Triangulation of 3D Surfaces Recovered from STL Grids

D. Rypl, Z. Bittnar

In the present paper, an algorithm for the discretization of parametric 3D surfaces has been extended to the family of discrete surfaces represented by stereolithography (STL) grids. The STL file format, developed for the rapid prototyping industry, is an attractive alternative to surface representation in solid modeling. Initially, a boundary representation is constructed from the STL file using feature recognition. Then a smooth surface is recovered over the original STL grid using an interpolating subdivision procedure. Finally, the reconstructed surface is subjected to the triangulation accomplished using the advancing front technique operating directly on the surface. The capability of the proposed methodology is illustrated on an example.

Keywords: 3D surface, stereolithography format, interpolating subdivision, advancing front technique.

1 Introduction

In recent decades, the finite element method (FEM) has become the most powerful tool for structural analysis. Automatic generation of consistent, reproducible, high quality meshes without user intervention makes the power of the finite element analysis accessible to those not expert in the area of mesh generation. Therefore tools for automated and efficient mesh generation, including the discretization of 3D surfaces, are important prerequisites for the complete integration of the finite element method with design processes in CAD, CAE, and CAM systems.

An important class of 3D surfaces is the group of surfaces described by the stereolithography (STL) file format. This format approximates 3D surfaces of a solid model with oriented triangles (facets) of different size and shape (aspect ratio) in order to achieve a smooth enough representation suitable for industrial processing of 3D parts using stereolithography machines. However, such a representation is not appropriate for computational analysis using FEM. The aim of this paper is to extend a recently developed algorithm for discretization of discrete 3D surfaces [1] to the class of surface of which the geometry is described by discrete data in the STL format.

The actual discretization consists of several phases. Initially, a boundary representation of the entire model is constructed from the STL file using feature recognition based on appropriate topological and geometrical operations. In this way, distinct model entities (vertices, curves, and surfaces) of a topological nature (topological features) or with important geometrical aspects (sharp features) are established. In the current implementation, the geometrical operations are based on the dihedral and turning angle, and on the aspect ratio of two neighbouring facets. Note that the current implementation makes no attempt to detect the volumes. In the next phase, a smooth (limit) surface is recovered over the original STL grid. This is accomplished using the interpolating subdivision based on the modified Butterfly scheme, which yields C^1 surfaces (even in a topologically irregular setting). Similarly, the limit boundary curves are recovered using one-dimensional interpolating subdivision producing C^1 curves. In the last phase, the reconstructed limit surface is subjected to triangulation accomplished using the advancing front technique operating directly on the limit surface. This avoids difficulties with the construction of smooth parametrization of the whole surface.

The paper is organized as follows. In Section 2, the STL file format is described. Section 3 outlines the extraction of the boundary representation of the model. The reconstruction of a smooth 3D surface from the discrete STL data using the subdivision technique is recalled in Section 4. Section 5 briefly describes the actual mesh generation and presents the capabilities of the algorithm on an example. The paper ends with concluding remarks in Section 6.

2 STL File format

An STL file is a triangular representation of a 3D surface geometry. The surface is tessellated logically into a set of oriented triangles (facets). Each facet is described by the unit outward normal and three points listed in counterclockwise order representing the vertices of the triangle. While the aspect ratio and orientation of the individual facets is governed by the surface curvature, the size of the facets is driven by the tolerance controlling the quality of the surface representation in terms of the distance of the facets from the surface. The choice of the tolerance is strongly dependent on the target application of the produced STL file. In industrial processing, where stereolithography machines perform a computer controlled layer by layer laser curing of a photo-sensitive resin, the tolerance may be in the order of 0.1 mm to make the produced 3D part precise with highly worked out details. However, much larger values are typically used in pre-production STL prototypes, for example for visualization purposes.

The native STL format has to fulfill the following specifications: (i) The normal and each vertex of every facet are specified by three coordinates each, so there is a total of 12 numbers stored for each facet. (ii) Each facet is part of the boundary between the interior and the exterior of the object. The orientation of the facets (which way is "out" and which way is "in") is specified redundantly in two ways, which must be consistent. First, the direction of the normal is outward. Second, the vertices are listed in counterclockwise order when looking at the object from the outside (right-hand rule). (iii) Each triangle must share two vertices with each of its

adjacent triangles. This is known as the vertex-to-vertex rule. (iv) The object represented must be located in the all-positive octant (all vertex coordinates must be positive).

However, for non-native STL applications, the STL format can be generalized. The normal, if not specified (three zeros might be used instead), can be easily computed by the application from the coordinates of the vertices using the right-hand rule. Moreover, the vertices can be located in any octant. And, finally, the facet can even be on the interface between two objects (or two parts of the same object). This makes the generalized STL format suitable for modelling 3D non-manifold objects.

3 Extraction of boundary representation

Although an STL file represents a fully conforming grid, the construction of a boundary representation suitable for further processing is not trivial. First of all, the STL file has to be converted to a topologically more consistent form (called a background (bg) file hereafter) containing initially a list of vertices (called bg nodes hereafter), each one defined by an identification number (id) and three coordinates, and a list of facets (called bg faces hereafter), each one defined by id and three bg nodes ids. The background file is then extended by a list of bg edges, each defined by id and two bg nodes ids. Note that each bg edge must coincide with a side (called a bg side hereafter) of at least one bg face. Firstly, bg edges corresponding to topological features are detected. These are all those bg sides that are not shared exactly by two bg faces. Then bg edges corresponding to geometrical (sharp) features are identified. Three criteria are used in the following order. The first criterion is based on finding all neighbouring bg faces that form a continuous plane. Then those bg sides that are shared only by one bg face corresponding to a particular plane formed by at least three bg faces is marked as a bg edge. The second criterion whether a bg side forms a sharp feature uses the dihedral angle (based on the angle of normals) between the two neighbouring bg faces sharing that bg side. Should this angle exceed a user specified threshold the bg side is marked as a bg edge. Taking into account that the aspect ratio of bg faces corresponds to the curvature of the surface, the third criterion is based on the ratio of heights of two neighbouring bg faces with respect to the shared bg side. Should this ratio exceed a user prescribed threshold and the normals of those bg faces are not same and do not exceed the angle threshold, then the bg side is marked as a bg edge. When all bg edges are identified, the bg nodes corresponding to topological features are detected and classified as a model vertex. These are all bg nodes that are not shared exactly by two bg edges. Then the bg nodes corresponding to geometrical features are searched for. Again three criteria are used. The first criterion is based on finding all neighbouring bg edges that form a continuous straight line. Then those bg nodes that are shared only by one bg edge corresponding to a particular line formed by at least two bg edges is classified as a model vertex. The second criterion classifies as a model vertex all those bg nodes that are shared by two bg edges at a turning angle above a user prescribed value. And thirdly, if the ratio of the lengths of two neighbouring non-colinear bg edges exceeds a user specified value, the bg node shared by those bg edges is also classified as a model vertex. Note that the current implementation makes no attempt to detect a sharp vertex not connected to any bg edge (e.g. the tip of a cone). Note also that each model vertex keeps a list of all bg edges connected to it. Once the model vertices are identified, model curves can be determined by traversing the chains of connected bg edges from a starting vertex until the ending vertex is reached. Every visited bg edge and its not yet classified end nodes are classified to the corresponding model curve. Loops of not visited bg edges (there is no model vertex on any of those bg edges) and their end nodes are also classified to a particular model curve. Finally, the model surfaces are identified. This is simply accomplished by assembling all neighbouring bg faces that do not share the same bg edge. Each face and its not yet classified corner nodes are classified to the corresponding model surface. Since the border of each model surface is formed by bg edges, which are in turn classified to model curves, it is easy to set up for each surface the list of boundary curves. This makes the extraction of the boundary representation complete.

The problem of the above concept consists in the fact that it gives no guide how to choose the individual thresholds. This is the consequence of the fact that the STL file does not possess information about the original tolerance used to generate it, and that the identification of the original geometry from the STL file is generally not unique. In other words, there may exist several geometries that are represented by the same STL file generated for the same tolerance. A reasonable strategy to tackle this problem is to use an iterative approach in which the algorithm accepts the already identified features from previous iterations. Initially, conservative values of thresholds, which yield only really "sharp" features, are chosen to produce the first boundary representation of the object. Alternatively, no values may be specified at all, resulting in boundary representation based solely on topological features. Next, suitable values are interactively specified for individual entities of the model to further define the boundary representation. However, even such an approach can fail to detect some significant features of the object (or can detect them only at the cost of detecting simultaneously some undesirable ones). Therefore the identification procedure cannot rely only on the threshold values and must also accept an interactive input of manually selected (or deselected) bg edges. This, however, makes the process of extracting the boundary representation tedious.

4 Reconstruction of a limit surface

The smooth surface over the original STL triangulation is reconstructed using a suitable subdivision technique based on the hierarchical recursive refinement of triangular simplices forming the STL mesh (Fig. 1). Each step of the refinement consists of two stages – splitting and averaging (Fig. 2). In the splitting stage, new bg nodes are introduced exactly in the middle of individual bg edges. During the averaging, the bg nodes are repositioned to a new location evaluated as a weighted average of bg nodes in the neighbourhood (according to the so called averaging mask). As the level of refinement grows, the resulting grid approaches the so called limit surface.

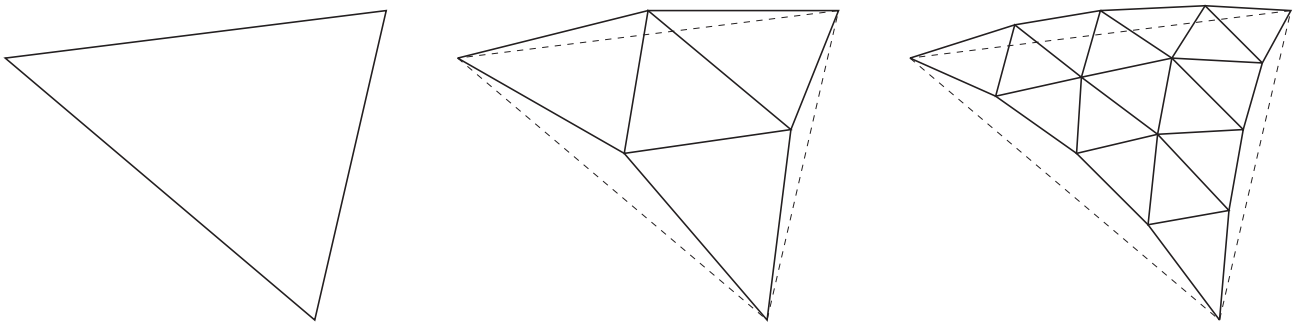


Fig. 1: Two levels of hierarchical refinement of a bg face

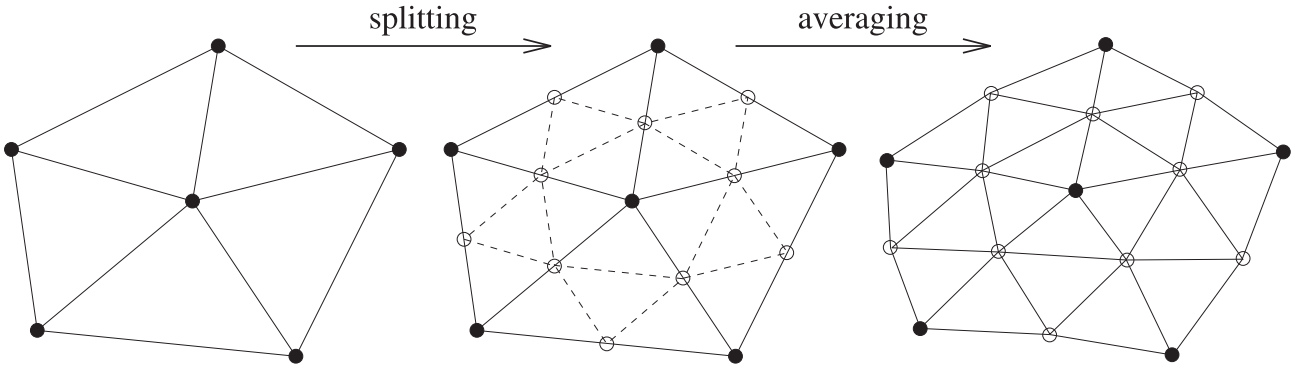


Fig. 2: Two stages of refinement – splitting and averaging

In the presented implementation, the recursive subdivision based on the modified Butterfly scheme [2] has been employed. It is the interpolating non-uniform stationary scheme, in which the existing bg nodes (on the current level of refinement) remain unchanged and the position of a new bg node S on the next level (Fig. 3a) is calculated as

$$S = w_R + \sum_{i=1}^{n} w_i P_i, \tag{1}$$

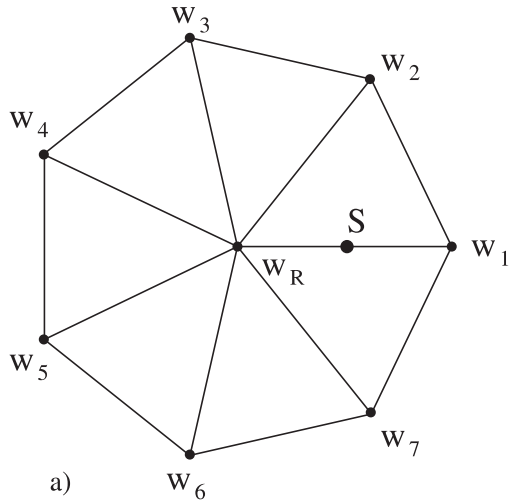
where P_i are bg nodes connected to the surface bg node R of valence n. The weights w_R and w_i corresponding to the surface averaging mask (Fig. 3a) are given by

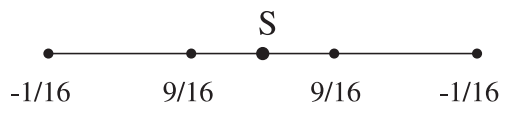
$$w_R = 3/4, \tag{2}$$

$$w_i = \frac{1}{n}\left(\frac{1}{4} + \cos\frac{2\pi(i-1)}{n} + \frac{1}{2}\cos\frac{4\pi(i-1)}{n}\right), \text{ for } n > 4, \tag{3}$$

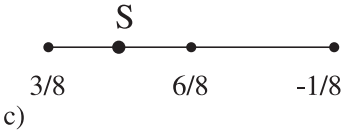
$$w_1 = 3/8,\ w_2 = 0,\ w_3 = -1/8,\ w_4 = 0, \text{ for } n = 4, \tag{4}$$

$$w_1 = 5/12,\ w_2 = -1/12,\ w_3 = -1/12, \text{ for } n = 3. \tag{5}$$





b)



c)

Fig. 3: Averaging masks: a) surface mask ($n = 7$), b) 4-point curve mask, c) 3-point curve mask

The modified Butterfly scheme exhibits favourable properties:

- generality – it works with a control grid of any topology,
- smoothness – it yields C^1 continuous limit surface,
- locality – it uses only a one-level neighbourhood and
- simplicity – it ensures easy and efficient evaluation.

Similarly, the limit boundary curves are recovered using a one-dimensional interpolating subdivision [3] producing C^1 continuous curves. The adopted 4-point (for a new bg node between two curve bg nodes) and 3-point (for a new bg node between a vertex bg node and a curve bg node) averaging masks are depicted in Fig. 3b and 3c.

The final interpolating procedure evaluates the position of a new bg node according to the classification and regularity of the end bg nodes of its parent bg edge (a surface bg node of valence 6 is called regular, otherwise it is called irregular):

1. for every surface bg edge bounded by an irregular surface bg node and a regular surface bg node, compute the bg midnode position using the surface averaging mask with respect to the irregular surface bg node,
2. for every surface bg edge bounded by two irregular or regular surface bg nodes, compute the bg midnode position using the surface averaging mask with respect to both end bg nodes and take the average,
3. for every surface bg edge bounded by a surface bg node and a non-surface bg node, compute the bg midnode position using the surface averaging mask with respect to the surface bg node,
4. for every surface bg edge bounded by two non-surface bg nodes, compute the bg midnode position as the average of the positions of the end bg nodes,
5. for every curve bg edge bounded by two curve bg nodes, compute the bg midnode position using the 4-point curve averaging mask,
6. for every curve bg edge bounded by a curve bg node and a vertex bg node, compute the bg midnode position using the 3-point curve averaging mask,
7. for every curve bg edge bounded by two vertex bg nodes, compute the bg midnode position as the average of the positions of these vertices.

An example of the application of the modified Butterfly scheme to a simple domain is depicted in Fig. 4. The control grid is derived from 24 triangular facets covering a regular 12-sided polygon by shifting two interior nodes (opposite

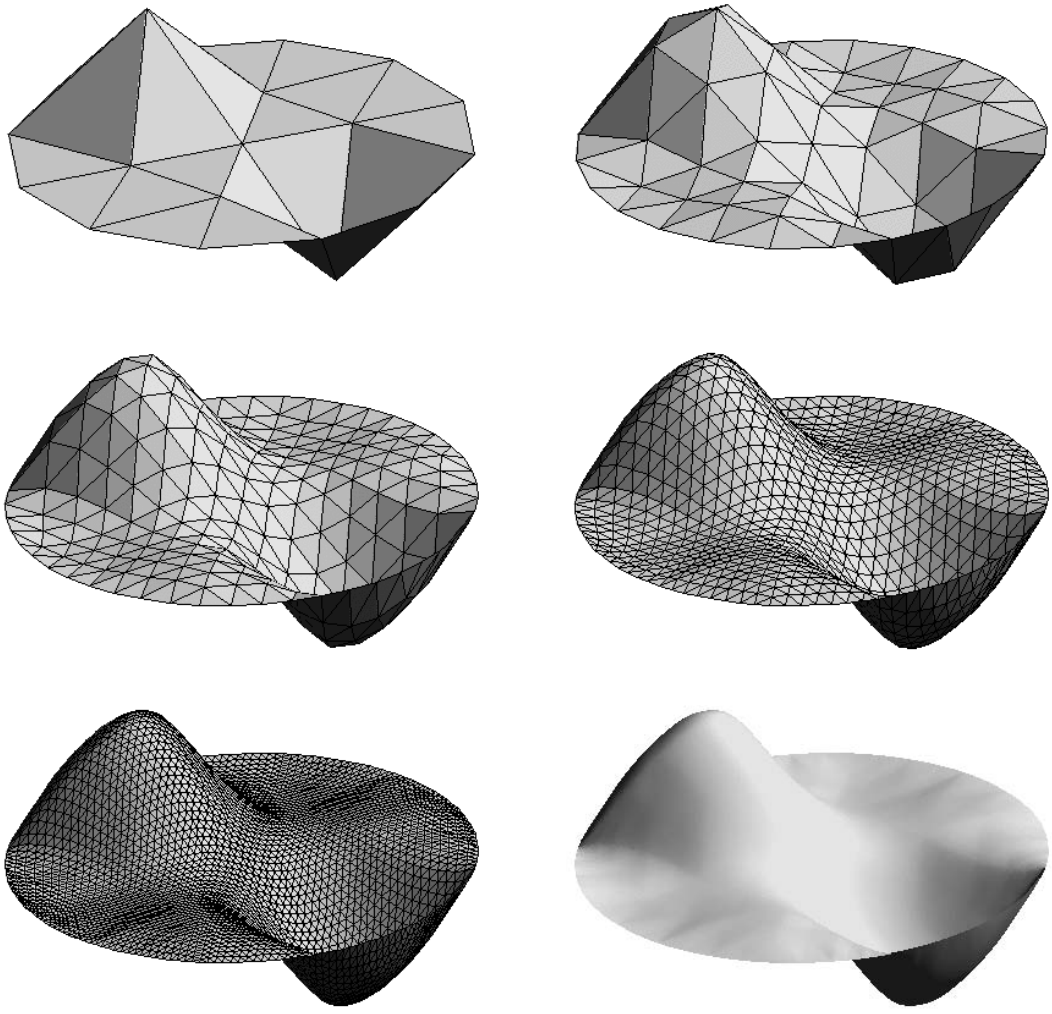


Fig. 4: Application of the modified Butterfly scheme to a simple domain: original control grid (top left), four refinement levels (top right, middle row, and bottom left), and the limit surface (bottom right)

with respect to the polygon center) out of the plane of the polygon (each one in the other direction).

The properties of the limit surface (and also the curve) can been deduced by a standard examination of the eigenstructure of the local subdivision matrix corresponding to the adopted subdivision scheme [4]. This includes especially the derivation of the so called derivative masks used for the calculation of vectors tangent to the limit surface at the limit position of a bg node of the control grid. These tangent vectors are then in turn used for evaluating the limit surface normal.

An important aspect is that the above seven rules for evaluating of the limit position of new bg nodes were derived for more or less isotropic control grids, i.e., for control grids with elements of approximately unit aspect ratio. However, STL meshes typically consist of elements of very large aspect ratio, which is the consequence of the curvature based tessellation procedure used to generate them. In this context, the 4th rule for the subdivision of a surface bg edge bounded by non-surface bg nodes was found too restrictive (rigid) and was replaced by the following rule:

4. for every surface bg edge bounded by two non-surface bg nodes, find the off-diagonal bg nodes of the quadrilateral formed by two bg faces sharing that bg edge and compute the bg midnode position
   (a) according to appropriate from rules 1 - 3 using the temporarily swapped bg edge, if any of the off-diagonal bg nodes is a surface bg node,
   (b) as the weighted average of positions calculated according to rule 4 for the original (weight 0.75) and temporarily swapped bg edge (weight 0.25), if both off-diagonal nodes are non-surface bg nodes.

It should be emphasized that the topological change to the STL mesh by swapping the bg edge should be of temporary character only (just for the evaluation of the position of the corresponding bg midnode). Otherwise the change of connectivity (in the case of STL meshes often regular) induced by the swapping may seriously deteriorate the quality of the recovered limit surface.

5 Mesh generation

Having obtainedthe limit surface, the goal is now to generate a new triangular mesh over it, respecting a given mesh density distribution (typically prescribed at the nodes of the STL mesh serving as the control grid and/or curvature based) that makes the mesh suitable for subsequent computational analysis. The discretization is carried out in a hierarchical manner. Firstly, the model vertices are discretized. Then the (limit) curves are segmented using the mass curve of the required element density along that curve. And finally, the individual (limit) surfaces are triangulated.

In order to control the element size distribution, an octree is built around the domain to be discretized. The size of the individual octants corresponds approximately to the required element spacing, while the nodes (corners) of the octants are storing the required spacing exactly. To ensure the gradual variation of the element size, the maximum one-level octree difference of octants sharing an edge is enforced. This will guarantee the creation of well shaped triangles. During the actual mesh generation the required element size is extracted from the octree for a given location using the interpolation of octree nodal values of the element size.

In the presented implementation, the surfaces are discretized by the advancing front technique constrained directly to the surface and modified to reflect surface curvature [5]. Firstly, the initial front consisting of mesh edges at boundary curves of the surface (including inner loops) is established. Once the initial front has been set up, mesh generation continues on the basis of an edge removal algorithm according to the following steps until the front becomes empty:

- the first available edge AB is selected from the front,
- the position of the "ideal" point P (forming the new ΔABC) is calculated taking into account the local curvature of the limit surface and element size variation,
- the projection P' of point P to the limit surface is evaluated,
- the local neighbourhood of point P' is established (in terms of a set of octants),
- the neighbourhood is searched for the most suitable candidate C to form a new ΔABC,
- the intersection check is carried out to avoid overlapping of the candidate triangle with an already existing one in the neighbourhood,
- the front is updated to account for the newly formed ΔABC.

The generated mesh is then subjected to an optimization in order to improve the quality of the final mesh. The Laplacian smoothing technique in combination with topological transformations (diagonal edge swapping) is adopted. This yields the optimized grid after only a few cycles of smoothing (typically up to six). Note that, unlike to the smoothing carried out in 2D, the repositioning of a node during the smoothing is likely to shift the node out of the surface. Therefore, the node has to be projected back to the surface to satisfy the surface constraint.

A crucial aspect of the proposed mesh generation strategy is related to the point-to-surface projection. Simple and efficient algorithms available for the projection to parametric surfaces cannot be adopted, simply because parametrization of the limit surface is missing. The situation is further complicated by the fact that the normal to the limit surface can be evaluated only at the bg nodes of the original or refined control grid. Therefore in order to make the projection sufficiently accurate (in terms of the distance from the limit surface and match with the exact normal), it is necessary to subdivide the original control grid up to a high level. This results in a huge amount of data to be stored, which is not acceptable. In [6], an efficient and reliable approach for projecting of a point to the limit surface has been proposed. This approach is based on localized progressive refinement of the control grid towards the actual projection. Recursive implementation of this algorithm enables virtually an unlimited number of refinements with constant memory requirements to be performed. Note that the refinement is of a temporary character, and it is discarded after the projection is completed. Since some of the projections during mesh generation can be accomplished with considerably lower accuracy (without any significant impact on the resulting mesh), an alternative approximate, but more efficient projection tech-

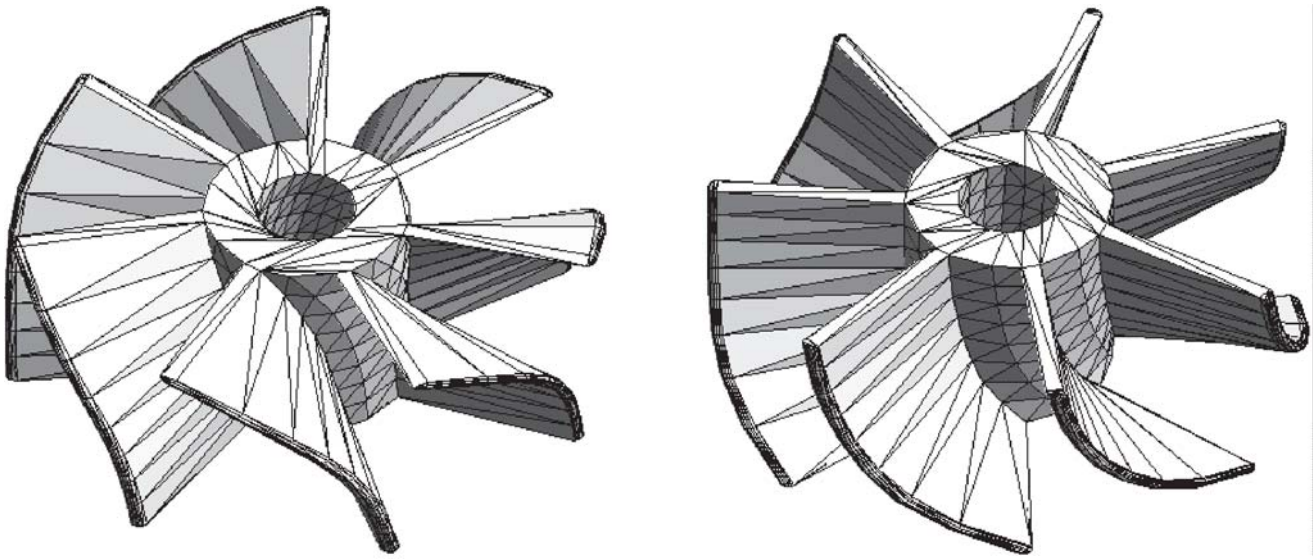


Fig. 5: STL mesh of a propeller

nique was suggested. This is based on approximating of bg faces of the control grid by quadratic Bezier triangular patches and on employing of a standard projection technique applicable to parametric surfaces (see [6] for details).

With respect to the application of the above meshing technique to the limit surfaces reconstructed from the STL meshes, it should be noted that planar surfaces (seemingly the simplest case) must be handled in a special way. The reason is that a planar surface, having zero curvature in all directions, prevents the use of a curvature based mechanism to tessellate it into STL bg faces. Therefore the aspect ratio and orientation (in plane) of the individual bg faces cannot be related to the curvature. This allows two neighbouring bg faces forming the same planar surface to considerably differ in the aspect ratio measured with respect to the shared bg side. As a direct consequence, the limit surface folds over itself, possibly crossing the boundary of the surface. Even though such a surface still seems to be planar, it is not any more, since the normal changes orientation from point to point, which is fatal for the meshing algorithm. Therefore, when triangulating a planar surface, the subdivision is not invoked and the bg faces of the control grid are used for localization purposes only.

The proposed methodology for the discretization of surfaces described by STL meshes is demonstrated on the example of a propeller. The original STL mesh is depicted in Fig. 5. The triangulation with curvature based element size control is presented in Fig. 6. Although the original STL representation is rather coarse, the final mesh captures the shape of the propeller well.

6 Conclusions

This paper has introduced an approach for the direct triangulation of 3D surfaces described by STL meshes. Although the STL mesh is a valid fully conforming triangulation, its special designation for rapid prototyping makes it very specific. The actual discretization consists of several phases. Firstly, a boundary representation of the object is

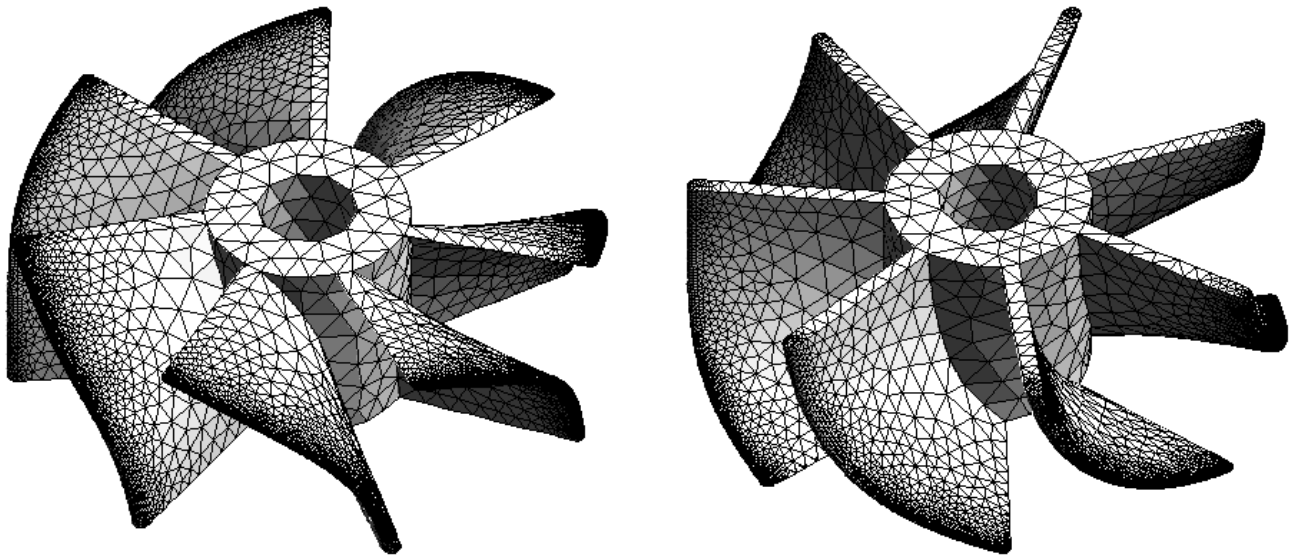


Fig. 6: Graded mesh of a propeller

constructed from the STL file using a feature recognition approach. It has been shown that this is an ambiguous task which cannot generally be fully automated. A successful completion of this procedure often requires user intervention in the framework of an interactive environment. In the next phase, a smooth (limit) surface is reconstructed over the original STL mesh using a subdivision technique yielding the differentiable C_1 limit surface. An interpolating subdivision based on the slightly modified Butterfly scheme has been adopted. The modifications make the limit surface smoother in situations where the original strategy seems to be insufficiently flexible, which is often the case for STL meshes containing elements of large aspect ratio spanning the whole extent of the surface in a particular direction. Finally, the limit surface is subjected to a triangulation based on the advancing front technique constrained directly to the limit surface. The vitality of the proposed approach has been demonstrated on some examples. Further research focus on primarily on additional improvement of the construction of the boundary representation in order to enable as much as possible automated processing of complex STL models.

Acknowledgment

This work was supported by Ministry of Education of Czech Republic project No. MSM 210000003.

References

[1] Rypl D., Bittnar Z.: "Triangulation of 3D Surfaces: From Parametric to Discrete Surfaces." CD-ROM Proceedings of the Sixth International Conference on Engineering Computational Technology, Civil-Comp Press, 2002.

[2] Zorin D., Schröder P., Sweldens W.: "Interpolating Subdivision for Meshes with Arbitrary Topology." Computer Graphics Proceedings (SIGGRAPH '96), 1996, p. 189–192.

[3] Dyn N., Gregory J. A., Levin A.: "A Four-Point Interpolatory Subdivision Scheme for Curve Design." *Computer Aided Geometric Design*, 1987, p. 257–268.

[4] Halstead M., Kass M., De Rose T.: "Efficient, Fair Interpolation using Catmull-Clark Surfaces." Computer Graphics Proceedings (SIGGRAPH '93), (1993), p. 35–44.

[5] Bittnar Z., Rypl D.: "Direct Triangulation of 3D Surfaces Using the Advancing Front Technique." Numerical Methods in Engineering '96 (ECCOMAS '96), Wiley & Sons, 1996, p. 86–99.

[6] Rypl D., Bittnar Z.: "Triangulation of 3D Surfaces Reconstructed by Interpolating Subdivision." to appear in *Computers and Structures.*

Doc. Dr. Ing. Daniel Rypl
phone: +420 224354369
Fax: +420 224 310 775
e-mail: drypl@fsv.cvut.cz

Prof. Ing. Zdeněk Bittnar, DrSc.
phone: +420 224 353 869
Fax: +420 224 310 775
e-mail: bittnar@fsv.cvut.cz

Department of Structural Mechanics

Czech Technical University in Prague
Faculty of Civil Engineering
Thákurova 7
166 29 Prague 6, Czech Republic

