Convegno Nazionale

xIV ADM
xxxIII AIAS

INNOVAZIONE NELLA PROGETTAZIONE INDUSTRIALE

Dipartimento di Disegno Tecnico, Industriale e della Rappresentazione
Politecnico di Bari
Dipartimento di Ingegneria Meccanica e Gestionale

Bari, 31 Agosto - 2 Settembre 2004
AN HISTORY-BASED BINARY TREE FOR ASSEMBLIES

Alfredo Liverani\(^{(1)}\), Giancarlo Amati\(^{(1)}\), Gianni Caligiana\(^{(1)}\)

\(^{(1)}\) Università degli Studi di Bologna, DIEM, alfredo.liverani@unibo.it

Abstract

Most of commercial CAD assembly modules are based on a geometric entities constraining. DOF between surfaces (or faces), axis alignments and point-to-point constraining are targeted to a final product digital mockup. The assembly components are not specifically time sequenced and ordered and the assembly simulation is targeted to improve and optimize the component interaction. The final assembly appears in a thorough group and subgroup tree, but a correct subsequent action based assembly is hardly deducible.

In this paper a binary tree for mechanical assemblies is proposed and implemented. The time (or action) sequence is recorded and stored in a parallel data structure called BAT (Binary Assembly Tree), directly by the operator, avoiding complex automatic assembly sequence extraction from geometry. The BAT has been also provided by an addictive algorithm, which allows the incremental interactive check of large mechanical assembly components in case of direct connection with Mixed Reality systems.

Keywords: CAD, Assembly, Geometric Modelling, CSG.

1. Introduction

The extremely competitive nature of the global market of manufactured products requires not only an high level of manufacturing velocity and engineering efficiency, but also an increased integration between different and concurrent tasks. On one hand, in order to shorten the time required for the development of the product and its manufacturing process in a concurrent engineering environment, it is desirable to automate and computerize the process planning activity. On the other hand, the same knowledge developed and matured in the engineering simulation stage has to be used directly and interactively in other project related tasks, like training, rapid component seek in the storehouse, etc... Any integration step ahead means a reduced time-to-market or cost compression.

Nowadays in the project design stage a big deal of work is spent in the assembly simulation for the complete product digital mock-up setting. But the assembly tools in modern CAD systems are not fully adequate to represent complex functional relationships or only an ordered assembly time sequence. There is no possibility to define a time sequence or a hierarchical tree in order to better organize the stuff. The operator is much more involved in the correct adjustment of constraints, which sometimes are used to disengage when parameters are modified, than prepare an efficient assembling simulation. The right assembly sequence may be stored in a movie or animation and sometimes requires extra working time.
In a solid part modeller the situation is different: a binary tree is used as well as solid representation and the B-Rep (Boundary Representation) geometry is completely driven by it.

Features are stored in a progressive sequence and the binary tree, the CSG (Constructive Solid Geometry) tree, is able to perform a full model reconstruction after a parameter modification. At the moment in any assembly module this is not possible and sometimes it is clear that consistent parameter adjustment in a component has bad reflection on assembly. This is due to the lack of a binary tree serving the assembly module.

Several authors have proposed efficient computer-aided process planning and automatic generation of assembly instructions and actions, but all these approaches seem not to be really integrated in the modern CAD system structure. Most important items on these topics will be briefly reviewed in the following two sections.

1.1. Automatic assembly sequence generation.

Computer-aided process planning for assembly involves the preparation of a detailed plan for the assembly of the product using its design as the starting point. Assembly sequence is the most important part of an assembly plan and the determination of proper assembly sequence is crucial. Since it may be costly to overlook a potential candidate assembly sequence, it is desirable to select a satisfactory sequence from the set of all feasible sequences. In the literature, several authors have developed assembly sequence determination methods. Bourjault [9] has developed structured methodologies in which a series of questions must be answered to generate the feasible sequences. Ghosh [7] has developed an algorithmic procedure, which uses information on contact and mobility constraints and it can generate all the feasible sequences. Attempts have also been made to generate sequences directly from the CAD model of the assembly [8].

The research has also run in the direction of an efficient classification of all feasible assembly sequences in order to make the operator able to evaluate the resources available and choose the best one. Several representation schemes have been proposed to represent the assembly sequences. These representations can be classified into two groups: ordered lists and graphical representations. The ordered list could be a list of tasks, list of assembly states, or list of subsets of connections. Each assembly sequence is represented by a set of lists.

The most common diagrammatic representation schemes are: precedence diagrams (Prenting and Battaglin) [13], state transition diagrams (Warrats et al.) [14], inverted trees (Bourjault) [9], liaison sequence graphs (De Fazio and Whitney) [11], and ‘and/or’ graphs (Homem de Mello and Sanderson) [10]. In the precedence diagram, assembly operations are represented by numbered circles. The circles are connected by arrows showing the precedence relations. Bourjault [9] represented all valid assembly sequences in the form of an inverted tree which describes the possible orders of assembly. The inverted tree gives the liaison sequences only, not the actual assembly sequences. Also, it does not contain any information about subassemblies.

1.2. The CSG schema similarity.

The graphical and dataset introduced in this paper are very similar to the CSG tree, very common solid representation in mechanical CAD software on the market. CSG has its origins in the work of Rvachev [2], Shapiro [3] and Requicha [4]. Objects are organized as a collection of primitive solids which are leaves of a tree, whose nodes correspond to Boolean operators that perform unions, intersections and differences. The number and types of allowed primitives control the scope of the representation. Topology is stored both implicitly (in the tree structure and set operators) and explicitly (in the primitive objects), and formally valid.
object representations are easily maintained. CSG is appealing due to its intuitive formulation which is directly analogous to physical manufacturing processes where complicated solids are created by "cutting and pasting" together primitive solids. On our case the CSG schema for graphing assemblies is limited in the usage of Boolean operations to only the addition/union, as better illustrated in section 3.

2. An open geometric modelling environment.

2.1. Software libraries review.

The BAT tool has been implemented over an heterogeneous software development environment which exploits several different geometric and graphical libraries:

1) Spatial ACIS Toolkit for geometric kernel and B-Rep mathematical manipulation;

2) Hoops 3D and TGS Open Inventor for geometric object visualization;

3) Trolltech QT for GUI (Graphical User Interface).

Figure 1 shows the several connections between modules and libraries used in AIS-CAD (ACIS Integrated CAD). The open architecture is a key-feature in this environment with something less in terms of CSG tree and feature management. On the other hand, AIS-CAD uses a synchronization routine to keep aligned the B-Rep database, stored in the ACIS SDK, and any other triangle stripe viewer, like Hoops 3D or Open Inventor. So any new geometry instancing doesn’t need a similar action in the OIV scenegraph, but the “mesh manager” routine generates an optimized mesh for the scenegraph based graphical viewer.

This configuration, a little slower than a direct B-Rep visualization, is particularly useful for software prototyping and keeps also all advantages of interactive and manipulation tools implemented in OOP native graphical libraries (i.e. OIV). Full 3D interaction, like in a VR setting and precise B-Rep geometry access, is definitely unified and integrated in the same environment.

Figure 1. The development CAD architecture used for BAT implementation.
2.2. Assembly constraints implementation.

Most of commercial CAD software exploits geometric models of individual components of the assembly that represent the geometry and their positions in the world coordinate space. This is a very common approach, but it is inadequate for a deeper control over the whole assemblage procedure. A more sophisticated model for assembly relations recording and also component assembly sequence storing has been introduced by Gottipolu and Ghosh’s one [7]. Their relational model has been implemented in AIS-CAD as a better schema for geometric relation storage. More in detail, the relational model of an N-component is a two tuple <P,U>, where:

- P = {P₁, P₂, . . . Pₙ} is a set of symbols and each symbol corresponds to one part in the assembly.

- U = {U₁, U₂, . . . Uₘ} is a set of 4-tuples, representing the relations between components in the assembly, where M=NP₂ = N (N-1).

- Uᵢ = <Pa, Pb, Cab, Tab >, where Pa, Pb ∈ P,

\[C_{ab} = (C_1, C_2, C_3, C_4, C_5, C_6)\] is a 1 x 6 binary function representing contacts between the components a and b. Hence it is called a contact function or C-function. It can be defined as:

\[C_{ab} : C_i \rightarrow \{0,1\} \quad i = 1 \text{ to } 6\]

- \(C_i = 1\) indicates presence of contact in the direction \(i\), i.e. part \(b\) is in contact with part \(a\) in the direction \(i\);

- \(C_i = 0\) indicates absence of contact in that direction.

\[T_{ab} = (T_1, T_2, T_3, T_4, T_5, T_6)\] is a 1 x 6 binary function representing translational motion between components a and b. Hence it is called a translational function or T-function. It can be defined as:

\[T_{ab} : T_i \rightarrow \{0,1\} \quad i = 1 \text{ to } 6\]

- \(T_i = 1\) if the part \(b\) has the freedom of translational motion with respect to the part \(a\) in the direction \(i\);

- \(T_i = 0\) if the part \(b\) has no freedom of translational motion with respect to the part \(a\) in the direction \(i\).

The six entries in the C and T functions correspond to the three directions (X, Y and Z), and two senses (positive and negative) of a tri-orthogonal Cartesian coordinate system. Here, directions 1, 2 and 3 indicate the positive sense of X, Y, and Z axes (X+, Y+, and Z+), respectively, whereas directions 4, 5 and 6 correspond to the negative sense of X, Y, and Z axes (X-, Y-, and Z-), respectively. Every component must be oriented along the three principal axes.

Furthermore the authors start from that model to introduce a C & T function matrix, which is useful to extract all possible assembly sequence combinations. In our case, instead, a different graphic tool and dataset for assembly sequence has been implemented, as reported in the following section.

3. The BAT (Binary Assembly Tree) tool.

The BAT (Binary Assembly Tree) is much more similar to a CSG tree nowadays common in all mechanical CAD systems, than to a dataset used for automatic assembly sequence extraction. This similarity comes from the need to match the above requirements/benefits:
1) optimal integration with parametric parts (time wasting in assembly adjustment after a component modification is dramatically reduced: i.e. faces involved in assembly constraining sometimes are no more coherent with constraints after parameter variational modification);

2) the operator can perform a realistic assembly simulation, defining in the design stage the correct procedure to obtain the final product.

The basic algorithm which is responsible of the BAT mechanism is a simple virtual node generator, associated with a pointer to a C++ structure storing component codes included in that node. A new virtual code is automatically assigned to the virtual node in order to strictly connect reference virtual node with its attached node. A such configuration is an effortless programming structure with incremental features. It has big properties and it influences the operator’s work in combination with a GUI and a graphing tool, targeted to summarize the real component and virtual node arrangement. In Figure 2 a simple graphing utility has been used to show the combination of real components, like SCREW 0012-4324 or FLANGE 0012-4323, and virtual nodes (noted with the string “COMP.”), like COMP. 0014-0001. All virtual codes are incrementally generated, as well as for real codes, but in a further software release, all these data will be obtained from the PDM (Product Data Management) for a better integration.

![Binary Tree with incremental feature](image)

**Figure 2.** Example of Binary Tree with incremental feature.

4. Tests and results.

The BAT tool has been introduced in AIS-CAD in order to store component coupling sequence during the normal assembly simulation in the CAD environment. The interface (see Figure 3) shows a side bar with the binary tree listing component couplings, instead of surface mating, axial match, etc…. The operator is only required to constraint new components one by one: the constraints are imposed simply between the last component and the incremental component array stored at BAT lower level.
In Figure 4 a simple example is described step-by-step like in an instruction manual. In fact BAT works in quite the same way: the first component is fixed (SCREW 0012-4324) and the RING 0012-4322 is simply added to the incremental array. Then a virtual item (COMP. 0013-0001) is created by the addition of the previous cited components (screw and ring). This is now a new single node in the tree and can be freely used to assemble other components. In that case, the SPACER 0013-4323 is connected to COMP. 0013-0001, leading to a new virtual item called COMP. 0014-0001 at the following assembling step.

The iterative procedure is implemented through a basic algorithm which can generate and store the tree virtual inclusive items like:

\[
M_3 = M_1 \cup M_2, \quad M_5 = M_3 \cup M_4 \quad \text{and} \quad M_7 = M_5 \cup M_6
\]

thus:

\[
M_7 = M_5 \cup M_6 \iff M_6 \cup M_4 \cup M_3 \iff M_6 \cup M_4 \cup M_1 \cup M_2
\]

At the end of the procedure, the final virtual node represents the complete assembly:

\[
M_{11} = M_1 \cup M_2 \cup M_4 \cup M_6 \cup M_8 \cup M_{10}
\]

The benefits of an iterative approach with virtual incremental nodes are several:

1) the assembling procedure is clearer to the operator and also to people collaborating to the project that need to quickly acquire information about the product;

2) the sequence is step-by-step evidently checked and approved;

3) the virtual node may be used to interactively check the operator’s work in a Mixed Reality (MR) environment (Figure 5).
Figure 4. Example of assembly sequenced (related to Figure 2).

The real-time connection between an MR application and a CAD tool like BAT seems to be ideal for training and human work control on site. In fact (see Figure 5) the operator, wearing an head-mounted display may receive assembly hints and suggestions by the assembly checking algorithm.

Figure 5. BAT for Mixed Reality application (training).

5. Conclusions

This paper reports the experiences and the implementation of an assembly sequence recording tool for mechanical multi-component product. The final sequenced assembly is manually recorded by the operator and appears as a binary tree, similar to most CSG and feature-based
geometric modeller. The direct and manual store of assemblage actions is much more efficient than complex automatic assembly sequence extraction from already modelled geometry and also simple and intuitive.

In that way the operator may not only normally simulate the assembly operations needed to assemble the final product by imposing constraints on faces or axis (which is sometimes not instinctive), but he may also setup a clear assembly sequence. The implementation of BAT may store both, traditional constraining and action sequence.

The overall result of this original piece of software is clearly valuable not only in normal CAD usage, but especially for training purpose, where the user is remotely driven, step by step, to the correct assemblage, or, is forced to the solution, through an automatic assembly check software, by exploiting the already optimized proper sequence.

References


