Automatic Synchronization of a Feature Model with Direct Editing based on Cellular Model

Abstract

As the de-facto standard in modern CAD systems, *feature-based modeling* is widely used for parametric design. *Direct modeling*, in contrast, is an emerging technology which allows users promptly edit B-rep models without involving the feature model, i.e. the history tree. The two modeling technologies have their own advantages but are hard to communicate. To exert the powers of both technologies in a single design session and enable the interoperability between them, we propose a method that automatically converts direct modeling operations into history tree modifications, for synchronizing the feature model. We sequentially adopt three synchronization strategies - feature definition modification, feature order change and feature add/remove - to preserve design semantics of a feature model as much as possible. With a series of experiments, we demonstrate that our synchronization method can successfully fuse direct modeling and feature-based modeling in a natural and efficient way.

Key words: direct modeling, feature-based modeling, automatic synchronization, cellular model, design semantics

1. Introduction

In the last two decades, parametric and feature-based modeling [1] played a great role in computer-aided design and brought huge economic benefits to manufacturing industries. Rather than low-level editing, the technology enables a designer to create or edit high-level design elements, i.e. "*feature*", which possesses rich engineering/manufacturing semantics. However, feature-based modeling has its own drawbacks [2], e.g. high complexity, order dependency and predefinition of changeable parameters, which can hamper the modeling flexibility. Recently, direct modeling technology is rapidly developing and getting used in commercial CAD systems like SpaceClaim [3], NX [4] and Creo [5]. Direct modeling is friendly to normal users, as one can easily manipulate geometric elements to edit a native model, without understanding the complex or even tricky design semantics in feature history tree. Compared with feature-based modeling, direct modeling is a lightweight editing tool, and behaves more efficient and flexible. Early/conceptual design or personal customization/fabrication can all benefit from it. While the above two technologies have their own merits, they can be complementary to each other. They are often used in different design phases or preferred by experts from different domains. Actually, the interoperability and communication between the two modeling methods is crucial to design success, as designers need to transfer design knowledge between different design phases, and domain experts need to share idea or edit models during collaboration. Unfortunately, this kind of interoperability is still hard and rare, which severely hinders design innovation or even fails a project.

While feature-based modeling uses feature model to record the parametric design process, direct modeling operation is an immediate event that makes a change to geometric model (B-rep). Once users edit a parametric model through direct modeling, the underlying B-rep is modified without involvement of the history tree. To keep the parametric information up-to-date and valid, the feature model should be synchronized accordingly, and the new feature model should be able to faithfully regenerate the modified geometric model. This is the key to maintain consistency and enable interoperability between the two modeling technologies.

In this paper, we present an automatic method to smoothly convert direct modeling operations into history tree modifications. Bases on feature representation harnessed on cellular model [6], we evaluate the influence of geometric editing on feature definitions and update the history tree. It must be emphasized that the history tree modification is not unique, as there are many possible ways to update the history tree for regenerating the latest geometric model. We thus design three synchronization strategies following the principle of least astonishment: a) modify feature definitions; b) adjust feature orders in history tree; and c) add/remove specific features. We executed them sequentially to ensure a successful synchronization,

while introducing least modifications to the original design history.

Contribution In summary, our work makes following contributions:

- We propose an automatic synchronization method to ensure consistency between the geometric model under direct editing and the feature definitions in history tree, and thus form a seamless bridge between direct modeling and feature-based modeling.
- We design three synchronization strategies to update the history tree, with a minimum deviation from the original design semantics.
- We leverage cellular model to efficiently detect geometric consistencies and modify feature information.

2. Related Work

Model Conversion In solid modeling, boundary representation (B-rep) and constructive solid geometry (CSG) are two dominating methods for representing shapes. The conversion technology between B-rep and CSG is of significance for developing a dual-representation modeling system. Requicha and Voelcker [7] presented the boundary evaluation and merging algorithms which described set membership classification and neighborhood manipulation in detail. Through the work, the problem of computing B-rep from CSG representation can be well understood. The inverse problem was systematically studied by Shapiro et al. [8–10]. They mainly considered two aspects of conversion from B-rep to CSG, i.e. the construction of separating half-spaces and the optimization of the resulting CSG.

The smooth exchange of part models among different vendor systems has practical importance. Kim et al. [11] put forward a foundation for the standardized intersystem exchange of parametric models. Several researches dedicated to exchanging parametric design information between heterogeneous CAD systems, such as macro-parametric approach [12], universal product representation (UPR) architecture [13], and neutral modeling commands [14].

Bronsvoort et al. [15,16] presented a multiple-view feature technology to convert various data between different design phases, which enabled an integral environment for product development.

In our work, we focus on smart conversion from direct modeling operations to feature model modifications, facilitating users with powers and flexibilities of both modeling technologies.

Feature-based Modeling Feature-based modeling, employing features as the elementary units to construct solid models, mainly includes two categories of methods: one is feature-based design, e.g. machine feature-based decomposition [17] and design feature based synthesis [18,19]; the other is feature recognition [20–22] which can be classified into volumetric decomposition, hint-based geometric reasoning and graph-based algorithms.

Direct Modeling Different from the features, parameters and constraints involved in feature-based modeling, the only input of direct modeling is a B-rep. Users can directly drag or rotate the geometric elements to edit the shapes. The efficiency, flexibility and simplicity drew great attentions from the industry. For more information, please refer to [23–26].

Cellular Model Cellular model was developed based on the non-manifold boundary representation, of which the boundary evaluation and boolean operations were reported in [27,28]. Bidarra et al. [6] first proposed to manage feature information with cellular model. This representation brought efficient boundary evaluation for feature modeling [29]. After that, extensive studies on application of cellular model were conducted, such as semantic feature modeling [2], multiple-view feature modeling [15], feature model visualization [30], progressive solid models generation [31] and feature-based multiresolution modeling [32]. In this paper, we adopt cellular model for detecting geometric consistencies and manipulating features.

3. Overview

Our synchronization method takes as input (i) a parametric model M, (ii) a direct modeling operation d that modifies the underlying geometries of M. In particular, M is defined as $\{F, O, G\}$, in which $F = \{F_0, F_1, \cdots, F_n\}$ is the sequence of features, $O = \{\otimes_1, \otimes_2, \cdots, \otimes_n\}$ is the sequence of boolean operations "$+$, $-$ and $\cap$" applied on features, and G is the geometric model computed by:

$$G = F_0 \otimes_1 F_1 \otimes_2 F_2 \cdots \otimes_n F_n.$$

After the direct modeling operation (Fig. 1(b)), we have a new geometric model $G' = d(G)$.

Our synchronization method computes a new feature sequence F' and corresponding boolean operation sequence O' such that the geometric model computed from F' is exactly the same as the edit result G':

$$F'_0 \otimes'_1 F'_1 \otimes'_2 F'_2 \cdots \otimes'_m F'_m = G'.$$

We note that there is usually no unique F' satisfying the above constraint. By following the principle of least modification to the original feature model, we design three synchronization strategies:

1. *Modification of feature definitions* Modify the parameters and sketch of the original features.
2. *Adjustment of feature orders in history tree* Adjust the positions of specific features in design history while maintaining the inherent feature-dependencies.
3. *Addition and deletion of features* Add new features into history tree or delete useless features from it, which ensures the success of synchronization.

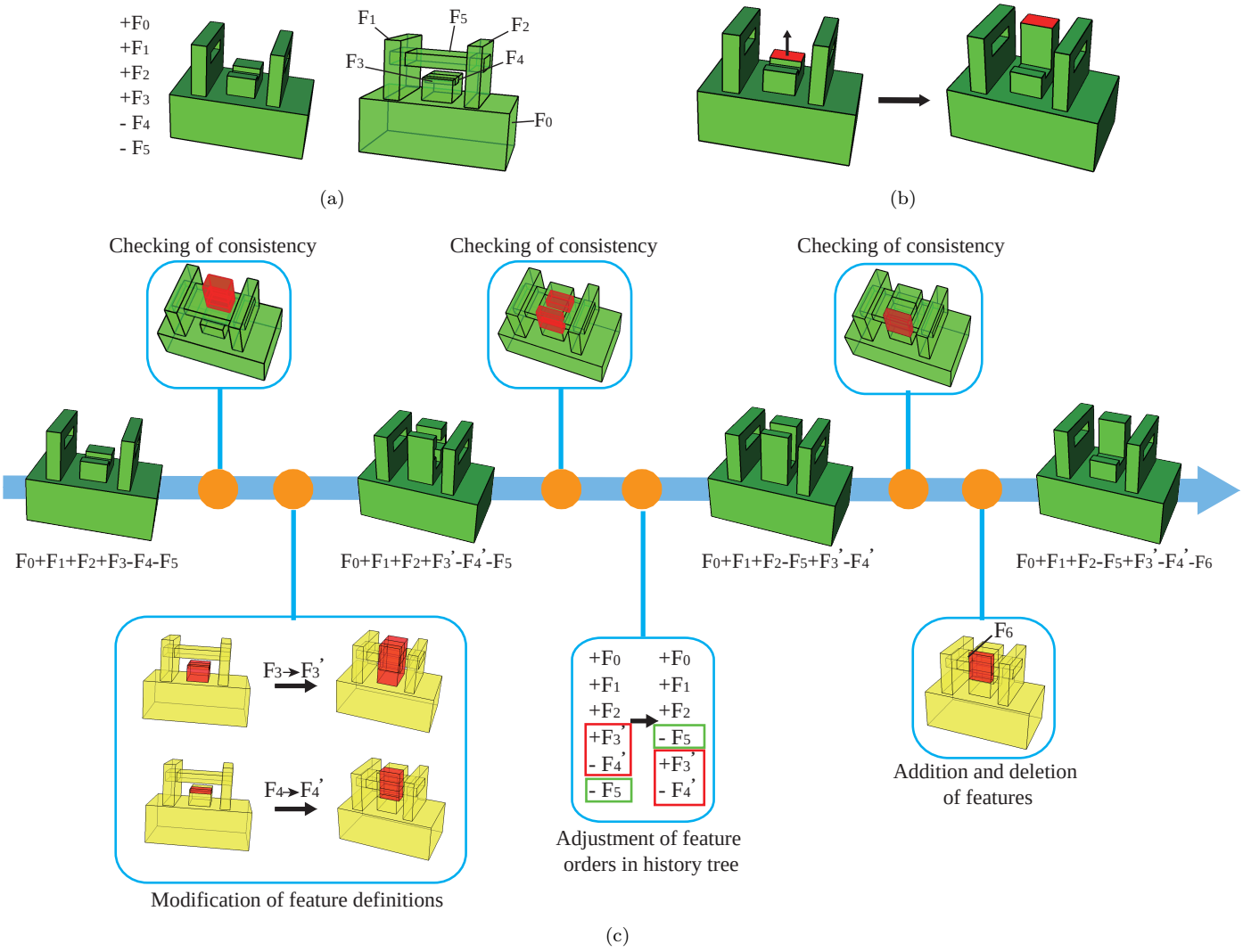


Fig. 1. Overview of synchronization method: (a) the original feature model; (b) the direct editing; (c) the synchronization process.

As shown in Fig. 1(c), the three strategies are sequentially applied to compute a valid F' that preserves the design semantics as much as possible.

In this work, we only consider extrusion feature, which is mostly used in practical, to simplify the synchronization problem. However, the main process is applicable to other kinds of features, e.g. rotations, with specific sub-algorithms carefully designed. We also assume that inter-feature geometric constraints do not exist in the input model, which is left as the future work.

4. Algorithm

Algorithm 1 lists the main pipeline of our synchronization processing. The three synchronization strategies are executed in order, until the underlying geometry of the new feature model is consistent with the direct editing result. Before giving details of each synchronization strategy, we first introduce the cellular-based feature model briefly, which is essential for checking geometric consistency and manipulating feature information.

Algorithm 1 Feature Model Synchronization

```
1: procedure SYNCHRONIZATION(M, G')
2:     CM ← ConstructCellularModel(M)
3:     CM.AddFeature(G')
4:     if IsConsistent(CM) then return
5:     ModifyFeatureDefinitions(CM)
6:     if IsConsistent(CM) then return
7:     AdjustFeatureOrders(CM)
8:     if IsConsistent(CM) then return
9:     AddDeleteFeatures(CM)
10:    return
```

4.1. *Cellular-Based Feature Model*

Representation Cellular model is a non-manifold geometric representation [33]. It consists of a connected set of volumetric quasi-disjoint cells. As shown in Fig. 2(a), cells

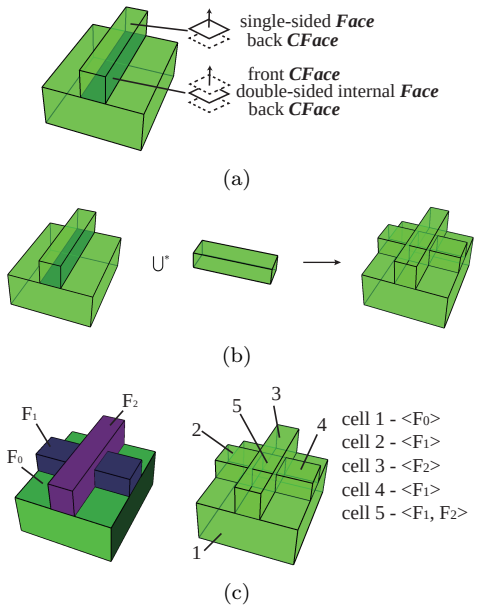


Fig. 2. Cellular model: (a) the representation of cellular topology; (b) the non-regularized Boolean unite operation; (c) the cellular-based feature model.

represent 3D (volumetric) regions closed by cell faces. Two cells can be separated by a double-sided internal face, with each side corresponding to a cell. Single-sided faces lie on the shape boundary. Cellular model also supports Boolean unite/intersect/subtract operations as B-rep model but are non-regularized version. As shown in Fig. 2(b), when non-regularized Boolean unite operation is performed on two cellular models, the overlapping region is split into a new cell.

Bidarra et al. [6] proposed a feature representation based on cellular model. As shown in Fig. 2(c), they associate an owner list to each cell storing what features the cell belongs to, and give each cell a nature attribute specifying whether the region of the cell is additive or subtractive, i.e. adding or removing material to the volume. A face is on the boundary of the feature model if its two sides have opposite nature.

Manipulation and Construction The cellular-based feature model supports feature addition, feature deletion, and feature modification operations.

When inserting a new feature, the model is updated as follows:

1. Instantiate the new feature as a cellular model with only one cell.
2. Perform a non-regularized Boolean union between the new cellular model and the original cellular model. Whenever two cells undergo overlapping and result in mutual decomposition, the new cell merges both owner lists of the overlapping cells.
3. Sort the features in each cell's owner list according to feature orders in history tree, and set each cell's nature as same as the last feature in it's owner list.

When deleting a feature, the model is updated as:

1. Remove all the references to the deleted feature from each cell's owner list.
2. Merge the adjacent cells with the same owner lists and discard the cells whose owner lists are empty.

Feature modification operation is achieved by removing an existing feature from the cellular model and then inserting a new feature. As only Boolean unite operations are used in cellular-based feature model, the boundary evaluation result is independent of the operation orders. This characteristic greatly reduces the time complexity of feature manipulations, since there is no need to re-execute the whole history tree. Due to the fact that feature operations are intensively used for synchronization, using the cellular-based feature model can largely improve the efficiency.

We construct the cellular-based feature model as the first step of the synchronization algorithm. During the construction, we insert the features of the history tree into a cellular model one-by-one, and update the feature information accordingly.

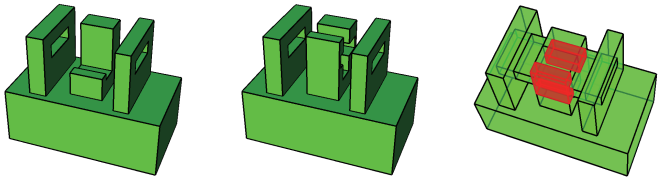
4.2. *Checking Geometric Model Consistencies*

The synchronization succeeds if the underlying geometry of cellular-based feature model is consistent with the direct editing result. By regarding the direct modeling result as a feature F_d and inserting it into the cellular model, we can efficiently check the geometric consistencies by comparing the current nature and the target nature of cells. The current nature of each cell is evaluated without considering F_d in the owner list, and the target nature of a cell is additive if its owner features contain F_d, and vice versa. We call a cell *conflict* if its current nature and target nature are opposite. Therefore, the synchronization is success iff there exist no conflict cells in the cellular model. The pseudo-code of consistency checking algorithm is shown in Algorithm 2.

Algorithm 2 Consistency Checking

```
1: procedure IsConsistent(CM)
2:     conflictCellList.clear()
3:     for each cell ∈ CM do
4:         realNature ← cell.getRealNature(CM)
5:         targetNature ← cell.getTargetNature(CM)
6:         if realNature ≠ targetNature then
7:             conflictCellList.push(cell)
8:     return conflictCellList.isEmpty()
```

Fig. 3 shows a checking instance. In this algorithm, we traverse the cellular model only once to detect conflict cells, which is highly efficient. Without cellular-based feature model, Boolean subtract operation has to be performed to detect B-rep consistency, which is not only slow, but also prone to fail.



(a) (b) (c)

Fig. 3. Detection of conflict cells: (a) the direct editing result; (b) underlying geometry of current synchronized feature model; (c) conflict cells in cellular model (red).

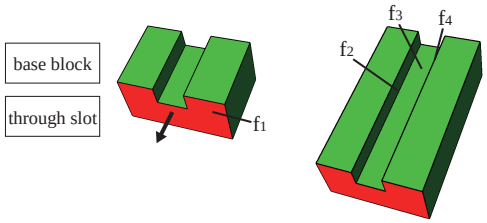
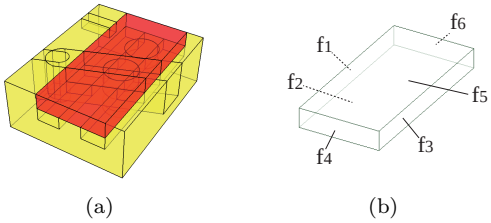


Fig. 4. Identification of affected features.



(a) (b)

Fig. 5. Feature faces: (a) A slot feature (red) in the cellular model; (b) the corresponding boundary feature faces and non-boundary feature faces.

4.3. *Modification of Feature Definitions*

To modify design semantics of the feature model as little as possible, we first try to synchronize through modifying definitions of the original features. After identifying the features affected by direct editing, we reconstruct these extrusion features through modifications on sketch shape and feature parameters. The detailed procedure is shown in Algorithm 3.

Algorithm 3 Feature Definition Modification

```
1:  procedure ModifyFeatureDefinitions(CM)
2:      afs ← IdentifyAffectedFeatures(CM)
3:      Sort afs by the order of history tree
4:      for each oldf ∈ afs do
5:          sg ← DetermineSurfaceGroup(CM, oldf)
6:          msg ← MaxSurfaceGroup(sg)
7:          oldsg ← oldf.GetOldSurfaceGroup()
8:          if msg ≠ oldsg then
9:              sketch ← ConstructSketch(msg)
10:             newf ← ReconstructExtrusion(sketch)
11:             UpdateCellularModel(oldf, newf)
12:     return
```

4.3.1. *Identification of Affected Features*

Since definition modifications are applied on the original features, we first identify the features possibly affected by direct modeling operation. During the direct editing process, faces are dragged or rotated, which results in conflict cells. According to the operated face and the conflict cells, three kinds of affected features are identified: (i) Corresponding features of the operated faces; (ii) The features whose faces overlapped or intersected with the operated faces; (iii) Features in the owner list of the conflict cells.

For the instance shown in Fig. 4, face f_1 is moved, so we identify the block feature as an affected feature. And by checking the adjacent faces f_2, f_3 and f_4 of operated face f_1, we also identify the through slot feature as an affected feature.

4.3.2. *Reconstruction of Extrusion Feature*

We reconstruct the affected extrusion features based on their surface group after direct editing, which is processed according to their orders in the history tree. By modifying the sketch and the extrusion direction, we try to define the maximum extrusion feature.

Surface Group Determination The surface in the extrusion feature is constructed using the geometry equation of *boundary feature face* and *non-boundary feature face*. Boundary feature faces are the primitive faces of the feature at the boundary of the B-Rep model, and non-boundary feature face are those not at the boundary. As shown in Fig. 5, faces f_1, f_2 and f_3 are boundary feature faces, while f_4, f_5 and f_6 are non-boundary feature faces.

The boundary feature faces are derived from those of the affected features in the original model. For instance, as shown in Fig. 6, the boundary feature faces of the slot feature in Fig. 5 after direct editing are derived from the corresponding boundary feature faces before direct editing. When only parts of the feature faces sharing the same surface are affected by direct modeling, we select the face group with larger area as the boundary feature face. This accords more with human visual perception. As shown in Fig. 6, four blue boundary faces are moved while the red face is not affected, and we choose the blue face group to form the boundary feature face.

We determine non-boundary feature face using reference feature face, which is coplanar with other feature faces in feature model. Fig. 7(a) shows a block feature cut by a through slot feature, and the through slot feature is bounded in the block feature. The three non-boundary feature faces of the through slot feature f_4, f_5 and f_6 are coplanar with the three feature faces f_4', f_5' and f_6' of the block feature respectively. To keep the original design semantics, we preserve this type of coplanar relationship. In particular, reference feature faces must satisfy two conditions: (i) be coplanar and overlapped with current non-boundary feature face; (ii) be constructed before current feature.

Since the reference feature face is possibly changed dur-

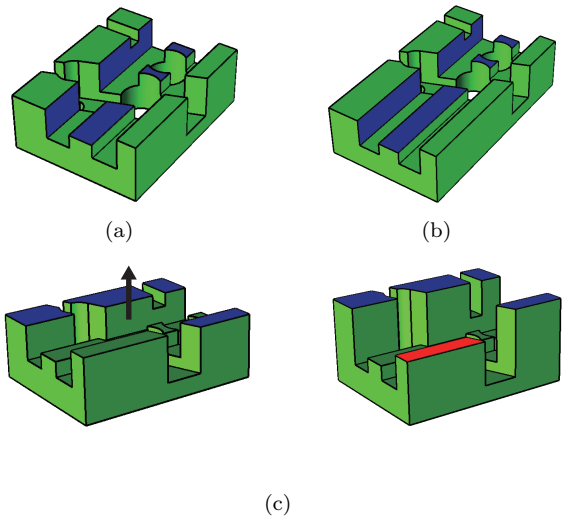


Fig. 6. Determination of boundary feature faces: (a) the corresponding boundary feature faces (blue) of the slot feature in Fig. 5 before direct modeling operation; (b) the corresponding boundary feature faces (blue) of the slot feature in Fig. 5 after direct modeling operation; (c) partially affected boundary feature faces.

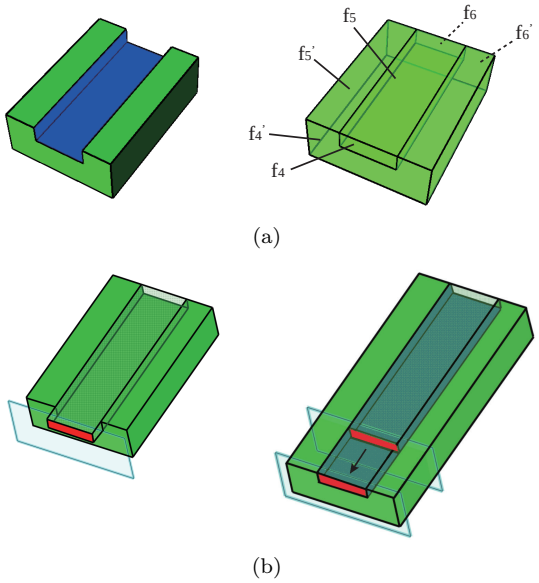


Fig. 7. Determination of non-boundary feature faces : (a) coplanar reference feature face; (b) non-boundary feature face without reference feature face.

ing the process of direct modeling, non-boundary feature face is determined after direct modeling. In the process of synchronization, the affected features are sorted by the order of history tree and reconstructed in turn, which guarantees the correctness of reference feature faces. The reference feature faces are efficiently determined using cellular model. In case that non-boundary feature face cannot be determined using reference feature face (Fig. 7(b)), we only estimate a reasonable geometric equation. Two heuristic rules are adopted:

1. Set the normal of feature face as unchanged, in order to introduce least modification to the original model.

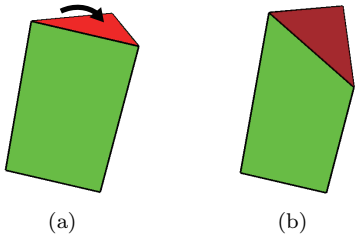


Fig. 8. Maximum extrusion surface group: (a) original extrusion feature; (b) direct editing result with the top face (red) rotated.

2. Put the feature face on either side of other feature faces, forming the minimum positive volume of the face group.

Maximum Extrusion Surface Group The surface group with maximum extrusion possibility is recognized if the feature is no longer a valid extrusion feature after direct editing. As shown in Fig. 8, the top face of the prism is rotated, resulting in a non-extrusion feature. In this case, the maximum extrusion surface group that satisfies two conditions is determined: (i) least number of faces are excluded; (ii) the face group can be extruded along one direction.

Therefore, the maximum extrusion surface group are determined as follows:

1. Collect a feature face set which consists of all the boundary feature faces and non-boundary feature faces with reference feature faces.
2. Enumerate all the possible extrusion directions for the feature face set of step 1, and determine the corresponding extrusion face groups.
3. Select the face group with maximum number of faces as the maximum extrusion face group.

In addition, all the feature faces not in the maximum extrusion face group are regarded as the non-boundary feature face without corresponding reference feature face. In Fig. 8, the maximum extrusion surface group contains all but the rotated feature face.

Sketch Construction Based on the maximum extrusion surface group, the extrusion feature is constructed by extruding the sketch along its perpendicular direction. As the extrusion direction is perpendicular to the sketch plane and the extrusion length is the distance from extrusion source face to extrusion target face, the remaining problem is how to construct the sketch. Based on the original topology of side faces in the extrusion feature, we construct the sketch as follows:

1. Project all the surfaces of side faces onto the sketch plane and obtain curves of the surfaces.
2. Intersect the curves based on the original topology of side faces and obtain the divided edges.
3. Connect the conjoint edges and complete the sketch.

As shown in Fig. 9, the blue faces in the top row are the corresponding boundary faces of the features, and the red wireframes are the constructed sketches. The red volumes

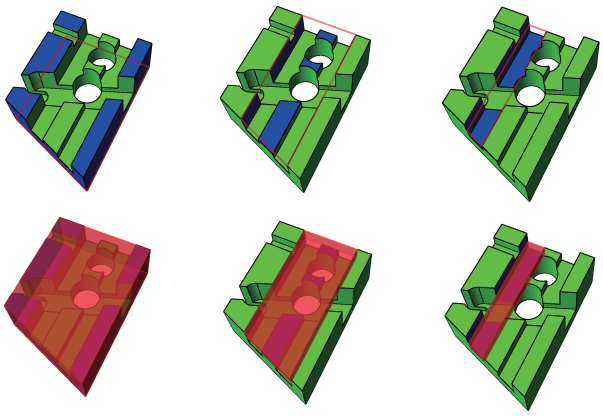


Fig. 9. Construction of sketch in extrusion feature.

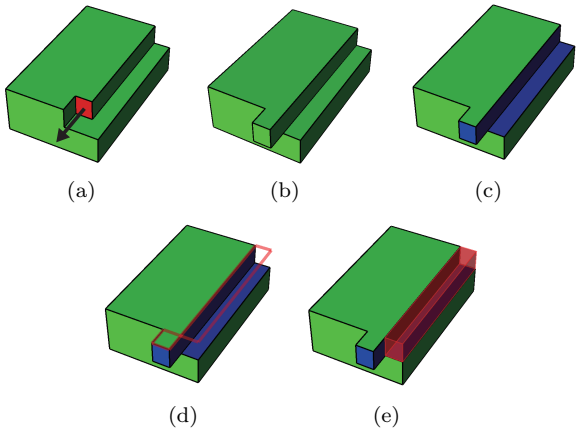


Fig. 10. Repair of self-intersected sketch: (a) the original model which includes a block feature cutting out a L-shaped feature; (b) the direct editing result; (c) the boundary faces of L-shaped feature marked in blue; (d) the self-intersected sketch; (e) reconstructed extrusion feature after repair.

in the bottom row are the reconstructed extrusion features based on the sketches.

When great changes are made during direct editing, the original design semantic is largely modified, which may cause self-intersected sketch. In this case, the sketch is repaired as follows:

1. Intersect non-adjacent edges each other, and split each edge into two new edges respectively if there is any intersection between two non-adjacent edges.
2. Repeat step 1 until there is no edge to split.
3. Find the wireframes that are not self-intersected.
4. Select the wireframe with the largest area as the sketch.

Fig. 10 shows an example of the above method. Combined the sketch of each extrusion feature with the extrusion direction and extrusion distance, we can reconstruct the extrusion feature.

Model Updating Once an extrusion feature is reconstructed, both of the feature model and its cellular-based representation are updated. The affected feature in the feature model is updated only if its feature face geometric

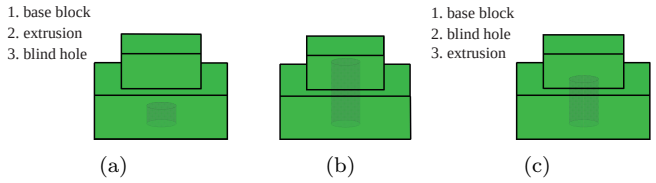


Fig. 11. Influence of feature orders on synchronization: (a) original feature model and corresponding history tree; (b) the depth of the blind hole is modified through direct modeling; (c) inconsistency appears for specific feature orders (the extrusion feature 'cut off' the blind hole).

equations are changed. The cellular representation is also updated during this process, and the nature of each cell turns to be opposite if the primitive of the feature is inverted.

When all of the affected features are reconstructed and updated, geometric consistency checking is performed to detect conflict cells. The synchronization is successful if there is no conflict cell; otherwise we proceed to the next synchronization step.

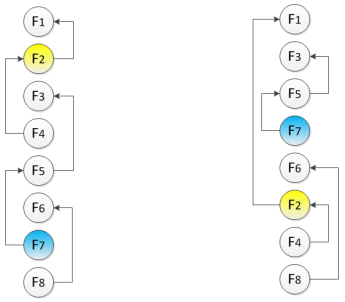
4.4. *Adjustment of Feature Orders in History Tree*

One reason of the geometric inconsistency after direct modeling is the incorrect feature orders in history tree. The nature of new cell introduced by direct modeling depends on latest feature in its owner list. Depending on history tree, the nature of the new intersection region may be different from that of direct modeling result, see Fig. 11 as an example. For successful synchronization, we first analyze the dependency relationship of features, and then swap necessary features to eliminate conflict cells.

4.4.1. *Analysis of Feature Dependency Relationship*

The dependency relationships between features reflect design semantics. Besides new intersections introduced by direct editing, the natures of original intersection regions depend on the related feature orders. Therefore, the dependency relationships should be kept during the process of feature order adjustment, otherwise new conflict cells may appear in the synchronized model.

We use the feature dependency graph which is defined as $G = \{V, E\}$ to described the feature dependencies. $V = \{v_1, v_2, \ldots, v_n\}$ represents the vertices of graph. Each vertex corresponds to a feature. And $E = \{e_1, e_2, \ldots, e_n\}$ represents the directed edges of graph. Feature v_1 depending on feature v_2 is denoted as $v_1 \rightarrow v_2$, which means feature v_1 must be constructed after feature v_2. In addition, the feature dependencies satisfy transitivity, i.e. $v_1 \rightarrow v_3$ if $v_1 \rightarrow v_2$ and $v_2 \rightarrow v_3$. The dependencies are obtained by analyzing the intersection of features, which is convenient in cellular model. Since the analysis of feature dependencies is processed after modification of feature definitions, the conflict cells are ignored to ensure the reasonability of the analysis. We sort the features in each cell's owner list as the order in history tree. If two features with opposite nature



(a) (b)

Fig. 12. Swapping of features: (a) original history tree; (b) new history tree with adjusted feature orders.

are adjacent in owner list, then the latter one depends on the former one.

We use a matrix D to store the feature dependencies: $D_{ij} = 1$ if feature i depends on feature j, and $D_{ij} = 0$ otherwise. Based on the matrix D, we use Warshall algorithm to compute the transitive closure.

4.4.2. *Swapping of Features*

We try to adjust the feature orders based on the feature dependencies to eliminate the conflict cells. Supposing the last feature in the owner list of a conflict cell is feature F_j, if there exists a feature F_i in the owner list that satisfies $i \neq j$ and $D_{ij} \neq 1$, then the conflict cell can be eliminated by adjusting the feature order between F_i and F_j. In order not to introduce new conflicts in the process of feature orders adjustment, we must preserve the original feature dependencies. For instance, Fig. 12(a) shows the original feature history tree, in which the directed edges represents the feature dependency relationships. Obviously, all the directed edges are pointed from the latter features to the former features, and there is no edge cycle. To swap the position of F_2 and F_7 while keeping the original feature dependencies, F_4 must be put after F_2, and F_3, F_5 must be put in the front of F_7 after swapping.

Our strategy for keeping feature dependencies is: collect features with transitive dependencies as a group, and keep the inner orders of features in the group during feature swapping. Given two features F_i and F_j to swap, the feature $F_k (i < k < j)$ between them in the history tree can be classified into three types: F_{k1} that satisfies $F_{k1} \rightarrow F_i$; F_{k2} that satisfies $F_j \rightarrow F_{k2}$; and F_{k3} that no feature $F_t (i < t < j)$ satisfies $F_{k3} \rightarrow F_t$ *or* $F_t \rightarrow F_{k3}$. Based on the classification, the swapping is executed as follows:

1. Collect features of F_{k1} and F_i as group g_1, collect features of F_{k2} and F_j as group g_2 and collect features of F_{k3} as group g_3.
2. Rearrange these features in the order: group g_2 first, then group g_3, and group g_1 last.

After feature swapping, the conflict cells caused by incorrect feature orders can be eliminated. If there exist no conflict cells any more, the synchronization is successful; otherwise we go to the next step of synchronization.

4.5. *Addition and Deletion of Features*

Addition of New Features The synchronization is not guaranteed to succeed through modifying feature definitions and adjusting feature orders in history tree. By adding new features to the feature model, we can always achieve a valid synchronization. The procedures are:

1. Select a conflict cell as the seed, and form a connected region with other adjacent conflict cells having the same nature.
2. Recognize extrusion feature for the conflict region. Terminate the algorithm if the recognition succeeds.
3. If the recognition failed, set the region as a user-defined feature, and set the feature's nature of as opposed to the nature of the conflict cell.
4. Insert the new feature at the tail of history tree and update the cellular model. Go to step 1 if there still exists any conflict cell.

Deletion of Ineffective Features During the synchronization process, the features do not contribute on the final geometry are useless and should be deleted from the history tree. The procedures are:

1. For each additive cell, mark the last additive feature in its owner list as useful.
2. For each negative cell, mark the last negative feature as useful if there exists an additive feature before the negative feature in the owner list.
3. Delete all the features without the usefulness marks from the history tree.

5. Experimental Results

We illustrate the effectiveness of our automatic synchronization method through three experiments. The original feature information are obtained from commercial system UG NX [4], and the synchronization process is implemented based on the cellular topology husk of geometric modeling toolkit ACIS.

The first example is shown in Fig. 1. When user applies a direct modeling operation on one part of a feature face in feature F_3, our method first tries to synchronize through modification of feature parameters. From Fig. 1(c), we can see that the extrusion lengths of feature F_3 and feature F_4 are increased. Then consistency checking is performed, which detects two conflict regions. One of the conflict regions is caused by intersection of feature F_3 and feature F_5, which can be eliminated by swapping F_3 and F_5. Finally, a new feature F_6 is added to guarantee the success of synchronization.

Fig. 13 shows the second example. The original feature model and the direct editing, which drags four co-planar

Table 1
Statistics of synchronization operations on the feature model, in which #Sync1 means the number of feature definition modifications, #Sync2 means the number of feature order adjustments and #Sync3 means the number of feature additions or deletions.

Example	#Sync1	#Sync2	#Sync3	#Total
no.1	2	1	1	4
no.2	4	0	1	5
no.3	3	0	2	5

faces at the same time, are shown in Fig. 13(a). The corresponding synchronization process is shown in Fig. 13(b). We can see that, during the synchronization, four features' definitions are modified and a new feature is added into the history tree. No features are swapped in the process. Analogously, Fig. 14 shows another synchronization example.

From these experiments, we demonstrate that our algorithm can effectively synchronize the feature model with the direct editing. More importantly, one direct modeling operation is usually converted into multiple equivalent modifications on the feature model, which can be seen from statistics in Table 1. That means, without our automatic synchronization method, users always need to manually modify many features' definitions and orders in the history tree to achieve the same editing effect as a single direct modeling operation. Even worse, sometimes they need to add or delete features. This indicates that our method indeed brings great convenience to the users who wants to edit feature models "directly".

6. Conclusion and Future Work

In this paper, we proposed a novel synchronization method to automatically keep the feature model consistent with the direct editing result. By successively executing modifications of feature definitions, adjustments of feature orders and additions/deletions of features, the method not only ensures a successful synchronization solution, but also preserves the design semantics of original model as much as possible. During the process, we leverage the cellular model to efficiently and effectively check geometric consistencies and manipulate features. The experimental results show that the synchronization algorithm works well for models with extrusion features. In all, we believe our method opens the possibility towards an elegant fusion between feature-based modeling and direct modeling technologies.

In the future, we plan to explore the support of geometric constraints during synchronization, which largely enriches the design semantics of feature model and hence requires more complex algorithms. In addition, we are also interested in synchronization of non-extrusion features and even assembly model.

References

[1] J. J. Shah, *Parametric and feature-based CAD/CAM: concepts, techniques, and applications*. John Wiley & Sons, 1995.

[2] R. Bidarra and W. F. Bronsvoort, "Semantic feature modelling," *Computer-Aided Design*, vol. 32, no. 3, pp. 201–225, 2000.

[3] "Spaceclaim corporation." [Online]. Available: http://www.spaceclaim.com/en/

[4] "Siemens corporation." [Online]. Available: http://www.plm.automation.siemens.com/en_us/products/nx/

[5] "Ptc creo elements/direct modeling." [Online]. Available: http://www.ptc.com/product/creo-elements-direct/modeling

[6] R. Bidarra, K. J. de Kraker, and W. F. Bronsvoort, "Representation and management of feature information in a cellular model," *Computer-Aided Design*, vol. 30, no. 4, pp. 301–313, 1998.

[7] A. A. Requicha and H. B. Voelcker, "Boolean operations in solid modeling: Boundary evaluation and merging algorighms," *Proceedings of the IEEE*, vol. 73, no. 1, pp. 30–44, 1985.

[8] V. Shapiro, "Representations of semi-algebraic sets in finite algebras generated by space decompositions," Ph.D. dissertation, Cornell Programmable Automation, Cornell Univ., Ithaca, 1991.

[9] V. Shapiro and D. L. Vossler, "Construction and optimization of csg representations," *Computer-Aided Design*, vol. 231, no. 1, pp. 4–20, 1991.

[10] V. Shapiro and D. L, "Separation for boundary to csg conversion," *ACM Transactions on Graphics (TOG)*, vol. 12, no. 1, pp. 35–55, 1993.

[11] J. Kim, M. J. Pratt, R. G. Lyer, and R. D. Sriram, "Standardized data exchange of cad models with design intent," *Computer-Aided Design*, vol. 40, no. 7, pp. 760–777, 2008.

[12] G. H. Choi, D. Mun, and S. Han, "Exchange of cad part models based on the macro-parametric approach," *International Journal of CAD/CAM*, vol. 2, no. 1, 2002.

[13] A. Rappoport, "An architecture for universal cad data exchange," *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pp. 266–269, 2003.

[14] M. Li, C. C. Wang, and S. M. Gao, "Real-time collaborative design with heterogeneous cad systems based on neutral modeling commands," *Journal of Computing and Information Science in Engineering*, vol. 7, no. 2, pp. 113–125, 2007.

[15] W. F. Bronsvoort and A. Noort, "Multiple-view feature modelling for integral product development," *Computer-Aided Design*, vol. 36, no. 10, pp. 929–946, 2004.

[16] de Kraker, K. Jan, M. Dohmen, and W. F. Bronsvoort, "Maintaining multiple views in feature modeling," *Proceedings of the fourth ACM symposium on Solid modeling and applications. ACM*, pp. 123–130, 1997.

[17] M. R. Cutkosky, J. M. Tenenbaum, and D. Muller, "Features in process based design," *ASME computers in Engineering (CIE) conference*, pp. 557–562, 1988.

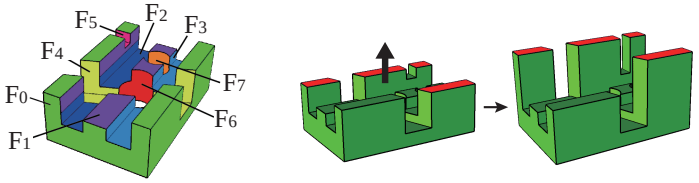
[18] J. C. H. Chung, R. L. Cook, and D. Patel, "Feature-based geometry construction for geometric reasoning," *Computers in Engineering (Eds. VA Tipnis, EM Patton) San Francisco: ASME*, pp. 497–504, 1988.

[19] J. J. Shah and M. T. Rogers, "Expert form feature modelling shell," *Computer-Aided Design*, vol. 20, no. 9, pp. 515–524, 1988.

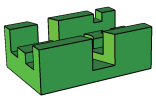
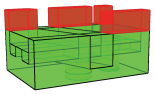
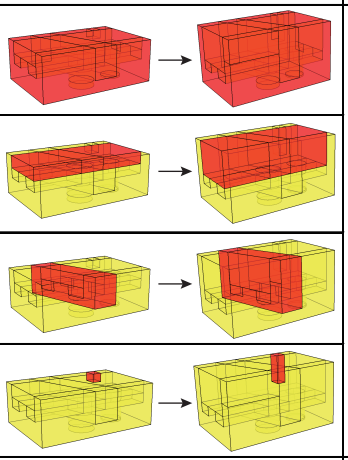
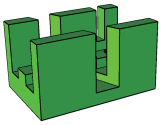
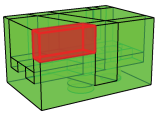
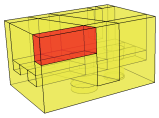
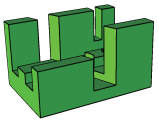
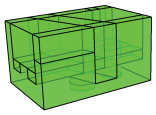
[20] S. M. Gao and J. J. Shah, "Automatic recognition of interacting machining features based on minimal condition subgraph," *Computer-Aided Design*, vol. 30, no. 9, pp. 727–739, 1998.

[21] J. Han, M. Pratt, and W. C. Regli, "Manufacturing feature recognition from solid models: a status report," *Robotics and Automation, IEEE Transactions on*, vol. 16, no. 6, pp. 782–769, 2000.

[22] B. B., N. N., and M. Z., "A review of automated feature recognition with rule-based pattern recognition," *Computers in Industry*, vol. 59, no. 4, pp. 321–337, 2008.



(a)

	Processes	Feature Sequence	B-rep Model	Conflict Cells(red)
Original Model	No	F0-F1-F2-F3-F4-F5-F6-F7		
Modification of Feature Parameters		**F0'-F1'**-F2-F3-**F4'-F5'**-F6-F7		
Addition and Deletion of Features		F0'-F1'-F2-F3-F4'-F5'-F6-F7-**F8**		

(b)

Fig. 13. Example2: (a) the original feature model and the direct modeling operation; (b) the synchronization process.

[23] "Direct modeling - who and why needs it? a review of competitive technologies." [Online]. Available: http://isicad.net/articles.php?article_num=14805

[24] "Parametric or direct modeling: why you may need both." [Online]. Available: http://www.econocap.dk/assets/files/creoparametric/6275_Parametric_Direct_CAD_WP.pdf

[25] "Spaceclaim in manufacturing." [Online]. Available: http://partnerfiles.spaceclaim.com/Collateral/White%20papers/Develop3D_SpaceClaim_in_MFG.pdf

[26] "Synchronous technology." [Online]. Available: http://www.plm.automation.siemens.com/legacy/docs/Synchronous_Technology_CPDA_WhitePaper.pdf

[27] G. A. Crocker and W. F. Reinke, "An editable nonmanifold boundary representation," *Computer Graphics and Applications, IEEE*, vol. 11, no. 2, pp. 39–51, 1991.

[28] H. Masuda, "Topological operators and boolean operations for complex-based nonmanifold geometric models," *Computer-Aided Design*, vol. 25, no. 2, pp. 119–129, 1993.

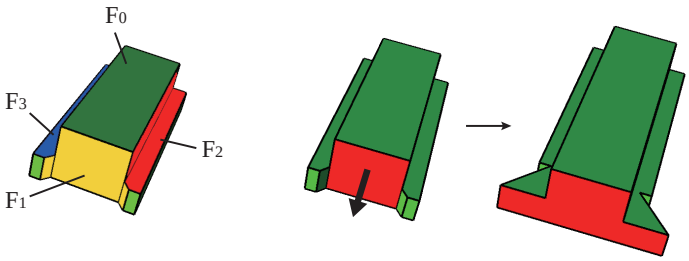
[29] R. Bidarra, J. Madeira, W. J. Neels, and B. W. F., "Efficiency of boundary evaluation for a cellular model," *Computer-Aided Design*, vol. 37, no. 12, pp. 1266–1284, 2005.

[30] W. F. Bronsvoort, R. Bidarra, and N. Alex, "Feature model visualization," *Computer Graphics Forum*, vol. 21, no. 4, pp. 661–673, 2002.

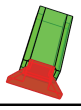
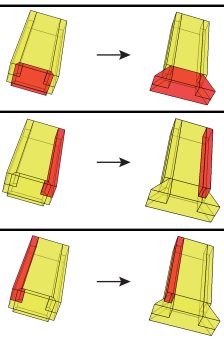
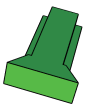
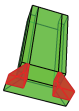
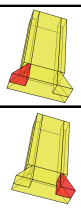
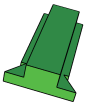
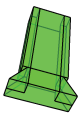
[31] J. Y. Lee, J. H. Lee, H. Kim, and H. S. Kim, "A cellular topology-based approach to generating progressive solid models from feature-centric models," *Computer-Aided Design*, vol. 36, no. 3, pp. 217–229, 2004.

[32] S. H. Lee, "Feature-based multiresolution modeling of solids," *ACM Transactions on Graphics (TOG)*, vol. 24, no. 4, p. 2005.

[33] "Cellular topology data structure." [Online]. Available: http://doc.spatial.com/index.php/Cellular_Topology_Data_Structure



(a)

	Processes	Feature Sequence	B-rep Model	Conflict Cells(red)
Original Model	No	F0-F1-F2-F3		
Modification of Feature Parameters		F0+**F1'-F2'-F3'**		
Addition and Deletion of Features		F0'+F1'-F2'-F3'-**F4-F5**		

(b)

Fig. 14. Example3: (a) the original feature model and the direct modeling operation; (b) the synchronization process.