ADVANCED DIRECT MANIPULATION OF FEATURE MODELS

Rafael Bidarra, Alex Noort
Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, The Netherlands
A.R.Bidarra@tudelft.nl, A.Noort@ewi.tudelft.nl

Daniel Lourenço, Pedro Oliveira
Instituto Superior Técnico, Technical University of Lisbon, Portugal
daniel.antunes.lourenco@gmail.com, pedro.vcm.oliveira@gmail.com

Keywords: Feature modelling, direct manipulation, user interaction, handles, constraints

Abstract: In current commercial feature modelling systems, support for direct manipulation of features is not commonly available. As a result, re-designing is time-consuming due to the inefficient feedback, the insight given is rather poor, and user interaction often lacks intuitiveness. This is partly due to the lack of speed of current constraint solvers, but also to deficient interactive facilities. In this paper, we argue that providing advanced direct manipulation facilities for feature models is possible and can significantly speed up the product design process, by giving designers a much more intuitive interface, with immediate feedback and deeper insight into the consequences of each modelling action. An approach to such a direct manipulation interface is presented that brings together the advantages of direct manipulation of feature models with the necessary emphasis on fundamental feature modelling paradigms like feature parametrisation and feature validity maintenance. In particular, it offers a powerful combination of various 3D handles for real-valued feature parameters, with a preview overlay facility for all modelling operations. Details are provided on how this approach was successfully implemented in a prototype feature modelling system.

1 Introduction

Feature modelling is a design paradigm that comes as an alternative to the traditional geometry-based design systems. The founding idea of feature modelling is to focus the modelling tasks of the designer on a higher level, facilitating the specification of many different aspects in a product model, and gaining insight into their inter-relations (Shah and Mäntylä, 1995). This is achieved by enabling the designer to associate functional information to the shape information in the product model.

Although one cannot find a consensual definition of the concept of feature, one that nicely fits to this research 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 life-cycle phase" (Bidarra and Bronsvoort, 2000). In contrast to conventional CAD systems, in which the design focus mainly lies on geometry, in a feature modelling system the designer builds a model out of features, each of which has a well-defined semantics. As an example, for manufacturing planning purposes it would be appropriate to provide the designer with features that correspond to the manufacturing processes available to manufacture the product being designed (e.g. slots and holes).

Feature model semantics is mostly represented by a variety of constraints. Constraints can be used in feature modelling systems to express characteristics of the model (e.g. to specify some feature faces to be co-planar, or restrict a given dimension to a certain range). But, above all, constraints are used as the internal constituents of features that express their semantics (e.g. a hole feature could have constraints to position and orient it, or constraints that express the physical limits of the drilling machinery available). Because of this central role of constraints, feature modelling systems have to make an intensive use of constraint solving techniques. In particular, geometric constraints and geometric constraint solving techniques are very common.

To ensure that feature model semantics is maintained, the validity of the feature model has to be checked after each model modification. Feature model validity is usually checked by solving the constraints in the model: a valid feature model is a feature model that satisfies all its constraints. Modelling systems which guarantee feature model semantics to be maintained throughout the modelling process are called semantic feature modelling systems (Bidarra and Bronsvoort, 2000). So constraints play an important role during model creation and modification.

Quite some research work has been done on techniques to enable constraint solvers to be used in interactive applications, such as user interface onstruction (Borning and Duisberg, 1986; Freeman-Benson, 1993; Hosobe, 2001), and geometric modelling systems (Hsu et al., 1997; van Emmerik, 1991). However, in current modelling systems, the specification and the modification of feature parameters that determine its position/orientation and its dimensions, is still mostly done through the input of values in dialog boxes, after which the model is updated accordingly (Parametric Technology Corporation, 2006; SolidWorks Corporation, 2006; UGS Corporation, 2006). 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 of constraints 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. In other words, there is no explicit feedback on which features were affected and how.

non-intuitiveness due to the fact that the user is simply editing values in dialog boxes that do not express how the feature is affected by the parameter.

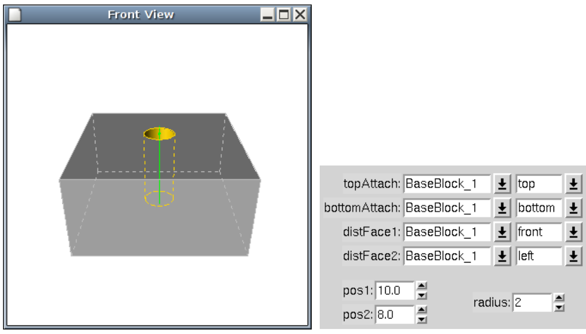
As a result of these drawbacks, all too often designers are forced into using a trial-and-error approach to find the right feature parameter to be changed or to find the right value for the parameter.

Good interactive facilities for direct manipulation of features should always deal with the three drawbacks mentioned above. In this research, we developed a new approach that allows the designer to select a parameter of a feature in the model, and subsequently modify its value interactively, while being provided with real-time feedback on the consequences of the operation. When the designer is satisfied with the model, he can choose to provisionally accept the changes and, eventually, let the system check the model validity.

The most crucial aspect of this approach consists of being able to provide real-time feedback on the changes effected to the feature model. Since this visual feedback has to be generated several times per second to support interactive modification of a feature parameter value, all geometric constraints have to be solved at that same pace. To achieve this, we developed a technique that (i) reduces the time needed to solve a geometric model, (ii) can be applied with a variety of constraint solvers, and (iii) can be easily implemented. This technique has been recently presented in (Lourenço et al., 2006), which contains a detailed description of our model compilation and constraint solving approach, together with a performance analysis of its prototype implementation.

In this paper we focus on how our approach solves the other two drawbacks mentioned above. Throughout the paper, we deal with the situation in which a real-valued feature parameter that determines a dimension, or the position or orientation of a feature in a feature model, is interactively manipulated by a designer. All aspects of our approach described in the paper were implemented in Spiff, a prototype feature modeling system developed at Delft University of Technology.

We first introduce various aspects involved in our approach to direct manipulation of features (Section 2). Next we propose several types of feature handles (Section 3), and describe how they are utilised to yield advanced interactive facilities (Section 4). Finally, some conclusions are drawn (Section 5).



(a) (b)

Figure 1: A through hole feature (a), and its parameters (b).

2 Direct manipulation of features

Features can be modified by manipulating their parameters. Although a parameter of a feature can also be a face of another feature to which it is attached, or with respect to which it is positioned, this paper only deals with manipulation of real-valued feature parameters, such as the dimension of a feature, the distance of a feature with respect to a face of another feature, etc. An example of a through hole feature with its parameters is given in Figure 1, showing the typical entry fields for the various parameter values.

2.1 Feature manipulation phases

Direct manipulation of a real-valued feature parameter consists of two phases. In the *selection phase*, the parameter to be manipulated has to be selected. In the *interaction phase*, the value of the parameter is interactively changed by the designer, and the feature model is updated accordingly by the system.

In the interaction phase, the designer changes the value of a feature parameter, by using the mouse to drag an icon that represents the feature parameter on the displayed feature model. During the dragging, the model and its visualization are updated continuously to reflect the modifications.

The interaction phase needs to be performed real-time, since the designer needs the feedback of the image of the changed model on the display while dragging the mouse. Real-time here means fast enough to preserve the illusion of movement, i.e. the illusion that consecutive images of the same object in a somewhat different position, show a moving object. This illusion is preserved when the system displays more than 10 frames (or images) per second (Card et al., 1983).

2.2 Model validity maintenance

Manipulating the parameter of a feature in a model may turn a valid feature model into an invalid one (see Section 1), e.g. because an undesirable interaction occurs between two features, or a dimension does not satisfy its dimension constraint anymore.

An invalid situation should preferably be detected during the manipulation of the model, and the designer should preferably be immediately informed on it. In case that it is not feasible to detect the invalid situation during the manipulation of the model, for example, because it takes too much time to check the validity of the model, the validity of the model should be checked as soon as the manipulation of the model is ended.

However, in case a model has become invalid during manipulation of a feature parameter, further manipulation should not be prohibited, because the model may turn valid again if the value of the parameter is changed even more. For example, if the model of Figure 1 would also contain a through slot that is positioned to the left of the hole, and the through slot would be moved to the right by manipulating its position parameter, then, as the through slot and the hole start to overlap, the model becomes invalid, but the model turns valid again when the through slot is moved beyond the hole.

If the model is invalid at the moment that the manipulation of a feature parameter is ended, then some validity maintenance mechanism (Bidarra and Bronsvoort, 2000) should be triggered to assist the designer to make the model valid again.

2.3 Constraint management

To solve the geometric constraints in the model, a constraint management scheme is used. The constraint management scheme maps a high-level constraint model, containing the complex design constraints, into a large low-level constraint model, containing primitive constraints, that can be solved by the constraint solvers used, and updates the high-level constraint model based on the solved low-level constraint model.

Fortunately, in the interaction phase of direct manipulation, only part of the constraint model needs to be solved. Since only one feature parameter is changed during the interaction phase, typically, large parts of the model do not change, i.e. they are rigid. Such rigid parts can, therefore, be represented by a single constraint variable in the low-level constraint graph, thus avoiding the need to solve all constraints within the parts.

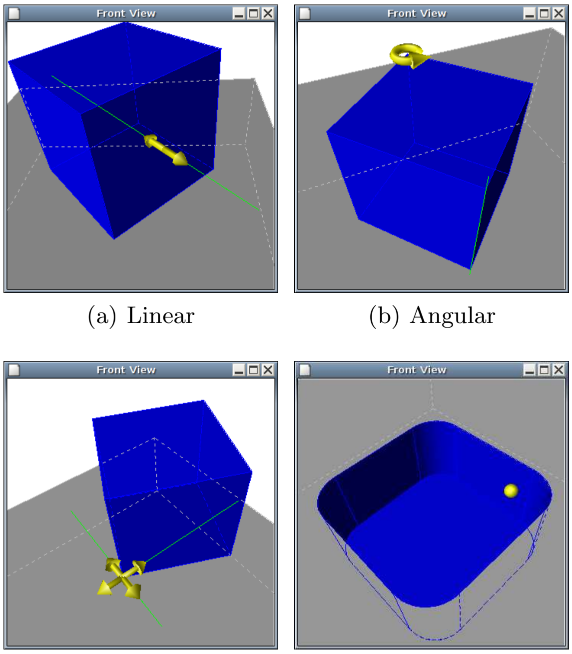
Constraint management for interactive feature manipulation identifies all rigid parts of the model, and maps each one to a separate constraint variable in the low-level constraint model that is solved in the interaction phase. The resulting, simple, constraint model is then used to find the relative position and orientation of the rigid parts during the interaction phase, given the current value of the feature parameter that is changed. Again, this model compilation and constraint solving approach is described in (Lourenço et al., 2006), to which the reader is referred for many details on its fundamentals, implementation and performance.

3 Types of feature handles

Not all parameters of features have a direct geometric meaning. For such parameters, it is not possible to assign a feature handle with a natural behaviour. A good example of this situation is a feature that has a volume parameter. On the other hand, most parameters do have a simple geometric meaning, e.g. the width of a protrusion or the rotation angle of the same protrusion. As stated before, in this paper we deal only with such real-valued parameters. For these, four types of handles were identified that cover most direct manipulation needs:

- Linear handles
- Angular handles
- Planar handles
- Slider handles

In this approach all handles of a feature are specified by their relation with reference elements (e.g. reference points and reference lines) of the feature. With these references the system will be able to know the placement of the handles as well as their behaviour as they are being manipulated. 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 solely determined by the constraint solver.



(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:

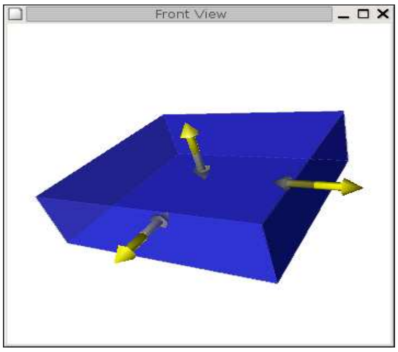
- A *linear handle* is a handle that moves along a straight line and reacts linearly with the mouse movement. Besides the reference point representing the position, it also contains a reference line representing the line on which the handle moves. When the handle is dragged, the new mouse position is projected against the reference line, and the parameter variation is given by the difference between the computed position and the previous position of the handle. See Figure 2(a).
- An *angular handle* is a handle that moves along an arc in a way that the user is able to specify an angle parameter. This handle has a reference line besides the position reference

that represents the axis of rotation around which the angular handle revolves. When the handle is dragged, the new mouse position is projected onto the plane perpendicular to the axis of rotation which passes through the current position of the handle thereby computing *point1*. To determine the change in the parameter affected by this handle an angle is computed. This angle is the angle between two lines in the plane just mentioned which pass each through the axis of rotation and through the original position of the handle and *point1* respectively. See Figure 2(b).

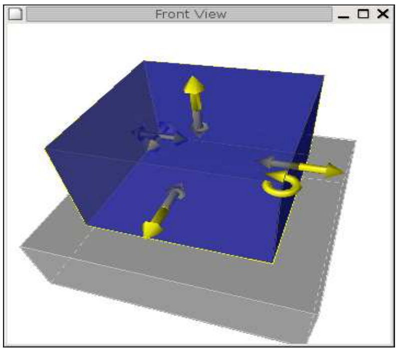
- A *planar handle* behaves somewhat like a linear handle with the difference that, instead of having its movement restricted to a line, its movement is restricted to a plane (having two linear degrees of freedom). For this type of handle two reference lines will be needed to determine the plane on which it can move. One thing that differs from this handle to all the others is the fact that by manipulating it the user is affecting two parameters of the feature at the same time instead of one. To determine the change in the parameter values a computation similar to the one used for the linear handle will happen with the difference that the mouse position will now be projected onto each of the reference lines corresponding to each of the parameters affected. 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 being changed by such parameter manipulation. Therefore this handle, unlike any of the other handles, will contain solely a reference point for its position and no reference lines. The slider handle suits nicely to provide direct manipulation to parameters that have no simple nor easy-to-localize geometric meaning, as the corner radius in the pocket of Figure 2(d).

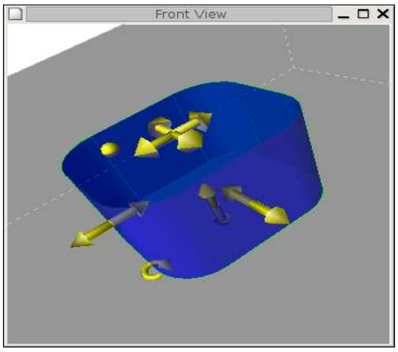
By combining these four kinds of handles in feature class definitions, the parameterization of a large variety of feature classes is easily made ready for the advanced direct manipulation facilities described in the next section. Figure 3 shows a few examples of such feature classes.



(a) Base Block feature with handles for width, height and length



(b) Block Protrusion feature with handles for width, height, length, rotation and position



(c) Rounded Rectangular Pocket feature with handles for width, height, length, rotation, position and corner radius

Figure 3: Examples of handle specifications for three different feature classes.

4 Advanced interactive facilities

The direct manipulation approach developed should overcome the three disadvantages of traditional feature manipulation mentioned in Section

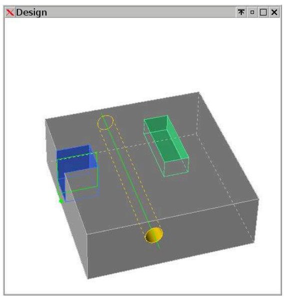
1. In other words, it is required that (i) feedback is efficient, (ii) that insight into the modelling operation consequences is provided, and (iii) that the relation between what's manipulated and the way it affects the feature becomes intuitive. In this section we describe how this was achieved.

For efficient feedback every modelling operation leads to an immediate preview of the result of the operation to the user. This preview is provided through a transparent overlay display of the resulting model. The user has insight on the consequences of the operation, which become clear through the comparison of the original model with the simultaneously displayed preview. The manipulation is intuitive because of the feature handles' characteristics. The deployed feature handles (see Section 3) take into account that each feature parameter has a specific semantics. This semantics will be expressed through the handle's behaviour, positioning and iconic representation.

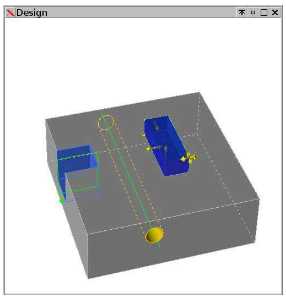
We now give a brief description of how the user interacts with the feature model under this approach, in a way that materializes the requirements presented previously. For this, the example given in Figure 4 will be used.

- When a user selects a feature to be edited it is highlighted with a transparent overlay with a distinguishing color and its available handles are displayed on the screen; see Figure 4.(b).
- The user may then edit the handles by dragging them. When the user does so the display is updated according to the change in the parameter. When this happens, all the features which were affected by the variation of the parameter also appear as a transparent overlay; ; see Figure 4.(c). These changes are not final and can be undone.
- When the user chooses to Apply the modelling operations, changes which have been made (and highlighted) previously become final and the full model display is updated with the new parameter values; see Figure 4.(d).
- When the user chooses to Dismiss the modelling operation, the changes which have been made since the last Apply are undone.

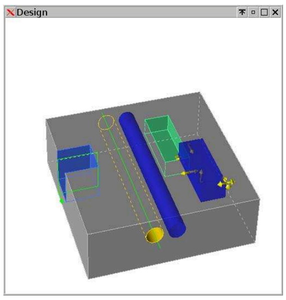
As described in Section 2.1, two important phases exist in the flow of events for handle manipulation: the selection phase, when the user selects a parameter of a feature to be edited, and the interaction phase, each time the respective handle is dragged.



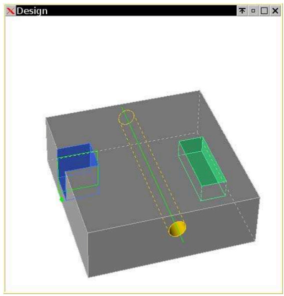
(a) Initial model



(b) selection of the pocket



(c) dragging of the pocket



(d) Final model

Figure 4: Interactively modifying a feature: the through hole, positioned relatively to the pocket, is highlighted and dragged together with it.

Figure 5 is a high level diagram of the flow of events that happens when the user selects a feature to be edited. Here, the user selection 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.

The flow of events that results from a single drag event of the user is depicted in figure 6. 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 3). After the new value is set, the Operator orders the Constraint Manager to solve the model. This solving process is done with the incremental constraint management solution specially optimized for direct manipulation described in (Lourenço et al., 2006). The simple idea of this process is to add a preprocessing step when a parameter is first manipulated that boosts the constraint solving performance to further changes in that same parameter. With the new model values resulting from the constraint

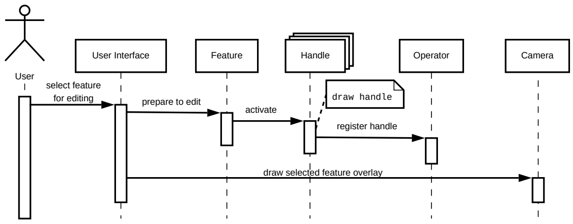


Figure 5: Interaction diagram of what happens when the user selects a feature for editing.

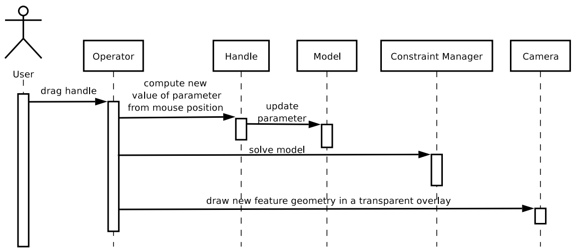


Figure 6: Interaction diagram of what happens when the user drags a handle.

solving process a transparent preview of the result of the operation is rendered and displayed.

5 Conclusions

In this paper a new approach to the direct manipulation of feature models was presented. Several types of handles for parameters with a simple and intuitive geometric meaning were introduced. In addition, a *slider* handle was developed, for situations in which the feature class designer wants to add direct manipulation to a generic parameter lacking an easy-to-localize geometric meaning. The main aspects of the new approach were discussed, together with its prototype implementation within the Spiff feature modelling system, demonstrating its value and feasibility.

The definition of handles at the feature class level is such that handles only depend on reference elements that are specified as any others in the system. This solution has the advantage of having the position and orientation of handle references automatically computed in the constraint solving process.

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 through its handles. In this way the user can clearly see the effects of the modelling operations and compare the result with the original situation.

The prototype implementation has been successful in making the manipulation of feature models a very intuitive process, effectively improving the user experience and, therefore, confirming the high potential of the approach.

REFERENCES

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

Borning, A. and Duisberg, R. (1986). Constraint-based tools for building user interfaces. *ACM Transactions on Graphics*, 5(4):345–374.

Card, S., Moran, T., and Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Lawrance Erlbaum Associates, Hillsdale, N.J.

Freeman-Benson, B. N. (1993). Converting an existing user interface to use constraints. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 207–215. ACM Press.

Hosobe, H. (2001). A modular geometic constraint solver for user interface applications. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 91–100. ACM.

Hsu, C., Huang, Z., Beier, E., and Brüderlin, B. (1997). A constraint-based manipulator toolset for editing 3d objects. In *Proceedings of the fourth ACM symposium on Solid modeling and applications*, pages 168–180. ACM.

Lourenço, D., Oliveira, P., Noort, A., and Bidarra, R. (2006). Constraint solving for direct manipulation of features. *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 20(4):369–382.

Parametric Technology Corporation (2006). Pro/ENGINEER product information. http://www.ptc.com.

Shah, J. J. and Mäntylä, M. (1995). *Parametric and Feature-based CAD/CAM*. John Wiley & Sons, Inc., New York.

SolidWorks Corporation (2006). Solidworks 2006 product information. http://www.solidworks.com.

UGS Corporation (2006). Unigraphics NX product information. http://www.ugs.com.

van Emmerik, M. J. G. M. (1991). Interactive design of 3D models with geometric constraints. *The Visual Computer*, 7(5/6):309–325.