STL Model Segmentation for Multi-Axis Machining Operations Planning

Pierre P. Lefebvre [1] and Bert Lauwers[2]

[1]Division PMA, KULeuven Belgium, pierre.lefebvre@mech.kuleuven.ac.be
[2]Division PMA, KULeuven Belgium, bert.lauwers@mech.kuleuven.ac.be

ABSTRACT

This paper aims a segmentation procedure for an STL model on behalf of an automated multi-axis milling operations planning system. The STL file that represents the existing product model has to be segmented into regions suited for the planning of milling operations. Originating from the domain of rapid prototyping, the STL data format nowadays appears in other domains of manufacturing. Within machining, STL is being used for three-axis tool path generation and some prototype systems are able to generate five-axis tool paths directly on STL. The algorithm presented in this paper is situated in the step prior to tool path generation. The dedicated segmentation algorithm consists of a phase of sharp edge detection followed by a workpiece setup dependent segmentation. The dynamically variable nature of this segmentation scheme is stressed.

Keywords: STL, geometry segmentation, CAPP

1. INTRODUCTION AND LITERATURE REVIEW

Multi-axis machining is used for the manufacturing of complex shaped products. The CAM operation planning for multi-axis milling is a very labor-intensive task. One starts from the CAD model of the freeform shaped product and analyses the shape in order to identify regions suited for a specific machining operation. Particularly for freeform shapes it mostly happens that the identified machining regions do not accord with CAD model design features. Therefore a workaround using drive surface geometry is needed [8]. This mismatch between design features and manufacturing features for complex shaped geometry makes the operations planning a complicated task. It also makes that the workpiece CAD geometrical description is not always useful for the CAM programming so that other geometrical descriptions may be used for CAM programming purposes. In this research work a triangular faceted geometrical description was chosen.

Triangular meshes are very popular data structures for product geometry description. They appear in visualization applications and computer graphics, CAE, RE (reverse engineering) and computer integrated manufacturing. In manufacturing engineering, triangular meshes for product geometry first appeared in the field of rapid prototyping as the STL (STereo Lithography or Standard Triangulation Language) data format. STL is a particular implementation of a triangular mesh. This data format was so successful that it became a de facto standard for geometrical data description. Its simplicity is probably the most dominant factor of success.

An STL model is a surface model based on 1 geometrical primitive, namely a triangle. This makes the geometry description homogenous, CAD-kernel independent, modeling history independent and it allows the description of whatever kind of shape complexity. The STL data format also has a lot of disadvantages, such as the limited accuracy, the high memory storage space and the complete loss of metadata of the product model. Despite its drawbacks, the importance of STL in CAD/CAM applications is still increasing. One sees nowadays STL appearing in traditional CAM applications (ex. tool path calculation based on STL) [6],[7]. Modeling and design based on faceted geometry is also gaining importance and is a topic of ongoing research.

At least three reasons were decisive to use the STL data format within this application domain of multi-axis CAM for complex shapes. First, a homogeneous STL model is beneficial for analysis routines compared to complicated heterogeneous geometrical descriptions (ex. a set of different connected surface patches, complex sweeps, etc.). Those analysis routines are necessary for automated part geometry investigation. Second, complex (freeform) product shapes also advocate for STL since it hides the complex modeling history. Freeform shapes are conceived from advanced operations in CAD, like surface modeling operations, solid-solid intersections, etc. Finally, because of the homogeneity, sub regions of the geometry can easily be

defined and eventually altered according to the needs of certain CAM operation types.
The drawback of this implicit simplification by STL is that the information content of the data is very limited. A phase of manufacturing feature identification on the product model is necessary before machining operations planning can be elaborated. Mind that this identification phase is also necessary if a classical CAD geometrical description is used, as already discussed before. The algorithm presented in this paper is part of such kind of manufacturing feature identification procedure. The algorithm aims to segment the STL into useful geometry sub regions for machining operations. This algorithm is developed specifically for multi-axis milling operations planning for complex shapes (ex. Fig. 1). Feature types like holes and regular shaped pockets are not considered as complex shapes. A lot of research for these regular feature types has already been done and is well documented in literature.

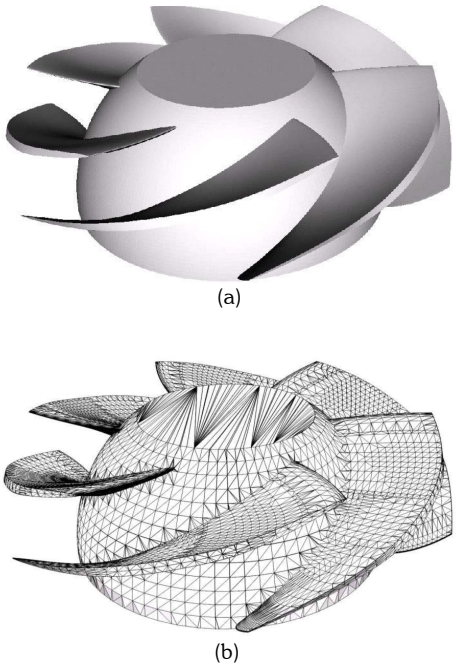


Fig. 1. Example of a complex shaped workpiece to be machined by multi-axis milling. Original CAD model (a), STL model (b).

Triangular meshes have already been studied extensively in literature on computer graphics, CAE and RE. Using STL for CAPP (more specifically operations planning for CAM) is a novel application domain.
The particular problem of splitting up a triangular mesh into meaningful subsets (segmentation) seems to be very application specific. The segmentation algorithms for point cloud datasets (in RE for example) have to deal with other issues than the ones for STL based datasets. Scan measurements typically contain dense, noisy, more or less equally spaced data points. Advanced algorithms are necessary to process those scan datasets [2], [13]. The RE application also requires the retrieval of design features like holes, blends, sweeps, etc. because a CAD model needs to be reconstructed from the scan data. This issue also requires very specific algorithms [1]. STL dataset properties are rather different. The triangles can vary intensely in shape and size, some regions of the geometry are densely sampled, others are not, and the noise level is limited to the triangulation accuracy.

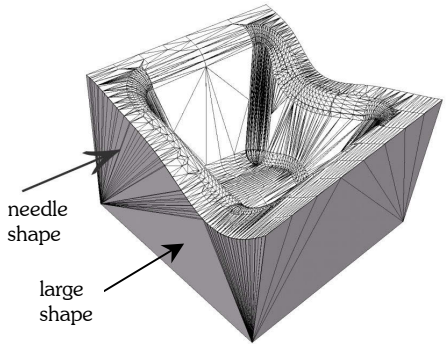


Fig. 2. STL model with different triangle shapes and sizes.

Segmentation of STL datasets specifically for operations planning has not been addressed in literature before. However some related topics on segmentation are already reported. Razdan et al. report in [12] about a mixed approach of the watershed method [11] combined with an edge based method to deal with irregular spaced meshes. This method is quite general to be used in different application domains, but it is not fully automated. Manual interaction is needed to control the segmentation process.
The algorithm in [12] uses local mesh adaptation. The issue of mesh adaptation appears very often in related literature, but is again application specific. Lee et al. describe a technique for global adaptation of triangular meshes in [9], while local adaptation techniques in the neighborhood of boundary features are developed in [4] and [15]. Sun et al. present in [14] an edge detection algorithm based on the concept of edge strength. This is used for surface segmentation as well as for adaptive surface smoothing in RE applications. Botsch et al. describe in [3] mesh adaptation to remove typical shaped triangles from STL models (called caps or needles, see
Fig. 2) for CAE applications.
The algorithm discussed in this paper will not change the topology of the basic triangular mesh. The mesh originates from a CAD model, so typical triangle shapes

will appear and this information can be used later on for the operations planning phase.

2. PROPERTIES OF THE ALGORITHM

The literature review in previous section illustrates the great variety of possible approaches to segment triangular meshes. This section explains the particularities of the developed algorithm.

The input of the algorithm is an STL file generated from a CAD model of the workpiece. Unlike many other segmentation methods, the algorithm should not modify the triangular mesh since it assumes that the CAD STL generator delivers an adequate model. The triangular mesh resulting from the STL file has its typical properties like highly varying triangle shapes and sizes (see Fig. 2). This irregular mesh is beneficial from the viewpoint of memory occupation and it delivers information on the underlying geometry structure. Flat or near flat regions are coarsely sampled with big triangles while highly curved regions are sampled with a lot of small triangles. Ruled surfaces are sampled with very sharp long triangles. So the variety in triangles represents the characteristics of the underlying geometry.

The core task of the algorithm is to segment the STL file into disjoint regions practically useful for multi-axis milling operations planning. By generation of the STL file, the boundaries between CAD features disappear. New segment boundaries have to be defined. This may look like a roundabout way but this is not the case. The algorithm will not try to find CAD feature boundaries, but it will look for boundaries of geometry regions that can be machined with a certain machining operation. A characteristic of complex shapes already mentioned is that machining regions and CAD entity boundaries seldom accord. In the current practice, the CAM programmer has to work around this mismatch by defining drive geometry on top of the design geometry during the definition of multi-axis milling operations. This means that for this segmentation algorithm, the boundary definition will not only rely on geometrical artifacts, but also on machining specific elements. The setup of the workpiece into the machine will influence the segmentation. This will be discussed in the detailed algorithm description.

Another important characteristic of the algorithm is that the segmentation does not result in only one exclusive solution for the workpiece STL model. Because of the application of multi-axis milling operations planning, a multitude of operation types are applicable for a certain workpiece. Each operation type may lead to another segmentation scheme. The algorithm needs to be able to calculate all those different segmentations. Because of that, it should work quite fast and concentrate on fast calculations rather than complex detailed mathematics.

3. DETAILED ALGORITHM DESCRIPTION

The segmentation algorithm for multi-axis machining operations planning is discussed in detail in this section. In the first subsection the data model being used is briefly introduced. Next the segmentation-processing algorithm is discussed. This processing contains two phases. In the first phase a preliminary segmentation is executed based on sharp boundary retrieval within the STL model. The second phase is the application specific phase where further segmentation of the model is based upon setup characteristics of the workpiece into the milling machine. Since during operations planning, multiple alternatives should be calculated and evaluated, this second phase has a dynamic behavior and is designed to quickly calculate different segmentations. Some examples conclude this section.

3.1 Data Model

The input of the algorithm is the STL file of the workpiece CAD model. This file actually contains only a collection of separate triangle descriptions (3 vertex coordinates and a facet normal vector for orientation purpose). On behalf of the segmentation algorithm an STL data model is built. An object-oriented design of the data model was selected because of the dynamic behavior of the system. The segmentation is not unique and alternative segmentations are needed for evaluation purposes during the multi-axis operations planning. An object oriented structure enables flexible iteration through the model and easy manipulation of several segments through referencing. Therefore, the system is implemented in JAVA using the VTK Visualization Toolkit (http://www.vtk.org) for the graphical inspection. Fig. 3 illustrates schematically the skeleton of the data structure model with its main concepts.

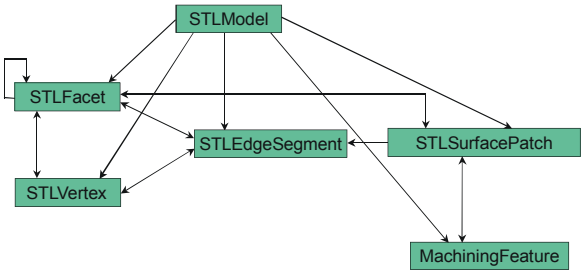


Fig. 3. Schematic data model diagram

The scope of this paper does not allow detailed discussion on the object structure of the data model itself.

3.2 First Phase: Sharp Boundary Segmentation

The first phase of the segmentation algorithm for an STL model is based on sharp boundary retrieval. A sharp

edge is a good first criterion for defining a multi-axis machining feature boundary. Sharp edge based segmentation methods are common in literature [12]. Such methods are based on the dihedral angle value or on the length of the difference vector between the normal vectors of neighboring facets. A common remark with such methods is the importance of the threshold value used for edge classification. This value is very case specific and highly influences the quality of the algorithm. In many applications, the user sets this threshold value or can manipulate the value interactively. This is not desirable for the application in automated operations planning. Manual interaction should be avoided as much as possible. Two methods to cope with this issue were investigated in this research work.

3.2.1 Edge Tracking Method

The first method is called edge tracking. This method detects edge segments within the STL triangles collection and attempts to combine these edge segments into a closed boundary delimiting a surface patch segment. A start threshold value is set (ex. $\sqrt{2}$). All edges of the model are tested if they are candidate segment edge i.e. the length of the difference vector between the facets normal vectors exceeds the threshold. These candidate edges are arranged into an edge segments graph. The STL vertices and the candidate edges define that graph. Segmentation is now done by finding cycles into the graph (DFS Depth First Search is used). When the edge segments graph still contains entries after cycles have been removed (i.e. according segments defined), the threshold is decremented and the procedure is repeated iteratively.

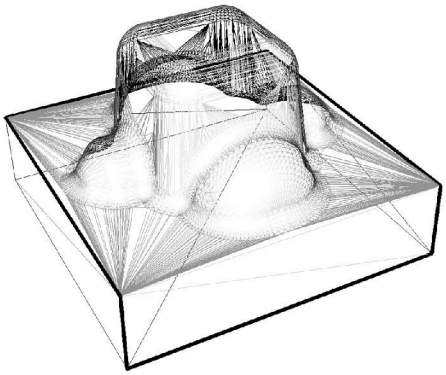


Fig. 4. Graph cycle that delimits an impractical surface patch.

While this algorithm solves the problem with fixed threshold value, experiments showed that the algorithm fails for some specific cases to isolate meaningful segments. The issue is that for some workpiece models the edge segments graph contains many possible cycles and that there is a high risk of selecting a cycle delimiting a segment that is impractical for machining operations. An example is given in Fig. 4 (edge depicted with a thick black line). The DFS cycle detection algorithm causes this malfunction. While branching the edge segments graph, the algorithm cannot take into account the topology of the resulting cycle. It is very hard to find a correct working mechanism to enhance the cycle detection algorithm to predict in what direction the branching should happen to extract correct boundary cycles. This is because an edge segment belongs to two surface patches and it is not known a priori which surface patch will result from the cycle tracking. The algorithm only works well if the number of multiple connected vertices (connectivity > 2) in the edge segments graph is strictly limited. This is the case for very complex sculptured surfaces. However industrial complex sculptured parts mostly contain also some regular geometry with multiple connected vertices. This restricts the usability of the described method. Instead of trying to patch the shortcomings of the edge tracking method, a second method is suggested.

3.2.2 Surface Flooding Method

The second method uses a different viewpoint than the first method. In the surface flooding method, a single triangle facet is considered as a seed for a segment. Neighboring facets are added to that surface patch if the difference vector of both normal vectors has a length below a threshold value. The problem of threshold adaptation is handled here totally different from the previous approach. There exists a risk that a facet just is not added to a patch because of minimal exceeding of the threshold. This kind of artifacts causes the generation of very small segments (typically containing only a few small facets). Also specific facet shapes (ex. needles and caps as mentioned in the introduction) can cause such surface patches.

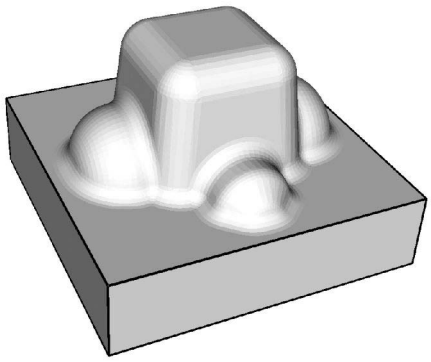


Fig. 5. Correct sharp edge detection.

Therefore the algorithm is extended with a recombinant phase, which filters out small surface patches. It does not make sense to identify small surface patches since they are useless from machining operation viewpoint. The facets of the filtered patches are added to the dominant neighboring surface patch. This algorithm now handles the same example as Fig. 4, and as Fig. 5 shows, the surface patch detection is correct. This latter approach seems to be quite robust to split up the STL product model into surface regions with sharp edges. Each facet of the model is guaranteed assigned to a surface patch by using this method and useless small surface patches are rearranged by the recombinant phase.
An interesting property of this method is that a triangle is added to the segment whenever it shares one of its edges below the threshold. This makes that a triangle being rejected during the flooding iteration can still be added in a later iteration step via an alternative edge.

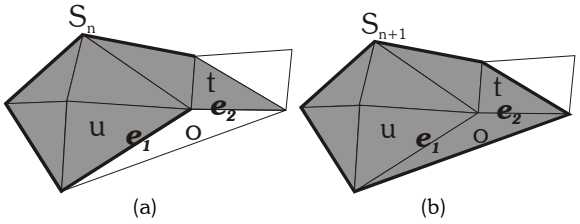


Fig. 6. Alternative triangle connection.

An example of this process is shown in Fig. 6. Let S_n be the segment at iteration step n of the surface-flooding algorithm. t is the triangle being processed during iteration step n. The gray area represents the triangles belonging to S_n. In Fig. 6(a) it is shown that triangle o is not added to the segment yet because during the processing of triangle u in an earlier step of the iteration the edge e_1 exceeded the threshold value. During the testing of the neighbor facets of triangle t, it is detected that the edge e_2 is below the threshold value, which makes that triangle o is added. In this way the spurious edge e_1 is annihilated and the segment S_{n+1} looks like Fig. 6(b) containing the triangle o.

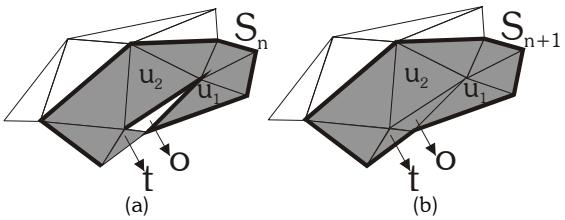


Fig. 7. Alternative triangle connection.

The same procedure can happen when a triangle was already rejected twice, while it is added during the testing of the third edge. This situation is pictured in Fig. 7.

Another interesting characteristic of this method compared to the edge tracking method is that each segment is bounded by a closed boundary. This property follows from the construction method of the segment.
One sees that the threshold value is no hard boundary in the surface-flooding algorithm. On the one hand the principle discussed in Fig. 6 and Fig. 7 can by-pass the threshold value. On the other hand too small isolated segments are attached to a neighbor by the recombinant phase. For the experiments and examples, the initial threshold is set to 0.4 and this value performs well for the models tested until present.

3.3 Second Phase: Machining Setup Based Segmentation

The second phase of the segmentation process is driven by the application of multi-axis machining operations planning. This phase is responsible for the dynamic behavior of the segmentation. The aim is twofold: further segmentation based on a particular setup of the workpiece into the milling machine environment, and second providing a mechanism to repeat this segmentation for different kinds of setups. In that way, multiple segmentation schemes can be elaborated so that operations planning can evaluate them and decide which setup to choose. To achieve these two targets, a flexible segmentation procedure is elaborated.
The concept of EGI (Extended Gaussian Image) is hereby used. Briefly stated, an EGI is a surface normal vector histogram on the Gauss unit sphere. This concept was introduced by Horn in [5]. An example is given in Fig. 8 where the EGI is 3D rendered and the histogram is colored. For algorithmic purposes, a corresponding Z-map representation of the EGI will be used.

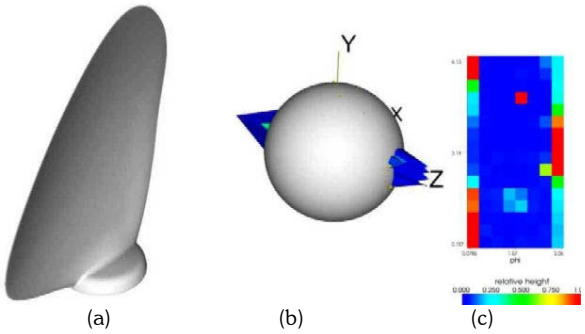
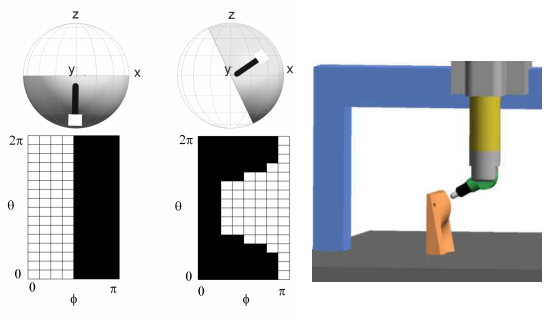


Fig. 8. Propeller blade model (a) with 3D rendered EGI (b) and Z-map representation EGI (c).

For each segment defined by the first phase of the algorithm, an EGI can be computed. Each STL triangle normal vector contributes to the discrete EGI.
The second phase will further split up the surface patch by pruning the EGI of that surface patch. This pruning happens by evaluating all triangles of a segment and test

if they pass through a filter. This filter characterizes the setup by indicating which machine directions are allowed and which are not. Two examples of such a filter are given in Fig. 9. The black cells indicate direction vectors that pass the filter while white cells indicate directions that are filtered out.



(a) (b) (c)

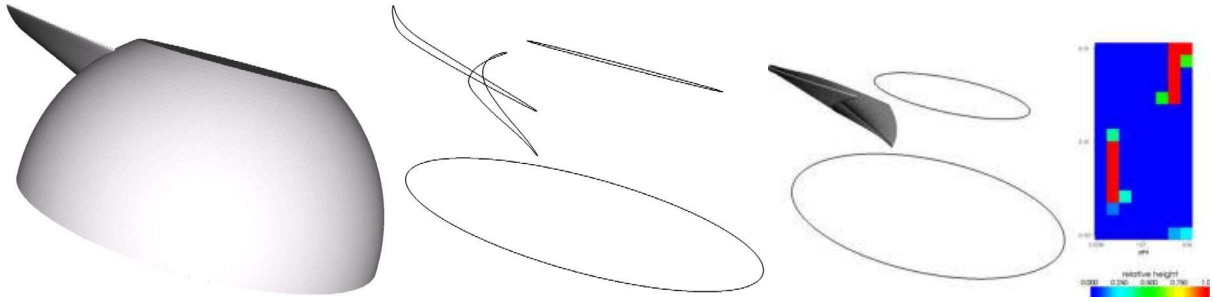
Fig. 9. Two sample filters (a) and (b). Machine setup for filter (b) pictured in (c).

The complete segmentation (both phase 1 and phase 2) will be illustrated by example in Fig. 10, which is a simplified version of the impeller model in order to keep the pictures clear. The input is the original STL file from the CAD model. This file is converted into a connected STL data model without edge information Fig. 10(a).

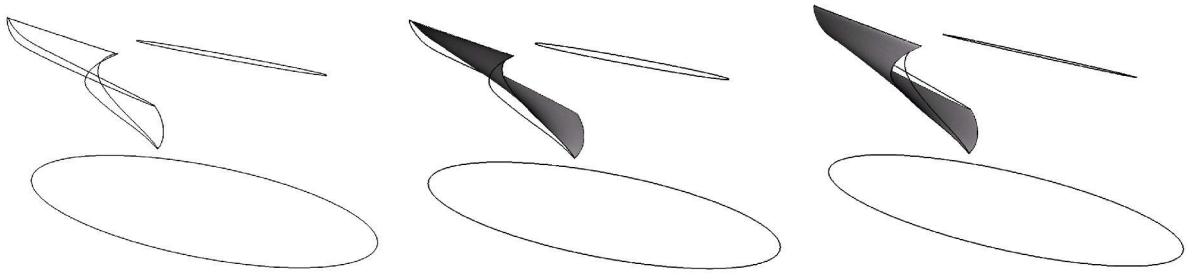
During the first phase of the segmentation the sharp boundaries of the model are retrieved. The skeleton of the sharp boundaries without the STL facets is visualized in Fig. 10(b). In Fig. 10(c) one particular segment (corresponding to an impeller blade) from the first phase segmentation is visualized. Also the Z-map representation of the EGI of the blade segment is shown in Fig. 10(c). Now the second phase of the algorithm is executed onto the blade profile segment. In this example the setup filter is set similarly to Fig. 9(a). Applying the pruning results in a further segmentation of the blade segment and the introduction of extra boundaries as shown in Fig. 10(d). The resulting segments are shown in Fig. 10(e) and Fig. 10(f).

The segmentation in the second phase is completely dependent on the pruning definition of the EGI, set by the chosen filter. After the pruning step is executed, the same risk exists as with the surface flooding in the first phase that some small triangles just fall outside the pruning boundary and leave practical useless segments behind. To cope this problem, the same strategy is used as in the first phase surface flooding. The recombinant step is reused and applied after the pruning step.

The dynamic behavior of this second step is already stressed several times. The concept of evaluating several segmentation schemes is illustrated by the next example. The original STL workpiece model is given in Fig. 11(a).



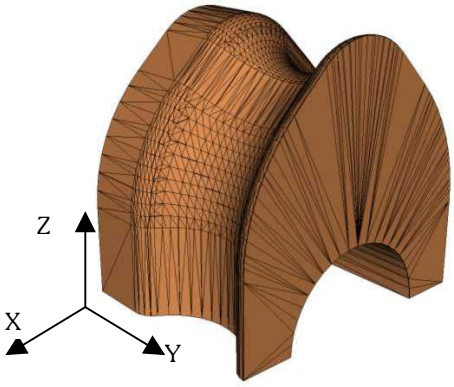
(a) STL data model. (b) First phase sharp edge detection. (c) EGI of a first phase segment.



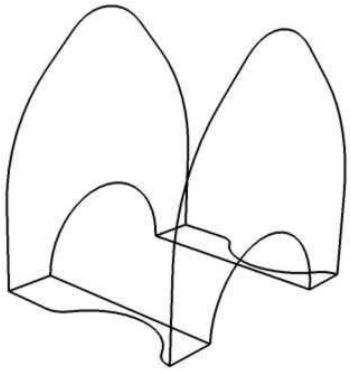
(d) Second phase filtered segment. (e) Resulting segment after second phase. (f) Resulting segment after second phase.

Fig. 10. Example of both first and second phase segmentation.

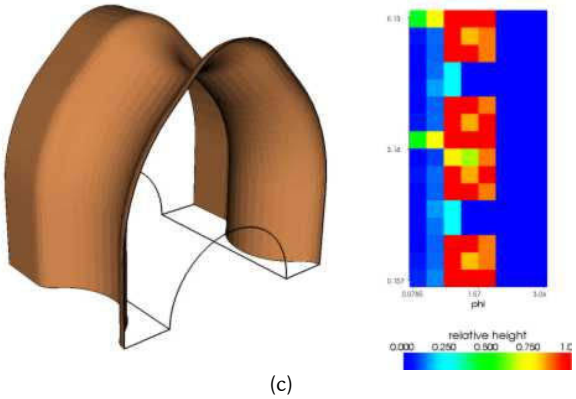
The first phase sharp edge retrieval results in the sharp edge boundary skeleton of the model. This skeleton is illustrated in Fig. 11(b).



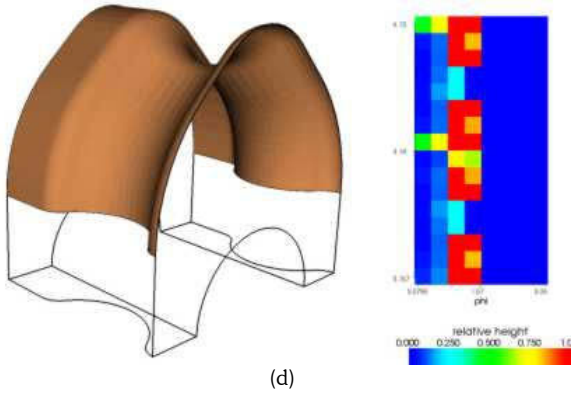
(a)



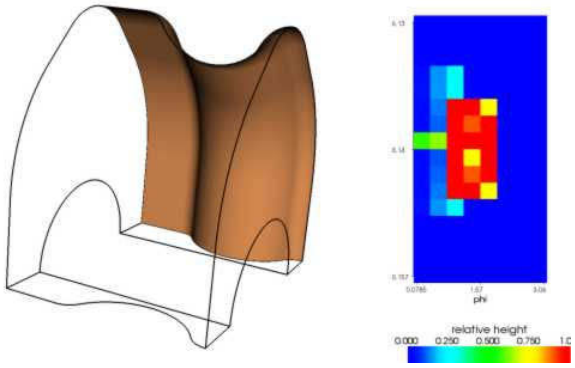
(b)



(c)



(d)



(e)

Fig. 11. Example of multiple setups segmentation.

For the second phase, the complex shaped feature in the middle of the part is further processed. The EGI of that segment is depicted in Fig. 11(c). Several setups for that segment can be considered. If a four-axis or five-axis setup is considered, the whole segment passes the setup filter, so that no further segmentation happens (also depicted in Fig. 11(c)). If a three-axis setup is considered along positive Z-direction, further segmentation happens. The resulting split-up into three segments and the filtered EGI are depicted in Fig. 11(d). Another possible three-axis setup along the negative X-direction is considered in Fig. 11(e) and results in a further segmentation into two segments.

One sees that the possibilities of setup-based segmentation quickly grow exponentially. It will be the responsibility of the operations planning algorithm to select some appropriate setups for input to the second phase segmentation.

The object-oriented data structure supports the multiple segmentation schemes. This is necessary to keep track and to evaluate the alternatives already segmented. By the concepts of STLSurfacePatch (see Fig. 3) and Machining Feature on top of the STL data model, the

original geometry is not altered and the different segmentation schemes can exist together concurrently.

4. CONCLUSIONS AND FUTURE WORK

In this paper, the development of a dedicated segmentation algorithm is described to split up an STL CAD model into surface patches for multi-axis milling operations planning. Particularities of the application area are used for the conception of the algorithm. A two-stage approach is deployed. First, sharp surface patch boundaries are detected. The second step further divides surface patches based on orientation information. That orientation information is captured in the EGI (Extended Gaussian Image). Segmentation happens by pruning that EGI. The pruning definition is characterized by the setup information of the workpiece into the milling machine tool. Multiple setup schemes can be processed which result in multiple segmentations of the same workpiece model. These multiple alternatives can be further processed by the operations planning system to effectively choose a suitable segmentation that will be used for multi-axis machining operations definitions.
Tests show the effectiveness of the algorithm, but it should be pointed out that the development was focused on applications in the domain of complex shaped CAD geometry to be machined by multi-axis milling. The algorithm concentrates on fast calculations and efficient processing in order to support its dynamic requirements to quickly evaluate multiple setup possibilities.

5. ACKNOWLEDGEMENTS

This research is funded by a specialization grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen).

6. REFERENCES

[1] Benko P., Martin R. R. and Varady T., Algorithms for reverse engineering boundary representation models, *Computer-Aided Design,* Vol. 33, No. 11, 2001, pp 839-851.

[2] Besl P. J. and Ramesh C. J., Segmentation Through Variable-Order Surface Fitting, *IEEE Trans Pattern Analysis and Machine Intelligence*, Vol. 10, No. 2, 1988, pp 167-192.

[3] Botsch M. and Kobbelt L., A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes, in *Proceedings of the Vision, Modelling & Visualisation Conference*, Stuttgart, Germany, 2001; VMV pp 283-289.

[4] Botsch M. and Kobbelt L., Resampling Feature and Blend Regions in Polygonal Meshes for Surface Anti-Aliasing, in *Proceedings of Eurographics*, 2001; 20(3).

[5] Horn B. K. P., Extended Gaussian images, *Proceedings of the IEEE*, Vol. 72, No. 12, 1984 pp 1671-1686.

[6] Jun C.-S., Kim D.-S. and Park S., Exact Polyhedral Machining, in *Proceedings SSM 1998*; pp 263-271.

[7] Lauwers B., Kiswanto G. and Kruth J.-P., Development of a Five-axis Milling Tool Path Generation Algorithm based on Faceted Models, *Annals of the CIRP*, Vol. 52, No. 1, 2003, pp 85-88.

[8] Lauwers B., Kruth J.-P. and Dejonghe P., An operation planning system for multi-axis milling of sculptured surfaces, *Int. J. Adv. Manufact. Technology,* Vol. 17, No. 11, 2001 pp 799-804.

[9] Lee A. W. F., Sweldens W., Schröder P., Cowsar L. and Dobkin D., MAPS: Multiresolution Adaptive Parameterization of Surfaces, *in Proceedings of SIGGRAPH Conference*, 1998; pp 95-104.

[10] Lefebvre P. P. and Lauwers B., Automated Part Geometry Analysis for Setup Selection in Multi-Axis Machining Operations Planning, *in Proceedings of the 36th CIRP International Seminar on Manufacturing Systems*, Saarbrücken, Germany, 2003; pp 293-298.

[11] Mangan A. and Witaker R., Partitioning 3D surface meshes using watershed segmentation. *IEEE Trans Visualization and Computer Graphics,* Vol. 5, No. 4, 1999.

[12] Razdan A. and Bae M., A hybrid approach to feature segmentation of triangle meshes, *Computer-Aided Design*, Vol. 35, No. 9, 2003, pp 783-789.

[13] Sapidis N. S. and Besl P. J., Direct Construction of Polynomial Surfaces from Dense Range Images through Region Growing, *ACM Trans on Graphics*, Vol. 14, No. 2, 1995, pp 171-200.

[14] Sun Y., Page D. L., Paik J. K., Koschan A. and Abidi M. A., Triangle mesh-based edge detection and its application to surface segmentation and adaptive surface smoothing, *in Proceedings of the IEEE International Conference of Image Processing*, 2002; III pp 825-828.

[15] Vorsatz J., Rössl C., Kobbelt L. P. and Seidel H.-P., Feature Sensitive Remeshing, *in Proceedings Eurographics*, 2001; Vol. 20, No. 3.