AIAA JOURNAL
Vol. 40, No. 3, March 2002

Direct Surface Triangulation Using Stereolithography Data

Yasushi Ito* and Kazuhiro Nakahashi†
Tohoku University, Sendai 980-8579, Japan

An efficient and user-friendly approach is presented for constructing unstructured surface grids directly on three-dimensional CAD surface data. This system has the following features: 1) use of stereolithography data as surface definitions, that is, background grids for surface meshing; 2) automatic reconstruction of geometric features for initial front setup; 3) graphical user interface for easily controlling surface mesh density by inserting line sources and point sources on the surface; 4) adoption of an advancing-front surface triangulation method, which minimizes the necessity to divide the surface into a number of patches; and 5) outer boundary generation from templates in the system. The resulting system has been applied to several airplane configurations, and a notable reduction in turnaround time has been achieved.

I. Introduction

COMPUTATIONAL fluid dynamics (CFD) has become an indispensable design and analysis tool for many different types of geometric configurations and field regimes. The effectiveness of CFD usage in an actual design environment, however, is often limited. Most of these limitations are due to difficulties in generating the computational grid in a reasonable amount of time.

The grid generation method should be automated as much as possible. This is true for volume grid generation; it is now possible to create volume grids automatically. For the surface grid generation, however, user interventions are often required because the surface mesh directly affects the solution accuracies in CFD problems, especially for the aerodynamic evaluation of airplanes. The aerodynamic performance is usually evaluated by surface values, such as pressure, skin friction, and so on. These values are highly affected by the definition of sound geometric features, as well as the surface grid density. This suggests that surface grid generation methods must easily facilitate changes in grid density to obtain the best result with limited computational resources. Although controllability is often incompatible with the automation of grid generation, it is essential for surface meshing.

Surface grid generators are naturally expected to be user friendly; however, this is not usually the case. The main reason for this stems from the employment of a so-called mapping method, in which a three-dimensional surface usually needs to be inputted not as one body, but as a collection of surface patches. Each patch should be defined as a topologically rectangular array of discrete data points for mapping from a three-dimensional free-form surface to a two-dimensional plane. The triangulation is then performed on the two-dimensional domain. This procedure often requires hundreds of hours of manipulation to obtain a satisfying surface grid from patched geometry. Surface decomposition is not essential to generate a high-quality surface mesh and should be avoided.

To define an object surface precisely and efficiently requires a close link between CAD and grid generator. CAD software has many output formats. One of the most common output formats is an initial graphics exchange specification (IGES) format. Several approaches for surface meshing have been proposed using the IGES format.[1–4] Although IGES files include detailed information about geometry, they must be converted to a suitable representation for surface meshing. Unfortunately, this conversion is tedious and time consuming.

Another famous CAD output format is the stereolithography (STL) format. The STL file format has become the de facto standard in rapid prototyping.[5] This file is made from pure geometric information from a model (this is composed of triangular facets and their associated unit normals) so that it is very simple and easy to manipulate the defined geometry. However, the STL file is not designed for surface meshing so that it often has ill-conditioned facets, such as overlapping facets and gaps. Because of this problem, the use of the STL file has been limited.

In this paper, the STL file is employed as a background grid for surface meshing. To revise the ill-conditioned facets included in the STL file, a strategy to use both the automatic revision and the graphical user interface- (GUI-) based interactive correction for the STL data is employed. Commonly encountered patterns of the ill-conditioned STL facets are recognized and fixed automatically. The remaining ill-conditioned facets are corrected utilizing a GUI tool developed here. This strategy of using both automatic and interactive procedures is the essence of the present approach for realizing surface grid generation for complex geometries.

To minimize user interventions in surface grid generation, the direct advancing front method for surface meshing[6] is applied directly to the three-dimensional surfaces. One of the advantages of this approach is that the surface does not need to be divided into many patches. Furthermore, the initial fronts are useful in determining the local grid densities by distributing the grid points, which have been numbered by the user. The initial fronts, however, must be set up beforehand. This front setup sometimes becomes tedious and time consuming.

In the present approach, an automatic reconstruction of the geometric features, such as wing leading and trailing edges, wing–fuselage junctions, and other body boundaries, is introduced to construct initial fronts automatically. The primitive idea of using geometric features was mentioned by Löhner[7]; however, he used them only to recover surface features. In this paper, we also use them to control surface grid density efficiently. The initial fronts are constructed on these geometric features, which are easy to reconstruct from the CAD surface data. To enhance the controllability of grids, initial fronts can also be specified by the user through the use of a GUI.

In this paper, we discuss a surface grid generation method in which user interventions are minimized, whereas the controllability of the grid is enhanced. This is achieved through the use of an advancing front algorithm[4,6–9] coupled with an automatic reconstruction of the geometric features from STL data. The easy controllability is also enhanced by the use of a GUI.

II. Surface Grid Generation Method

A flowchart of the surface grid generation procedure is shown in Fig. 1. After reading geometry data in STL format, geometric

Presented as Paper 2000-0924 at the AIAA 38th Aerospace Sciences Meeting, Reno, NV, 10–13 January 2000; received 3 June 2000; revision received 4 September 2001; accepted for publication 12 October 2001.

*Graduate Student, Department of Aeronautics and Space Engineering, Aoba-yama 01; itoh@ad.mech.tohoku.ac.jp.

†Professor, Department of Aeronautics and Space Engineering, Aoba-yama 01; naka@ad.mech.tohoku.ac.jp. Associate Fellow AIAA.

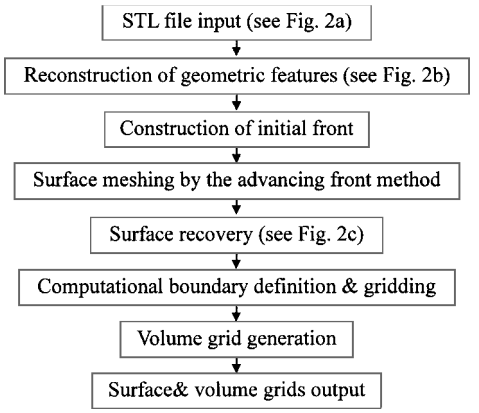


Fig. 1 Flowchart of surface grid generation.

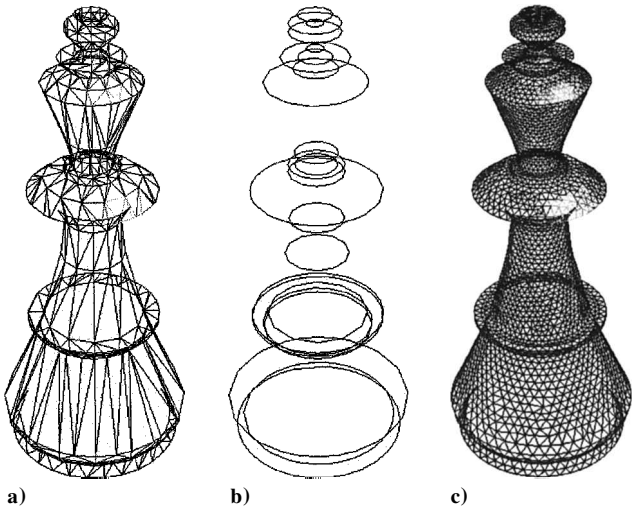


Fig. 2 Surface grid generation process for a chess piece: a) typical STL data having very coarse triangles, stretched triangles, and some gaps between triangles; b) ridge reconstruction from STL data; and c) generated surface mesh.

features are reconstructed. Grid points are distributed on these curves to set up initial fronts for the advancing front method. Point sources and line sources may be added to control local grid density. The advancing front method is then applied directly on the STL-defined background grid. After that, the surface recovery procedure is applied to the generated surface grid for accurate representation of the original surface model. For a flow computation around an object, the remaining work defines the computational outer boundaries and the volume grid generation.

It is desirable to automate the grid generation procedure. However, the controllability of the local grid density is also very important. In the present approach, users can interactively perform the following steps in the surface meshing process using the mouse and keyboard: 1) addition of point sources and line sources, that is, ridges (see Sec. II.C); 2) specification of division parameters at each ridge (see Sec. II.C); and 3) specification of the maximum and minimum length of the grid (see Sec. II.D). The GUI environment has been constructed with Microsoft Visual C++ 6.0 and OpenGL.

The remaining steps are 1) correction of STL data (see Sec. II.A), 2) reconstruction of geometric features (see Sec. II.B), 3) initial front setup from the user-specified parameters (see Sec. II.C), 4) surface meshing (see Sec. II.D), and 5) surface recovery (see Sec. II.E).

These steps are not essential to control the surface mesh that will be generated, and so they should be performed automatically. An example of the surface grid generation of a chess piece is shown in Fig. 2.

A. STL Data

The present approach uses an extracted STL format for surface meshing. An STL output is designed for rapid prototyping. Arbitrary three-dimensional free-form surfaces are approximated as a set of triangular facets. The output is available in either ASCII or binary format and in either metric or imperial units. The STL can also be scaled automatically to accommodate the dimensions of the prototyping equipment. The output file contains only pure geometric information. In other words, only the coordinates of each triangular facet and its corresponding unit normal vector indicate the external objects' surface. Because of the simplicity of the STL file, other geometry data such as a structured surface grid can be easily converted to this format if necessary. This is another advantage of using STL data as an input data format for surface meshing.

The process used to generate an STL file from a CAD dataset is called tessellation. The accuracy of an STL file is determined from two user-specified tolerances: the size of each triangle and the maximum distance between the surface and a triangular facet (sag value). Large flat areas are represented by a small number of very coarse triangles as shown in Fig. 3a, and cylindrical and conical surfaces are represented by stretched triangles as shown in Fig. 3b. Therefore, the triangulated surface given by the STL file must be retriangulated for CFD use.

In the present approach, STL data are used as a background grid for surface meshing. For this purpose, a set of STL facets must cover the original surface without overlaps or gaps. STL data, however, often contain some ill-conditioned facets due to errors in the tessellation process because the STL file format does not provide for consistency and completeness tests.[10] Decreasing the sag value ensures the greatest accuracy of the three-dimensional geometry; however, it increases the likelihood of triangulation errors. These facets often cause some conflicts in the following steps, for example, when ridges are automatically reconstructed from the original STL data, or when the local direction of surface meshing is calculated. In the present method, most of the ill-conditioned facets are automatically corrected or removed by the following procedures.

The ill-conditioned facets can be classified into two types: highly stretched facets (Fig. 4) and sticking facets (Fig. 5). The highly stretched facets, whose minimum interior angle is less than 1 deg and whose area is almost zero, are considered ill conditioned.

The highly stretched facets are detected and corrected as follows:

1) Calculate lengths of all edges.

2) At each triangular facet, three edges are lined up in ascending order by their lengths.

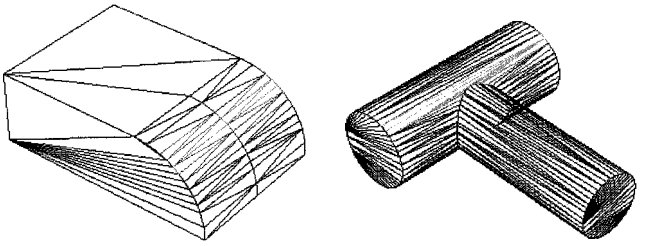


Fig. 3 Examples of STL data.

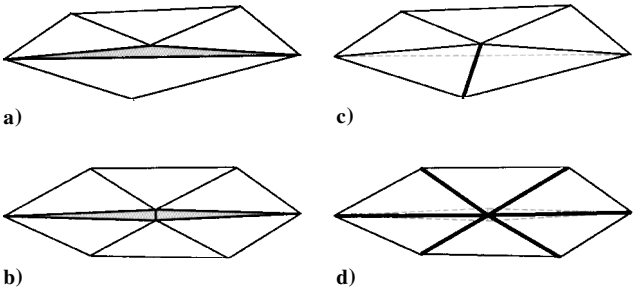
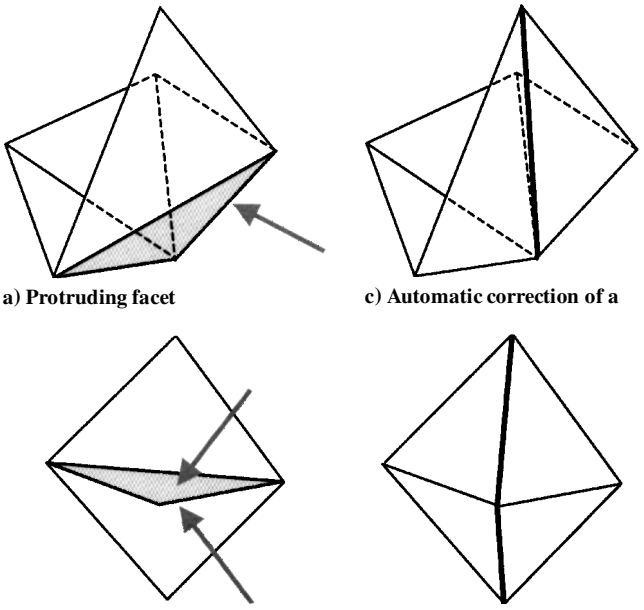


Fig. 4 Models of highly stretched facets in STL data: a) type 1, isolated highly stretched facet; b) type 2, connected highly stretched facets; c) automatic correction of type 1; and d) automatic correction of type 2.



a) Protruding facet

c) Automatic correction of a

b) Overlapping facets

d) Automatic correction of b

Fig. 5 Examples of sticking facets in STL data.

3) The following reference value ξ is used for the detection of a stretched facet:

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

where l_1, l_2, l_3 ($l_1 > l_2 > l_3$) are lengths of its three edges and α is a small angle such as 1.5 deg. If the value ξ is less than unity, the facet is considered to be highly stretched.

4) If $l_3 > 0.05\, l_1$, the facet is type 1, as shown in Fig. 4a. This facet is corrected by swapping the longest edge (Fig. 4c).

5) If $l_3 \leq 0.05\, l_1$, the facet is type 2, as shown in Fig. 4b, and is removed together with the neighboring facet (Fig. 4d).

Because the area of the stretched facet is almost zero, as mentioned, these revisions do not reduce the accuracy of the modeling.

The sticking facets can be found using the facet normals. When an angle between two normal vectors of adjacent facets is larger than, for example, 160 deg, they are flagged. If three flagged facets share three nodes, one of them is a protruding facet, as shown in Fig. 5a and revised in Fig. 5c. If two flagged facets share three nodes, they are overlap facets, as shown in Fig. 5b and revised in Fig. 5d.

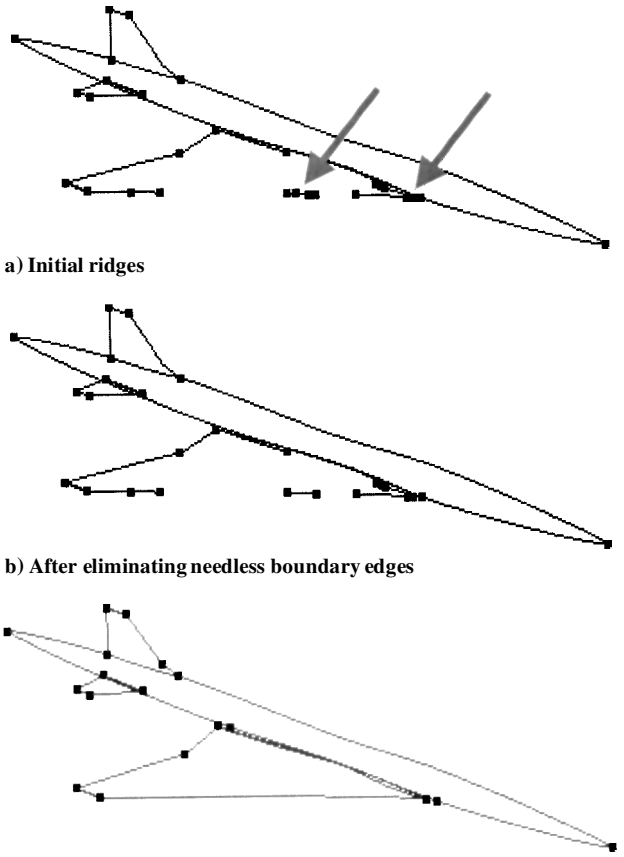
The remaining ill-conditioned facets that do not meet the commonly encountered patterns will be revised by means of the GUI tool developed here.

B. Reconstruction of Ridges

In the second step, geometric features are reconstructed from the STL-defined configuration. The geometric features include ridges and surface boundaries, such as wing leading and trailing edges. An example of the ridge reconstruction is shown in Fig. 6c. The reconstruction of these curved features is important in accurately representing the original configuration. These ridges also become the initial fronts of the surface meshing procedure in the next step.

The ridges are defined as aggregates of boundary edges. Boundary edges consist of two types of edges: surface boundary edges and inner boundary edges. Surface boundary edges are defined as edges that have only one neighboring facet. Inner boundary edges have two neighboring facets and are searched using a folding angle that is calculated using the normal vectors of two facets on both sides of each edge. The edge is considered a candidate edge for constructing a ridge if the folding angle is larger than the specified value. (Default value is 30 deg.)

Figure 6 shows the reconstruction process of the geometric features for an airplane configuration. Figure 6a shows the ridges detected from the STL data using only the folding angles. These ridges, however, have not yet been connected to each other in this stage,



a) Initial ridges

b) After eliminating needless boundary edges

c) After the recovery of boundary edges

Fig. 6 Ridge reconstruction of an NAL experimental supersonic airplane model.

but may be isolated, and do not represent the model properly. Some manipulation is required to eliminate or reconnect some ridges to recover the proper features accordingly.

First, distances between every two candidate edges at their midpoints are calculated. If the distance is less than half of the smaller edge length between two edges, either of the edges is removed from the candidate list. This removal is based on the number of connecting candidate edges on both ends of the edge or the strength of the folding angle.

Second, the connecting angles of two candidate edges are calculated. If the angle is less than 3 deg, either of the edges with the same criterion is removed. Note that some exception is needed to avoid removing the essential candidate edges, for example, at the wing tip. The ridge in this phase is shown in Fig. 6b.

Third, several inner boundary edges are added from the ends of the ridges in consideration of the folding angle and the direction. The edge addition is stopped when the ridge encounters a ridge, or the folding angle is less than a specified value (default value is 3 deg).

Fourth, kinks in the ridges should be identified automatically. The kinks divide the ridges into a set of smooth ridges. The smooth ridge is required so that spline interpolations can be used in the following step.

The final ridges reconstructed in the aforementioned procedures are shown in Fig. 6c. These steps can be performed automatically.

C. Initial Front Setup

In the third step, the initial front for the advancing front method is constructed. Node points are distributed on each curved feature by specifying division parameters, the number of points and the spacing parameters used in the Vinokur's stretching function,[11] using a GUI command.

According to demand, point sources and line sources can be easily added on the user-specified region to control the local grid density. Figure 7 shows a simple example of a square area. If ridges are

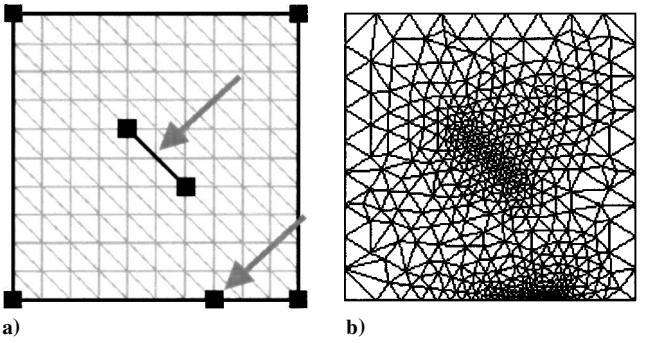


Fig. 7 Control of local grid density: a) line source and point source insertion for a square area and b) triangulation of part a.

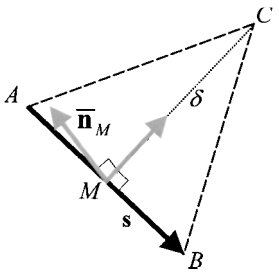


Fig. 8 New temporary node position.

automatically generated on this area, they are made of the only edges surrounding it. It is relatively easy to control the grid density along the boundaries. However, it is difficult to cluster the grid points inside of the domain or at an intermediate point of the boundary. Adding a point source and a line source, as shown in Fig. 7a, users can gather the grid points anywhere, as shown in Fig. 7b.

D. Triangulation

Many researchers have described in detail the surface-meshing algorithm using an advancing front method.[6,8,12] Here, we discuss an outline of the triangulation procedure. After the initial front is set up on the background grid, the following steps are repeated while any front edge exists:

1) Search for the shortest front edge that is used as a base for the triangle to be generated from the list of front edges.

2) Generate a new triangular cell properly in consideration of already created nodes, lengths of neighboring front edges, local curvature, and so on.

3) Remove the selected front edge from the list and add the new front edges to the list if generated.

Surface triangulation for three-dimensional surfaces is done using the direct advancing front method[6] with several important improvements in the reliability and efficiency by the following modification of the algorithm, which is how to determine a new temporary node position in item 2. First, the new node position $\boldsymbol{r}_c^*$ is determined as follows:

$$\boldsymbol{r}_c^* = \boldsymbol{r}_M + \delta(\bar{\boldsymbol{n}}_M \times \bar{\boldsymbol{s}})$$

where $\boldsymbol{r}_M$ denotes the position vector of the midpoint M of the front edge AB, $\bar{\boldsymbol{n}}_M$ denotes the unit normal vector of the cell of the background grid that contains the midpoint M, and $\bar{\boldsymbol{s}}$ denotes the unit side vector of the edge AB (Fig. 8). A length δ is determined with the curvature of the background grid at the point, then the node is mapped to the background grid to verify that it is on the surface. There are, however, many cases where the node is not located on any cell. By gradually shortening the length δ, we can obtain the node on the surface without failure in these cases.

The maximum and minimum lengths of the grid are specified at each closed region surrounded by the initial front edges. The surface meshing process is then advanced, for example, as shown in Fig. 9.

E. Surface Recovery

The new node position of the surface grid is temporarily determined with a first-order correction not to fail in the triangulation.

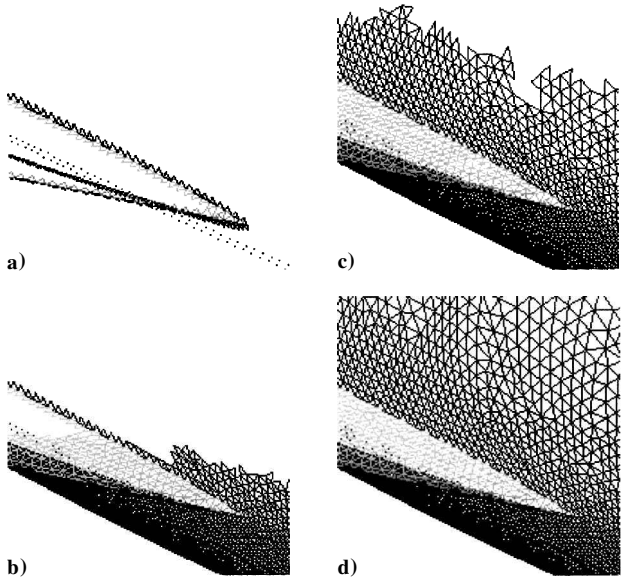
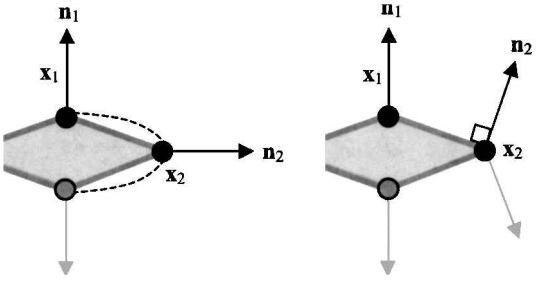


Fig. 9 Surface meshing process for an NAL experimental supersonic airplane model at the junction of the wing leading edge and the fuselage; the front is advanced from panel a to panel c, and the triangulation is terminated at panel d.



a) Quadratic recovery **b) Surface recovery on a ridge**

Fig. 10 Surface recovery.

The generated surface grid may not, however, represent the original configuration accurately if the STL-defined original data is not fine enough. To improve the grid quality, a surface recovery algorithm is applied after the surface triangulation.

The recovered point location is usually given by a quadratic triangle shape function. In Ref. 7, a midpoint location $\boldsymbol{x}$ is estimated from a Hermitian polynomial as

$$\boldsymbol{x} = 0.5\boldsymbol{x}_1 + 0.125\boldsymbol{r}_1 + 0.5\boldsymbol{x}_2 - 0.125\boldsymbol{r}_2$$

where

$$\boldsymbol{r}_i = |\boldsymbol{s}|\frac{\boldsymbol{n}_i \times (\boldsymbol{s} \times \boldsymbol{n}_i)}{|\boldsymbol{n}_i \times (\boldsymbol{s} \times \boldsymbol{n}_i)|} \qquad (i = 1, 2), \qquad \boldsymbol{s} = \boldsymbol{x}_2 - \boldsymbol{x}_1$$

where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are nodes location of the edge and $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are unit normal vectors at each node (Fig. 10a). Note that these nodes and edges belong to the background grid. However, if $\boldsymbol{x}_1$ or $\boldsymbol{x}_2$ is on a ridge such as a wing trailing edge, the obtained midpoint location, which is on the broken lines of Fig. 10a, is a great distance away from a background grid. This implies that the way to estimate the normal vector at each node, which is mentioned as follows, is very important:

1) Flag edges if the folding angles are larger than a certain value. The default value is 0 deg for boundary edges, 30 deg for the others.

2) At each node, select the cells and the flagged edges surrounding the node. The cells are divided into local zones by the flagged edges. The normal vector at the node is calculated at each zone.

3) Apply the quadratic triangle shape function for each cell. The normal vectors at three nodes depend on the cell to be applied as shown in Fig. 10b.

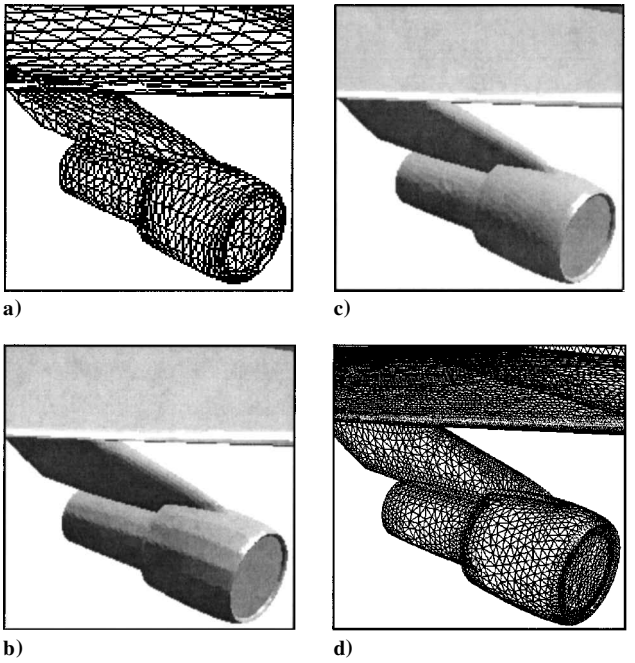


Fig. 11 Enlarged view of a Boeing 747-200 inboard nacelle: a) background grid (this grid is defined as a set of triangular panels and is very coarse), b) faced shading of the generated surface grid (the coarse background grid affects this grid), and c) and d) surface recovery (this smooth grid can be obtained after the surface recovery).

Figure 11 shows an example of the surface recovery for a Boeing 747-200 inboard nacelle. The background grid is very coarse, as shown in Fig. 11a, and it affects the generated surface grid, as shown in Fig. 11b. After the surface recovery, the surface grid, shown in Fig. 11c, becomes very smooth.

F. Outer Boundary and Volume Grid Generation

For volume grid generation in CFD applications, outer boundaries must also be created to cover the computational region. Here, the outer boundary generation is simplified just by choosing an appropriate boundary template, such as a half/full sphere, a half/full cone, and so on. After the outer boundary is generated, the volume grid is generated using Delaunay tetrahedral meshing.[13]

III. Applications

A. National Aerospace Laboratory Experimental Airplane Model

The unpowered National Aerospace Laboratory (NAL) experimental airplane will be launched to altitude of 15,000 m by a solid rocket booster in 2001 (Ref. 14). It was designed for aiming the natural laminar flow on the wing. The configuration data were prepared on CATIA CAD software and the STL output was generated. After the reconstruction process of the geometric features, as shown in Fig. 6, surface triangulation is performed (see Fig. 9).

The generated surface grid has 100,044 node points and 200,084 cells. The required CPU time on a Pentium III (900-MHz) personal computer is about 10 min for the surface grid generation process. This application required an additional 30–60 min for the interactive operations on a GUI screen, such as creating initial fronts in this case. By saving all of the preprocessing work in a file, a user can restart the surface grid generation from an intermediate step.

B. Blended-Wing-Body Airplane Model

The blended-wing-body (BWB) airplane was modeled on the CATIA CAD software as shown in Fig. 12. After the reconstruction of geometric features, several ridges were added at the engine nacelles to control the local grid density (Fig. 13). Surface triangulation was then performed as shown in Fig. 14.

The generated surface mesh has 46,933 nodes and 93,431 cells. The required CPU time on a Pentium III (900-MHz) personal computer is less than 1 min for the surface grid generation process in this case. Total required time for this surface mesh is about 1.5 h.

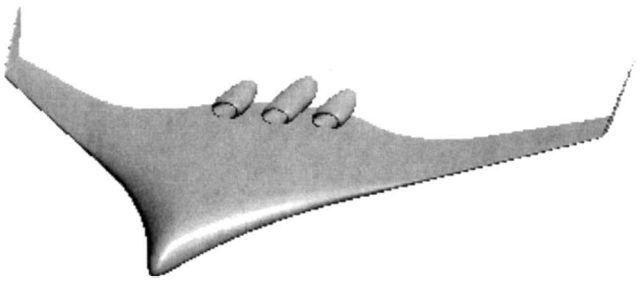


Fig. 12 STL data of BWB airplane.

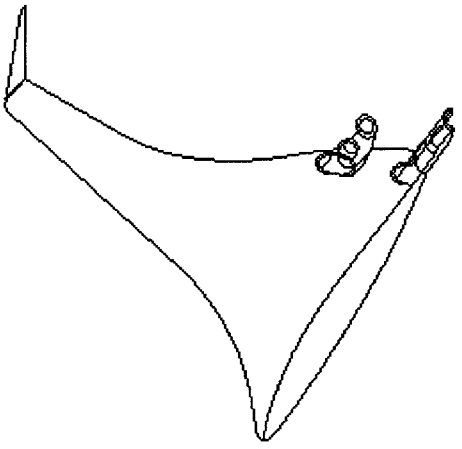


Fig. 13 Reconstruction of geometric features for BWB airplane.

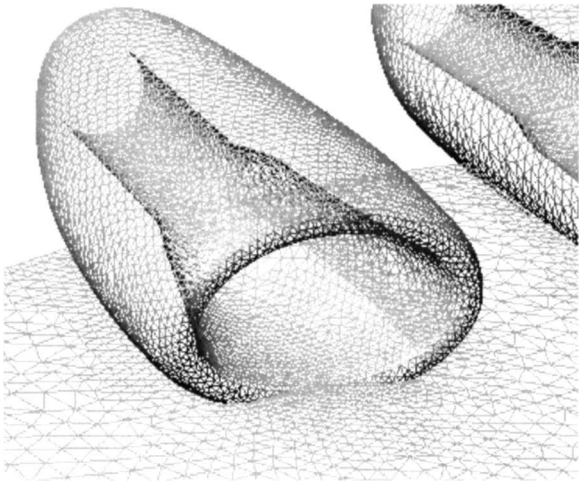


Fig. 14 Surface mesh of BWB airplane at the outboard engine nacelle.

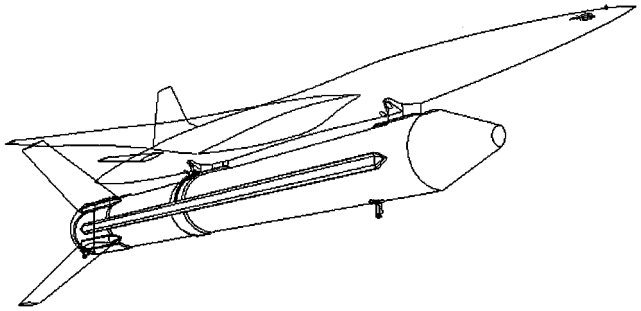


Fig. 15 Reconstruction of geometric features for the NAL experimental airplane model and the solid-rocket booster.

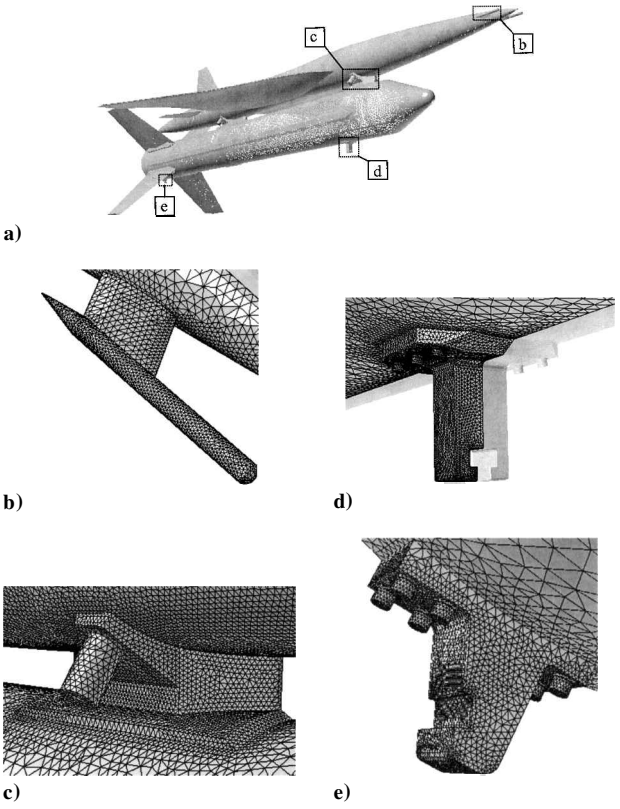
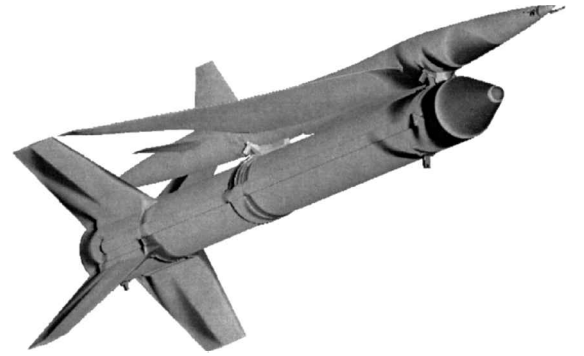
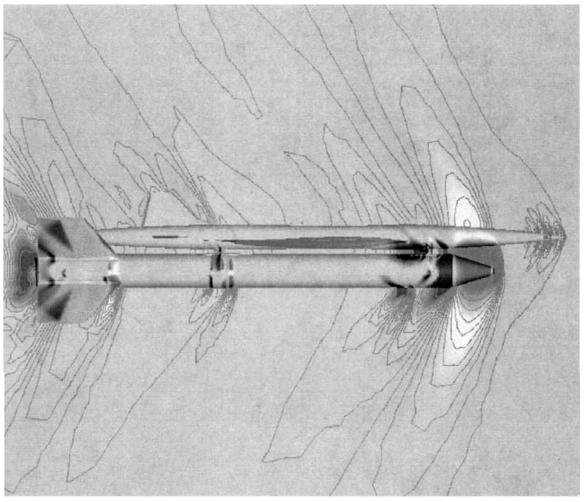


Fig. 16 Triangulation for the NAL experimental airplane and the solid-rocket booster: a) general view, b) pitot tube, c) front connection of the airframe and the rocket, d) front fitting, and e) rear fitting.



Surface pressure distribution



Pressure distribution on the symmetry plane

Fig. 17 Euler calculation for the NAL experimental airplane and the solid-rocket booster.

Most of the time has been spent specifying the division parameters at each ridge by trial and error.

C. NAL Experimental Airplane Model and Solid-Rocket Booster

Figures 15–17 show the CFD calculation process of the NAL experimental airplane piggybacked on a solid-rocket booster for launch. This airframe itself is the same one mentioned in Sec. III.A, but some detailed components, such as a pitot tube, are added. The reconstruction of geometric features and the addition of the ridges were performed as mentioned before (see Fig. 15). The number of the ridges becomes 501, and 213 closed regions are made of the ridges. This configuration is so complex that about 4 h were needed for mouse operation, such as the addition of ridges and specifying division parameters at each ridge. This half-model surface mesh, shown in Fig. 16, has 105,112 nodes and 208,497 cells. The required CPU time on a Pentium III (900-MHz) personal computer was about 3 min for the surface grid generation process.

Two types of the outer boundary shape were prepared for the calculation in transonic flowfields (hemisphere) and supersonic flowfields (half cone) to obtain better results. We needed only 10 additional minutes for each of the outer boundary generation processes. The volume grid, which consists of tetrahedra, was then generated by a Delaunay method. The half-sphere-type grid has 658,532 nodes and 3,570,327 tetrahedra.

Euler calculations were performed based on these volume girds.[15] Figure 17 shows the pressure distribution at a freestream Mach number of 1.2 and an angle of attack of zero based on the hemisphere-type grid.

IV. Conclusions

An efficient and effective surface triangulation method has been developed. STL data are employed to define a configuration as output of CAD system. Because the original STL data are not always suitable as a background grid for the surface meshing, ill-conditioned facets are automatically detected from the geometrical information and revised. The automatic reconstruction process of geometric features ensures accurate representation of the STL-defined geometry. The initial front setup based on the geometric features minimizes user interventions. The easy control of the surface grid density has been shown. The significant reduction of time for surface grid generation and the improvements in quality of surface grids were demonstrated.

Acknowledgments

The authors wish to thank Y. Shimbo and T. Iwamiya of the National Aerospace Laboratory of Japan for providing us the geometry data of the airplanes and the rockets. The authors also wish to thank T. Fujita, Graduate Student of Tohoku University, for his help in generating the stereolithography data.

References

[1] Hufford, G. S., Mitchell, C. R., and Harrand, V. J., "Trimmed NURBS, Unstructured Grids and CFD," AIAA Paper 96-1996, 1996.

[2] Kania, L., and Warsi, S., "Curvature Adapted Triangulation of NURBS Surface," *Proceedings of the 13th AIAA Computational Fluid Dynamics Conference*, AIAA, Reston, VA, 1997, pp. 790–799.

[3] Ladeinde, F., "Truly Automatic CFD Mesh Generation with Support for Reverse Engineering," AIAA Paper 99-0828, 1999.

[4] Woan, C. J., "Unstructured Surface Grid Generation on Unstructured Quilt of Patches," AIAA Paper 95-2202, 1995.

[5] *Stereolithography Interface Specification*, 3D Systems, 1989.

[6] Nakahashi, K., and Sharov, D., "Direct Surface Triangulation Using the Advancing Front Method," *Proceedings of the 12th AIAA Computational Fluid Dynamics Conference*, AIAA, Washington, DC, 1995, pp. 442–451.

[7] Löhner, R., "Regridding Surface Triangulation," *Journal of Computational Physics*, Vol. 126, No. 1, 1996, pp. 1–10.

[8] Löhner, R., "Extending the Range of Applicability and Automation of the Advancing Front Grid Generation Technique," AIAA Paper 96-0033, 1996.

[9] Sharov, D., and Nakahashi, K., "Curvature Adapted Triangulation of Surface Models via Incremental Insertion Algorithm," *Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations*, International Society of Grid Generation, 1998, pp. 695–704.

[10] Jacob, G. G. K., Kai, C. C., and Mei, T., "Development of a New Rapid Prototyping Interface," *Computers in Industry*, Vol. 39, No. 1, 1999, pp. 61–70.

[11] Vinokur, M., "On One-Dimensional Stretching Functions for Finite-Difference Calculations," *Journal of Computational Physics*, Vol. 50, No. 2, 1983, pp. 215–234.

[12] Morgan, K., Peraire, J., and Peiró, J., "Unstructured Grid Methods for Compressible Flows," *Special Course on Unstructured Grid Methods for Advection Dominated Flows*, Rept. 787, AGARD, 1992, pp. 5.1–5.39.

[13] Sharov, D., and Nakahashi, K., "A Boundary Recovery Algorithm for Delaunay Tetrahedral Meshing," *Proceedings of the 5th International Conference on Numerical Grid Generation in Computational Field Simulations*, International Society of Grid Generation, 1996, pp. 229–238.

[14] Iwamiya, T., "NAL SST Project and Aerodynamic Design of Experimental Aircraft," *Proceedings of the 4th ECCOMAS Computational Fluid Dynamics Conference*, Wiley, Chichester, England, U.K., 1998, pp. 580–585.

[15] Fujita, T., Ito, Y., Nakahashi, K., and Iwamiya, T., "Aerodynamics Evaluation of NAL Experimental Supersonic Airplane in Ascent Using CFD," AIAA Paper 2001-0564, 2001.

J. Kallinderis
Associate Editor