13º Encontro Português de Computação Gráfica

12 - 14 Outubro 2005

Direct Manipulation of feature models using Handles

Daniel Lourenço
DEI / IST
Lisbon, Portugal
daniel.antunes.lourenco@gmail.com

Pedro Oliveira
DEI / IST
Lisbon, Portugal
pedro.vcm.oliveira@gmail.com

Rafael Bidarra
EEMCS / TU Delft
Delft, The Netherlands
r.bidarra@ewi.tudelft.nl

Abstract

In feature based systems support for direct manipulation is not commonly available. This is partly due to the strong reliance of feature modelling systems on constraints and on the lack of speed of current constraint solvers. In this paper an approach to the optimization of the geometric constraint solving for the specific situation of direct manipulation is described. Also a solution for a direct manipulation interface was designed that brings to feature modelling the advantages of direct manipulation while taking into account the main feature modelling paradigm concepts. Details are provided on how it was implemented successfully in the SPIFF feature modelling system.

Keywords

Feature-modelling, Direct manipulation, CAD/CAM

1.INTRODUCTION

1.1Feature Modelling

Feature modelling is a design paradigm that aims to bring the modelling task to a higher level in a way that it is easier for the designer to specify and understand elements and their relations in a model. This is done by associating functional information to the shape information in the product model.

The feature modelling process is done based on features. [Bidarra99] defines a feature as a representation of shape aspects of a product that are mappable to a generic shape and are functionally significant for some product lifecycle phase. This way, unlike when the design is based in geometry alone, in feature modelling the designer builds a model out of features that have a well-defined semantics.

A feature class library is a collection of feature classes that are made available to the designer. Each feature class has a specified semantics (e.g. a specific machining operation) and can be instantiated as a feature in the model by providing a set of instance parameters.

Constraints are a crucial concept for feature modelling systems and can be used to express high-level characteristics of the model (e.g. restrict the volume of a product to a certain maximum). Constraints can also be used as internal elements of features that express the feature semantics. Systems that force the maintenance of each feature's semantics in a feature model throughout the modelling process are called semantic feature modelling systems [Bidarra99]. Because of this intensive usage of constraints, feature modelling systems have to make an extensive usage of constraint solving techniques.

1.2Limitations of Traditional Feature Manipulation

In the SPIFF system the specification of feature parameters is done solely through the input of values in dialog boxes. When a user edits a feature, the model is updated with the new user values and then fully solved to determine the new geometry.

The main disadvantages of this approach are:

- inefficient feedback, making the design task much slower. Each time the designer changes the parameters of a feature he has to wait for the whole system to be solved and only then can he see the effect of his changes and check the validity of the model.
- lack of insight on the consequences of the modelling operation. When changing a parameter the user can only see the original and resulting model of the operation. This means that there is no explicit feedback on which features were affected and how.
- non-intuitiveness due to the fact that the user is simply editing dialog boxes that do not express how the feature is affected by the parameter.

1.3Project Goals

The main goal of this project was the development of a direct manipulation interface for form-features in the system. In particular feature handles were added as a way of specifying feature parameters. As the modelling operation occurs real-time feedback has to be provided to the user[1]. With this improvement the design process will be improved so that the user may interactively change

[1] by real-time we mean fast enough for the user to have a direct manipulation experience and not the strict meaning of the real-time expression

the value of the parameters of features in the model while he is being provided with feedback of the changes. When the user is satisfied with the model he can choose to make the changes final and check the model validity.

Two critical issues had to be tackled in order to provide the real-time interaction required -- efficient model constraint handling and efficient display of the model geometry being manipulated by the user[2].

One other orthogonal problem that had to be solved was to determine which should be the characteristics of the feature handles available for each feature (where they are positioned; how they are specified; how they behave; how does their manipulation affect the parameter being changed).

1.4Paper Overview

On this paper we will first start by giving an overall insight of the what the SPIFF system is, its architecture, main concepts and how they relate and are implemented internally. This will be done in section 2.

In section 3, after a brief introduction on how constraint solving works in SPIFF, we will present a solution for constraint solving that copes with the efficiency requirements of real-time feedback for direct manipulation.

In section 4 we will state the designed target manipulation requirements and how these where materialized in the SPIFF system.

In section 5 we give a brief overview of what was done while concluding whether the main objectives were achieved and identifying the limitations of the solutions found.

2.THE SPIFF SYSTEM

The SPIFF system [SPIFF] is an advanced feature modelling system developed at the Delft University of Technology, the Netherlands. Development on this system started in 1994 and work on it has been going on since then leading to a system which incorporates many different ideas and concepts on feature modelling and which is therefore quite complex.

The underlying geometric kernel of this system is the ACIS modelling kernel.

This system supports many advanced feature modelling concepts including the concept of semantic feature modelling [Bidarra00].

2.1Feature Classes in the SPIFF System

A feature class in the SPIFF system can be seen as a specification of its shape and the set of parameters needed to instantiate it.

The shape of a feature class is determined based on the concept of basic shape. A basic shape works as a meta-feature that only specifies the geometry it represents (e.g. cylinder). This representation is a set of model elements (e.g. faces) and constraints that specify their relations. Features acquire their geometry by deriving from a specific basic shape.

[2] we will not approach the latter issue on this paper

Besides its shape, a feature class also contains a number of constraints and variables which further define its shape and semantics and the way it relates with other features in the model.

2.2Feature Model

The feature model in the SPIFF system has multiple internal representations. One of these, on which we will focus, consists of the High Level Constraint Graph. Its nodes contain the variables and the constraints applied upon them and their relationships are established through the edges. This graph aggregates the definitions of every feature in the model and the relations between them. Each feature is present as a set of shape constraints, variables (e.g. faces and parameters) and other constraints. In figure 1 we can see a simplified version of a High Level Constraint Graph.

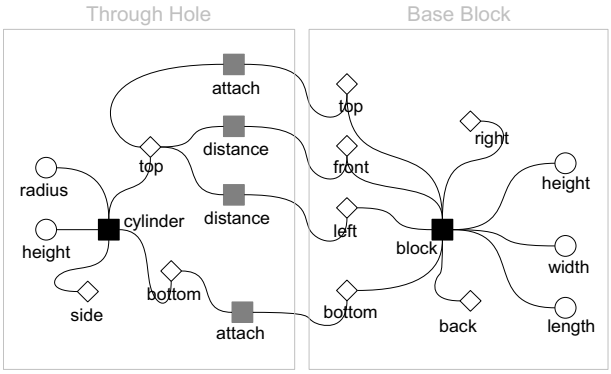


Figure 1: Sample constraint graph

Only two types of variables exist:

- geometric variables which represent geometric entities of the model. These variables are defined by a position and an orientation. From the point of view of our problem, we can assume that only two types of geometric variables exist: faces and references. References are model elements like lines, points or surfaces that can be added to the model.
- value variables which contain several types of numeric values.

As for the constraints they can be of several types but, for this context, it suffices to explain the following three:

- geometric constraints which restrict the way geometric variables are positioned (e.g. distance between two faces).
- shape constraints which are similar to geometric constraints but specify the relation between all the faces of a basic shape and between its value variables.
- algebraic constraints which specify algebraic relations between value variables (e.g. equal, division).

Other constraints that exist that are not relevant for this paper are the boundary constraints and dimension constraints. These constraints are not used for the definition of model values but only to further enforce the semantics of each specific feature. One example of a boundary con-

straint is that the top face of the through hole in Figure 1 must not be on the boundary of the model.

Each time a feature is added to the model, its corresponding variables and constraints are added to the High Level Constraint Graph. It is by solving this graph that the position and orientation of each geometric variable is determined and, thereafter, the full geometry of the model is known and can be shown to the user.

One other internal model of the SPIFF system is the Cellular model. This model is a geometric model which is used both for display and for the validity checking of boundary constraints. It is implemented on top of the geometric kernel using its data structures and operations.

3.REAL-TIME CONSTRAINT SOLVING FOR DIRECT MANIPULATION OF FEATURES

As was seen in the introducing section, a feature model in the SPIFF system is internally represented as a constraint graph. To know the geometry and to check the semantics of the model, the whole graph has to be solved. After the solving process, if the model is valid, the value of all the variables in the system is determined.

In order for the system to solve the graph it follows a number of steps of which the most important are the solving of specific mappings of the High Level Constraint Graph by both an algebraic and a geometric constraint solver. These solvers are, respectively, the SkyBlue solver [Sannella92,Sannella93] and the Kramer Solver [Kramer92].

A performance analysis has shown that the two most important bottleneck of this solving process are the execution time of the geometric solving and the evaluation of the cellular model.

The evaluation of the cellular model can be skipped by displaying only a simplified model. As the boundary constraints validation depends on the cellular model it cannot be performed.

Since the cellular model evaluation was avoided the optimization efforts were focused on the Kramer Solver. We next briefly describe how it works and what was changed in order to make the system faster for the direct manipulation situation.

3.1The Kramer Graph

The specific mapping of the High Level Constraint Graph which is used by the Kramer Solver is called Kramer Graph. It is composed of constraints and variables. In this graph the variables are called geoms and represent a coordinate system which is in a specific position and orientation in space. This position and orientation are not fixed[3] but are restrained by the constraints in the graph. It is by determining which is the position and orientation for each geom that satisfies all constraints that the final values for the variables are determined.

Each constraint reduces the relative degrees of freedom between geoms by specifying restrictions to their position and orientation.

To each geometric variable of the High Level Constraint Graph will correspond exactly one geom and to each constraint will correspond one or more constraints in the Kramer Graph.

3.2Solving Strategy

To explain how the solving actually works we first need to define the concept of joint. A joint is no more than the set of constraints between two geoms. This means a joint is what restricts how two geoms are positioned in relation to each other. We call a joint rigid if it leaves no relative degrees of freedom between the two geoms it relates i.e. their position and orientation is fixed in relation to each other.

The Kramer Solver works by iteratively determining which joints are rigid and joining them into new geoms called macro-geoms. By progressively joining geoms a point is reached when there is only one geom. At this point the system is solved.

One final remark is that the Kramer Solver is able to handle models which are not fully constrained. In this situation it merges geoms up to the point where it can no longer find rigid joints. The result will be a graph with more than one remaining macro-geom. This fact is crucial for this work as will be explained in the next section.

3.3Geometric Constraint Solving for Direct Manipulation

The key remark to understand the optimization proposed is that, when the user is directly manipulating a parameter, only its corresponding variable is being changed during the direct manipulation session[4]. As a result, all its solving operations are highly repetitive because most of the graph is exactly the same. From here we can devise a solution that attempts to avoid this repetitiveness.

The idea is to put the ability of the solver to handle under-constrained models at the service of the repeated solving for the same parameter. Our solution will be to add a preprocessing step to the direct manipulation session that executes the computations that otherwise would have been repeated and encapsulates their result in the constraint graph. For each interaction step only the operations that depend on the parameter are executed.

This solution will be composed of two steps: 1) the preprocessing step solving and the 2) interaction step solving.

In the preprocessing step the High Level Constraint Graph is mapped without the constraints which are determined by the parameter being changed by the user. The removal of these constraints makes the mapped Kramer Graph under-constrained and therefore, after solving, we will have a solution with a set of macro-geoms. We call this solution the preprocessed graph. This graph is precisely the structure that encapsulates the repeated part of the solving operations. It will be used as the starting point for solving in subsequent changes of the parameter.

[3] apart from a small number of geoms which have an absolute position and orientation in space

[4] by direct manipulation session we mean the period of time while the user directly manipulates one parameter

For each of the subsequent solving operations the interaction step solving is executed. In this step the constraints determined by the parameter being changed will be mapped to the preprocessed graph. This graph is now fully constrained and can be solved to give us the solution for the current value of the parameter. This solving step is much faster than solving the whole model as most of the repeated solving is skipped.

4.MANIPULATION OF FEATURES USING FEATURE HANDLES

4.1Target Manipulation Interface

The direct manipulation facilities designed must deal with the three disadvantages of traditional feature manipulation mentioned in section 1.2.

For efficient feedback every modelling operation has to lead to an immediate preview of the result of the operation to the user. This preview will be provided through a transparent overlay display of the resulting model. The user will have insight on the consequences of the operation through the comparison of the original model with the preview which are displayed simultaneously. The manipulation will be intuitive because of the feature handles' characteristics.

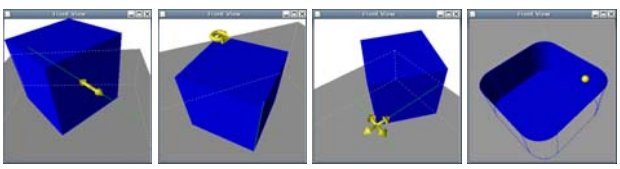
The deployed feature handles must take into account that each feature parameter has a specific semantics. This semantics will be expressed through the handle's behaviour, positioning and graphical representation.

4.2Types of Feature Handles

In this work it was decided to deal only with parameters which contain numerical values which generally have a simple geometric meaning.

In SPIFF the following four types of handles are enough to cover the direct manipulation needs: linear handles; angular handles; planar handles; slider handles.

In this proposal all handles of a feature are specified by their relation with reference elements (e.g. reference points and reference lines) of the feature. With them the system will be able to know the placement and behaviour of the handles. One reference that is needed for the specification of any type of handle is the point that specifies its position in the model. These references are specified at the feature definition like any other references - using constraints. One very important consequence of the definition using references is that the position and orientation of the handles will be determined by the constraint solver alone.



(a) Linear **(b) Angular** **(c) Planar** **(d) Slider**

Figure 2: Types of Handles

We will now look with more detail into each of the proposed handles:

- A linear handle is a handle that moves along a straight line and reacts linearly with the mouse movement. It contains a reference line which represents the line along which the handle moves. See figure 2(a).
- An angular handle is a handle that moves around a rotation axis in a way that the user is able to specify an angle parameter. This handle has a reference line which represents the axis of rotation around which the angular handle revolves. See figure 2(b).
- A planar handle behaves somewhat like a linear handle with the difference that, instead of having its movement restricted to a line, its movement is restricted to a plane (having two linear degrees of freedom). For this type of handle two reference lines will be needed to determine the plane on which it can move. One thing that differs from this handle to all the others is that when manipulated it affects two parameters of the feature instead of one. The motivation for the existence of this handle is to enable the user to change the position of features that have it specified by two distances to external faces. See figure 2(c).
- A slider handle is also similar to a linear handle but, in this case, the line will simply be a vertical line on the viewport with no connection to the actual feature geometry. Therefore this handle will contain no reference lines. The slider handle adjusts nicely to parameters that have no simple geometric meaning but should be directly manipulatable. See figure 2(d).

4.3Handle Manipulation Flow of Events

Two important moments exist in the handle manipulation flow of events: the moment when the user selects a feature to be edited and each time a handle is dragged.

When the user selects a feature to be edited this leads to the activation of all the handles of the feature -- the handle is displayed and registered in the Operator (a control entity). The feature is also highlighted with a transparent overlay.

When a single drag event of the user occurs the drag event is reported to the Operator with the information of the new mouse position. The operator sends a message to the Handle which leads to the computation of the new value of the parameter and to its update in the model. This computation is obviously done taking into account the handle behaviour (see section 4.2). After the new value is set, the Operator orders the Constraint Manager to solve the model. With the new model values resulting from the constraint solving process a transparent preview of the result of the operation is rendered and displayed.

5.CONCLUSIONS

An approach for 3D geometric constraint solving for direct manipulation was developed. This approach has been successful at increasing the speed for the direct manipulation situation and, therefore, improving the user experience. The average improvement on the speed of geometric solving for the tests performed was of 18 times. With full frames being generated in 290ms for medium sized

models (13 features) we believe the objective of providing the user with real-time manipulation has been achieved. Of these 290ms only 50ms are for geometric solving therefore further optimization work mostly has to be performed on other parts of system[5].

One problem concerns the large amount of time required for preprocessing before the first frame of the manipulation session can be shown. This happens because of the necessary underconstrained solving process which is slower. Unfortunately, at this point, we have no promising idea on how to improve this time.

In spite of the problems that still stand, the overall outcome regarding the work on geometric constraint solving was clearly positive with impressive gains at the geometric constraint solving performance.

A whole new architecture for the direct manipulation with handles for a feature modelling system was designed and implemented. Four types of feature handles were designed and no situation was found in which these four types of handles were not enough. For parameters with a clear geometric meaning this solution was fully successful in making the manipulation process intuitive[6].

For an effective and insightful feedback on the modelling operations, a transparent preview of the model is overlaid as the user directly manipulates a feature. This way he can clearly see the effect of the modelling operations and compare the result with the original situation.

Through out this project separate parts were developed. They build on top of each other to provide a system which allows the user to perform direct manipulation of features in real-time. Although many improvements can still be made the product of this project provides a sound foundation for further developments.

[5] tests performed on an Intel Centrino 1.3Ghz with 512Mb of physical memory and an ATI Radeon Mobility 9000 graphics card

[6] in truth this conclusion could only be assured based on usability tests with real users

6.REFERENCES

[Bidarra99] Bidarra, R. (1999), Validity Maintenance in Semantic Feature Modelling, PhD Thesis, Delft University of Technology, Delft, The Netherlands

[Bidarra00] Bidarra, R. and Bronsvoort, W.F. (2000) Semantic feature modelling. Computer-Aided Design 32(3): 201-225

[Bunnik00] Bunnik, A. (2004), Web-based direct manipulation of feature models, MSc Thesis, Delft University of Technology, Delft, The Netherlands

[Dohmen97] Dohmen, M. (1997), Constraint-based feature validation, PhD Thesis, Delft University of Technology, Delft, The Netherlands

[Kramer92] Kramer, G. A. (1992), Solving geometric constraint systems: a case study in kinematics, The MIT Press.

[Noort02] Noort, A. (2002), Multiple-View Feature Modelling with Model Adjustment, PhD Thesis, Delft University of Technology, Delft, The Netherlands

[Sannella92] Sannella, M. (1992), The SkyBlue constraint solver, Technical Report 92-07-02, Dept. of Computer Science and Engineering, University of Washington, USA.

[Sannella93] Sannella, M. (1993), The SkyBlue constraint solver and its applications, in "First Workshop on Principles and Practice of Constraint Programming".

[SPIFF] The SPIFF system homepage, <http://www.cg.its.tudelft.nl/~spiff/spiff.html>

[Wikipedia] Direct manipulation interface, Wikipedia, <http://en.wikipedia.org/wiki/Direct_manipulation_interface>