


# A slice based approach to recognize and extract free-form volumetric features in a CAD mesh model

Nepal Adhikary and B. Gurumoorthy 

Indian Institute of Science, India

## ABSTRACT

This paper describes an algorithm to recognize and extract volumetric features by directly clustering the triangles constituting a feature in a CAD mesh model. The algorithm involves two steps – isolating features in 2D slices followed by a 3D traversal to cluster all the triangles in the feature. The main advantage of this approach is that processing of the three dimensional mesh occurs only for the triangles associated with its signature identified on the 2D slice. Results from an implementation tested on various mesh models are presented.

## KEYWORDS

Volumetric feature recognition; mesh model; slicing

## 1. Introduction

Volumetric features in a model are regions of interest created by adding or removing volumes to or from a solid. Recognizing and extracting volumetric features from a mesh model is important in applications such as mesh simplification [6] and direct editing of mesh models at higher levels of abstraction. Mesh models are typically obtained from one of two broad methods – scanning a physical prototype of an object or by tessellating a digital model (usually constructed in a CAD system). The latter type of model is referred to as CAD mesh model and is, the focus of this paper

Mesh simplification is also required in the use of mesh models in computer graphics and animation. In these domains, simplification is done by segmenting the mesh model. Segmentation partitions a mesh model into meaningful connected regions based on geometric properties, typically the nature of the underlying surface [4]. As a volumetric feature is often composed of regions with many different underlying surfaces, the approaches developed for segmentation in the domain of computer graphics and animation are not of much use [4]. Segmentation of a CAD mesh model has been a preferred approach for extracting surface features [17]. Mesh segmentation procedures available in the literature can be divided into two main categories, surface based and part based mesh segmentation [4, 2].

Surface based approach depends on geometric properties like uniform distance from a fitted surface or uniform

curvature deviation. Triangles are grouped together into regions based on the point distance or curvature properties. A collection of these regions can represent some meaningful features. This is why surface based approach can be considered as pre-processing step before detecting features from a mesh model [4]. There are two types of surface segmentation. The first type works by grouping triangles that belong to specific surfaces [4]. Criterion for grouping is distance of a triangle from a particular surface. The second type of segmentation is based on the nature of continuity of differential properties (typically curvature) [3].

In the part-based approaches [10], the mesh model is subdivided into meaningful regions. The mesh is processed based on semantics of local regions in the mesh. To extract meaningful regions in part based scheme, different criteria can be used. Minima rule introduced by Hoffman and Richards [8] is the most common criterion used by researchers. According to the minima rule, segmentation should be done along ‘negative minima of the principal curvatures on surfaces’ as this is based on how humans perceive boundaries.

As the segmentation of mesh model does not reveal any information about the existence of volumetric feature, it is necessary to develop direct feature extraction algorithm for mesh model without segmentation. One approach possible is to reverse engineer a smooth boundary representation of the object [5] and then extract volumetric features from the boundary model. This approach

however introduces additional (and still evolving) steps in both the extraction process and also in further manipulation of the features. For instance, if the motivation for feature extraction is feature suppression then this approach would require remeshing after the extracted feature (in the boundary model) has been removed.

Available commercial software, such as RapidForm and GeoMagic, convert mesh representation obtained from point cloud into CAD model by surface fitting (typically using NURBS). Once the CAD representation of a discrete model is available, existing feature recognition algorithms / software's can be used to recognize the features. For any application in this approach, that wants to modify features in the mesh model would have to modify the CAD model and remesh the CAD model. This cycle has to be repeated till the requirement is met. It must be noted that the process of constructing a smooth representation from a discrete mesh based representation is not an exact process and is usually error prone.

The approach proposed in this paper directly extracts volumetric features without segmenting the mesh model. These features are associated with either volume addition or volume removal [6, 15] to or from a model. The algorithm presented here automatically detects features from a mesh model by processing triangles in the features zones after eliminating the non-feature triangles during a pre-processing step. The main objective of this paper is to eliminate the dependency of CAD model while processing the mesh model. Proposed algorithm does not depend on associated CAD model and extracts features directly from mesh model. It does so without processing all mesh elements. Converting mesh model into its CAD representation is time consuming and also an unnecessary overhead operation if users intend to continue working with mesh model and not with the CAD model. This paper is organized as follows: Section 2 reviews related literature and Section 3 defines related terminologies. Details of the algorithm are presented in Section 4. Section 5 provides illustrative examples and results on typical mesh models. Discussion based on results is provided in Section 6. Section 7 provides conclusion and future work.

## 2. Literature review

The problem of extracting volumetric features directly from a mesh model has not received attention in the literature. There have been some efforts that extract volumetric feature by first reconstructing a smooth CAD model from the input mesh and then using available feature recognition algorithms for smooth models [16]. Literature on extracting features using segmentation of the mesh model is reviewed briefly.

Patane and Spagnuolo [14] proposed segmentation based on region growing. Their approach processes the scan lines to first simplify and then classify the points on the scan lines based on similarity with a library of basic shape types. Feature lines across scan lines are then extracted based on proximity analysis. Segmentation of CAD mesh model based on clustering is proposed by Xiao et al. [19]. Some of the problems with the region growing or clustering based approaches are that typically the surface is over-segmented and the segmentation boundaries are not smooth. Both the above clustering based segmentation methods are not good for identifying volumetric features in the mesh model.

Huang and Menq [9] have proposed automatic segmentation of 3D point cloud to extract geometric surface features by grouping the triangles (created from point cloud) separated by discontinuous mesh boundaries. Algorithm detects discontinuous boundaries using both curvature and tangent discontinuities. Here the term, 'feature' is only limited to boundaries and mesh patches which do not directly convey any information about the presence of volumetric features. Weber et al. [18] proposed another automatic algorithm to detect sharp edges and vertices from point cloud. Algorithm computes Gauss map clustering on local neighborhoods to discard all points which are unlikely to belong to a sharp feature and then apply iterative selection process on the remaining points to identify points that belong to sharp edges or corner vertices based on local sensitivity parameter. This parameter is the threshold value defined by the minimum distance among all resulting clusters. Based on tensor voting technique, a method is proposed by Kim et al. [11] to extract vertices, edges and faces from triangle mesh.

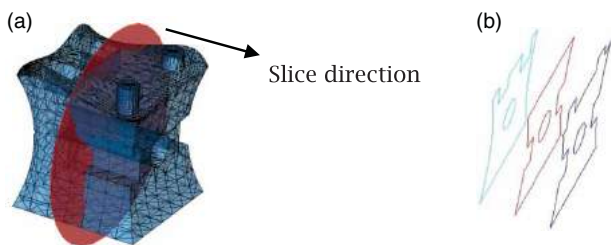
Garg et al. [7] proposed a fillet/blend suppression algorithm only for known types of underlying surface (Cylindrical, Spherical and Toroidal). Major limitation in their algorithm is that it considers planar adjacent faces around a blend. This is not always true for real world applications. A user assisted algorithm to identify planar, spherical, cylindrical and conical surfaces from a mesh model around a given seed point is proposed by Lai et al [12]. Reverse engineering framework for reconstruction of 3D solid model by extracting primitives from clean mesh model is proposed by Roseline et al.[5]. Gao et al. [6] use modified Watershed algorithm for mesh segmentation to obtain a CAD model that is then used to interactively identify features for suppression. The proposed modification involves first placing additional vertices in the mesh to enable the watershed algorithm to detect hard boundaries. This, however, results in new regions that have to be then merged so that segmented regions do not have any new entities. This approach also

has problems with cases where there are smooth boundaries between different regions. Moreover, the case where a newly added vertex becomes part of the segmented boundary is not addressed.

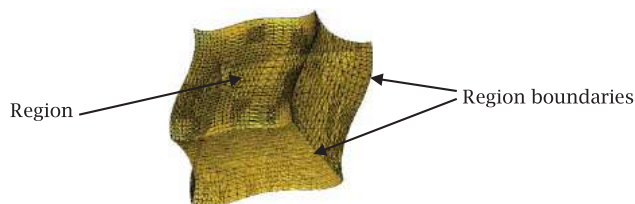
From the literature it is clear that there has not been much work done on extracting volumetric features directly from a mesh model.

### 3. Definitions and terminology

Following terminology is used in this paper – Slice and Wire - A cross section of the mesh model obtained by its intersection with a plane is referred to as a slice (Figure 1b). A closed contour of edges on a slice is called a wire. A slice may have multiple outer wires (but at least one) and each outer wire may contain multiple inner wires (could be none). A region (Figure 2) is a part of the mesh where all the triangles maintain discrete  $C^2$  continuity along their common boundaries. A region corresponds to a face in B-rep model.



**Figure 1.** (a) Mesh model and slicing plane, (b) slices.



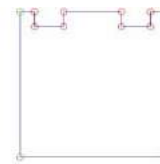
**Figure 2.** Mesh model with region and region boundaries.

**Sharp vertex** – A vertex is classified as sharp vertex, if the turning angle between two edges incident on it is greater than  $MIN\_THRESHOLD$  and less than  $MAX\_THRESHOLD$ . The turning angle is measured from first edge (E1) to second edge (E2) along counter clockwise direction with respect to the normal of the slicing plane.

**Smooth vertex** - If the turning angle between two edges of a vertex is less than  $MIN\_THRESHOLD$  or more than  $MAX\_THRESHOLD$ , the vertex is classified as smooth vertex.

$MIN\_THRESHOLD$  and  $MAX\_THRESHOLD$  have been defined based on an acceptable error of tessellation 'e' for a circle of radius 'r'. It can be shown from the tessellated representation of a circle that the turning angle between two facets is  $2\theta$ , where,  $\cos\theta = (r - e) \div r$  for counter clockwise orientation of edges with respect to normal to the plane of the circle. The value of  $MIN\_THRESHOLD$  is represented by  $2\theta$  and  $MAX\_THRESHOLD$  is represented by  $(360 - 2\theta)$ . In this paper,  $MIN\_THRESHOLD$  is set to be 50 degrees and  $MAX\_THRESHOLD$  is set as 310 degrees for a tessellation error 'e' being 10% of 'r'.

**Seed Point (SP):** Seed points are those points on a slice where there is a sudden change in shape in mesh model. In Figure 3, red and green circles represent seed points. Some of these seed points are associated with 3D features and are called potential-feature-seed-points, while the remaining seed points are known as ordinary seed points. An ordinary seed point belongs to a region boundary that is not a part of feature boundary. A seed point by construction is either associated with an edge shared by two triangles or a vertex shared by more than two triangles.



(Red circle - potential-feature-seed-points, green circle - ordinary seed points)

**Figure 3.** Typical slice section after removing co-linear vertices.

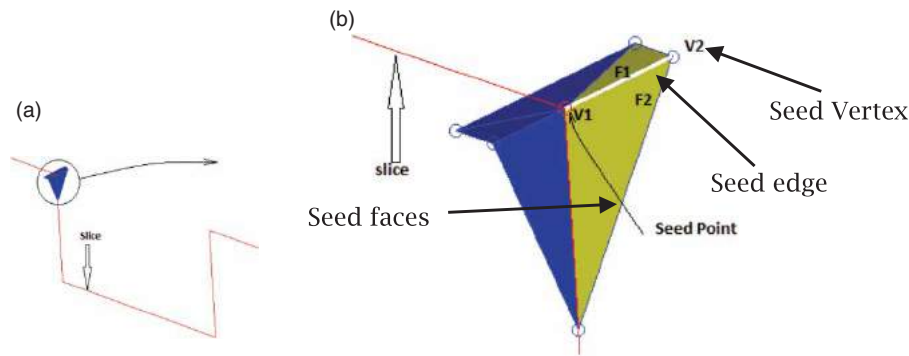
**Seed-Edge (SE)** - An edge in the mesh model that is responsible for potential-feature-seed-points (by intersection with the slicing plane) is called a seed-edge. In Figure 4b, edge V1V2 is a seed edge.

**Seed-Vertex (SV)** - A mesh vertex, associated with a seed edge and lying on the positive side of the slicing plane is called seed-vertex. It is associated with the potential-feature-seed-point. In figure 4b, vertex V2 is a seed vertex.

**Seed-Faces (SF)** - Triangles in the mesh model that are incident on the seed-edge are referred to as seed-faces. The seed-face is associated with potential-feature-seed-point. In Figure 4b, the faces shaded yellow are seed-faces. Figure 5 shows a typical slice. It consists of seven (7) wires, of which three are inner-loops and four are outer-loops.

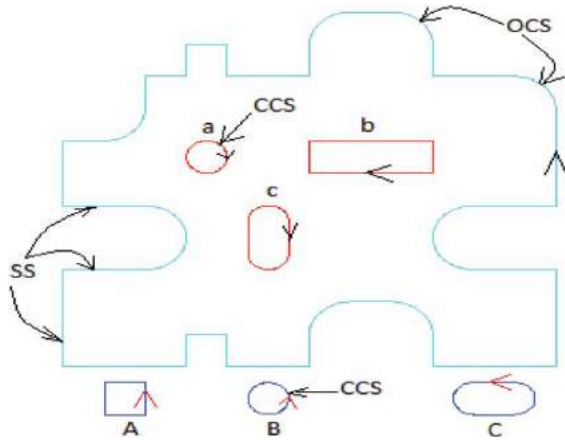
### 4. Algorithm to recognize features

This algorithm recognizes volumetric features in a mesh model by directly clustering the triangles constituting a



(Seed point of slice is associated with mesh vertex V1, V2 – Seed vertex, V1V2 – Seed edge, F1,F2 – Seed faces)

**Figure 4.** (a) Slice section with partial mesh, (b) enlarged view of circled region in (a).



**Figure 5.** - Slice segments (SS – straight segments, OCS – open curve segments, CCS – closed curve segments. a, B – closed feature made of single CCS. b, c, A, C – multi-segment closed feature).

feature in the mesh model. The algorithm involves two broad steps. In the first, presences of features are identified in 2D slice. The mesh model is sliced to obtain 2D slices and the contour forming the slice is processed to identify feature zones based on seed points and seed triangles. Seed points are points on a slice where there is a significant and characteristic change in the contour. The feature is recognized and extracted by a traversal from the seed point in one slice to the immediate neighboring slice. The traversal step clusters triangles belonging to the feature.

The main steps in the algorithm are:

- (i) Pre-processing to calculate bounding box and slicing plane.
- (ii) Two-dimensional slicing to generate 3D-seed-information for feature boundaries in mesh model.
- (iii) Three-dimensional traversing along the feature boundaries in mesh model using 3D-seed-information.

- (iv) Extracting the features separated by feature boundaries in mesh model.

These are explained in detail in the following subsections.

#### 4.1. Pre-processing

The mesh in the .stl format is read in and adjacency relation between the triangles is determined and stored. A min-max box aligned with the global coordinate reference frame is determined for the vertices in the mesh. The slicing direction is chosen to be the co-ordinate axis along which the min-max box has the smallest dimension. This is done to minimize the number of slices that need to be processed in the algorithm. The slicing plane is normal to the slicing direction.

#### 4.2. Two dimensional slicing of mesh model

The slicing operation starts from one end of the bounding box and the slicing plane is offset along its normal by a pre-defined distance and slicing repeated till the entire model is covered. The pre-defined distance is set to be smaller than the minimum feature dimension (MFD) expected in the model. For each slice the following processing is done.

##### 4.2.1. Classify vertices and create slice segments in each slice

In this step, co-linear vertices on the slice are first removed. The remaining vertices are then classified as smooth, sharp, convex or concave. The classification is based on the turning angle between the edges incident on the vertex. Algorithm creates a list of slice vertices using this classification information.

Based on the available vertex classification information stored in the list of slice vertices, slice segments are



identified. A closed curve slice segment (ccs) gets created if all the slice vertices of the list are classified smooth and the first and last vertices in the list are identical. If two consecutive slice vertices of the list are classified as sharp-sharp or sharp-smooth, a straight slice segment (ss) gets created by using those two consecutive slice vertices. An open curve slice segment (ocs) gets created by a set of smooth slice vertices such that the start and end vertices are different. Figure 5 illustrates the different type of slice segments.

#### 4.2.2. Identify two-dimensional (2D) feature segments

This step identifies 2D-features by grouping slice-segments. A group of slice segments in a wire of a slice makes a 2D feature.

A closed feature in 2D is made of single segment closed curve or multi-segment closed curve. A multi-segment closed feature will have a combination of open curve (OCS) and straight segments (SS). In Figure 6, red circle is a closed feature made of single CCS. In Figure 5, wires 'b', 'c', 'A' and 'C' are multi-segment closed features. A closed feature inherits the classification of a closed wire as either inner-loop or outer-loop based on the convexity information of slice vertices of that closed wire. The closed feature (red circle) in Figure 6 is the inner-loop, whereas the closed features 'A', 'B' and 'C' in Figure 5 are the outer-loops.

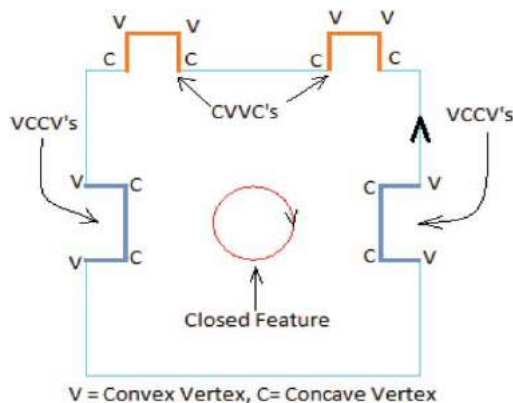


Figure 6. 2D features.

VCCV feature – If a 2D feature consists of slice segments with a chain of concave vertices bounded by convex vertices, the feature will be classified as VCCV's type. Here, 'V' denotes a convex vertex and 'C' denotes a concave vertex. In Figure 6, convex and concave vertices are shown with "V" and "C". Typically, this is a 2D feature created by material removal.

CVVC feature – If a 2D feature consists of slice segments with chain of convex vertices separated by concave vertices, the feature will be classified as CVVC's type. Both 'V' and 'C' have the same meaning as above. This is a 2D feature created by material addition.

Blend feature – An open curve segment (OCS) is called a blend feature. In Figure 5, open curve segments are marked OCS.

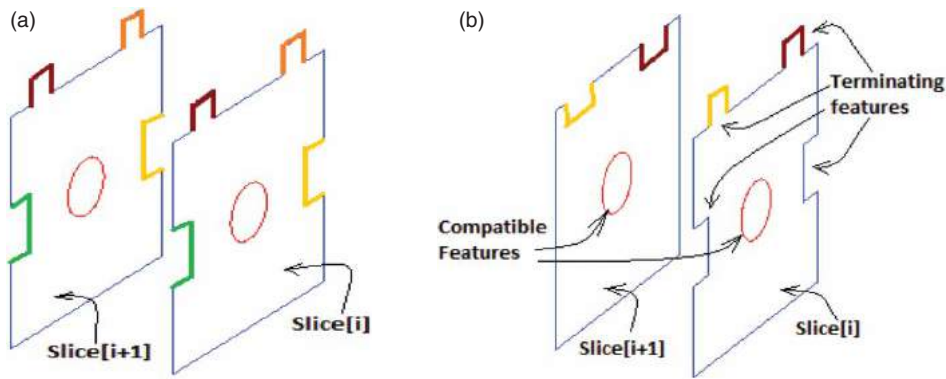
A 2D feature of type VCCV's and CVVC's directly classifies the type of the 3D feature as depressions and protrusion respectively. A closed 2D feature may be related to a 3D feature of type through hole or protrusion or depression. The type of 3D feature, related to a closed 2D feature, gets classified based on the convexity of the feature boundary.

#### 4.3. Three dimensional traversal

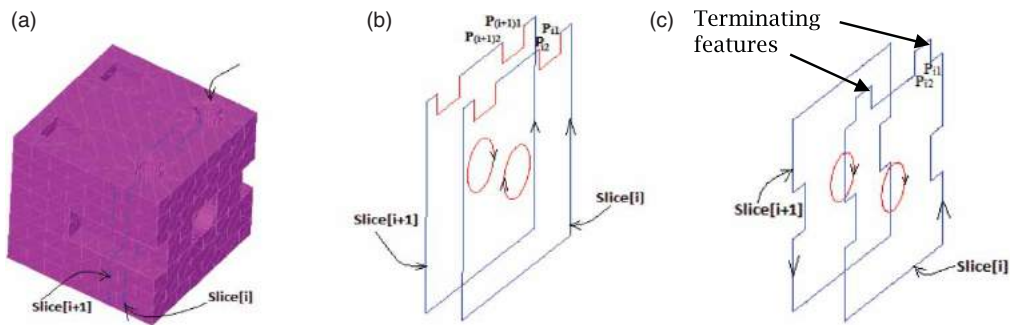
The 2D features identified in the slices form the starting point for the 3D traversal algorithm. The 3D traversal algorithm traces the boundary of the 3D feature associated with the 2D feature between slice[i] and slice [i + 1]. The complete feature is obtained by traversing across all slices. As the offset between two consecutive slicing planes is less than the minimum feature dimension, it is guaranteed that all the 3D features in the mesh model will be captured in one or more slicing planes. At each slice, the procedure traverses the boundary edges of every feature other than closed 2D-feature between the current slice and the next. This is done if the check for compatibility of slices or 2D features in the slices is established. Any feature that does not have a compatible 2D feature in the next slice will be treated as potential terminating feature and the procedure treats this case to find the boundary edges for the portion of the remaining feature from this slice. In the same manner any incompatibility of 2D features between the present slice and the previous one is also investigated to trace the boundary edges for the portion of the feature that terminates before the present slice. Closed features are handled separately. Finally, these boundary edges are used to extract the triangles associated to this 3D feature. The main steps in the procedure are:

- (i) Determination of compatible slices and compatible 2D features
- (ii) Traversing of a 2D feature from slice[i] to slice [i + 1].
- (iii) Traversal of a terminating 2D feature from slice[i] in either direction.
- (iv) Traversal of closed features.

Each step is described briefly below.



**Figure 7.** (a) Compatible slices, (b) incompatible slices.



**Figure 8.** (a) Object (b) slices with seed points for compatible features (c) and terminating features.

#### 4.3.1. Determination of compatible slices and features

If the numbers of 2D features in the two slices are the same and if the type of corresponding features in the two slices is the same, the two slices are said to be compatible. Since the distance between two successive slices is less than minimum feature dimension, the adjacent features with same type on slice[i] and slice[i + 1] must be associated with the same 3D features. Two successive slices are classified as incompatible when they have different types of 2D features. In Figure 7a, all 2D features of slice[i] are of same type as those in adjacent slice[i + 1].

In a pair of incompatible slices, a 2D feature on slice[i] is said to be compatible to a corresponding 2D feature on slice[i + 1], if they are of same type. The closed 2D feature in red in Figures 7a and 7b are compatible features in compatible and incompatible slices respectively.

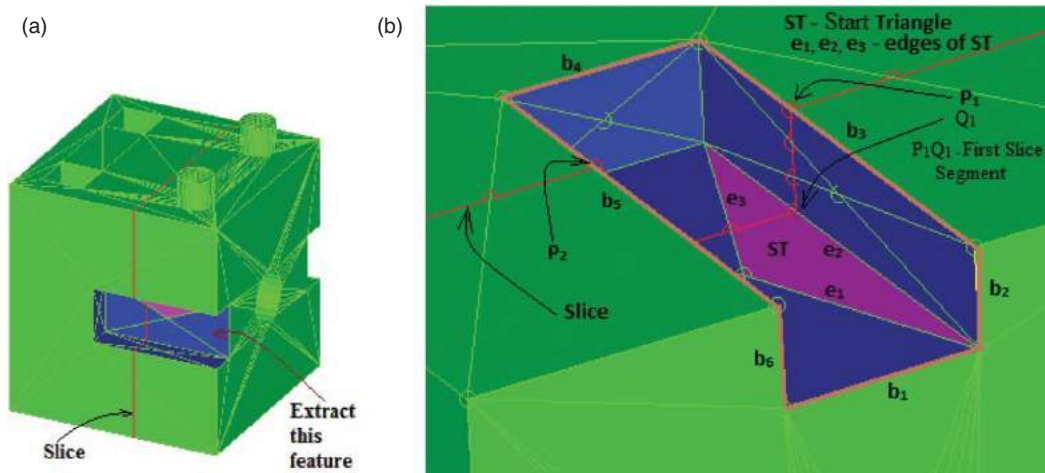
A 2D feature on a slice that corresponds to a given 2D feature on a neighbouring slice is determined as follows. A bounding rectangle that is aligned with the slice is constructed for the given 2D feature. The bounding rectangle is then projected on the plane of the next slice. If the projection intersects with the bounding rectangle of a feature on the next slice then the two features are said to be corresponding.

#### 4.3.2. Traversal from slice[i] to slice[i + 1]

This step identifies boundary edges of a 3D feature corresponding to a compatible pair of 2D features in the two slices. From each feature, the procedure extracts the corresponding seed points (start and end respectively). It traverses from both the start and end seed points  $P_{i1}$  &  $P_{i2}$  of a 2D feature on the slice[i] (Figure 8b), to terminate at the corresponding start and end seed points  $P_{(i+1)1}$  and  $P_{(i+1)2}$  respectively on slice[i + 1]. The traversal starts with the seed vertex of the edge incident on the seed point. It then identifies edges incident on the seed vertex of this edge and selects the edge that has its dihedral angle between MIN\_THRESHOLD and MAX\_THRESHOLD. Traversal then continues with the other end vertex of the selected edge till the edge containing a seed point on the next slice is reached. The list of boundary edges identified during the traversal forms the boundary edges of the 3D feature associated with the 2D feature.

#### 4.3.3. Traversal of terminating feature from slice[i]

A terminating 2D feature on slice[i] does not have corresponding compatible 2D feature on slice[i + 1] (Figure 8c). This step traces the remaining boundary edges of the 3D feature associated with the 2D feature. The traversal starts with the seed vertex associated



**Figure 9.** (a) Extraction of triangles of a 3D feature, (b) enlarge view of slot shown in (a).

with  $P_{i1}$ . From the mesh edges incident on this vertex, the next boundary edge of the feature would be one whose di-hedral angle is between  $MIN\_THRESHOLD$  and  $MAX\_THRESHOLD$ . If there are more than one edge satisfying this condition the procedure will pick an edge that is closely aligned with the vector defined by the vertices associated with the start and end seed points. The traversal terminates when the last boundary edge encountered contains the vertex associated with the end seed point.

There may be a situation where a terminating 2D feature is incompatible with respect to the 2D features on its previous slice. For those cases, algorithm traverse on both sides (positive & negative) of the slicing plane to extract the boundary edges of the 3D feature.

#### 4.3.4. Traversal for closed features

The 3D feature corresponding to closed 2D features have boundary edges only at the end faces and not between slices. Therefore, in the case of such features only the first and the last slice containing the corresponding closed 2D feature need to be processed. For closed feature, traversal is from the last slice in the positive direction of slicing plane to the boundary edge. Similarly, traversal is done from the first slice containing the 2D-closed feature in negative direction of the slicing plane. At the two boundaries, the boundary edges at the two ends of the 3D closed feature are obtained as described for the traversal between two seed points. Closed feature may have one or many slice segments. For single segment closed feature, seed points will be at same location.

It may be observed that for closed 2D-features associated to protrusion, one of the terminal slices will be the closed feature itself and in this case, it will be sufficient to traverse the boundary edges from the other terminal slice. There may be arbitrary oriented cases where none

or few of the 2D-features associated to a 3D feature are compatible. Above traversal algorithms are able to extract all feature-boundary edges of the volumetric feature by using the 3D-seed-information of those 2D-features.

#### 4.4. Extraction of 3D features

The feature boundary edges identified in the traversal steps above, separate a 3D mesh feature from its base mesh model. Extraction of the feature is the process of isolating the triangles of a 3D feature. Algorithm starts with a triangle  $ST$  that belongs to the feature, shown in Figure 9b, and grows the triangles from it in such a way that all triangles in the set are bounded by the feature boundary edges.

### 5. Implementation and results

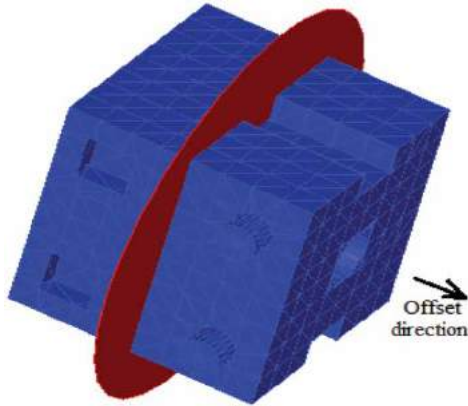
The proposed algorithms have been implemented using VC++ in VISUAL Studio 2008 environment in a computer with following configuration: Intel core i3 processors, 3GB RAM, 64 bit windows 7 operating system. APIs of 3D ACIS kernel modeler [1] has been used. Schema DL and GL viewer of acis have been used for visualization. The effectiveness and efficiency of the algorithm have been verified by testing on different mesh models obtained from .stl files containing facets of CAD mesh models. An illustrative example and results of testing on different mesh models are presented in the following sections.

#### 5.1. Illustrative example

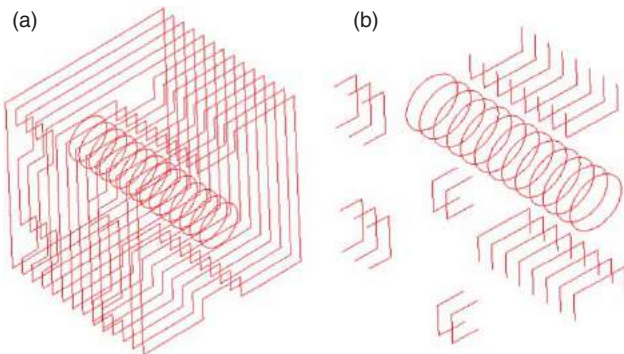
Figures 10 to figure 14 briefly illustrate the feature extraction procedure described in this paper. Algorithm starts with an input mesh model along with a slicing plane. Both



are shown in figure 10. Slices are generated by moving the slicing plane along the offset direction. Figure 11a shows all the 2D slices created by slicing with the slicing plane along the offset direction shown in figure 10. The offset between two successive slicing planes is set to be 3.5 units in this case. 2D features are extracted from these slices. All 2D features identified in each of these slices are shown in figure 11b.



**Figure 10.** Mesh model, slicing plane and offset direction.

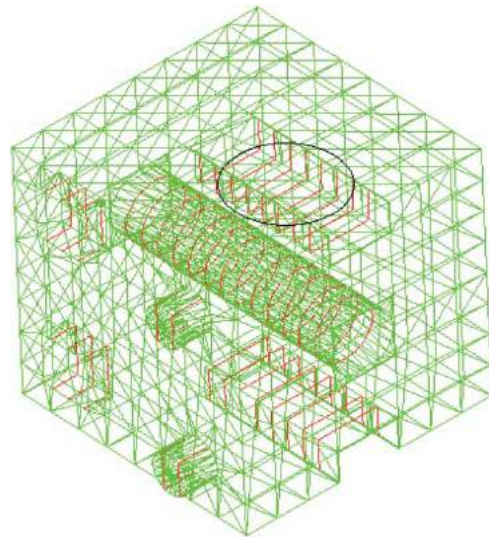


**Figure 11.** (a) Slices, (b) 2D features.

In Figure 12, extracted 2D features are shown along with the mesh model. Compatible slices and compatible 2D features are clearly visible in this figure. 3D features are extracted by traversing along the 3D feature boundaries of the mesh model by using the seed data available with each 2D feature. Traversal between slices is illustrated in Figure 13. Identified 3D features are shown in the Figure 14a. Finally, extracted features are shown in the mesh model in Figure 14b.

## 5.2. Results

Table 1 summarizes the model details as well as time performance of the algorithm for the examples shown



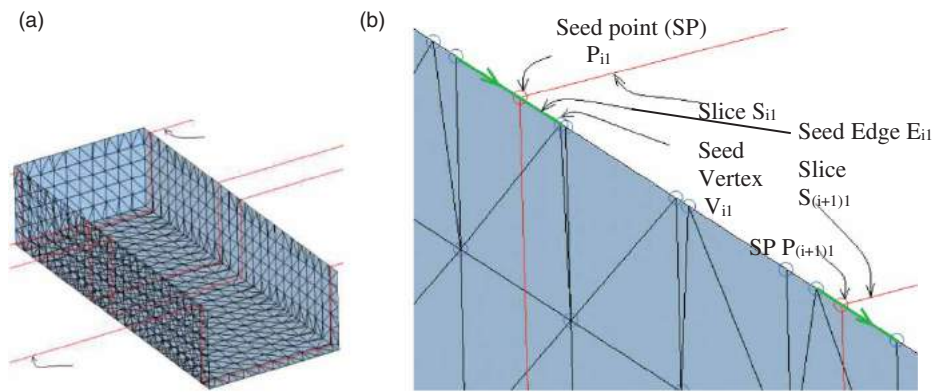
**Figure 12.** Mesh model and 2D features.

in Figures 15a, 16a and 17a respectively. The features identified in each case are shown in Figures 15b, 16b and 17b respectively. Table 2 shows the model details for the mesh model in Figure 17a with varying mesh density. Here MFD is known for each mesh model. Numbers of slices are related to the MFD as well as the minimum dimension of the mesh model. For a typical CAD mesh model, numbers of slices are independent of the number of triangles of that CAD mesh model. Number of slices is fixed for a given MFD. Two dimensional processing is proportional to the number of slices and speed of slicing depends on the number of triangles. As the number of slices is fixed for a given MFD, 2D processing timing depends only on the number of input triangles. Again slicing identifies the feature sensitive zones in a mesh model to be processed in three dimensions. Since the numbers of slices are fixed, increasing the number of triangles does not increase the timing of overall feature extraction (2D and 3D processing) linearly. As can be seen from the table 2, overall timing grows very slowly with respect to the growth in the number of triangles.

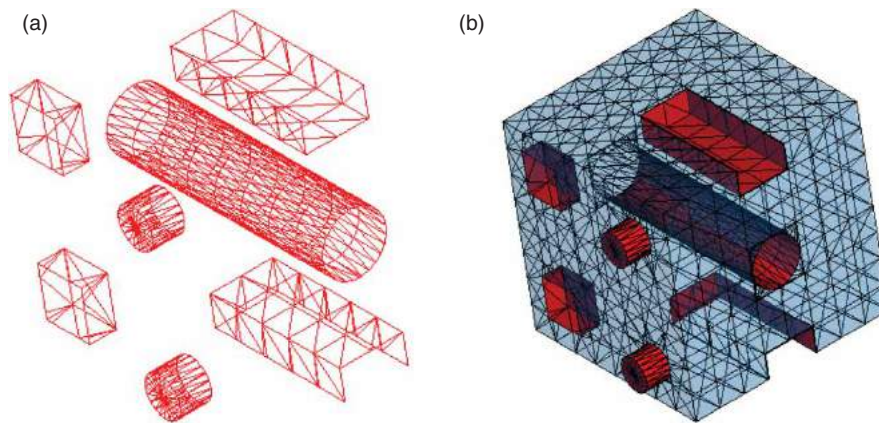
While the slicing plane and offset direction are iso-oriented with respect the global reference frame, the orientation of the plane does not influence the result in any way. The orientation of the slicing plane with respect to the object can be arbitrary. An example is shown in figure 18 that shows a few slices and the 2D features for a case where the relative orientation of the slicing plane and the object is arbitrary.

Here, all the 2D-features are incompatible with each other. Hence, each of them is treated as potential terminating feature and traversed in both the slicing direction (positive as well as negative) to extract feature-boundary





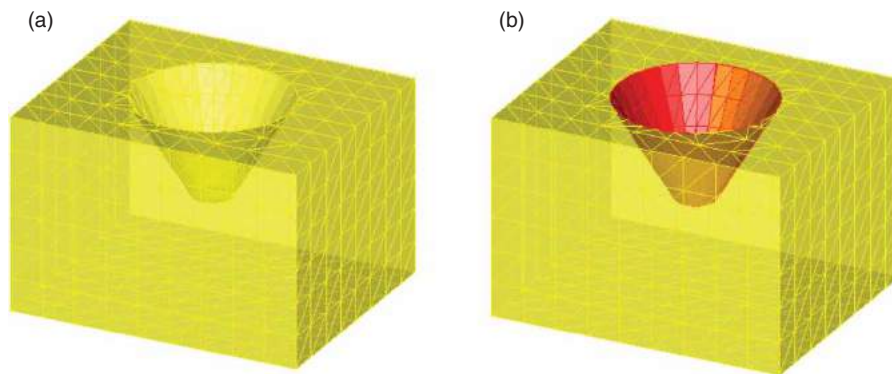
**Figure 13.** (a) Volumetric feature with two intermediate slices and two terminating slices, (b) traversal for a volumetric feature from slice[i] to slice[i + 1].



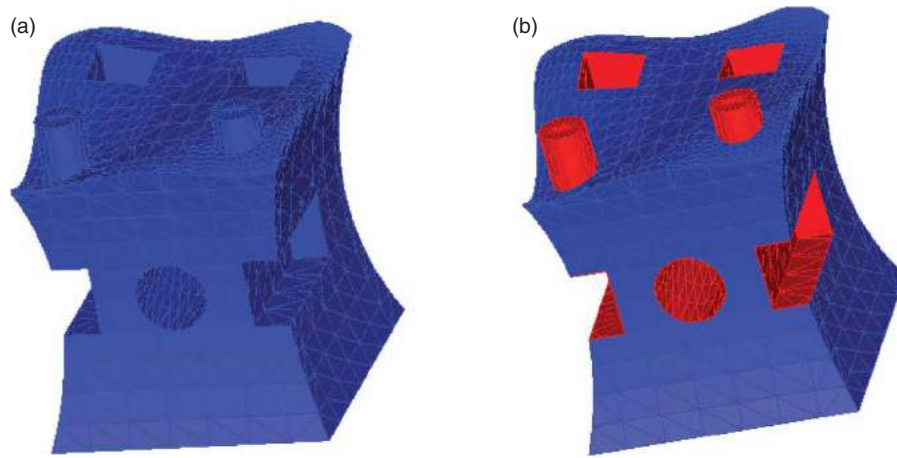
**Figure 14.** (a) identified 3D features in wire frame view, (b) extracted features in the mesh model.

**Table 1.** Experimental results for example mesh models shown in Figures 15a, 16a and 17a.

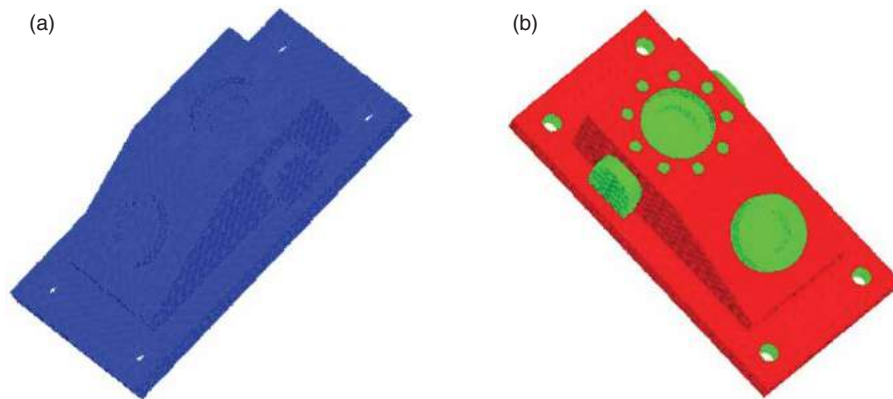
Examples	Number of Triangles	Number of Features	Minimum Feature Dimensions (MFD)	STL Data Size (in KB)	Number of Slices	Overall Timing (in ms)
Example 1 Figure 15a	978	1	3	236	9	608
Example 2 Figure 16a	2712	7	5	708	14	2059
Example 3 Figure 17a	7846	17	3	2078	12	3136



**Figure 15.** (a) Example 1 - input mesh model, (b) mesh model and extracted feature.



**Figure 16.** (a) Example 2 – input mesh model, (b) mesh model and extracted features.



**Figure 17.** (a) Example 3 - input mesh model, (b) mesh model and extracted features.

**Table 2.** Experimental results for example 3 mesh model shown in Figure 17a with varying mesh density.

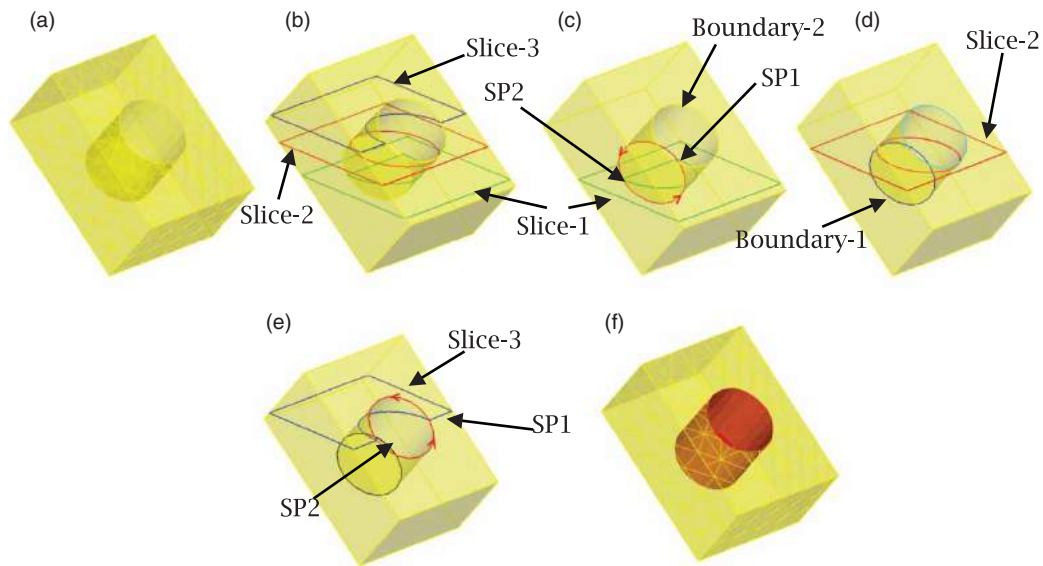
Cases	Number of Triangles	Number of Features	Minimum Feature Dimensions (MFD)	STL Data Size (in KB)	Number of Slices	Overall Timing (in ms)
Case 1	79786	17	3	21143	12	9813
Case 2	26578	17	3	7042	12	5413
Case 3	7846	17	3	2078	12	3136
Case 4	2950	17	3	781	12	2589

edges as described earlier. After traversal, algorithm verifies that the boundary edges extracted for 2D-features on Slice-1 and Slice-3 are same as the boundary edges extracted for the 2D-feature on Slice-2. The algorithm therefore associates these incompatible 2D-features with the same 3D-feature and extracts the triangles bounded by those feature boundary edges. Figure 18f shows the output of the arbitrary oriented mesh model shown in figure 18a.

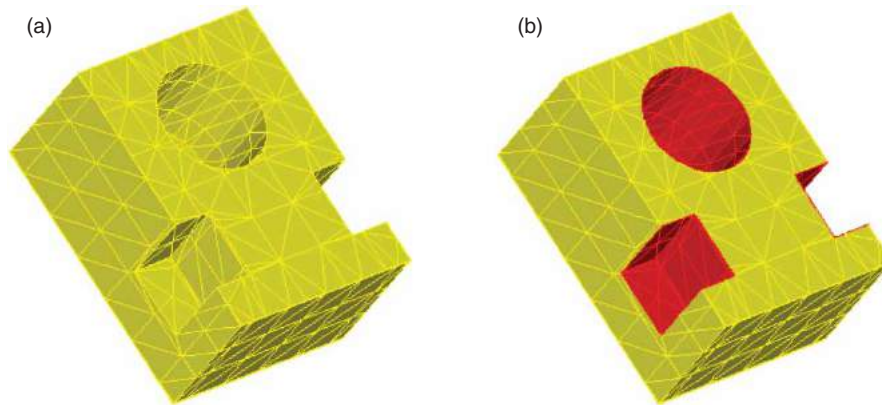
Another example of arbitrary oriented mesh model is shown in figure 19a. Extracted features along with the mesh model are shown in figure 19b.

## 6. Discussion

The proposed approach is able to identify all the boundary edges of a volumetric feature from a CAD mesh model and isolates all the triangles of that feature. The most interesting aspect of this approach is that the algorithm does not depend on the underlying surface definition of each region and also does not process the entire mesh; instead it simply narrows down the extraction process to only in the feature regions. This is because of the elimination of most of the regions in two-dimensional processing. Completeness of the feature



**Figure 18.** (a) Arbitrary oriented input mesh model to extract feature, (b) model and slices with all incompatible 2D-features, (c) model and slice-1 with 2D-feature, (d) model and slice-2 with 2D-feature, (e) model and slice-3 with 2D-feature, (f) mesh model and extracted feature (mesh edges are suppressed in the model from frame 'b' to frame 'e' for better visualization).



**Figure 19.** (a) Arbitrary oriented input mesh model with three features, (b) mesh model and extracted features.

extraction algorithm depends on the choice of MFD, which is taken as an input at present.

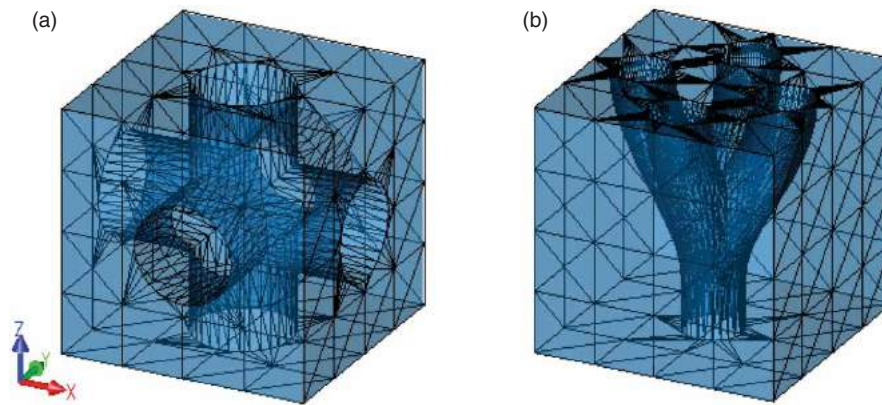
Since mesh models are imported data from various sources, noise and geometric errors may be present [1]. The algorithm has been tested with noisy input data (obtained by randomly altering the vertices in a mesh obtained from a CAD model). Noise is defined as percentage deviation of each mesh vertex with respect to the minimum feature dimension. It was found through experiments on a sample part that the procedure has no difficulty in extracting the features correctly for noise of upto 6.5%.

The characteristic size of the problem is the number of triangles in the mesh model (say  $M$ ). The steps deciding the complexity of the algorithm are those involving slicing and their processing. The worst case complexity of the slicing step is  $O(n_s * M)$  where  $n_s$  is the number

of slices. This can be reduced by culling the number of triangles to be considered using a bounding box defined by the slicing plane. The complexity of the steps involving classification of the slice vertices and slice segments is controlled by the number of triangles intersected by the slicing plane. If this number is  $M_s$  then complexity is linear with respect to  $M_s$  and  $M_s \lll M$ . The 3D traversal phase does not involve any combinatorial step as triangles are traversed between slices based on adjacency information that is determined once and stored in the mesh representation. The whole procedure therefore, is linear with respect to the number of triangles or faces and compares well with some of the reported complexity [4]  $O(f^2)$  for Katz and Tal [10] and  $O(\log f)$  for Liu and Ma [13] where  $f$  is the number of mesh faces.

Minimum feature dimension (MFD) is the smallest dimension associated with a feature in a mesh model.





**Figure 20.** (a) Interacting feature, (b) branching features.

Its value is assumed to be known. Slicing planes will be placed at an interval of less than minimum feature dimension. This is to ensure that each feature is intersected by at least one slicing plane. Numbers of slices are inversely proportional to the MFD. Two dimensional processing generates more slices from mesh model with smaller features than the model with large features. Moreover, more slices speed up the process of three dimensional traversal as two adjacent slices may be topologically connected by a mesh edge. This topological information eliminates the computation of dihedral angles to reach to the next slice while 3D traversing along feature boundaries. Therefore, more the number of slices, means more is the time for two dimensional processing, but less time for three dimensional traversing. The proposed algorithm does not depend on any underlying geometric information or on the density of triangles. Input mesh model may have either coarse or dense triangles. Actually coarse triangles help to speed up three-dimensional traversal since the adjacent slices may be topologically connected through bigger mesh edges. This helps to directly jump on to the next slice during three-dimensional traversing without doing any computation. Inversely, dense triangle takes relatively more time during three-dimensional traversal, as it also requires vector computation apart from the available topological information through mesh edges and vertices. Two dimensional slicing mostly depends on vector analysis, whereas three dimensional processing mainly depends on topological connectivity available in the mesh model and very least on vector analysis. Any algorithm based on topological information and vector computation is efficient and robust. Hence, the proposed algorithm is fast and robust. This is also verified by experimental output for example mesh models shown in tables 1 and 2.

Sometime 3D feature does not leave any signature on the slice. A through slot or a through hole can yield such cases. While heuristic procedures can be devised to

handle these cases, the most straightforward way to handle these would be to repeat the slicing process with an orthogonal slicing plane.

The algorithm at present cannot identify intersecting features as shown in Figure 20a. Multi-direction slicing can be explored to extract intersecting features. Proposed algorithm will be able to detect branching feature for simple branching objects for limited number of slicing directions, for example slicing direction along the axis of sweep for multi branch sweep object. An example of such a case is shown in Figure 20b. For this example, a 2D feature in a slice will have multiple compatible 2D features in the next slice. Algorithm will fail in 3D-traversal for branching features when slicing is in some arbitrary directions.

## 7. Conclusion

An algorithm to recognize volumetric features by directly clustering the triangles constituting a feature in a mesh model has been presented in this paper. The novelty of this approach is that it partially solves the problem in 2D and uses the results to reduce the effort in solving the 3D problem. The algorithm involves two steps – isolating features in 2D slices followed by a 3D traversal to cluster all the triangles in the feature. Algorithm also identifies the potential feature type from the 2D slices. Based on experimental results and time complexity analysis, it is concluded that this feature extraction algorithm is efficient and fast in identifying volumetric features from mesh model over the existing algorithms. Current work is focused on handling interacting features and in parameterizing the extracted features.

## ORCID

B. Gurumoorthy  <http://orcid.org/0000-0001-9857-9011>



## References

- [1] ACIS 3D Kernel, <http://www.spatial.com>; <http://doc.spatial.com/index2.php>
- [2] Agathos, A.; Pratikakis, I.; Perantonis, S.; Sapidis, N.; Azariadis, P.: 3D Mesh Segmentation Methodologies for CAD applications, *Computer-Aided Design & Applications*, 4(6), 2007, 827–841. <http://dx.doi.org/10.1080/16864360.2007.10738515>
- [3] Angelo, L. D.; Stefano, P. D.: C1 continuities detection in triangular meshes, *Computer-Aided Design*, 42(9), 2010, 828–839. <http://dx.doi.org/10.1016/j.cad.2010.05.005>
- [4] Attene, M.; Katz, S.; Mortata, M.; Patane, G.; Spagnuolo, M.; Tal, A.: Mesh segmentation – A comparative study, *Proceedings of the IEEE International Conference on Shape Modeling and Applications*, 2006, 14–25. <http://dx.doi.org/10.1109/smi.2006.24>
- [5] Bénéière, R.; Subsol, G.; Gesquière, G.; Le Breton, F.; Puech, W.: A comprehensive process of reverse engineering from 3D meshes to CAD models, *Computer-Aided Design*, 45, 2013, 1382–1393. <http://dx.doi.org/10.1016/j.cad.2013.06.004>
- [6] Gao, S.; Zhao, W.; Lin, H.; Yang, F.; Chen, X.: Feature suppression based CAD mesh model simplification, *Computer-Aided Design*, 42(12), 2010, 1178–1188. <http://dx.doi.org/10.1016/j.cad.2010.05.010>
- [7] Garg, A.; Krishnan, S. S.; Gurumoorthy, B.: Recognition and suppression of fillets/rounds in a tessellated solid model, *International CAD conference and exhibition, CAD'04*, May 2004. Thailand.
- [8] Hoffman, D.; Richards, W.: Parts of recognition, In S. Pinker, editor, *Visual Cognition* 1985; 18, 65–96.
- [9] Huang, J.; Menq, C. H.: Automatic data segmentation for geometric feature extraction from unorganized 3-D coordinate points, *IEEE Transactions on Robotics and Automation*, 17(3), 2001, 268–279. <http://dx.doi.org/10.1109/70.938384>
- [10] Katz, S.; Leifman, G.; Tal, A.: Mesh segmentation using feature point and core extraction, *The Visual Computer*, 21(8), 2005, 649–658. <http://dx.doi.org/10.1007/s00371-005-0344-9>
- [11] Kim, H. S.; Choi, H. K.; Lee, K. H.: Feature detection of triangular meshes based on tensor voting theory, *Computer-Aided Design*, 41(1), 2009, 47–58. <http://dx.doi.org/10.1016/j.cad.2008.12.003>
- [12] Lai, H.-C.; Chang, Y.-H.; Lai, J.-Y.: Development of feature segmentation algorithms for quadratic surfaces, *Advances in Engineering Software*, 40, 2009, 1011–1022. <http://dx.doi.org/10.1016/j.advengsoft.2009.03.017>
- [13] Liu, S.; Ma, W.: Seed-growing segmentation of 3-D surfaces from CT-contour data, *Computer-Aided Design*, 31(8), 1999, 517–536. [http://dx.doi.org/10.1016/S0010-4485\(99\)00050-0](http://dx.doi.org/10.1016/S0010-4485(99)00050-0)
- [14] Patane, G.; Spagnuolo, M.: Multi-resolution and slice-oriented feature extraction and segmentation of digitized data, *Proceedings of the seventh ACM symposium on Solid modeling and applications*, 2002, 305–312. <http://dx.doi.org/10.1145/566282.566326>
- [15] Razdan, A.; Bae, M.: A hybrid approach to feature segmentation of triangle meshes, *Computer-Aided Design*, 35(9), 2003, 783–789. [http://dx.doi.org/10.1016/S0010-4485\(02\)00101-X](http://dx.doi.org/10.1016/S0010-4485(02)00101-X)
- [16] Sunil, V. B.; Pande, S. S.: Automatic recognition of features from freeform surface CAD models, *Computer-Aided Design*, 40, 2008, 502–517. <http://dx.doi.org/10.1016/j.cad.2008.01.006>
- [17] Wang, J.; Yu, Z.: Surface feature based mesh segmentation, *Computers & Graphics*, 35, 2011, 661–667. <http://dx.doi.org/10.1016/j.cag.2011.03.016>
- [18] Weber, C.; Hahmann, S.; Hagen, H.: Sharp feature detection in point clouds, *IEEE International Conference on Shape Modeling and Applications*, 2010. <http://dx.doi.org/10.1109/smi.2010.32>
- [19] Xiao, D.; Lin, H.; Xian, C.; Gao, S.: CAD mesh model segmentation by clustering, *Computers & Graphics*, 35(3), 2011, 685–691. <http://dx.doi.org/10.1016/j.cag.2011.03.020>