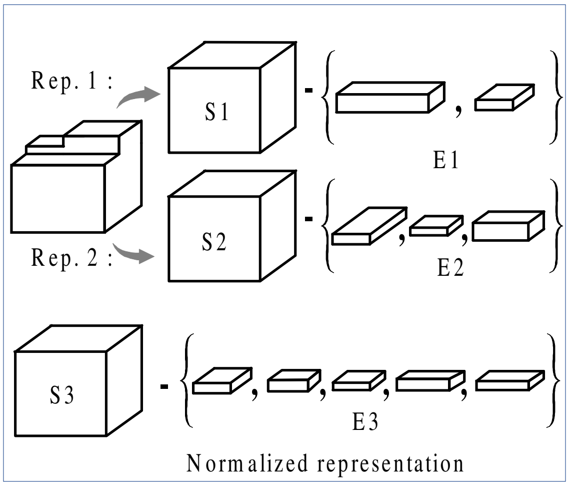
Alignment simplification
Figure 11

With these various rules, we manage to demonstrate the equality between two sets of elements so to demonstrate the equality between two DSG trees i.e. between two 3D solids.

The figure Fig.13. shows the example of a same built object in two different ways. We obtain two different sets of withdrawals E1 and E2 from the same initial solid representing the final solid merging. These two sets can be transformed into a same set of normalized withdrawal. The rules used to obtain this set are decomposition rules. Indeed, when the intersection of faces of two solids representatives of the withdrawals is not an edge, a new plan of cut is defined (cf Fig. 12).



Cut plan
Figure 12



Example
Figure 13

CONCLUSION

In this article, we were interested in the addition to the CAD system of a new assistance functionality to the user. Indeed, we studied the concept of re-use of elements resulting from libraries, during the design of new parts and under automatic suggestions of the system.

We tackled this problem in 2D and 3D. In 2D, the method consists in a normalization of the graphs of constraints of the objects; then on a representation of problem of comparison of representation standardize using formalism CSP. In 3D, the comparison of object himself base on a representation volumical in the form of tree DSG. Then of a comparison of tree DSG by search of pairing of whole of rule of decomposition, of correspondence and of simplification.

We tackled this problem in 2D and 3D. In 2D, the method consists of a normalization of the constraints graphs of the objects; then in a representation of the comparison problem of normalized representations using formalism CSP. In 3D, the objects comparison is based on a volumical representation in the form of DSG tree. Then of a comparison of DSG tree by search of matching of the rules sets of decomposition, correspondence and simplification.

REFERENCES

[Chelg98] Chelghoum, K., Jung, J-P., Leinen, S., 1998, *Constraints manipulation in CAD using ECA rules*, 2nd international conference on integrated design and manufacturing in mechanical engineering, Compiègne France, May 27-29, pp.951-958.

[Ceri96] Ceri, S., Ramakhrishnan, R., 1996*, Rules in Database Systems*, ACM Computing Surveys, Vol. 28, No.1, march.

[Haral80] Haralick, R.M., Elliott, G.L., 1980, *Increasing tree search efficiency for Constraint Satisfaction Problems*, Artificial Intelligence, No.3, pp.263-313.

[Leine97] Leinen, S., Jung, J-P, Gardan, Y., 1997, Comparaison de modèles CAO par normalisation de graphes. International Journal of CADCAM and Computer Graphics, Vol.12, No.1-2, MICAD 97, pp153-167.

[Nie89] Nie, Z., Peroche, B., 1989, *Déformations libres et arbres de construction*, BIGRE No.67, Journées AFCET GROPLAN, december, pp.98-109.

[Perng90] Perng, DB., Chen, Z., Li, RK., 1990, *Automatic 3D machining feature extraction from 3D CSG solid input*, Computer Aided Design, Vol.22, No.5, june, pp.285-295.

[Pross93] Prosser, P., 1993, *Hybrid algorithms for the constraint satisfaction problem*, Computational Intelligence, Vol.9, pp.268-299.

[Pross95] Prosser, P., 1995, *Forward checking with backtracking*, Constraint Processing, LNCS 923, pp.185-204.

[Requi80] Requicha, AAG. 1980 *Representations for Rigid Soldis : Theory, methods and systems*, ACM Computing Surveys, Vol.12, No.4, december, pp.437-464.