ELSEVIER

Computer-Aided Design 38 (2006) 627–640

COMPUTER-AIDED
DESIGN

www.elsevier.com/locate/cad

Repairing CAD model errors based on the design history

Jeongsam Yang [a,*], Soonhung Han [b,1]

[a] *Division of Industrial and Information Systems Engineering, Ajou University, San 5, Wonchun-dong, Yeongtong-gu, Suwon, Kyungki-do 443-749, South Korea*
[b] *Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, 373-1, Gusong-Dong, Yusong-Gu, Daejeon 305-701, South Korea*

Received 27 April 2005; accepted 25 February 2006

Abstract

For users of CAD data, few things are as frustrating as receiving unusable, poor quality data. Users often waste time fixing or rebuilding such data from scratch on the basis of paper drawings. While previous studies use the boundary representation (B-Rep) of CAD models, we propose an approach to repairing CAD model errors that is based on the design history. CAD model errors can be corrected by an interdependency analysis of the feature commands or of the parametric data of each feature command, as well as by a reconstruction of the feature commands through rule-based reasoning of an expert system. Unlike other correction methods based on B-Rep models, our method repairs parametric feature models without translating them to a B-Rep shape, and it also preserves parametric information.

Keywords: Boundary representation; CAD model; Correction; Design history; Interoperability

1. Introduction

Because of competition in the market, the lead time for designing and manufacturing products is decreasing, whereas the complexity of products is increasing. Furthermore, as the globalization of companies continues, the development of products surmounts the limitations of geographical boundaries and is enhanced by collaborative design in a distributed environment where there are frequent exchanges of product data for common parts and components.

When developing products, designers and engineers often encounter poor quality product data, and they waste considerable time and money overcoming this problem. One company, for instance, reported that it spent as much as 50% of its time repairing product models in downstream functions, such as rapid prototyping, finite element analysis, and CNC programming. The company had to deal with poor quality CAD data that were not properly modeled for use in these downstream functions. In another example, an original equipment manufacturer estimated that, within the company and between the company and its suppliers, it had as many as 453,000 exchanges of product data each year, and that it spent an average of 4.9 h solving each poor-quality problem. As for the cost of imperfect interoperability, the members of the US automotive supply chain reportedly spend at least $1 billion a year [1]. Similarly, in the Japanese automotive industry, which has as many as 250,000 product data exchanges a year, the cost of these exchanges is approximately $68 million a year, and 1.5 h of lead time are lost whenever one item of poor quality or unusable data needs to be repaired or replaced [2].

Previous studies on poor-quality CAD models have focused on the boundary representation (B-Rep) of 3D shapes [3–8]. They corrected errors by mathematically analyzing the topological and geometric elements of the B-Rep model. Most commercial applications use the B-Rep data to analyze and correct CAD model errors. However, although the B-Rep approach is effective for locating errors with the maximum or minimum tolerance as defined by the designer, an unstable topological structure can unintentionally distort a repaired B-Rep shape or cause the shape to collapse. As a result, only limited types of errors can be corrected, and automation of the correction process is difficult. Designers are therefore reluctant to correct errors with commercial applications that are based on the B-Rep approach.

To repair errors, we now propose a method that can reconstruct the design history of a CAD model. The design history refers to the chronological order in which a designer created the various features of a 3D shape. In a CAD system, there are different ways of defining a shape. Thus, the design includes the geometry-controlling parameters, the geometric

---

* Corresponding author. Tel.: +82 31 219 1879; fax: +82 31 219 1610.
E-mail addresses: jyang@ajou.ac.kr (J. Yang), shhan@kaist.ac.kr (S. Han).
[1] Tel.: +82 42 862 9226; fax: +82 42 862 9224.

0010-4485//$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2006.02.007

design features, the feature information, the design history tree, the parameterization data, and the constraints. We therefore analyzed several 3D parametric models from automotive companies to define the relationship between the various feature commands, as well as the relationship between the parametric data of each feature. We also defined a design history schema in order to structure the design history information extracted from a 3D model created in a commercial CAD system. We then used the schema to repair the CAD model through the rule reasoning of the design history. Finally, to verify the proposed method, we developed a CAD model correction system called *Q-Raider*, which can repair the following six types of CAD model errors: tiny faces, narrow regions, non-tangent faces, narrow steps, sharp face angles, and narrow spaces.

2. Related works

2.1. Error corrections based on B-Rep

Previous studies on correcting CAD model errors can be classified into an exact B-Rep approach, a faceted B-Rep approach, and a boundary curve-based approach. In the exact B-Rep approach, CAD model errors are checked and corrected through a mathematical computation of a data structure that represents the topological and geometric elements of the B-Rep model. Hoffman et al. [3] proposed the architecture for a master model, and their architecture combines with downstream application processes to produce different views of features. To correct the problem of geometric tolerance between different downstream applications, the architecture of Hoffman et al. uses a call-back mechanism that the user can control. Gu et al. [4] sequentially sorted topological entities by using a complementary model object tree, and they used the tree to correct topological errors.

The faceted B-Rep approach, which approximates the exact B-Rep model in terms of a polyhedron, corrects errors faster than the exact B-Rep approach. Barequet et al. [5] proposed a geometric hashing algorithm that sequentially reconfigures polygons after dividing the trimmed surface of a complex 3D shape into unordered lists of polygons. The hashing algorithm corrects errors by stitching the small gaps between the polygons.

The boundary curve-based approach can only be used for 3D shapes made of surfaces. Steinbrenner et al. [6] checked and corrected the gaps or overlaps between adjacent curves in a 3D shape that consisted of curved surfaces with various degrees. This method checks and corrects the gaps or overlaps through an edge-splitting and merging process after the boundary curve has been divided into small edge curves. To correct the G^1 discontinuity on surfaces (that is, the non-tangent angle between adjacent surface patches), Volpin et al. [7] simplified the original free-form surface model: they first divided the regions on the basis of curvature variation; next, they generated a boundary-conforming finite element quadrilateral mesh of the regions; and, finally, they fitted a smooth surface over the quadrilateral mesh.

Most studies discuss the B-Rep shape of CAD models and a B-Rep correction can be applied to a limited number of error types such as gaps and overlaps. Moreover, there is inevitably a loss of data when the shape is being simplified by the B-Rep approach.

2.2. Persistent naming problems

Although there is no previous study on checking and repairing CAD models on the basis of the design history, considerable attention has been given to research on the persistent naming problems associated with the exchange and modification of parameterized feature-based CAD models [8–16].

The literature reveals two approaches to persistent naming problems: topological and geometric. With respect to topological naming, Kripac [8] proposed a topological ID system in which IDs are assigned to topological entities (such as faces, edges, and vertices) in solid models. When the design history is modified and then automatically reevaluated to produce a new solid model, the Kripac system uses two searching algorithms (namely a forward algorithm and a backward algorithm) to ensure that the IDs of the topological entities in the previous model are mapped to the IDs of the corresponding topological entities in the new model. However, Kripac’s algorithm is difficult to implement and he failed to address details such as the mechanism of naming faces.

After proposing a topological naming method based on faces, Wu et al. [9] presented the concept of parametric space information to solve the topological ambiguity that occurs when naming, recording and retrieving topological entities; however, they ignored name matching and the ambiguity that arises from face merging.

Mun et al. [10] proposed a disambiguation method based on object space information and secondary names. Instead of using parametric space to compare topological entities with the same name, Mun et al. used object space information, which is similar to Wu’s parametric space information. Basically, they used the object space of a feature’s 3D Cartesian coordinates to record and retrieve topological entities.

Capoyleas et al. [11] proposed a topological naming method that exploits feature-specific information such as the profile and path of extrusions. To solve the ambiguity problem, Capoyleas et al. used information on either the *local orientation* of adjacent topological entities, such as the edge or the face, or on the *feature orientation*, such as the extrusion path or rotational axis.

Vergeest et al. [25] endeavored to anticipate the feasibility of interoperability in an approach that systematically analyzes and models the requirements of a shared infrastructure. An inherent incompatibility between different CAx models exists in CAD translations due to system-specific modeling functionalities. For example, the hole creation command of Pro/E cannot be directly translated to CATIA because Pro/E provides more diverse ways to generate hole features than CATIA. Such incompatible modeling commands can be mapped through 1:N or N:1 mappings.