An Efficient Algorithm for Recognizing and Suppressing Blend Features

Xiufen Cui[1], Shuming Gao[2] and Guangping Zhou[3]

[1]Zhejiang University, xfcui@cad.zju.edu.cn
[2]Zhejiang University, smgao@cad.zju.edu.cn
[3]Zhejiang University, gpzhou@cad.zju.edu.cn

ABSTRACT

This paper presents an algorithm for efficiently recognizing and suppressing blend features. The algorithm first recognizes all of blend faces from the Boundary Representation of a part; then distinguishes them as edge-blend feature, vertex-blend feature and mixed-blend region; furthermore, divides the mixed-blend region into pure edge-blend feature and pure vertex-blend feature; lastly each recognized blend feature is suppressed as a whole by means of Euler Operators. The novelty of the presented algorithm lies in that all the entities of a recognized blend feature are suppressed in a global way, by which the efficiency of suppression is improved. In addition, the algorithm can deal with all types of blend faces generated by various methods. In the end, some test results are given.

Keywords: Blend feature, feature recognition, feature suppression, CAD/CAM/CAE

1. INTRODUCTION

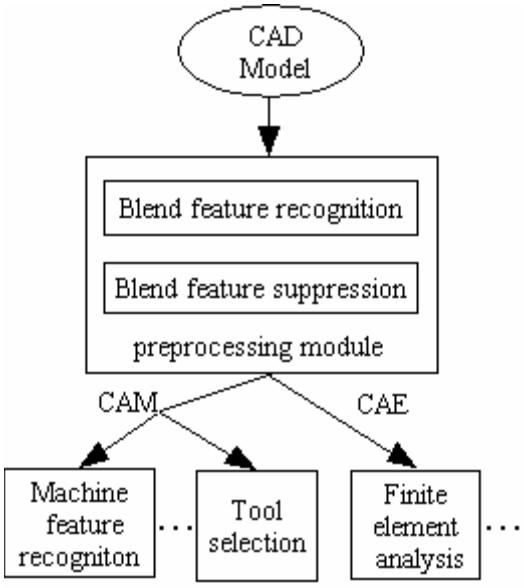


Fig.1. Applications of blend feature recognition and suppression

In mechanical design, blending is a common operation used to improve the strength, the aesthetics, and ensure the manufacturability of a part. In blending operation, certain smooth faces are added into the part's B-rep to smoothen sharp edges and vertices, which are termed as blend faces. The task of blend feature recognition is to recognize all blend faces from a part's B-rep and divide them into blend features. And the goal of blend feature suppression is to restore the original B-rep of the part by removing all the recognized blend features.

Some application domains of blend feature recognition and suppression (BFRS) are shown in Fig. 1. It is well recognized that BFRS is of significance for the integration of CAD/CAM[1] and the integration of CAD/CAE.

Several approaches to BFRS have been developed in the past decades[2~7]. Among them the most representative work is the algorithms presented by Sashikumar and Sohoni [2~4]. Their algorithms deal with the blend faces generated by the rolling-ball based blending technique. The algorithms first detect all the blends; Then generate the blend chains and deduce the chains' sequence in which they were created based on local heuristics around them; Lastly, the blend chains are suppressed one by one in the reverse order of the sequence.

In 2002, H. Zhu and C. H. Menq presented a method of simplifying B-Rep models by automated suppression of fillet/rounds[5]. Their method firstly recognizes three types of faces, i.e. toroidal faces, cylindrical faces and spherical faces as fillets or rounds in terms of certain rules; then connects them to form ring-type chains and disc-type chains; lastly, suppresses all the chains by an incremental knitting process. The method only deals with constant-radius fillets and rounds.

An approach to feature simplification for freeform surface models is presented by Joshi and Dutta in 2003[6]. Their approach recognizes and suppresses two types of basic features, holes and fillets in freeform surface models, as well as the combination of the basic features. Among them, the fillets are recognized by curvature calculations similar to the blend recognition algorithm of Sashikumar and Sohoni[2~4], and suppressed by skinning surfaces between the spring edge and the spine curve.

In the paper, a new algorithm for recognizing and suppressing blend features is presented. The algorithm is intended to improve the efficiency of blend feature recognition and suppression and enable the handling of all types of blend faces generated by various CAD systems.

The remainder of the paper is organized as follows: In Section 2, we introduce some basic concepts about the blend. In Section 3, we give the blend feature recognition algorithm. In Section 4, we discuss the handling of mixed blend region. We use Section 5 to present the blend feature suppression algorithm. In Section 6, we discuss the implementation of the presented algorithm. Finally we give the conclusion in Section 7.

2. BASIC CONCEPTS

Before describing the algorithm, we first introduce some basic concepts.

(1) Blend face: the face generated by blending operation, such as f_3 in Fig. 2. The surface of a blend face may be a NURBS surface or a complex quadric surface[7,8].

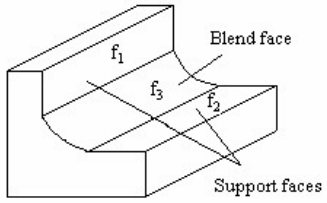


Fig.2. Blend face and its support faces

(2) Support face: the face that is not a blend face but adjacent with a blend face at its smooth edge. Taking blend face f_3 as an example, f_1 and f_2 are two support faces of it.

(3) Edge blend face: the blend face that replaces a sharp edge between two faces.

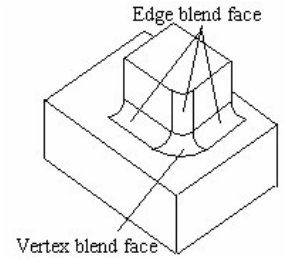


Fig.3. Vertex blend face and edge blend face

(4) Vertex blend face: the blend face that is not edge blend face, it connects smoothly several edge blend faces that meet at a common vertex (see Fig. 3).

(5) Cliff blend face: the blend face that takes place between a face and an edge. The corresponding edge that supports the cliff blend face is termed cliff edge.

(6) Blend on blend faces: the two blend faces that interact with each other. The blend faces must be generated by different blending operations, and the interacting edge is termed blend on blend edge (See Fig. 4).

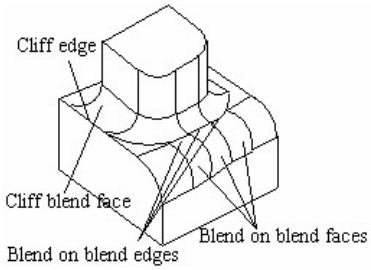


Fig.4. Cliff blend face and blend on blend faces

(7) Blend feature: a set of connected blend faces corresponding to a vertex or an edge of the suppressed model.

(8) Edge blend feature: the blend feature corresponding to an edge of the suppressed model, such as the grayer face in Fig. 5 that forms an edge blend feature.

(9) Vertex blend feature: the blend feature corresponding to a vertex of the suppressed model, such as two blue faces in Fig. 5 that form a vertex blend feature.

(10) Mixed blend region: a set of connected blend faces that cannot be determined as a vertex blend feature or an edge blend feature currently.

(11) Blend feature's internal edge: the smooth edge whose two adjacent faces are both blend faces of the

blend feature. In Fig. 5, e_9 is the internal edge of the vertex blend feature.

(12) Blend feature's boundary edges: the smooth edge that is adjacent with only one blend face of the blend feature. In Fig. 5, e_1, e_2, e_3 and e_4 are boundary edges of the edge blend

feature (the grayer face). And e_4, e_5, e_6, e_7, and e_8 are boundary edges of the vertex blend feature (the blue faces) in Fig. 5.

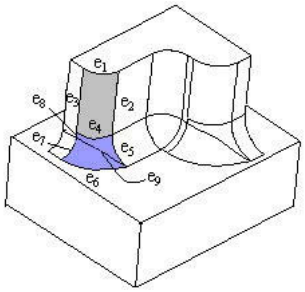


Fig. 5 Edge blend feature and vertex blend feature

Among all the concepts defined above, blend feature together with vertex blend feature, edge blend feature and mixed blend region plays a key role in this work. We introduce these concepts and organize all the recognized blend faces as these high level features with more semantics to make the BFRS algorithm efficient and simple.

3. BLEND FEATURE RECOGNITION ALGORITHM

In this section, we describe the blend feature recognition algorithm. Fig. 6 shows the flowchart of the algorithm.

3.1 Blend Face Recognition

(1) Blend Face Detection

The algorithm first recognizes all blend faces from the model. Specifically, we check every curved face to see whether it satisfies the following conditions:

1) The face itself is smooth without any sharp edge and sharp vertex inside it.
2) The face has at least one smooth edge.
3) For any two faces adjacent with the face by smooth edges, they aren't parallel.
4) Area ratio between the blend face and any of its adjacent planar face is less than a given threshold.

All the curved faces satisfying above conditions are recognized as blend faces.

(2) Blend Face Classification

For each blend face, its support faces are determined by finding out all its adjacent faces that are not blend faces but share smooth edges with the blend face. According to the number of its support faces, each blend face is classified as one of the following two classes:

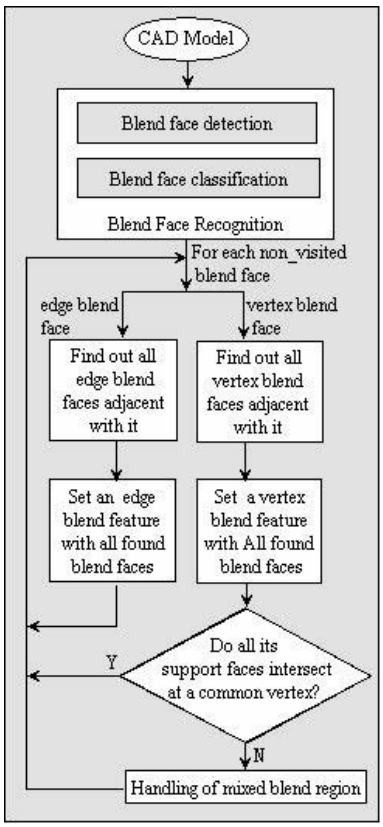


Fig. 6. The flow chart of the algorithm

1) Vertex blend face with only one or even no support face.
2) Edge blend face with two support faces.

For the blend face that has more than two support faces, the algorithm further detects if these support faces intersect at a common vertex. If they intersect at a common vertex, the blend face is classified as a vertex blend face; otherwise it is classified as an edge blend face.

(3) Blend Face Structure

The recognized blend faces are represented by the following structure:

```
Class BLEND_FACE {
FACE* blendfac;
      //the blend face's corresponding face in the model;
int type;
      //the blend face's type, 0: edge blend face,
      // 1: vertex blend face;
vector<FACE*> supportfac;
      //the vector depositing all the support faces of the
      //blend face;
bool visited;
      //a mark used to indicate whether the blend face is
      //visited or not
```

}
vector<BLEND_FACE> BLENDFS;
//the vector depositing all the blend faces in the model.

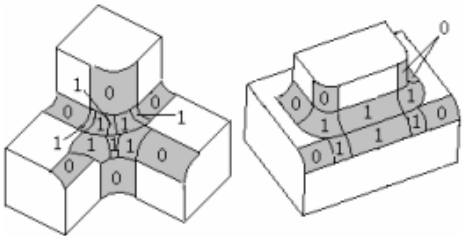


Fig.7 .The recognized and classified blend faces

Two examples are used to illustrate the results of the blend face recognition as shown in Fig. 7 where all the recognized blend faces are highlighted with "0" indicating edge blend faces and "1" indicating vertex blend faces.

3.2 Blend Feature Recognition

(1) Blend Feature Recognition Algorithm

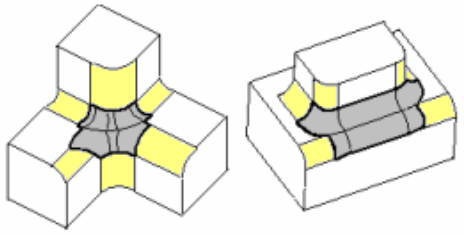
The input of the blend feature recognition algorithm is all blend faces of the model and their attributes set up in 3.1, and the output is all the blend features in the model. The specific algorithm consists of following steps:

Step 1: Take a non-visited blend face from all the input blend faces as a seed blend face, and set the visited item of the blend face to TRUE. If the seed blend face is a vertex blend face, then an initial vertex blend feature is generated; otherwise an initial edge blend feature is generated.

Step 2: Find out all those blend faces that have the same type as the seed blend face and are adjacent with it by smooth edges. These smooth edges are set as the blend feature's internal edges. Then take each found blend face as a new seed blend face and repeat the above process iteratively until no new seed blend face can be found. At this time, a complete blend feature is obtained.

Step 3: For each initial vertex blend feature, we further check whether its all support faces intersect at a common vertex. If not, it indicates that the blend feature cannot correspond to a vertex after suppression, and it is not a real vertex blend feature but a mixed blend region. For this case, the algorithm handles the mixed blend regions with the method given in the next section.

Step 4: After the generation of a blend feature is complete, retrieve all blend faces to see whether there still exists a non-visited blend face. If so, the above step 1-3 is repeated to generate the remainder blend features. The algorithm ends up when all the blend faces of the model are marked as visited.



(a) (b)

Fig.8. The recognized blend features

Fig. 8 illustrates the recognized blend features of the two models shown in Fig. 7. The model shown in Fig. 8(a) has one and only one vertex blend feature (the gray region) that is composed of its all vertex blend faces. In addition, it has six edge blend features (the yellow regions) each of which includes only one edge blend face. For the model shown in Figure 8(b), it has a mixed blend region composed of six blend faces that are all blend on blend faces (the gray region). Besides, it contains six edge blend features (the yellow regions).

(2) Blend Feature Structure

To effectively support blend feature suppression, after a blend feature is recognized, it is represented by the following structure.

```
class BLEND_FEATURE {
int type;
        //the blend feature's type, 0:edge blend feature,
        //1:vertex blend feature, 2:mixed blend region;
vector<FACE *>  blendf;
        //all the blend faces involved in the blend feature
vector<FACE *>  supportf;
        //all the different support faces of the blend faces
        // involved in the blend feature
vector<EDGE *>  interedge;
        //all the internal edges of the blend feature
VPOINT suppression_vertex;
        //the suppression vertex of the vertex blend
        //feature
}
class VPOINT{
 APPOINT coors;
        //the suppression vertex's coordinates of the
        // vertex blend feature
vector<EDGE *>  coes;
        //all the boundary edges of a vertex blend feature
}
```

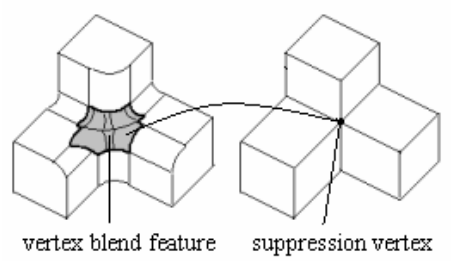


Fig.9. The suppression vertex of the vertex blend feature

In the above structure, the suppression vertex of a vertex blend feature refers to the vertex in the suppressed model corresponding to the vertex blend feature. In Fig. 9, the suppression vertex of the vertex blend feature in the left part is the highlighted vertex of the right part.

4. HANDLING OF MIXED BLEND REGIONS

As discussed above, the mixed blend region results from cliff blend faces and blend on blend faces. Because such kinds of blend faces make the detection of their support faces much more difficult, resulting in the trouble for classifying blend features, therefore, the first step of our algorithm for handling mixed blend regions is to detect cliff blend faces and blend on blend faces involved in each mixed blend region.

4.1 Detection of Cliff Blend Faces and Blend On Blend Faces

(1) Cliff Blend Face Detection

According to the cliff blend face definition that a blend face is a cliff blend face if it has a cliff edge, we check whether a blend face is a cliff blend face by checking if it has a cliff edge as follows: For each edge of the blend face, check if it is straight and non-smooth, if so, it is a cliff edge since the normal non-smooth edges of a blend face must be curved.

(2) Blend on Blend Face Detection

Similar to the cliff blend face detection, we convert the blend on blend face detection to the detection of blend on blend edges. According to the definition of blend on blend edge that it is the interacting edge between two blend faces (see 2(6)), obviously a blend on blend edge must be an internal edge of a mixed blend region. Also it is observed that a blend on blend edge should be very close to the common support face of the two interacting blend faces it belongs to. This is because if its two blend faces didn't intersect, the corresponding edge of the blend on blend edge in its two blend faces would be on the common support face.

Based on the above observations, we detect the blend on blend edges of a mixed blend region as follows: Given a threshold δ, for every internal edge of the mixed blend region, find out the support face closest to it by calculating the distances between its two vertices and every support face of the mixed blend region. If the distance between the edge and its closest support face is less than δ, then this edge is determined as a blend on blend edge of the mixed blend region, and the closest support face is taken as the common support face of the blend on blend faces adjacent with the blend on blend edge.

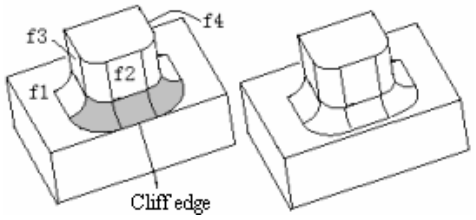
A problem with above method is how to set a reasonable threshold δ. In this work, we set the threshold δ to be the half of the least curvature radius of all the blend faces involved in the mixed blend region. Such threshold is reasonable for the mixed blend region because for any of its normal internal edges that is not a blend on blend edge, the distance between the edge and any support face of the mixed blend region is equal or almost equal to the curvature radius of the blend face adjacent with it. Experiments have also verified the validity of such defined threshold δ.

Using the above detection algorithms, one cliff edge and the corresponding cliff blend face are detected from the mixed blend region in Fig. 10(a), and three blend on blend edges and six blend on blend faces are detected from the mixed blend region in Fig.10 (b).

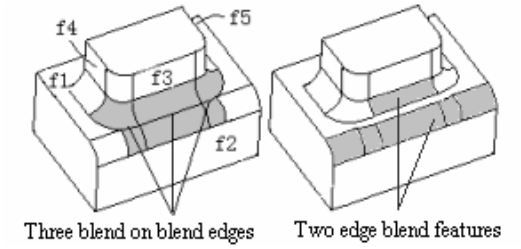
4.2 Mixed Blend Region Handling

After all the cliff blend faces, blend on blend faces as well as their corresponding cliff edges and blend on blend edges involved in the mixed blend region are determined, they are dealt with as follows:

1) Topology separation. Topology separation is performed on the mixed blend region along the cliff edges and blend on blend edges using the Euler Operation:



(a) Topology separation of a cliff blend face



(b) Topology separation of blend on blend faces

Fig.10. Illustration of topology separation of cliff blend face and blend in blend face

separate_topology(). This Euler Operation separates the adjacent faces along one or more edges by creating new edges, new loops, and re-organizing the edges of original loops.

2) Re-classify cliff blend faces and blend on blend faces. After the topology separation, the cliff blend faces and blend on blend faces become normal blend faces, so their support faces and their types are re-determined using the method described in 3.1(2).

3) Blend feature re-recognition. Set the visited values of all blend faces of the mixed blend region to FALSE, and re-recognize the blend features involved in the mixed blend region using the blend feature recognition algorithm presented in 3.2(1).

After the above handling, any mixed blend region is decomposed into vertex blend features and /or edge blend features. As two examples shown in Fig.10, the cliff blend face in Fig. 10(a) becomes a edge blend face since it has two support faces, and the mixed blend region is partitioned into two vertex blend features and one edge blend feature; four blend on blend faces in Fig. 10(b) become edge blend faces since all of them have two support faces, and the mixed blend region is decomposed two vertex blend features and two edge blend features.

5 BLEND FEATURE SUPPRESSION ALGORITHM

Blend feature suppression is to delete all blend features of the model to obtain a simplified model without blend faces. It involves the modification on both the topology and geometry of the model. In this work, Euler Operations are adopted to make topological modification, which modify topology locally and guarantee the modified model valid.

Specially, Three Euler Operations are used. One is DEV () that is employed to delete an edge and make the two vertices of the edge merged. Another one is DEF () that is utilized to collapse a face and delete one edge on it. The third is DFS () that deletes several faces from the model. Compared with topological modification, geometry modification is relatively simple. By calculating the intersection line between two related support faces and the intersection point among three or more related support faces, we obtain the required new geometry entities. Fig. 11 shows the flowchart of the blend feature suppression algorithm (topology part), the right grids denote the used Euler Operations in the corresponding step. The algorithm is divided into two stages:

Stage 1: Suppression of all the vertex blend features

For each vertex blend feature VBF

(1.1) Delete all the blend faces of VBF using DEF (). And determine the geometry of the VBF's corresponding vertex by calculating the intersection point among all the support faces of the VBF.

(1.2) Collapse all the common edges between the blend faces and support faces of VBF using the DEV(). The deleted edge's remaining vertex is set as the suppression vertex of VBF.

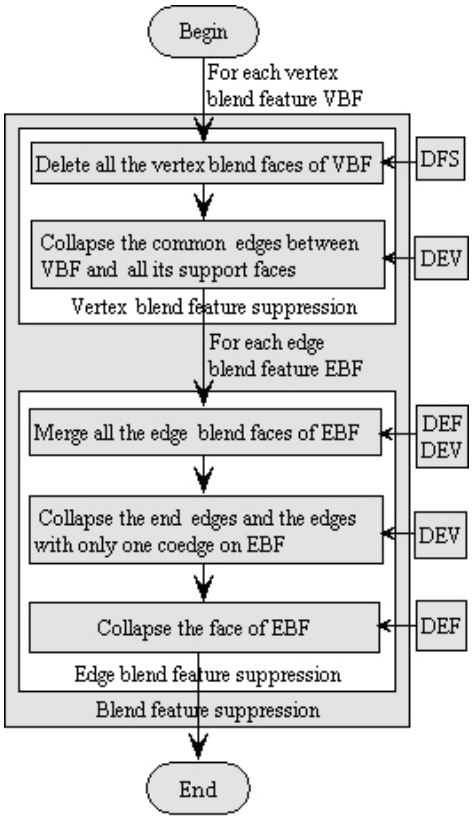


Fig.11 The flowchart of blend feature suppression algorithm

Stage 2: Suppression of all the edge blend features

For each edge blend feature EBF

(2.1) Merge all the blend faces of EBF into a single face using DEF () and DEV () if EBF contains more than one blend faces. During this process, the edges that have the same adjacent faces are merged into a single edge using the DEV ().

(2.2) Collapse the end edges of EBF and the edges with only one adjacent face, i.e. the blend face of EBF using the DEV (). For the end edge, its remaining vertex's geometry after collapsed is set as the intersection point among EBF's support faces and the face that is adjacent with the edge but not the blend face. For the other collapsed edge, since it must be on the boundary of a VBF, its remaining vertex's geometry is set as the geometry of the corresponding vertex of the VBF.

(2.3) Collapse the blend face of EBF into a single edge using DEF (), set as the intersection line between the EBF's support faces.

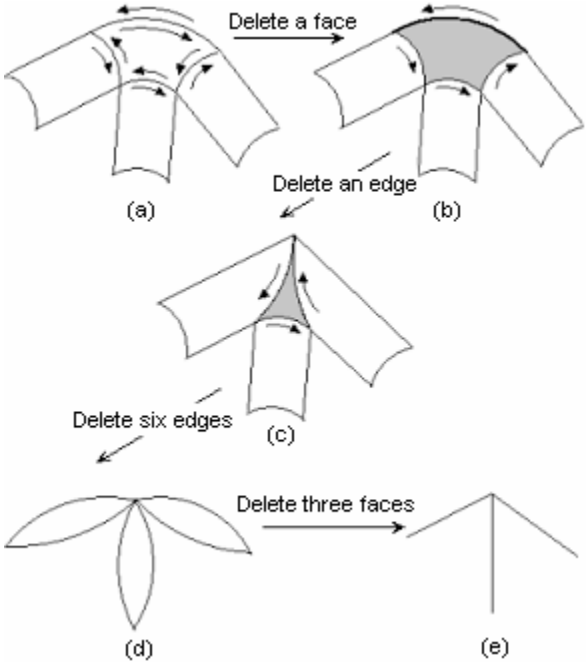


Fig.12. Process of suppressing a vertex blend feature and three edge blend features.

Fig. 12 shows the process of topologically suppressing a vertex blend feature and three edge blend features. The shaded region showed in Fig. 12(b) is a deleted the blend face; The Fig. 12(c) shows the result after deleting the edges between the blend face of the vertex blend feature and its support faces. Fig. 12(e) shows the result of suppressing three remaining edge blend features by first deleting six edges (Fig. 12(d)), then deleting three faces (Fig. 12(e)).

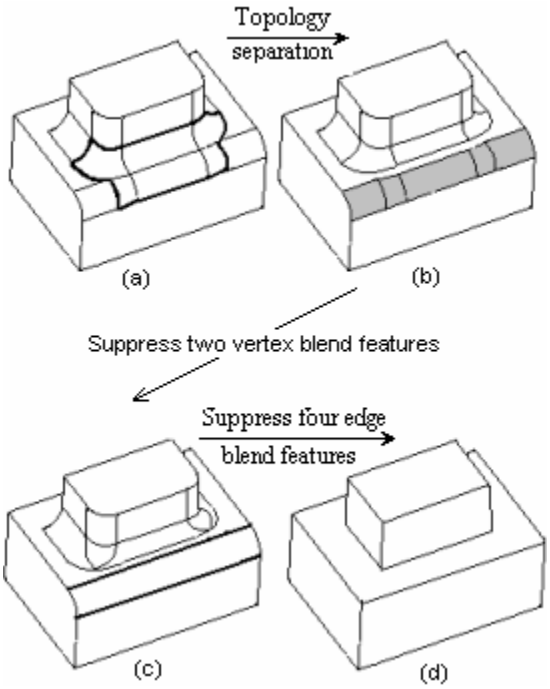


Fig.13. Blend feature suppression of a model with blend on blend faces.

In Fig. 13, the blend feature suppression of a complete model is shown. Specifically, Fig. 13(a) shows the detected mixed blend region composed of six blend faces; Fig. 13(b) shows the result after topological separation of the mixed blend region; Fig. 13(c) shows the result after the two vertex blend features are suppressed; Figure 13(d) shows the final result.

6. IMPLEMENTATION

The proposed algorithm has been implemented using ACIS 6.0 solid modeler, C++ language, running on win2000 operation system, and tested by some examples. Fig. 14 shows six tested examples, each of which consists of two parts: a part with recognized blend features (shaded areas) and the corresponding suppressed part. The execution time for recognizing and suppressing blend features of these models are respectively: (a) --- 0.063s, (b)---0.078s, (c) --- 0.063s, (d) --- 0.062s, (e) --- 0.048s, (f) --- 0.079s.

7. CONCLUSIONS

In this paper, an efficient algorithm of blend feature recognition and suppression is presented. The algorithm gives the concept of blend feature and the approach to recognize blend feature, presents a new idea about blend

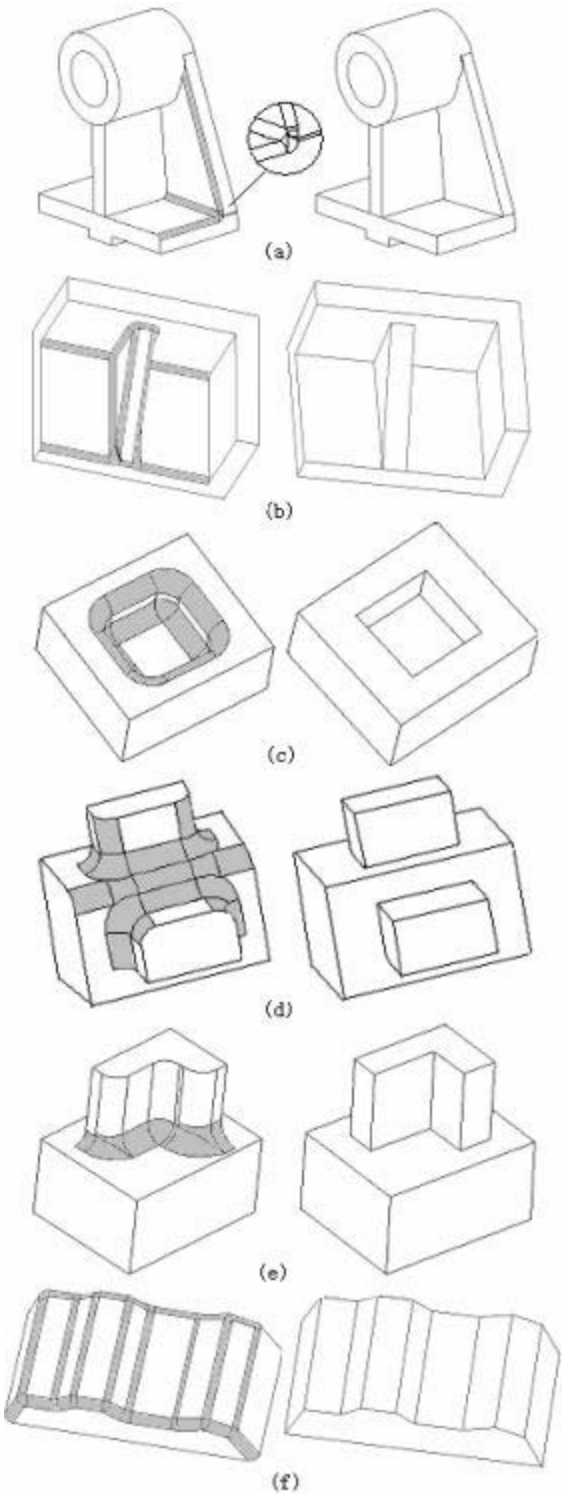


Fig.14. Tested examples of blend feature recognition and suppression.

suppression that suppresses the blend feature's all blend faces as a whole. Our algorithm has several merits, it is of efficiency, and works for all types of blend faces generated by various blending methods.

The major contributions of the work lie in:

1) The concept of blend feature with higher-level semantics is proposed;
2) Blend features are suppressed as a whole, making the blend suppression efficient and simple.

Future work will be focused on following aspects:

Enable the algorithm to deal with blend features between

1) sculpture surfaces.
2) Facilitate the algorithm to recognize and suppress the blend feature interacted by non-blend features.
3) Enable the algorithm to handle the blend feature with its support face missed.

8. REFERENCES

[1] Gao S. and Shah J. J., Automatic recognition of interacting machining features based on minimal condition subgraph. Computer Aided Design, 1998, Vol. 30(9), pp. 727-739.

[2] Venkataraman S, Sohoni M, Blend Recognition Algorithm and Applications. Proceedings of the sixth ACM Symposium on Solid Modeling and Applications, D.C. Anderson and K. Lee, eds., ACM press, Ann Arbor, June 2001, Michigan, pp.99-108.

[3] Venkataraman S, Sohoni M, Rajadhyaksha R, Removal of Blends from Boundary Representation Models. ACM Symposium on Solid Modeling and Applications, ACM press, June 2002, Germany, pp. 83-94.

[4] Venkataraman S, Sohoni M, Reconstruction of Feature Volumes and Feature suppression. ACM Symposium on Solid Modeling and Applications, ACM press, June 2002, Germany, pp: 60-71.

[5] H. Zhu and C. H. Menq, B-Rep model simplification by automatic fillet/round suppressing for efficient automatic feature recognition. Computer Aided Design, 2002, Vol. 34, pp. 109-123.

[6] N.Joshi and D.Dutta, Feature simplification techniques for freeform surface models, Journal of Computing and Information Science in Engineering, September 2003, Vol 3, Num 3, pp. 177-186.

[7] T.lim, J.R.Corney, D.E.R Clark, A laminae approach to constructing geometric feature volumes, Solid modeling, 2001,pp. 183-192.

[8] Vida J, Martin R.R. Varady T., A survey of blending methods using parametric surfaces. Computer-Aided Design, Vol.26, No 5.February 1994, pp. 341-364.

[9] I C Braid, Non-local blend of boundary models. Computer-Aided Design, Vol. 29, No 2, 1997, pp. 89-100.