INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2002; **39**:75–96 (DOI: 10.1002/fld.281)

Surface triangulation for polygonal models based on CAD data

Yasushi Ito*,† and Kazuhiro Nakahashi

Department of Aeronautics and Space Engineering, Tohoku University, Aoba-yama 01, Sendai 980-8579, Japan

SUMMARY

This paper presents an approach to the generation of unstructured surface meshes for Computer-Aided Design (CAD) surface models represented as lists of polygons with minimum user interventions. Stereolithography (STL) data are adopted as an interface between a CAD system and the surface grid generator. STL files often include problems such as overlapping surfaces, gaps, and intersections. They have to be revised quickly and automatically before the surface models are used for the background grid of the surface grid generation. In this paper, we describe an automatic revision method for use with STL-defined surface models. The advancing front method using geometric features is adopted directly on the modified STL surfaces. The capability of the method is demonstrated for several 3D surface models. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: surface mesh generation; CFD; CAD; stereolithography (STL)

1. INTRODUCTION

Computer-Aided Design (CAD) systems have become indispensable for Computational Fluid Dynamics (CFD) grid generation to obtain high quality meshes and to reduce user interventions. CAD systems support many types of output formats, one of the most common being the stereolithography (STL) format, which has become the *de facto* standard in rapid prototyping [1]. Because it is based on pure geometric information of a model, it is very simple and easy to manipulate the defined geometry. Thus, the STL format is one of the promising candidates for use in CFD grid generation.

A surface meshing method based on the direct advancing front method [2] has been developed [3] using STL data as a background grid. The key point of this approach is the use of geometric features, such as ridges and corner curves extracted from a given geometry, as the initial fronts for the advancing front method. By use of the geometric features, the original configuration is precisely preserved and the surface meshing procedure is significantly

*Correspondence to: Y. Ito, Nakahashi Lab, Department of Aeronautics and Space Engineering, Tohoku University, Aoba-yama 01, Sendai 980-8579, Japan.
†E-mail: itoh@ad.mech.tohoku.ac.jp

Received June 2001
Revised October 2001

simplified. The advancing front method is directly applied to the 3D surfaces. This method, however, requires continuous surfaces for the background grid.

If a CAD surface model is obtained as the Non-Uniform Rational B-Splines (NURBS) model, smooth STL surface data can be created. However, there are many cases in which we can only obtain polygon models such as the Drawing Interchange Format (DXF) file [4], the 3D Studio file (3DS) and the PLOT3D file for structured surface grids [5]. In these cases, surfaces with intersections, gaps, and so on often exist, which must be revised beforehand. CAD systems are not good for such operations in the case of polygon surface models. Because polygons can be divided into triangular facets, this paper only focuses on triangulated surface models.

Two approaches can be considered for removing intersections: (a) re-triangulation near the intersections after separate surface meshing for each part and (b) conversion of intersecting surfaces into continuous surfaces before the surface meshing. The former is described in References [6] and [7]. This approach is effective for simple geometries such as spheres, cylinders and cubes. For complex geometries, however, this approach has limitations because it is difficult to adjust the sizes of triangles of each part near the intersections. In the latter approach, the intersecting faces are eliminated beforehand in order to obtain continuous surfaces. If the continuous surfaces represent the model correctly, a surface mesh of good quality can be generated.

If a background grid has gaps, such as holes and cracks, on the defined surface, then there are potential difficulties in constructing valid surface meshes. Therefore, they are also revised beforehand.

This paper presents a surface triangulation method for STL-defined surfaces. We focus on automated revision of triangulated surface models. They are converted to continuous surfaces before the surface meshing method is applied. The modified surface is used as a background grid of the surface meshing. Several results of the grid generation and CFD computation are shown to demonstrate the capability.

2. OUTLINE OF GRID GENERATION

Figure 1 shows the outline of the grid generation process. First, a surface model should be prepared with CAD [8]. Details of the process of the surface triangulation have been discussed in Reference [3]. The method used for the input and that for the revision of STL surfaces are discussed in the following chapters.

3. STL DATA INPUT

STL files are adopted as an interface between a CAD system and the surface grid generator. The STL format contains a list of triangular facets, each facet being described by coordinate values of its three nodes — they are stored in counter-clockwise order when the facet is looked down on from outside the object — and a normal direction pointing towards the exterior of the model (see Figure 2). Based on these characteristics, coordinate values of nodes are listed duplicately. The same nodes should be identified as quickly as possible.

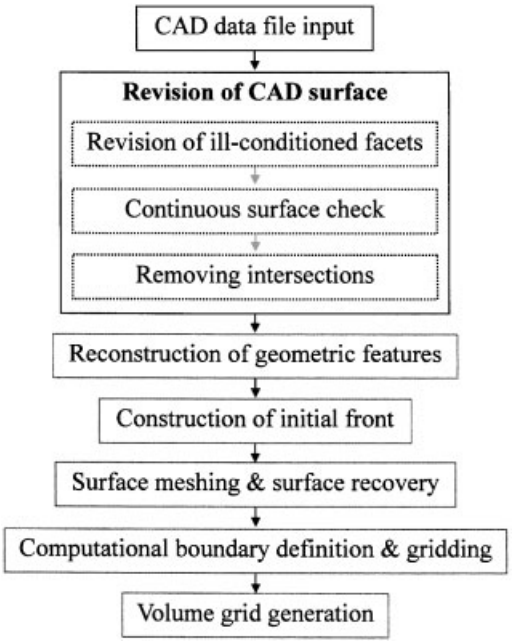


Figure 1. Outline of grid generation process.

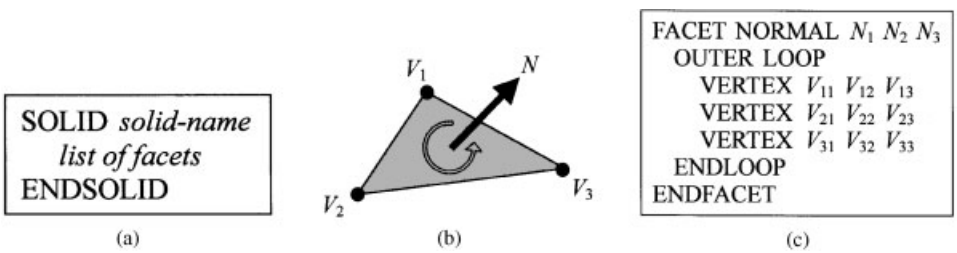


Figure 2. ASCII STL file format: (a) overview; (b) a facet, which is made of a normal vector $N=(N_1,N_2,N_3)$ and three vertices: $V_i=(V_{i1},V_{i2},V_{i3})$, $(i=1,2,3)$; (c) format for each facet.

The same node is defined as follows:

$$|\mathbf{x}_a-\mathbf{x}_b|\leqslant\varepsilon \tag{1}$$

where $\mathbf{x}_a$, $\mathbf{x}_b$ denote position vectors of two arbitrary nodes in an STL file, and ε denotes a small value. If the comparison is carried out through the positions of all the nodes, excessive CPU time is required as

$$T\propto(3m)^2,\quad 3m\geqslant n \tag{2}$$

where T denotes the required CPU time for the comparison, and m,n denote the number of facets and the number of actual nodes, respectively. n is approximately equal to $m/2$ if the model has no boundaries. $3m$ represents the number of nodes in the STL file.

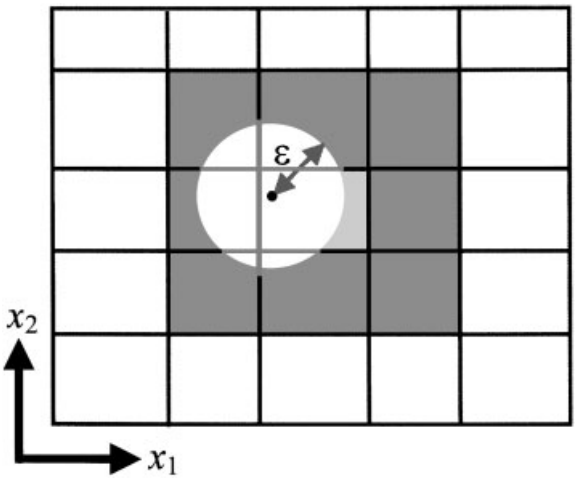


Figure 3. Node identification: in 2D, we only check the nodes in the nine boxes that surround the selected node.

To avoid wasting CPU time, the 3D space occupied by the input model is divided into a number of boxes before the comparison. The method of division in each direction (x_1, x_2, and x_3 in 3D) is as follows:

(1) The number of divisions n_{di} ($i = 1, 2, 3$ in 3D) is specified. n_{di} should be

$$n_{di} \leqslant \frac{l_i}{\max(\varepsilon, 10^{-12})} + 1 \tag{3}$$

where l_i denotes the length of the model in the x_i-direction. Coordinate values of the nodes are sorted in descending order.

(2) The nodes are divided into n_{di} groups by using the sorted list. Each group has about $3\,m/n_{di}$ nodes with the difference between the maximum and minimum coordinate values needing to be greater than $\max(\varepsilon,\ 10^{-12})$.

The comparison is then carried out. If one node is selected, we only compare the nodes in the 27 boxes that surround the selected node in 3D. Figure 3 demonstrates this procedure in 2D. The CPU time is greatly reduced if n_{di} is sufficiently large, for example, 40. Dynamic memory allocation allows this algorithm to be realized without unnecessary memory usage.

4. REVISION OF STL DATA

Before the surface grid generation method is applied, we have to revise the STL surface models automatically in order to use them as background grids. As shown in Figure 1, these operations consist of three parts: (a) revision of ill-conditioned facets, (b) continuous surface check, and (c) intersection removal. The former two operations are necessary due to errors

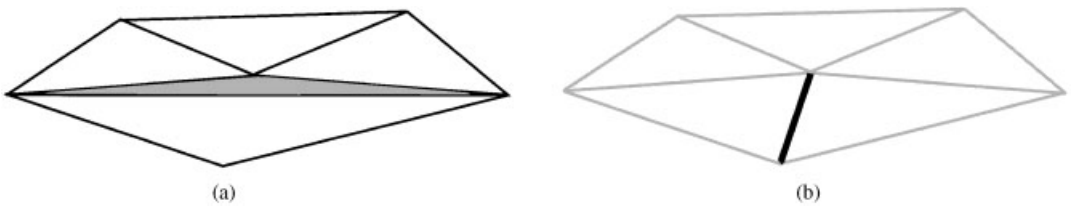


Figure 4. Isolated type of high aspect ratio facet: (a) model case; (b) correction of (a).

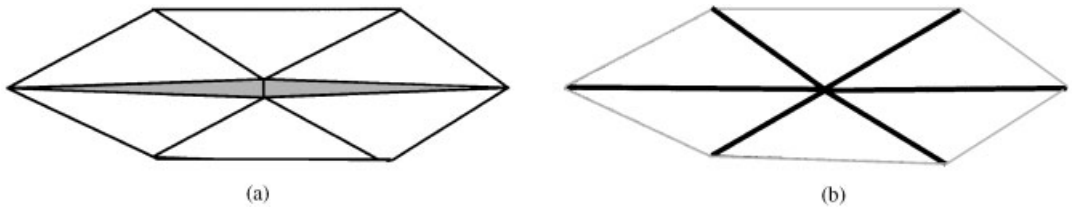


Figure 5. Connecting type of high aspect ratio facets: (a) model case; (b) correction of (a).

in the tessellation process in a CAD system because the STL file format does not provide for consistency and completeness tests [9]. The latter one is a weak point of CAD systems.

4.1. Revision of ill-conditioned facets

STL data often contain two types of ill-conditioned facets: high aspect ratio ones including zero-area ones, and sticking ones. These should be automatically revised beforehand or else their normal vectors cannot be estimated properly. The following revisions do not reduce the accuracy of the modeling because the ill-conditioned facets have almost no areas.

4.1.1. High aspect ratio facets. The high aspect ratio facets as shown in Figure 4 are detected and corrected as follows:

(1) Calculate lengths of all the edges.
(2) At each triangular facet, the following reference value, ξ, is used for the detection.

$$\xi = \frac{l_2 + l_3}{l_1 \cos \alpha} \tag{4}$$

where l_1, l_2, l_3 $(l_1 \geqslant l_2 \geqslant l_3)$ are lengths of three edges of the facet, and α is a small angle such as 1.5 degrees. If ξ is less than unity, the facet is considered as a high aspect ratio one.

(3) If $l_3 > 0.05 l_1$, the facet is an isolated one as shown in Figure 4(a). This facet is corrected by swapping diagonals with the neighboring triangular facet of the longest edge, as shown in Figure 4(b).
(4) If $l_3 \leqslant 0.05 l_1$, the facet is one of a pair of connecting ones as shown in Figure 5(a), and is removed together with its neighbor as shown in Figure 5(b).

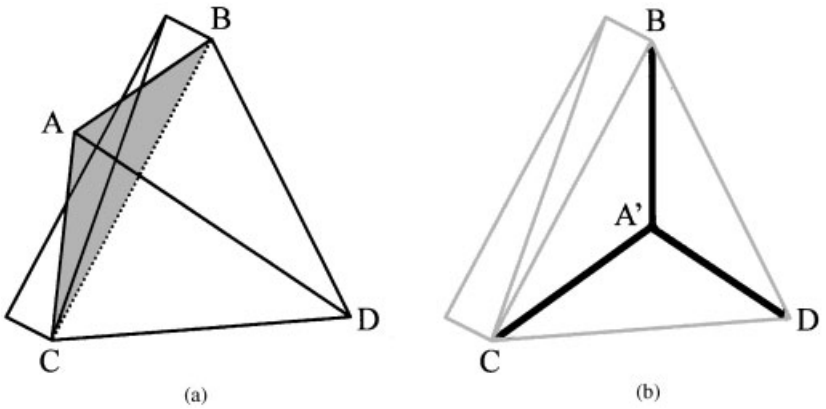


Figure 6. Sticking facet due to incorrect position of a node: (a) position of node A is incorrect. Nodes A, B, C, and D are on the same plane. (b) Correction of (a).

4.1.2. Sticking facets. A sticking facet means that a part of or all of the facet clings to another facet and their normal directions are contrary. The sticking facets can be classified into four types: (a) sticking ones due to the incorrect position of nodes, (b) completely overlapping ones, (c) outside protruding ones, and (d) inside protruding ones. The sticking facets can be found by using normal vectors of all the facets. When an angle between two normal vectors of adjacent facets is larger than, say, 160 degrees, they are flagged. The detection criteria for sticking facets are as follows.

- If a flagged facet is surrounded by three flagged facets, it is a sticking facet due to the incorrect position of a node (as shown in Figure 6(a)), which are revised by moving the node correctly as shown in Figure 6(b).
- If two flagged facets share three nodes, they are completely overlapping facets as shown in Figure 7(a), which are revised by swapping their longest edges with the neighboring facets, as shown in Figure 7(b).
- If three flagged facets share three nodes, one of them is an outside protruding facet as shown in Figure 8(a), which is revised by swapping the longest edge as shown in Figure 8(b). The other two facets are retained.
- If two flagged facets share two nodes and face front-to-front, they are inside protruding ones as shown in Figure 9(a), which is revised by swapping the shared edge as shown in Figure 9(b). The swapping often produces another inside protruding facet. The correction needs to be repeated until no protruding facets are produced.

4.2. Continuous surface check

The problems with STL files can be classified into three types: (a) overlapping facets, (b) gaps, and (c) incorrect orientation of facets. They prevent efficient surface grid generation, and hence we revise them beforehand.

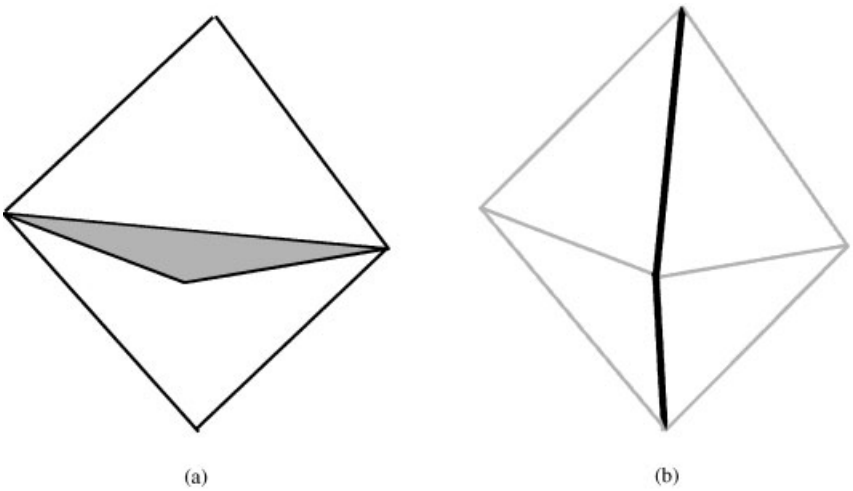


Figure 7. Completely overlapping facets: (a) the gray facets represent the overlapping ones. (b) Correction of (a).

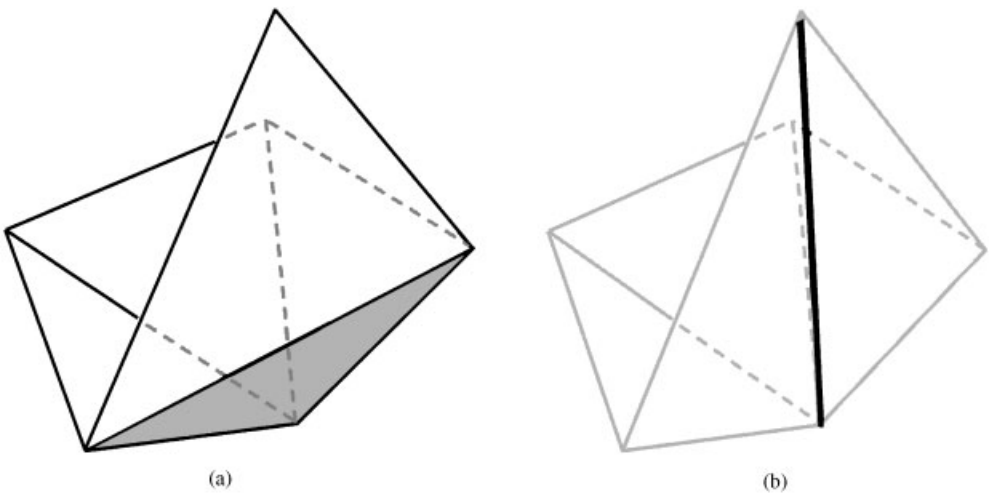


Figure 8. Outside protruding facet: (a) the gray facet represents the protruding one. (b) Correction of (a).

4.2.1. Removing overlap. Overlapping facets are removed in the following way.

(1) Information on edges is constructed. Each edge is identified by the nodes on its ends, and must have only one or two neighboring facets. If an edge has two neighboring facets, the facets must have the same orientation (see Figure 10).
(2) Facets are grouped into several zones of continuous facets. Each zone is surrounded by edges that have only one neighboring facet. If the number of zones becomes one, there are no overlapping facets (see Figure 11).

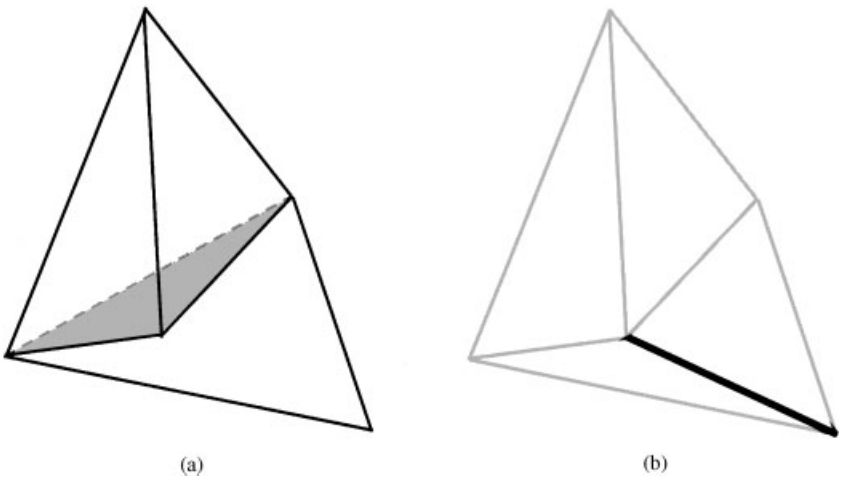


Figure 9. Inside protruding facet: (a) the gray facet represents the protruding one. (b) Correction of (a).

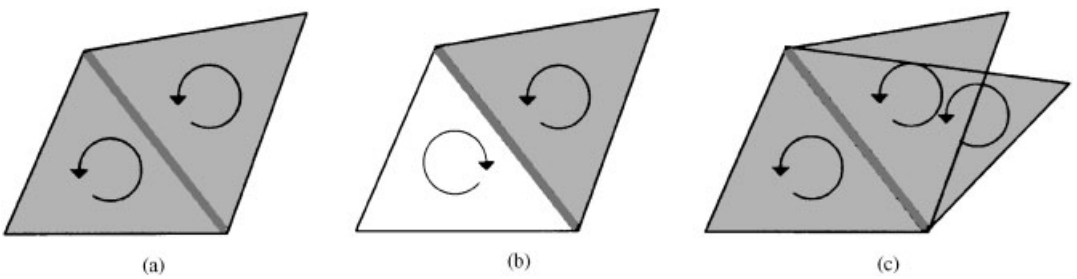


Figure 10. Definition of edges: the thick gray line in (a) represents one edge, the thick gray line in (b) represents two edges that have only one neighboring facet, and the thick gray line in (c) represents two edges; one of them has two neighbors and the other has one neighbor.

(3) If there are edges that have the same two nodes, overlapping facets may exist. Note that if each of them has only one neighboring facet, such as in Figure 10(b), this is another problem: the facet's orientation is incorrect (see Section 4.2.3). The zones that contain such edges are selected. The zone among them with the largest area is retained, but the others are removed.

4.2.2. Filling gaps. Gaps, such as holes and cracks, are filled in the following way.

(1) The edges that have only one neighboring facet are identified.
(2) The selected edges are linked, and gaps are detected (see Figure 12(a)).
(3) At each gap, the edges that border the gap are divided into several groups based on their folding angles (see Figure 12(b)).
(4) At each gap, the shortest edge is identified, and its length l_s is calculated. Let d be the distance between two nodes that have a different group number, and let r be the length of the surface model. If $d < \max(l_s, 10^{-7}r)$, the nodes are merged.

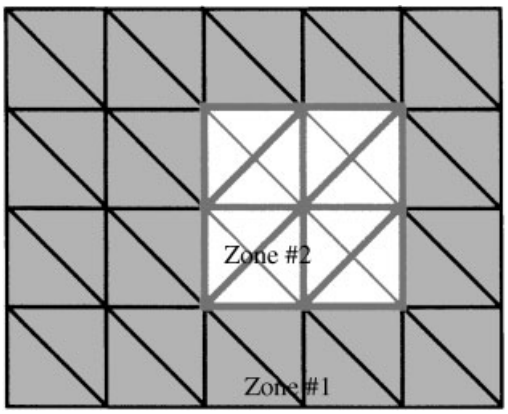


Figure 11. Definition of zones: each zone is surrounded by edges that have only one neighboring facet. Zone #2 is removed because it covers zone #1.

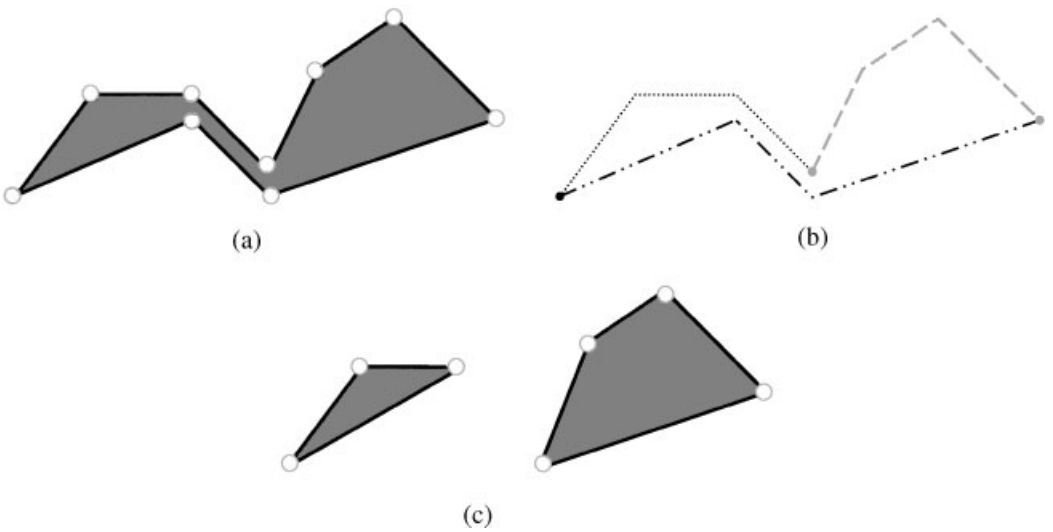


Figure 12. Pretreatment of gaps to be filled: (a) detecting gaps; (b) grouping of edges; (c) connecting nodes and updating gaps.

(5) The gaps are updated if merged nodes exist (see Figure 12(c)).
(6) If a partial gap is a large one (hole), as shown in Figure 13(a), the gap is covered with triangular facets as shown in Figure 14(a). If the gap is a narrow one (crack), as shown in Figure 13(b), the neighboring facets are divided as shown in Figure 14(b).

It is important that the gaps be classified into two types: holes and cracks in Step (6). If the cracks are caulked in the same way as the holes, the new facets become bad.

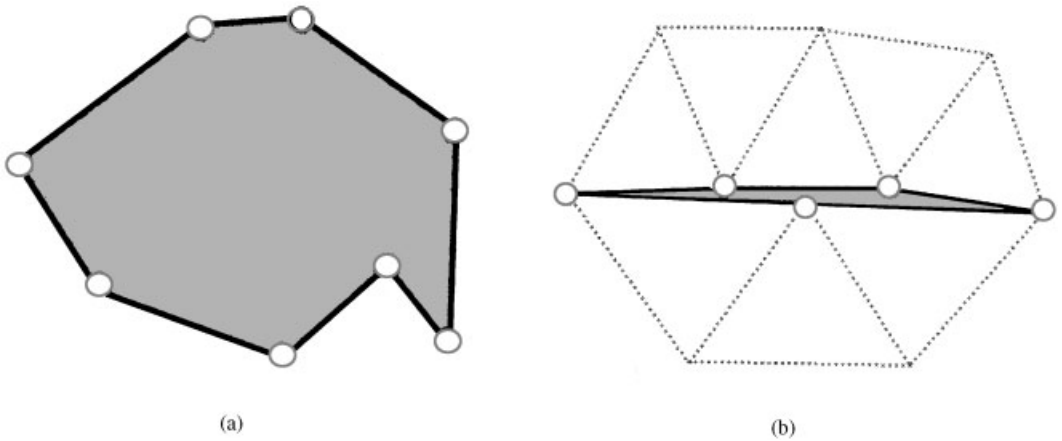


Figure 13. Examples of gaps: gray polygons represent gaps, black solid lines represent boundaries of the gaps, and gray dashed lines represent neighboring facets; (a) large gap (hole); (b) narrow gap (crack).

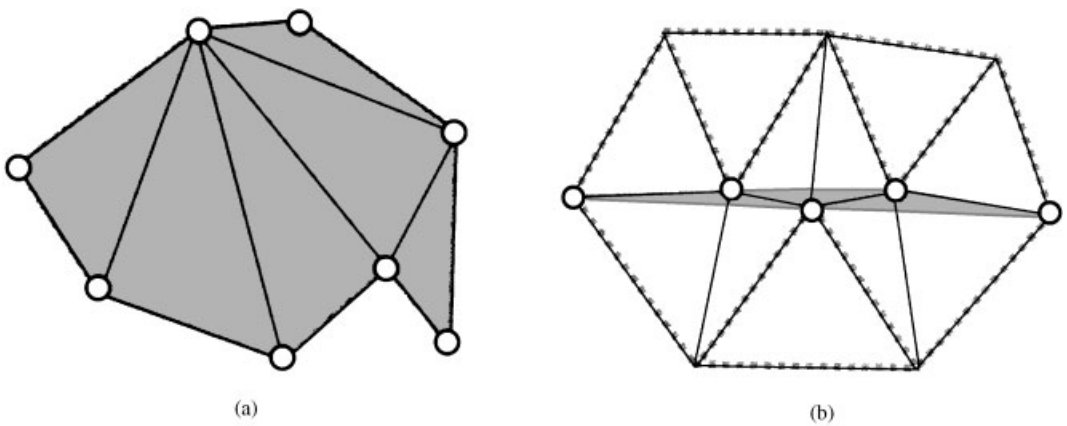


Figure 14. Filling gaps: (a) hole: solid black lines represent new facets;
(b) crack: solid black lines represent revised facets.

4.2.3. Revising incorrect orientations. Incorrect orientation of a facet means that its three nodes are stored in reverse order (see Figure 15). This problem is revised in the following way after the surface gaps have been filled.

(1) Zones are defined in the same way as in Step (2) of Section 4.2.1. If only one zone exists, there is no facet that has an incorrect orientation.
(2) At each node, connecting zones are listed. If a node is on only two zones, the entire orientation of the facets on one of the zones is changed. Zones are then updated.
(3) The item 2 is repeated until only one zone remains.
(4) The outside of the revised model is checked.

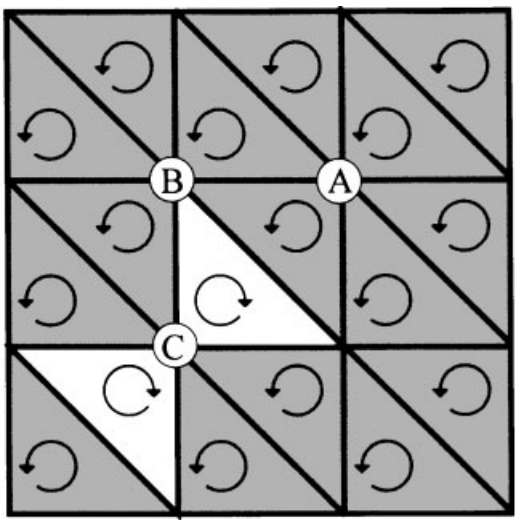


Figure 15. Incorrect orientation of facets: white facets have incorrect orientation. Nodes A, B, and C are in one, two, and three zones, respectively.

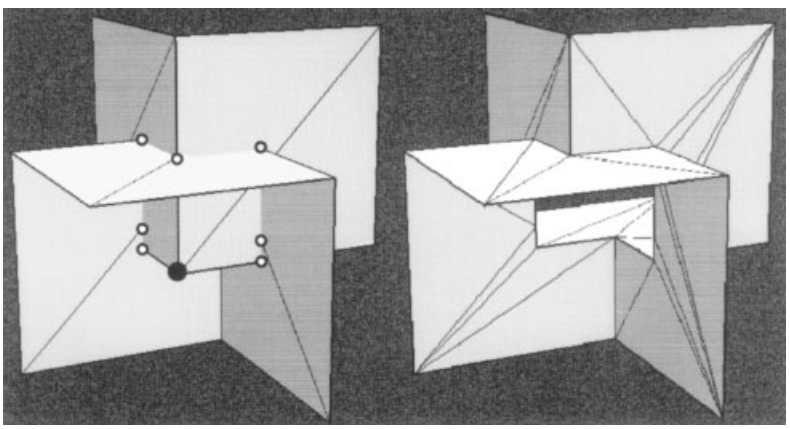


Figure 16. Boolean union: (a) two cubes that intersect; (b) after removal of the intersection.

4.3. Removing intersections

Intersections are removed in the following way.

(1) Replace each part with a bounding box [10]. If the boxes intersect, the following detailed check is needed.
(2) For each edge, identify crisscrossing facets or edges, and calculate the intersecting points, such as the white points in Figure 16(a). If edge–edge intersections exist, stretched facets may hinder the accuracy of the search due to round-off errors. They should be removed beforehand as shown in Figure 17.

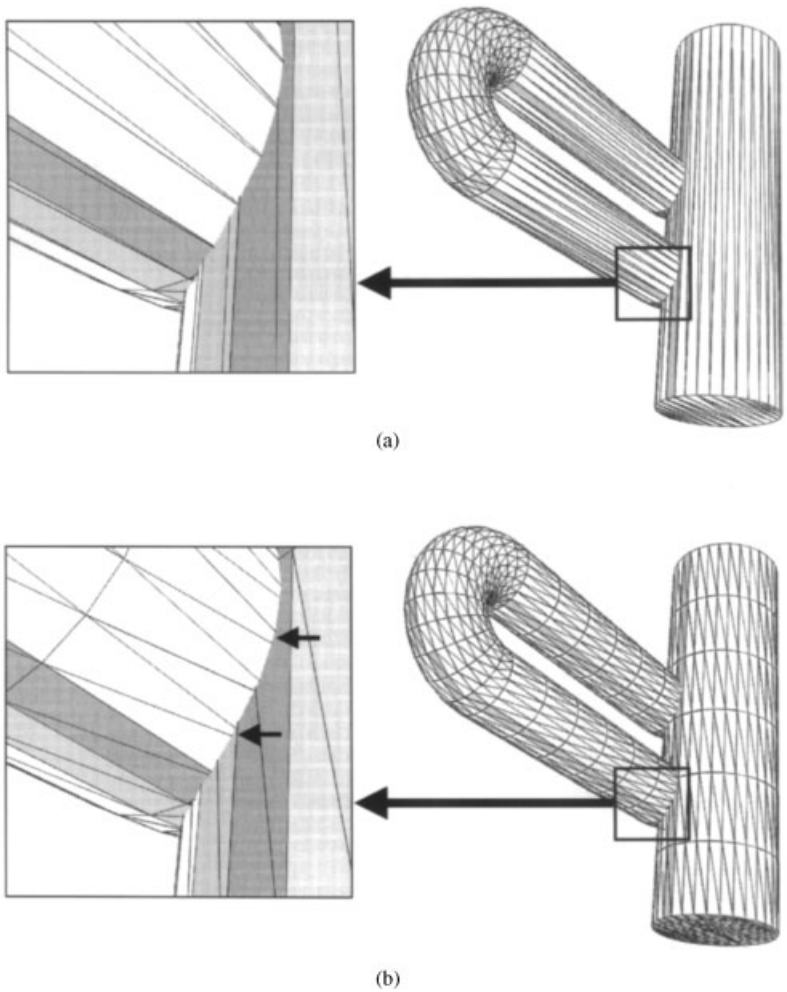


Figure 17. Division of stretched facets: (a) before the division; (b) after the division.

(3) Remove nodes that are inside the object, such as the black point in Figure 16(a).
(4) Add intersecting nodes calculated at Step (2) and edges.
(5) Divide polygons, which are made of divided original edges and newly created edges at Step (4), into triangular facets.
(6) Remove stretched facets if necessary.

5. APPLICATIONS

Three cases are presented to show the problems encountered with STL surface models and to demonstrate how to overcome them. The surface grid generation was run on a PC with a 1 GHz Pentium III processor, with 512 MB of memory.

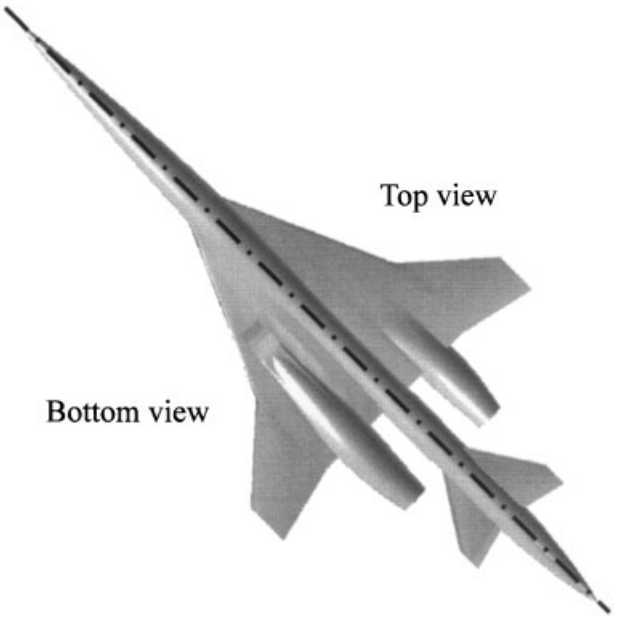


Figure 18. NAL jet-powered experimental supersonic airplane model.

Tetrahedral volume grids were then generated for the first two models using a Delaunay method [11] on a workstation with a 500MHz Alpha 21264 processor, having 4GB of memory. The volume grid generation was done fully automatically from a surface grid, and the required CPU time was between 30–60 min.

5.1. NAL experimental supersonic airplane model

Figure 18 shows the experimental supersonic airplane designed at the National Aerospace Laboratory of Japan (NAL) [12]. NAL has been working on the next generation supersonic transport, and the project includes the development of two types of experimental airplanes: a rocket-launched one with no engine and a jet-powered one. The model shown here is the latter one.

The engine nacelles have several surfaces, and the sides of the intakes are so thin that a trimmed surface model could not be obtained (see Figure 19). Accordingly, the STL data inside of the flow-through nacelle were created separately from the semi-span model as shown in Figure 20. Cracks existed between them.

The STL file had 164 422 facets, and its size was 7.82 MB in binary format. The required CPU time for the identification of nodes in the STL file, which is described in Section 3, was about 3 s. The cracks were then revised automatically.

Figure 21 shows the surface mesh, which had 88 977 nodes and 177 954 faces. In Figure 21(c), a fine mesh can be seen near the intake region where the cracks existed in the original STL model. The required CPU time for the mesh generation was 3.5 min. The turnaround time from the STL data input was about 3 h.

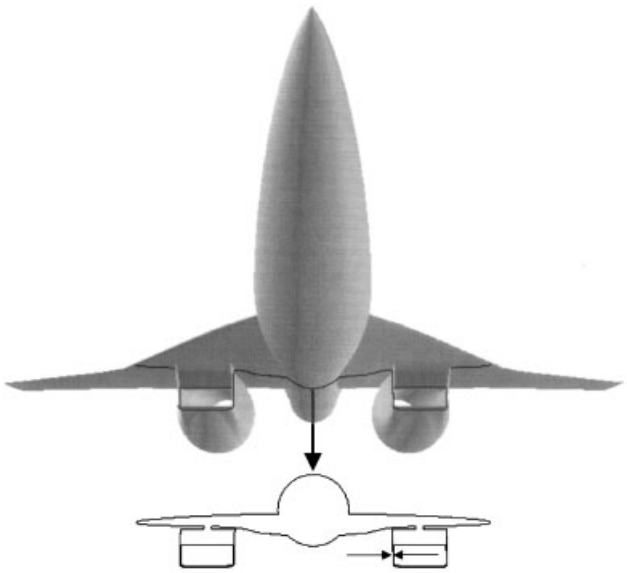


Figure 19. Cross-sectional view across the intakes: the intakes have thin sides.

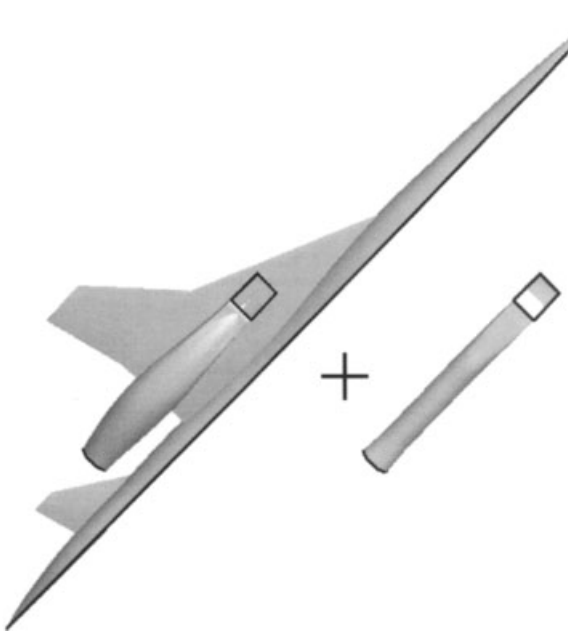


Figure 20. STL data of the NAL experimental supersonic airplane model: black lines represent surface boundaries.

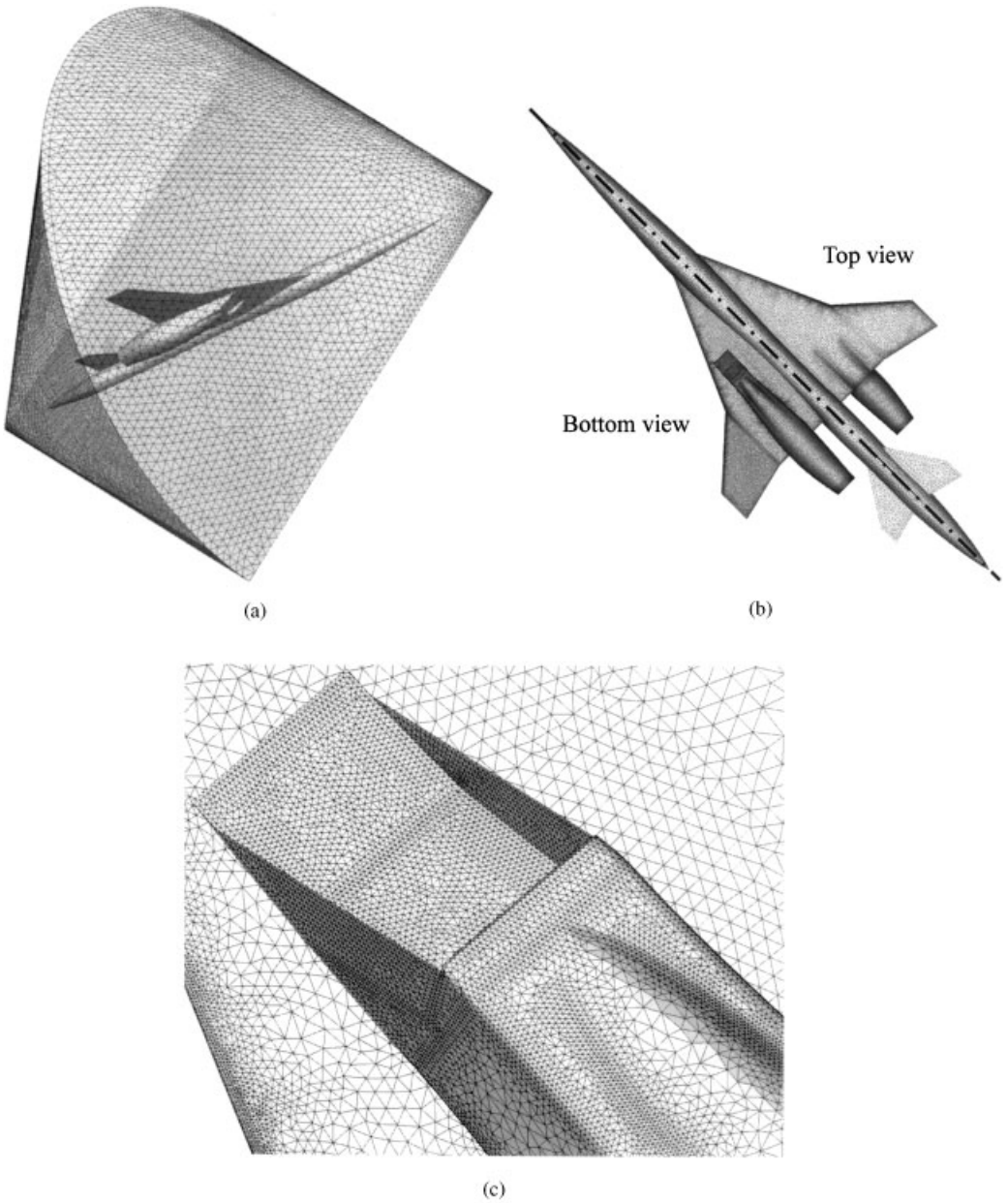


Figure 21. Surface mesh of the NAL experimental supersonic airplane: (a) general view; (b) top/bottom view of the airplane; (c) the mesh near the intake where cracks existed.

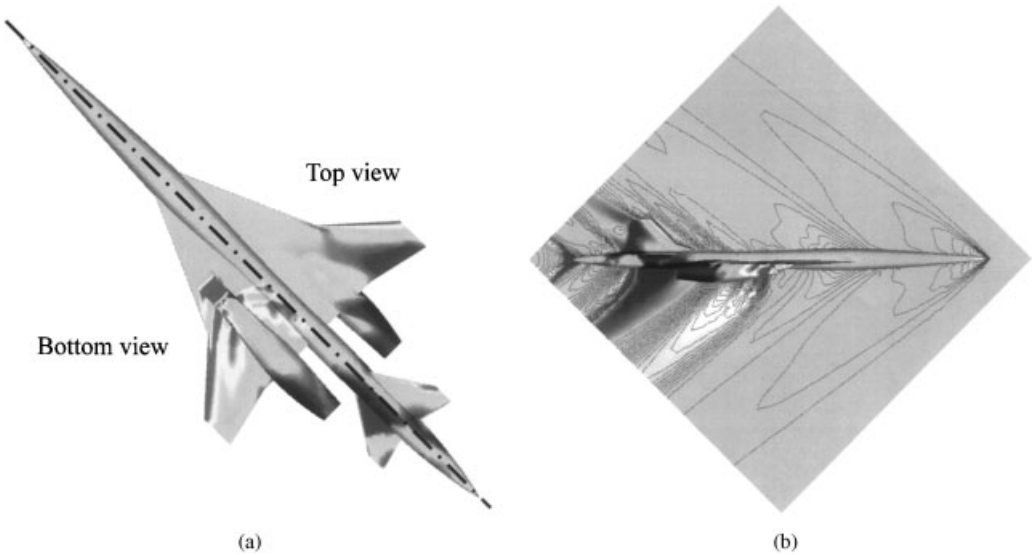


Figure 22. Euler calculation for the NAL experimental supersonic airplane model ($M_\infty = 1.7,\ \alpha = 0.0$): (a) surface pressure distribution; (b) pressure contours on the symmetry plane.

Figure 23. Hornet model.

The tetrahedral volume grid based on the surface mesh was generated and had 710 988 nodes and 3 962 982 tetrahedra. Euler calculations were performed, and Figure 22 shows the computed pressure distributions at a free-stream Mach number of 1.7 and an angle of attack of 0.0°.

5.2. Hornet model

Figure 23 shows a model of a hornet, which has antennae, a stinger, and legs with claws. The configuration data were prepared as a DXF file. This format was developed by AutoDesk

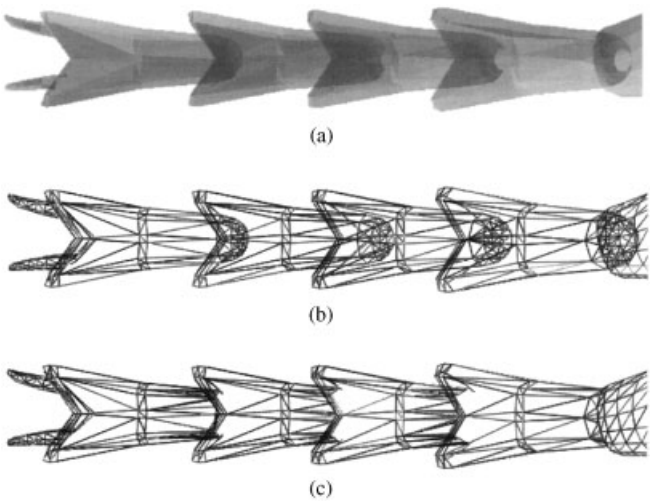


Figure 24. Foreleg of the hornet: (a) and (b) original model, which has intersections; (c) revised model.

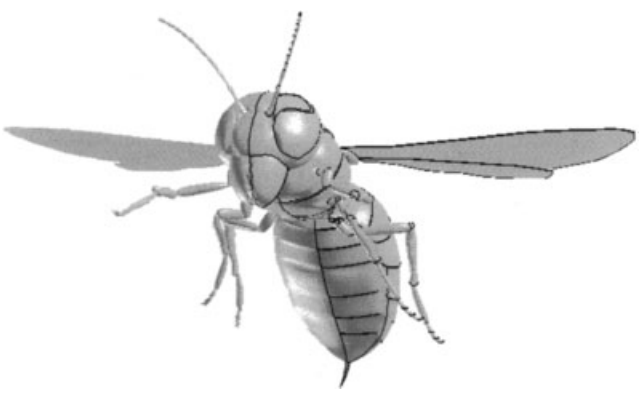


Figure 25. Geometric features of the hornet model (black lines).

to allow CAD data to be interchanged between different applications. The DXF file generally represents a configuration as a collection of parts, each part consisting of polygons. It is easy to subdivide polygons into triangular facets, but the intersecting facets among the parts as shown in Figure 24 have to be eliminated.

The model was converted to a semi-span and a continuous surface model automatically. The original model also had a problem: coarseness. However, the surface meshing method using the geometric features shown in Figure 25 ensured that the mesh generation provides the required resolution even if the background grid is not fine. Note that if the geometric feature lines are intricate, many nodes are required to represent them correctly.

Surface triangulation was then performed as shown in Figure 26. The generated surface mesh has 126 600 nodes and 253 196 faces. The required CPU time was about 3 min for the

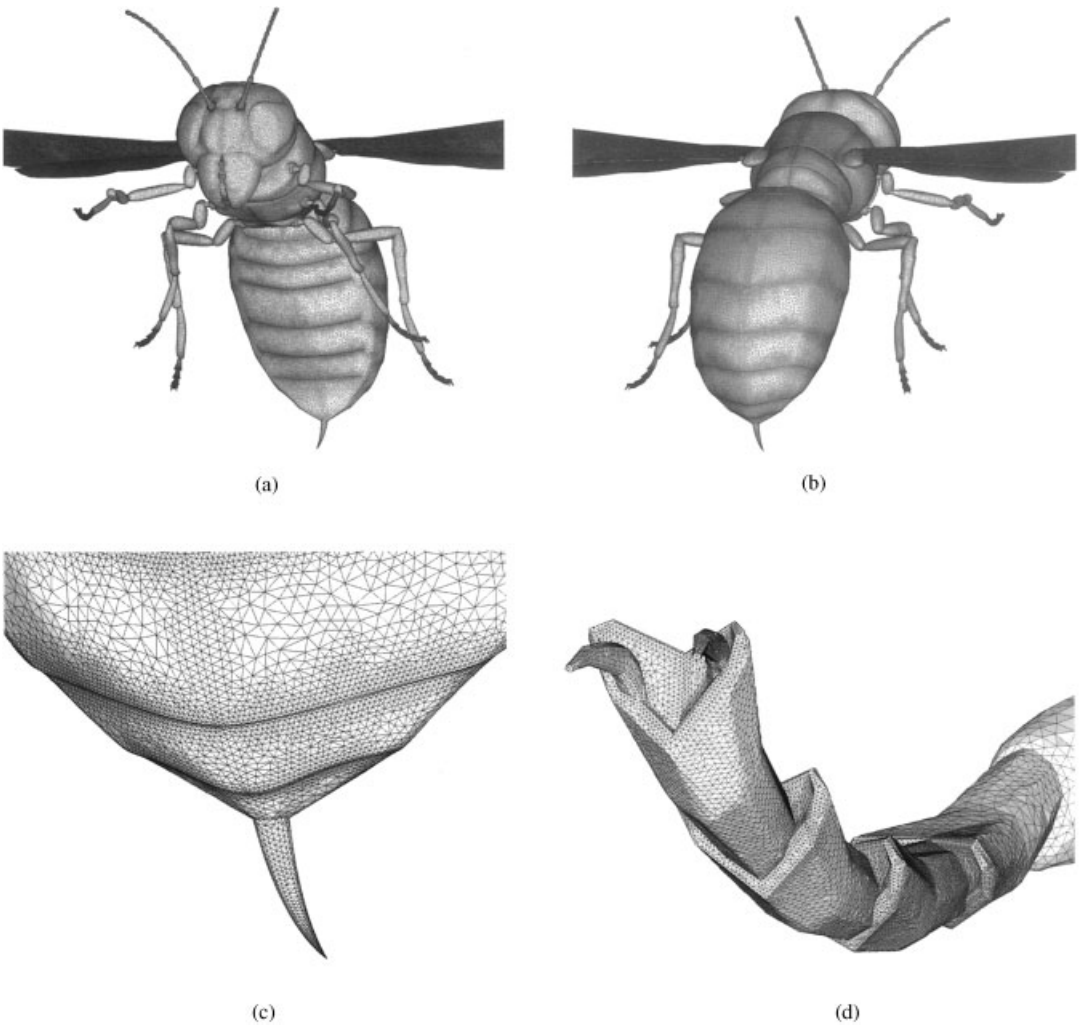


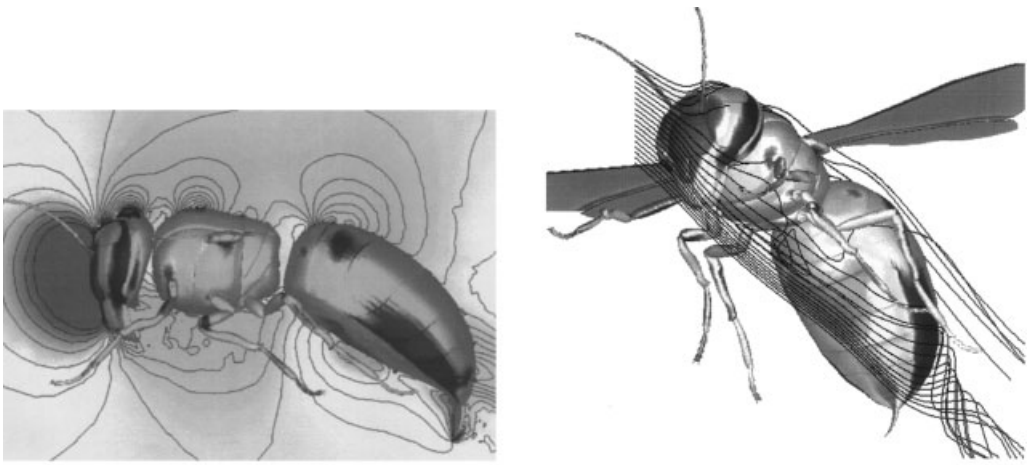
Figure 26. Surface mesh of the hornet: (a) front view; (b) back view; (c) stinger; (d) foreleg.

surface grid generation process in this case. The turnaround time was about 8 h from the input DXF file to the surface mesh.

The volume grid was generated and had 1 242 800 nodes and 7 016 043 tetrahedra. Euler calculations were performed, and Figure 27 shows the computed results at a Mach number of 0.2. The wing was fixed at a zero dihedral angle.

5.3. Rotor model

Figure 28 shows the component parts of a car engine. This model was prepared as an STL file, but the surface had many problems. Figure 29 shows the boundaries of the patches and



(a) (b)

Figure 27. Euler calculation for the hornet ($M_\infty = 0.2$, $\alpha = 0.0$): (a) pressure contours on the symmetry plane; (b) surface pressure distribution and streamlines.

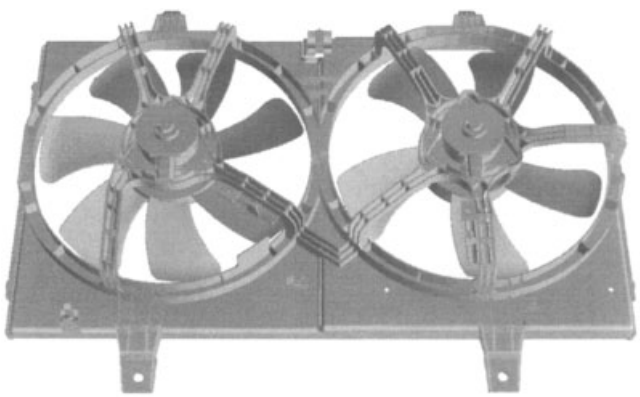


Figure 28. STL data of the rotor model.

gaps. We had to fill these gaps to obtain a continuous surface. These gaps were automatically corrected, but there was another problem: messy facets as shown in Figure 30. These facets were found by checking folding angles, and were revised manually.

After the revision, geometric features were automatically extracted. As the surface was also rough and course, we had to remove some ridges. The geometric features are as shown in Figure 31.

Surface triangulation was then performed as shown in Figure 32. The generated surface mesh had 325 649 nodes and 651 434 faces. The required CPU time was about 11 min for the surface grid generation process in this case.

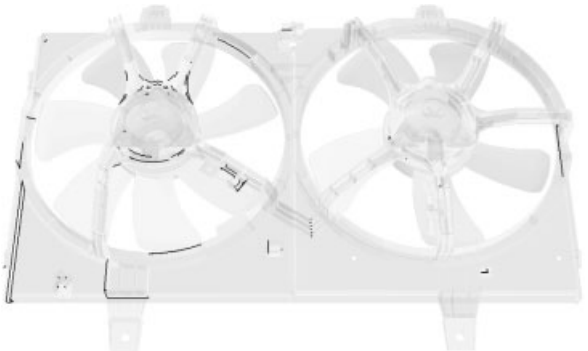


Figure 29. Cracks and gaps on the surface of the rotor model (solid lines).

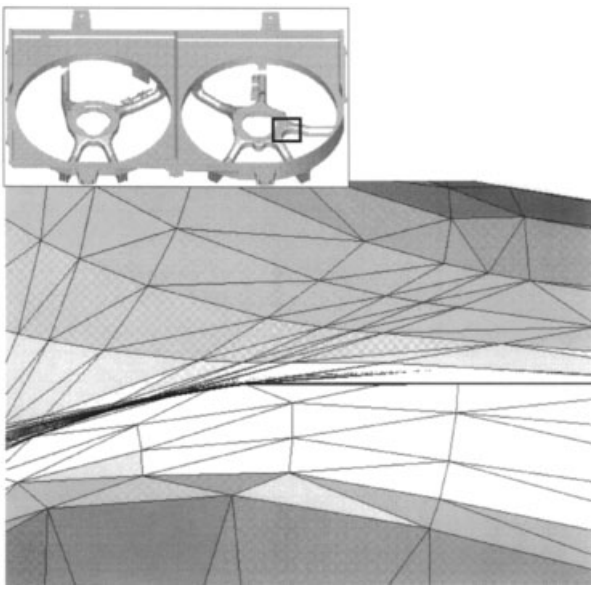


Figure 30. Messy facets: facets and nodes are intricate.

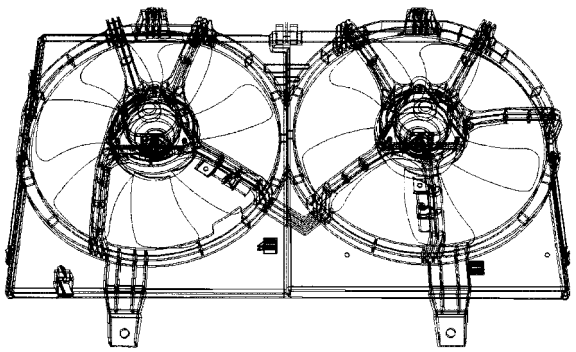


Figure 31. Geometric features of the rotor model.

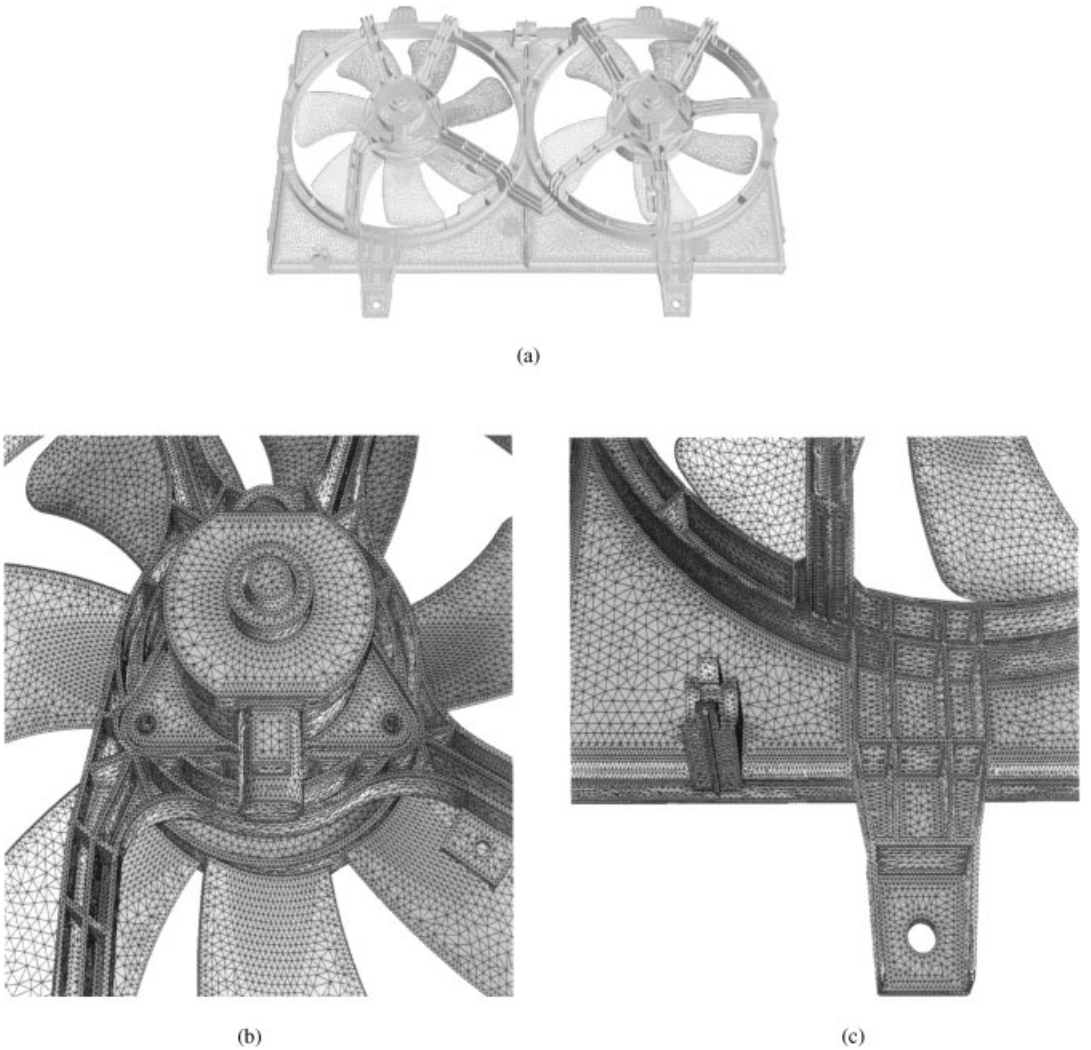


Figure 32. Generated surface mesh: (a) general view; (b) enlarged view around the center of the left rotor; (c) enlarged view near the left lower corner.

6. CONCLUSIONS

A surface triangulation method for CAD surfaces represented as lists of polygons was presented. STL-defined surfaces, which contain several problems such as overlaps, gaps and incorrect orientations, were converted to continuous surfaces automatically before the surface meshing method was applied. The modified surfaces were used for the background grid of the surface meshing. The ability to generate high quality surface grids was demonstrated by applying this method to several surface models.

ACKNOWLEDGEMENTS

The authors wish to thank T. Iwamiya of the National Aerospace Laboratory of Japan for providing them with the geometric data of the experimental supersonic airplane; T. Fujita, a graduate student of Tohoku University, for his help in generating the STL data; and the authors also wish to acknowledge K. Fujitani, Nissan Motor Co., Ltd., for providing them with the rotor model.

REFERENCES

1. *Stereolithography Interface Specification*. 3D Systems, Inc.: CA, 1989.
2. Nakahashi K, Sharov D. Direct Surface Triangulation Using the Advancing Front Method. *AIAA Paper 95-1686-CP*, 1995.
3. Ito Y, Nakahashi K. Direct Surface Triangulation Using Stereolithography (STL) Data. *AIAA Paper 2000-0924*, 2000.
4. *AutoCAD Reference Manual*. AutoDesk, Inc.: CA, 1990.
5. Walatka PP, Buning PG, Pierce L, Elson PA. PLOT3D User's Manual. *NASA TM 101067*, 1990.
6. Lo SH. Automatic mesh generation over intersecting surfaces. *International Journal for Numerical Methods in Engineering* 1995; **38**:943–954.
7. Shostko AA, Löhner R, Sandberg WC. Surface triangulation over intersecting geometries. *International Journal for Numerical Methods in Engineering* 1999; **44**:1359–1376.
8. Fujita T, Ito Y, Nakahashi K, Iwamiya T. Aerodynamics Evaluation of NAL Experimental Supersonic Airplane in Ascent. *AIAA Paper 2001-0564*, 2001.
9. Jacob GGK, Kai CC, Mei T. Development of a new rapid prototyping interface. *Computers in Industry* 1999; **39**(1):61–70.
10. Gottschalk S, Lin MC, Manocha D. OBB-Tree: A Hierarchical Structure for Rapid Interference Detection. *Proceedings of the SIGGRAPH 1996 Annual Conference on Computer Graphics*, 1996, pp. 171–180.
11. Sharov D, Nakahashi K. A boundary recovery algorithm for delaunay tetrahedral meshing. *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Field Simulations*, Mississippi State University, 1996, pp. 229–238.
12. Iwamiya T. NAL SST Project and Aerodynamic Design of Experimental Aircraft. *Proceedings of the 4th ECCOMAS Computational Fluid Dynamics Conference*, Athens. Wiley: Chichester, 1998, pp. 580–585.

