INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2013; **71**:891–909
Published online 25 May 2012 in Wiley Online Library (wileyonlinelibrary.com/journal/nmf). DOI: 10.1002/fld.3691

Hybrid grid generation for viscous flow analysis

Seyoun Park[1], Byungduk Jeong[1], Jin Gyu Lee[2] and Hayong Shin[1,*,†]

[1]*Department of Industrial and Systems Engineering, KAIST, Daejeon, South Korea*
[2]*R&D Institute Agency for Defense Development, Deajeon, South Korea*

SUMMARY

Cartesian grid with cut-cell method has drawn attention of CFD researchers owing to its simplicity. However, it suffers from the accuracy near the boundary of objects especially when applied to viscous flow analysis. Hybrid grid consisting of Cartesian grid in the background, body-fitted layer near the object, and transition layer connecting the two is an interesting alternative. In this paper, we propose a robust method to generate hybrid grid in two-dimensional (2D) and three-dimensional (3D) space for viscous flow analysis. In the first step, body-fitted layer made of quadrangles (in 2D) or prisms (in 3D) is created near the object's boundary by extruding front nodes with a speed function depending on the minimum normal curvature obtained by quadric surface fitting. To solve global interferences effectively, a level set method is used to find candidates of colliding cells. Then, axis-aligned Cartesian grid (quadtree in 2D or octree in 3D) is filled in the rest of the domain. Finally, the gap between body-fitted layer and Cartesian grid is connected by transition layer composed of triangles (in 2D) or tetrahedrons (in 3D). Mesh in transition layer is initially generated by constrained Delaunay triangulation from sampled points based on size function and is further optimized to provide smooth connection. Our approach to automatic hybrid grid generation has been tested with many models including complex geometry and multi-body cases, showing robust results in reasonable time. Copyright © 2012 John Wiley & Sons, Ltd.

Received 22 November 2011; Revised 15 March 2012; Accepted 30 April 2012

KEY WORDS: mesh generation; computational fluid dynamics; FVM; hybrid grid; viscous fluid

1. INTRODUCTION

Grid generation is a troublesome area in numerical analysis. This is particularly true of the field of CFD where automatic creation of high quality mesh in a robust fashion still remains a difficult and time-consuming step. The quality and density of the grid used in CFD directly affects the accuracy of the numerical solution. However, an unnecessarily fine grid slows down computation and the arrival at a solution [1].

Grid types in CFD analysis can be grouped in two categories: structured grid and unstructured grid. The pros and cons of each type are well discussed in the literatures such as Baker [2]. It is believed that a structured grid created in body-fitted fashion is superior for capturing viscous flow near the objects' surface because of the orthogonality of elements to the dominant flow direction [3]. However, a structured grid makes unnecessarily dense elements in areas far from focal objects to achieve the resolution required in the critical part of the domain. Moreover, it is not a simple task to automatically create a body-fitted structured mesh for objects with complex geometry. For this reason, a hybrid mesh consisting of a body-fitted structured mesh layer and an unstructured background mesh has gained the attention of CFD researchers for viscous flow analysis in recent years [3–13] including its use in several commercially available software programs such as VisCART, Harpoon, and GDT. Chand [4] proposed a structured–unstructured hybrid mesh for overlapping multi-body

*Correspondence to: Hayong Shin, Department of Industrial and Systems Engineering, KAIST, Daejeon, South Korea.
†E-mail: hyshin@kaist.ac.kr

objects. In [5–7], Wang and his colleagues reported their work on solving the moving body problem using a hybrid grid with quadrangle/triangle/Cartesian elements, as is the target of this paper. Their work proposed an efficient time-integration algorithm for a moving object (unsteady) flow problem discretized with a hybrid grid and showed the advantages of using a hybrid grid. However, their work focuses on efficient numerical solutions, and the hybrid grid generation method is only briefly introduced Also, thinner portions of the viscous layer are generated at the concave regions to avoid local interferences of the propagating nodes Dawes *et al*. [8, 9] also introduced a hybrid mesh with a viscous layer based on Cartesian cells as background mesh. They projected the nodes on the Cartesian cells to the viscous layer and inserted unstructured cells where these two layers met to connect them, To do this work without skewed unstructured cells, the surface mesh should have a uniform size similar to Cartesian cells. Table I shows the comparison of several existing methods used to generate grids for viscous flow analysis using mixed types of cells.

In this paper, we will focus on the robust generation of a hybrid mesh with high quality based on adaptive Cartesian grid in detail, which can handle two-dimensional (2D) and two-dimensional (3D) objects in the same fashion. As shown in Figure 1, we automatically generate a hybrid grid consisting of the following three layers : a body-fitted layer (quadrangles in 2D and prisms in 3D), a transition layer (triangles in 2D and tetrahedrons in 3D), and a Cartesian layer (quadtree in 2D and octree in 3D). We aim to develop a method that can ensure more than a certain thickness of viscous layer on the whole input surface regardless of convexity and smooth transition between two structured layers.

Our mesh has smaller number of cells than the approaches using body-fitted cells and tetrahedrons [4, 6, 10] because of an adaptive Cartesian mesh. Unlike several other approaches such as [5], which stop stacking body-fitted cells and fill with unstructured cells in that region, the body-fitted layer avoids local interferences and global interferences in a systematic way by using level set approach without losing parametric information of the surface boundary. In a transition layer, our approach allows arbitrarily sampled points in a space, and this means we can generate tetrahedral cells denser, where we want, regardless of surface mesh quality, unlike a general advancing front method. Even though an initial triangulation result has many skinny cells, the optimization step improves the quality and gives good result experimentally.

Table II summarizes the mesh element types and methods used for each layer, each of which will be explained in the subsequent sections. Section 2 will overview the overall procedure to create a hybrid grid, followed by detailed methods for the body-fitted layer, the Cartesian layer, and the transition layer in Sections 3, 4, and 5, respectively. Section 6 includes resulting meshes and computation time, and the conclusion is in Section 7.

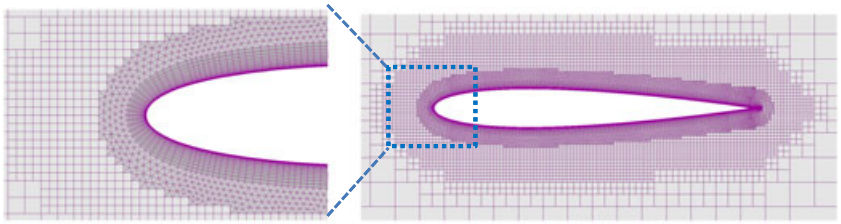
2. OVERALL PROCEDURE

As mentioned earlier, one of the advantages of our approach is that it can be applied to 2D and 3D within the same framework. Although the remainder of this study will deal primarily with 3D cases, it can be applied to 2D cases in the same way with the exception of the curvature estimation section, which will be explained in Section 3.

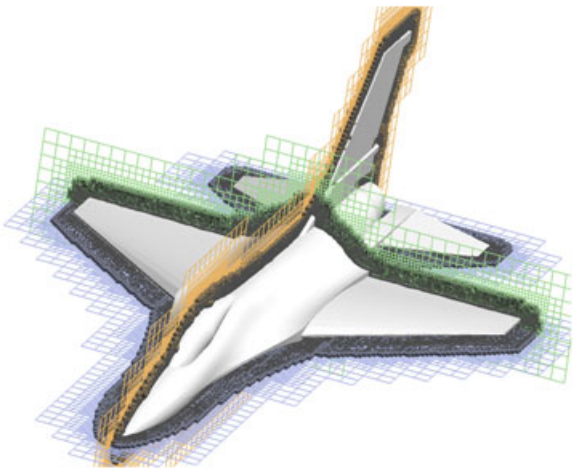
Table I. Hybrid approaches for viscous grid generation.

Group	Major cell type	Approach	Dimension
Zhang *et al.* [5]	Quad/Tri/Cartesian	Advancing front method	2D
Wang *et al.* [6]	Hexa/Tetra	Advancing front method	3D
Khawaja *et al.* [10, 11]	Prism/Tetra	Automatic receding + advancing front method	3D
Dawes *et al.* [8, 9]	Hexa/Cartesian	Level set method	3D
Gloth *et al.* [12]	Quad/Tri/Cartesian	Level set method	2D
Pattinson *et al.* [13]	Cartesian	Cut-cell, dual-mesh	2D
Ebeida *et al.* [3]	Quad/Tri	Spatial decomposition	2D

DOI: 10.1002/fld



(a) Hybrid mesh of the 2D NACA0012 model; the darkest region is the body-fitted layer, the brightest region is the Cartesian layer and the middle layer is the transition layer



(b) Hybrid mesh of a 3D fightter airplane model

Figure 1. Hybrid grids.

Table II. Element types and generation methods for each layer.

		Body-fitted layer	Transition layer	Cartesian layer
Data Structure	2D	Quadrangles	Triangles	Quadtree
	3D	Prisms	Tetrahedrons	Octree
Method		Level set	Size function based optimization	Scan conversion

We assume that the input object is given as a piecewise linear representation, such as the point sequence curves in 2D and triangular meshes in 3D. Let $B = \{ \Delta_1 \ldots, \Delta_t \}$ denote the input object's boundary, and the spatial region of concern where we need to generate cells may be either the inside or the outside of the objects. Let $S = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [z_{\min}, z_{\max}]$ denote the spatial area of interest within which mesh is to be made. C = <**BC**, **TC**, **CC**> where **BC** represents the set of cells in the body-fitted layer, **CC** is the set of Cartesian cells, and finally **TC** denotes the set of cells in the transition layer.

Figure 2 shows the overall procedure for generating hybrid mesh. The three primary sequential steps to generating the mesh for viscous flow analysis are as follows :

- *[Body-fitted layer generation]* The first step is to generate the body-fitted structured layer directly touching the surface of the object. This layer is composed of quad-cells in 2D and prisms in 3D. This layer is stacked outward from the objects' surface boundary, until the pre-defined conditions are satisfied. We adopted the level set approach for this step.
- *[Cartesian layer generation]* The next step is to create an adaptive Cartesian layer generated by using a quadtree/octree structure. This is a trivial task with a certain amount of offset model from the input geometries.

DOI: 10.1002/fld

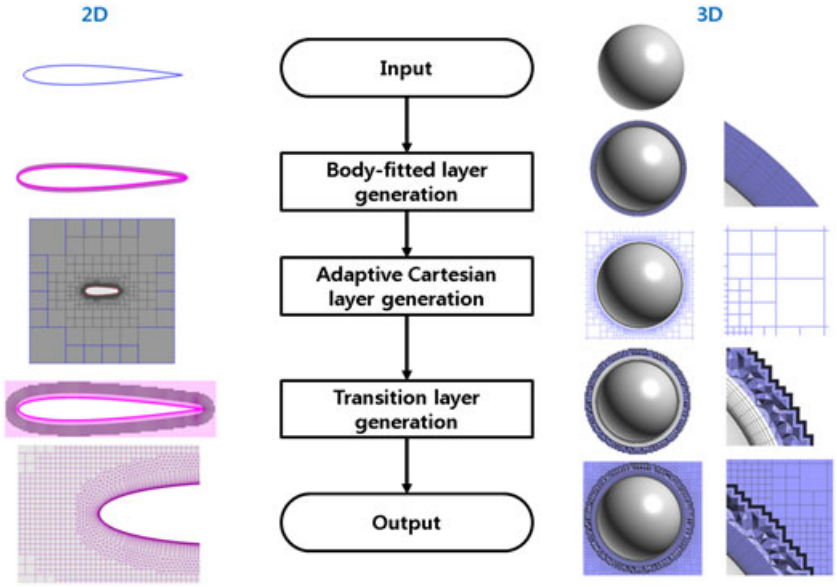


Figure 2. Overall procedure.

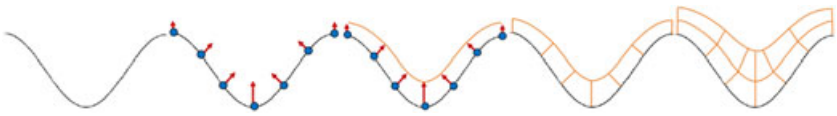


Figure 3. Grid generation with a concave curve.

- *[Transition layer generation]* The final step is to smoothly fill in the area between the two differently oriented structured layers with unstructured grids. To maintain the faces already created by the two structured meshes, we need to solve a constrained problem. Also, in order to build up the mesh as regularly shaped as possible, we apply an optimization step to the initial construction.

Again, Figure 1(a) shows these three kinds of layers for the NACA0012 model in 2D and Figure 1(b) shows the 3D result of a fighter airplane model.

3. BODY-FITTED LAYER GENERATION

In this paper, the target body-fitted layer mesh should satisfy the following properties as with other previous studies:

- The interlayer spacing is dense enough near the surface boundary to satisfy the required accuracy for the viscous fluid, and farther from the surface, the interval is wider.
- Fronting nodes propagate along the normal vector directions of a current layer.
- Each layer avoids local interferences at concave regions.
- Each layer avoids global interferences between multiple separated objects.

To satisfy the third condition, our solution is to make the propagation amount at each node according to its curvature value. Figure 3 shows the conceptual way to avoid local interferences with a 2D concave curve.

Those properties can be satisfied systematically by applying the level set method with curvature correction term. The level set [14, 15] is the mathematical model, which describes the behavior of fronting boundaries varied by time. The concept of level set is simple and there are many applications in CFD [34]. Given an interface $\Gamma(t)$ in R^n (n=2, 3) bounding an open region $\Omega(t)$ and moving along time t, let an implicit function $\Phi(\mathbf{p}, t)$ have the signed distance from $\mathbf{p}$ to $\Gamma(t)$, and

DOI: 10.1002/fld

the plus sign is chosen if $\mathbf{p}$ is outside $\Omega(t)$. With a speed function F, which gives the speed of $\Gamma(t)$ in its normal direction, the evolution of the interface is defined by the level set equation shown in Equation (1).

$$\Phi_t + F|\nabla\Phi| = 0 \tag{1}$$

In our case, the initial interface $\Gamma(0)$ is given as B. The term F governs the actual behavior of the evolving boundary [17]. This may be different in application, and in our case, we adopted F as Equation (2).

$$F(\mathbf{p}, t) = f(t) \cdot (F_A + F_G(\mathbf{p})) \tag{2}$$

where F_A is the advection term causing the interface to uniformly expand, usually set as 1, and $f(t) = f_0 * a^t$, where f_0 and a are user-defined constants, and the a^t term accelerates the propagation velocity by time to satisfy the first property. When L is the longest axial length of the axis-aligned bounding box of the input object, we set $f_0 = 10^{-5}L$ and a= 1.2 as default values in subsequent examples.

F_G is a geometric term depending on the geometric property of the propagating fronts, and here, we set $F_G(\mathbf{p}) = -\epsilon\kappa(\mathbf{p})$ where $\kappa(\mathbf{p})$ is the minimum normal curvature at $\mathbf{p}$ to make the fronting nodes in the concave region faster and the nodes in convex region slower. ϵ is the user-defined parameter, and we used 0.1 as a default value. F_G term makes the layer avoid local interferences, and makes curvature values at the same isocurve/isosurface similar to each other, farther from the boundary of the input model as shown in Figure 4.

3.1. Numerical implementation

The level set method is generally implemented by extracting isosurfaces from a scalar field evaluated along time evolution on a finite grid rather than extruding boundary nodes directly. Even though the level set method has nice properties, there are two kinds of practical shortcomings to apply to the viscous layer. First, it is very hard to use parametric information of boundary surfaces, which is critical in general to numerical solvers because we do not use boundary nodes directly. The other significant problem is time and memory efficiency. Because we generated extremely thin layer near the body wall as mentioned above, we need too many finite grid cells or too much iterative computations with manageable number of cells.

In this paper, we adopt the following strategy. We generate a body-fitted layer by moving forward front nodes along their normal directions directly following Equation (3). This is the first-order approximation of $\dot{\mathbf{p}} = F(\mathbf{p}, t) \cdot \nabla\Phi(\mathbf{p}, t)$

$$\mathbf{p}' = \mathbf{p} + \Delta t \cdot F(\mathbf{p}, t) \cdot \nabla\Phi(\mathbf{p}, t) \tag{3}$$

Even though local interferences do not appear by F term and small enough value of Δt, we cannot avoid global interferences as shown in Figure 5. We adopt a hybrid mesh in this system, find the

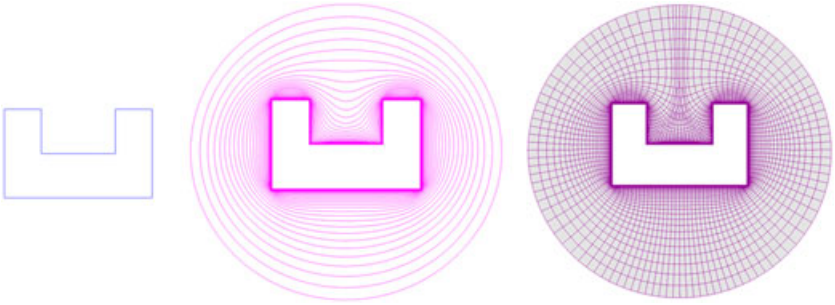


Figure 4. Grid generation using level set; input object(left), isosurface extraction(middle), grid generation by linking fronting nodes(right).

DOI: 10.1002/fld

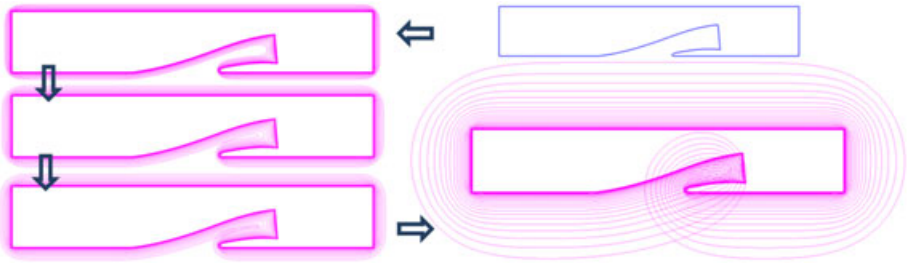
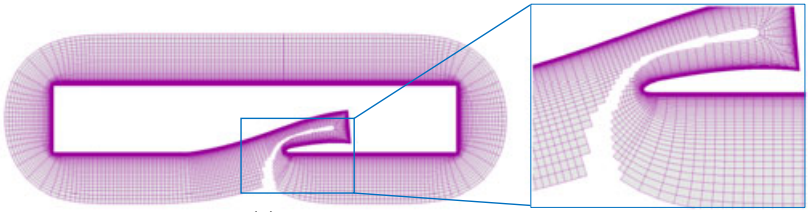
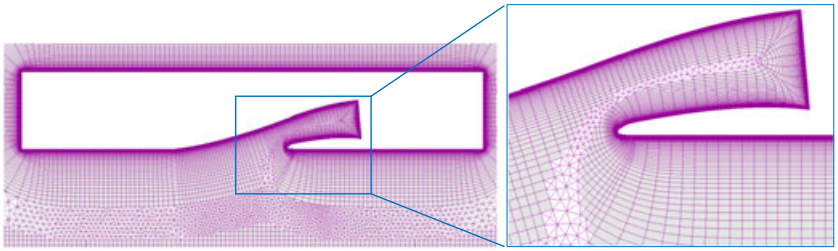


Figure 5. Global interferences (a sectional view of a submerged inlet).



(a) Interference removal using a level set



(b) Hybrid mesh generation

Figure 6. Hybrid mesh generation using level set.

intersecting cells and remove after marching one layer. The blank space is filled with unstructured cells at the transition layer generation step as shown in Figure 6. The most naïve method is to check all the paired combinations of cells, which takes too much time especially in 3D. To find the cells colliding by global interferences efficiently, we apply the *real* level set implementation, and early exclude non-intersecting cells.

Here, we do not need very small cells or an infinitesimal time step to satisfy the accuracy for viscous flow, but only need moderate size of finite difference grid, and we use one-step or two-step bigger size than the minimum cells of the quadtree/octree used for Cartesian cell generation in the following step. Where the interface collides, there is discontinuity in gradient, $\nabla\Phi$, and then numerically $\nabla^2\Phi \to -\infty$ near that region. This is called local Laplacian, and Xia *et al.* [18] suggest to find the cells with $\nabla^2\Phi < -\delta(\delta > 0)$ where $\delta = |\min\{\nabla^2\Phi\}|/10$. We apply this approach to find candidate region of collision using a quadtree/octree as a finite difference grid numerically.

The total procedure for body-fitted layer generation is shown in Figure 7. Let n be the number of body fitted layers to generate and T=$\{t_1, \ldots, t_n\} = \{1, \ldots, n\}$ the set of generation time of each layer. From t=0, time t is updated as $\Delta t(<1)$ at every iteration. At each iteration, we calculate speed at each node on $\Gamma(t)$ using Equation (2), then calculate a Euclidean distance field as Φ for the narrow band [19] of $\Gamma(t)$ for computational efficiency. Update Φ and find finite difference cells where $\nabla^2\Phi < -\delta$ and mark edges(2D)/faces(3D) on $\Gamma(t + \Delta t)$ passing these cells. An additional body-fitted layer is only generated when $t = t_i$, and intersection test is conducted for the cells from marked edges/faces after generating a layer. Whole steps are conducted iteratively until n layers are generated

DOI: 10.1002/fld

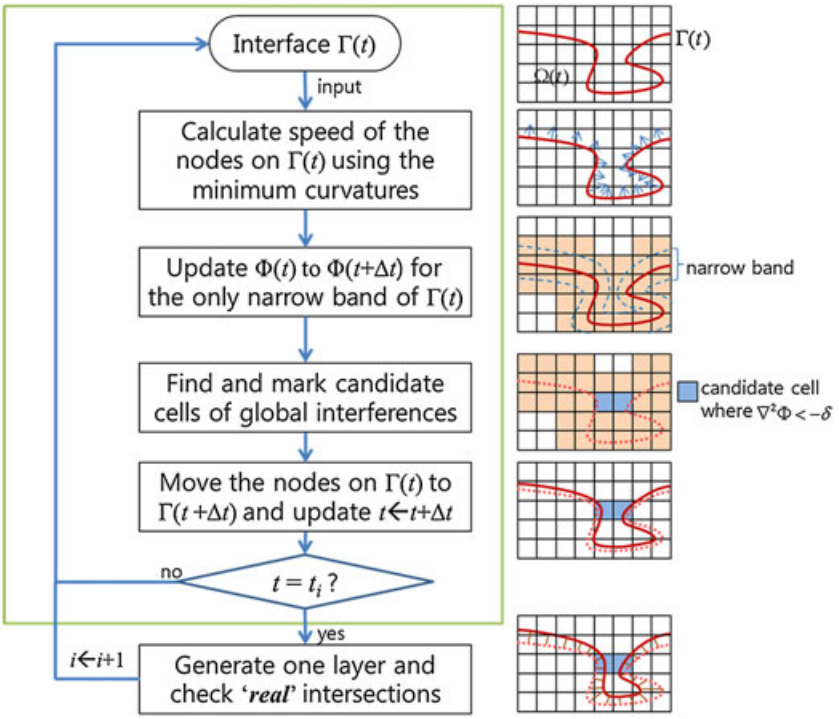


Figure 7. The overall procedure; steps in a green box are repeated until the time t reaches the predefined time step $t_i\,(i = 1,..n)$.

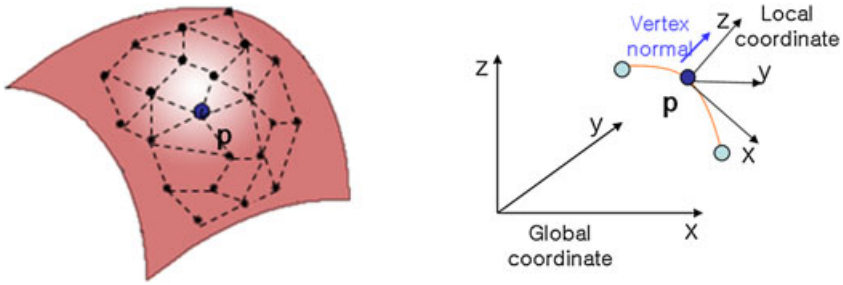


Figure 8. Quadric surface fitting (3D case).

3.2. *Curvature estimation of surfaces in 3D*

When we compute mesh structure using a level set, it can be more valuable to obtain the trends of large and small relationships rather than the exact curvature values. In our work, we calculated curvature by quadric surface fitting in order to obtain the values robustly, and the quadric surface model is simplified for computational efficiency [20]. As shown in Figure 8, we assume the local coordinate whose origin is the vertex **p** and the z-axis is the direction of the normal vector of **p**. Further, the tangent plane of the quadric surface at the **p** becomes local xy-plane.

The equation of the quadric surface at the local coordinate is represented as Equation (4). Note that there are no x, y and constant terms because the tangent plane of the surface equals xy-plane at the origin.

$$z = f(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & c \\ c & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = ax^2 + by^2 + 2cxy \tag{4}$$

We fit a quadric surface, represented as Equation (4), at each vertex **p** in B with its *two-ring* neighbor points by weighted least squares method whose weights are the inverse function of the distance

DOI: 10.1002/fld

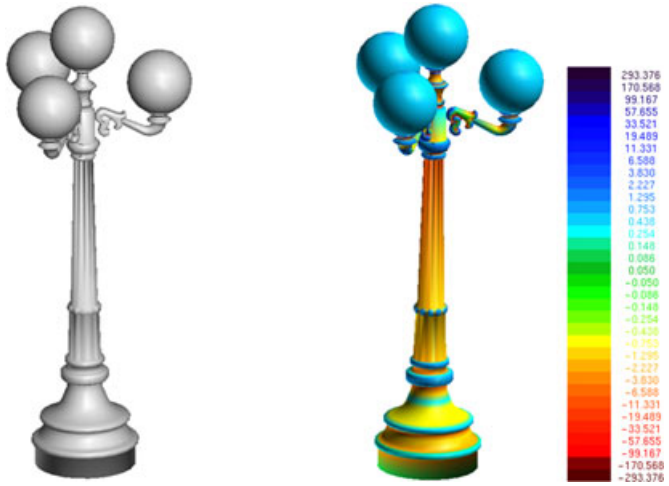


Figure 9. Minimum curvature calculation; an input geometry of a lamp (left) and its rendering colored by minimum curvature values (right).

between **p** and its neighbor vertex. After fitting a local surface, we can easily calculate the curvature at **p**. With a continuous parametric surface, the normal curvature is defined as Equation (5) [21].

$$\kappa = \kappa(\mathrm{d}x, \mathrm{d}y) = \frac{L\mathrm{d}x^2 + 2M\mathrm{d}x\mathrm{d}y + N\mathrm{d}y^2}{E\mathrm{d}x^2 + 2F\mathrm{d}x\mathrm{d}y + G\mathrm{d}y^2} \tag{5}$$

In our case, the constants E, F, G, L, N, and M are calculated as Equation (6) from surface formula (Equation (4)).

$$\begin{aligned}
E|_{(x.y)=(0,0)} &= \mathbf{P}_x \cdot \mathbf{P}_x = (x\mathbf{i} + (2ax + 2cy)\mathbf{k}) \cdot (x\mathbf{i} + (2ax + 2cy)\mathbf{k}) = 1 \\
F|_{(x.y)=(0,0)} &= \mathbf{P}x \cdot \mathbf{P}y = (x\mathbf{i} + (2ax + 2cy)\mathbf{k}) \cdot (y\mathbf{j} + (2by + 2cx)\mathbf{k}) = 0 \\
G|_{(x.y)=(0,0)} &= \mathbf{P}y \cdot \mathbf{P}y = (y\mathbf{j} + (2by + 2cx)\mathbf{k}) \cdot (y\mathbf{j} + (2by + 2cx)\mathbf{k}) = 1 \\
L|_{(x.y)=(0,0)} &= \mathbf{n} \cdot \mathbf{P}_{xx} = (\mathbf{k}) \cdot (2a\mathbf{k}) = 2a \\
M|_{(x.y)=(0,0)} &= \mathbf{n} \cdot \mathbf{P}_{xy} = (\mathbf{k}) \cdot (2c\mathbf{k}) = 2c \\
N|_{(x.y)=(0,0)} &= \mathbf{n} \cdot \mathbf{P}_{yy} = (\mathbf{k}) \cdot (2b\mathbf{k}) = 2b
\end{aligned} \tag{6}$$

where $\mathbf{P} = x\mathbf{i} + y\mathbf{j} + z\mathbf{k} = x\mathbf{i} + y\mathbf{j} + f(x, y)\mathbf{k}$. By substituting Equation (6) for Equation (5), the curvature becomes Equation (7), which is a function of dx and dy terms. We can directly compute the maximum or minimum curvature value from the equation. After dividing numerator and denominator by $\mathrm{d}x^2$, Equation (7) becomes the function of $t = \mathrm{d}y/\mathrm{d}x$ term, which means the direction at the tangent plane to calculate curvature. Then, we can find the minimum and maximum curvatures by differentiating $\kappa(t)$, and we use the minimum curvature to calculate F_G in Equation (2).

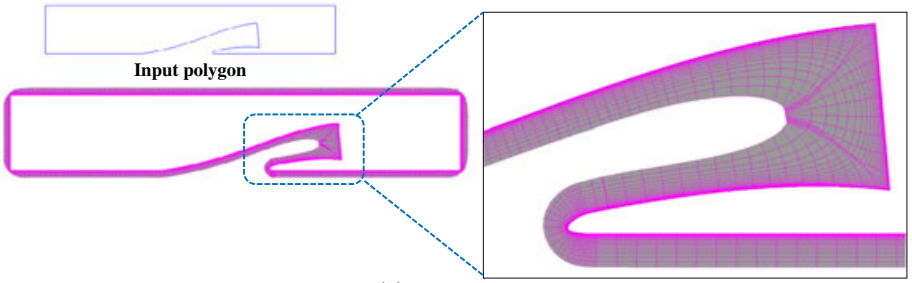
$$\kappa = \kappa(t) = \frac{2a + 4ct + 2bt^2}{1 + t^2} \tag{7}$$

Figure 9 shows a lamp model colored by the minimum curvature at each vertex.

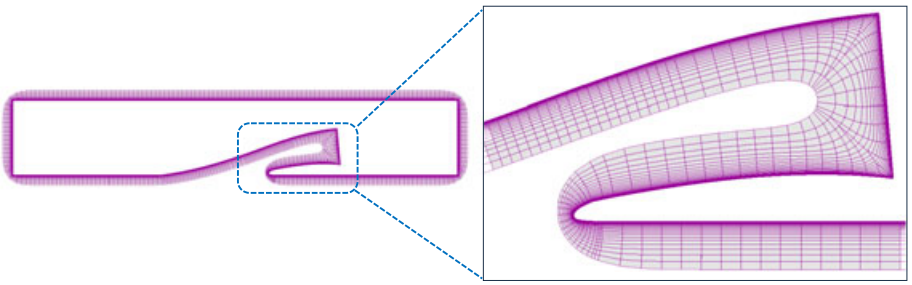
3.3. Density control for quality improvement

One effect that appeared when using level set approach is that the node points propagating at the concave region get closer as layers are stacked up as shown in Figure 10. This generates very thin cells, and they can affect the efficiency or convergence of the solver. On the other hand, the nodes at the convex region get farther from each other. This kind of effect can degrade the accuracy of the solution. This problem has been handled in the aspect of quality improvements [22–24]. Especially, Chalasani *et al.* [24] proposed a way to refine faces at convex regions and to collapse edges

DOI: 10.1002/fld



(a) Before merging



(b) After merging nodes

Figure 10. Density control result at concave region.

based on aspect ratios; however, in our case, we only deal with vertex split and merge rather than edge or face split because it is more efficient to maintain mapping information between vertices for many solvers. To cover these cases, we simply merge the nodes whose distances are within a user-defined threshold, and split a node at the low density region in a layer. Figure 10 shows the difference between the result of density control before and after.

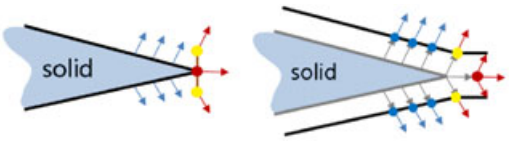
At the specific feature with special properties such as the end of the aircraft's wing, where the flow converges, it is hard to get a good solution in the manner of simply splitting a node. In this case, we insert multiple pseudo-edges and expand the centric node, the red point in Figure 11, to the middle direction of the normal vectors of the two actual neighbor nodes. Users can select this special node and define the expand direction before mesh generation. In Figure 11, we insert two pseudo-edges at the sharp corner (red point) of the NACA0012 model, and expand the centric node to the average direction of normal vectors of the two neighbors, and propagate two end nodes (yellow points) in the pseudo-edges to the normal vectors of the fronting nodes in the same direction. This scheme is applied iteratively to the central node until the propagation of the body-fitted layer ends. The following Figure 11(b) shows a result mesh with a trailing edge of the NACA0012 model.

We apply the similar approach for 3D cases. There exist similar multi-normal split approaches for grid generation [25, 26] and for surface offset problem [27]. Our method is a simplified version of Kim's work [27] and only applied to convex areas. At a sharp convex edge where its dihedral angle is less than the user-defined threshold, we split the edge normal to three vectors; two of them are neighbor-face normal vectors, and the other is an average of them as the default value. At each node, all the connected edges are checked one by one whether it is sharp or not, and if true, then the vertex normal is additionally split to the three sharp edge normal vectors. Note that the original vertex normal vector remains as it is. Figure 12(a) shows the conceptual way in 3D case, and Figure 12(b) shows an example with a simple box model.

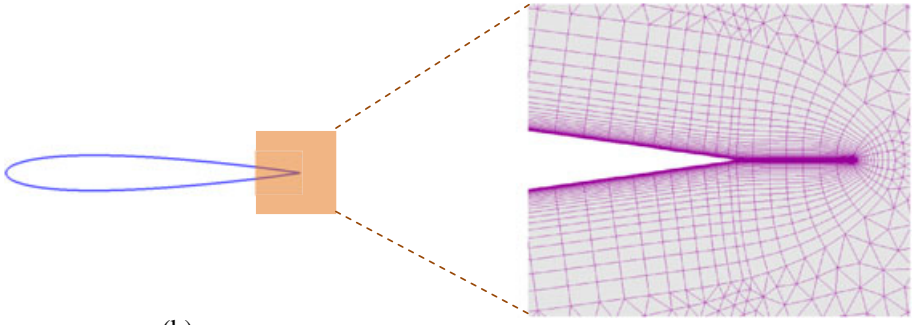
4. ADAPTIVE CARTESIAN LAYER GENERATION

To make a Cartesian mesh a certain distance away from the surface of the object, first we offset B as much as d and obtain B*. This is simply obtained by extruding nodes using Equation (6) from the last layer of the previous step as much as the remaining distance, Here, we do not need to consider

DOI: 10.1002/fld

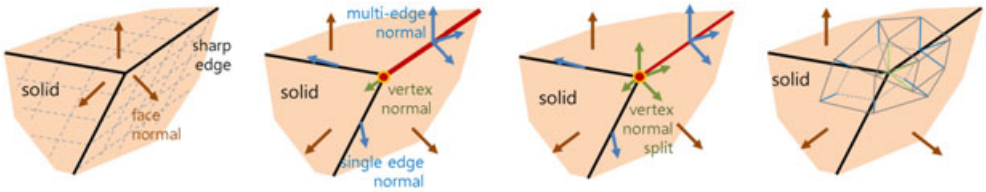


(a) 2D case; red edges means pseudo-edges

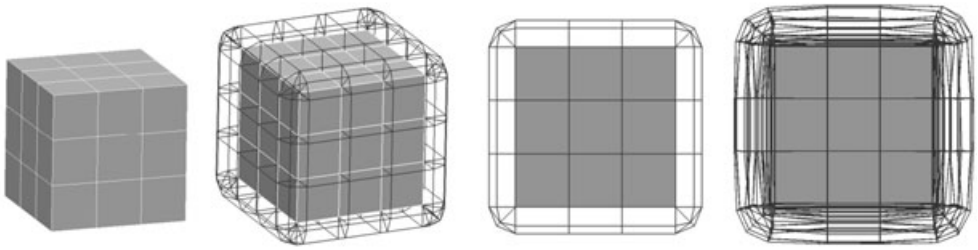


(b) Trailing edge handling with the NACA0012 model

Figure 11. Multi-spilt nodes in a 2D case.



(a) 3D case; a vertex is split only when it is connected to a sharp edge



(b) 3D case; a box input, two different views of one layer and 5 layers generation (from left to right)

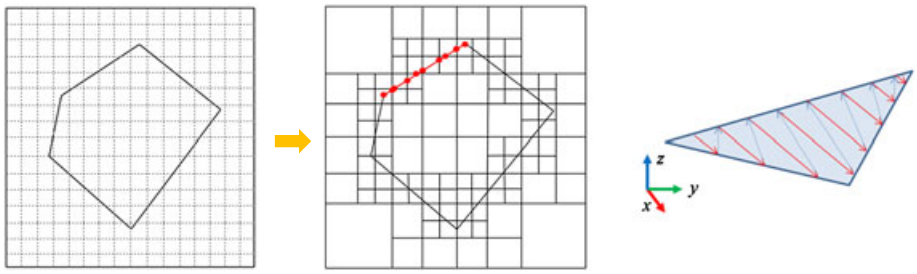
Figure 12. Multi-split examples in a 3D vase.

local and global interferences because the purpose of this step is to obtain cells outside B* and our approach only collect the outside cells of the outermost boundary of B* regardless of its topology or inner intersections.

We made Cartesian cells using a quadtree structure in 2D and an octree structure in 3D. This is a trivial task even though there are several issues with generation such as a top-down approach or bottom-up construction. We applied the scan conversion method shown in Figure 13 to make a quadtree or an octree through a bottom-up method. A 2D case is made up by the line scan conversion [28], and a 3D case can be done by a polygon scan conversion algorithm [29].

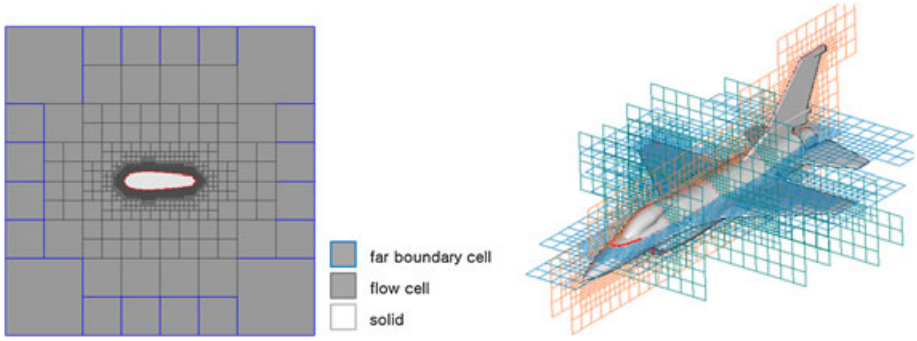
Figure 14 shows the 2D and 3D adaptive Cartesian cells obtained from a quadtree with the NACA0012 model. Only the gray-shaded cells are used for analysis. These cells are collected by a simple flood fill algorithm propagating from any seed cell outside B*

DOI: 10.1002/fld



(a) 2D line scan conversion (b) 3D polygonal scan conversion

Figure 13. Scan conversion in 2D (left and middle) and in 3D (right).



(a) A quadtree of a NACA0012 model in 2D (b) Sectional planes of the octree in 3D

Figure 14. Cartesian layer.

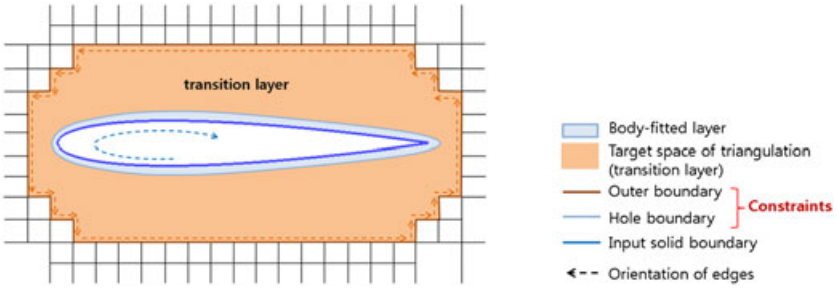


Figure 15. Transition layer.

5. TRANSITION LAYER GENERATION

In order to connect two types of grids with different orientations, we use a relatively easy-to-generate unstructured mesh like Zhang *et al.* [5]. However, in the case of the traditional advancing front method used in [5], it is hard to control the density of the points. We used the size function based mesh generation approach [30] especially introduced by Person *et al.* [31, 32] to get over this problem. This is very simple and produces good quality. Also, it can be applied to 2D and 3D case equally.

Our approach to generate cells in the transition layer is divided into two steps. First, we generate the scalar field whose value represents a cell size in the transition layer, and initial points are sampled based on this size function. All the nodes move forward until they meet the Cartesian layer. Then, we get an initial triangulation by constrained Delaunay triangulation with sampled points, and finally, we optimize the triangular mesh from the initial result. This step is repeated iteratively until the nodes become stable. Figure 15 shows geometric features in the transition layer in our problem.

DOI: 10.1002/fld

5.1. Size function and initial point sampling

Before making triangular cells, first, we need points sampling in the transition layer. To sample points, we calculate a scalar field in the transition layer by linearly combining the length of the nearest edge from the generation point and the minimum cell size of the Cartesian layer. Euclidean distances from two layers are used as weight of linear combination. Figure 16 shows the size function distribution by colored map display. Where the function value is low, we need dense points and vice versa. Basically we sample points by iteratively marching nodes on the most exterior curve/surface of the body-fitted layer along their normal directions as much as the size values rather than the speed function until all the nodes reach the Cartesian cells. Figure 17 shows the sampling result in the transition layer of the NACA0012 model.

5.2. Triangular/Tetra mesh generation

When we make triangular cells from the point samples, we need to maintain the faces already made from two structured layers in the process of creating the most regularized combination. This kind of problem is known as constrained Delaunay triangulation (CDT), where *Delaunay triangulation* means the triangulation method to maximize the minimum angle of all the angles of the triangles in the triangulation. The constraints of our problem are the boundary segments(2D)/faces(3D) from the Cartesian layer and the holes are the inside region of the input solids as explained in Figure 15. Figure 18 shows the CDT result of Figure 17

For 2D CDT and 3D DT problems, there are several open libraries such as CGAL [33]. However in a 3D case, constrained DT still remains a problem difficult to solve. In our case, similar to Cavalcanti *et al.* [34], we flip faces and edges according to the constraints after a general Delaunay triangulation.

Even though Delaunay triangulation guarantees regularized triangles from given points, the quality is affected by initial point sampling. Figure 18 shows the initial CDT from the sampled points. Person *et al.* suggest modifying this result by moving points to the equilibrium positions. For each

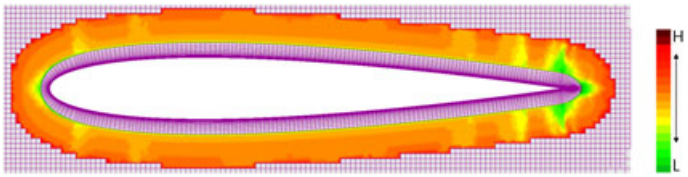


Figure 16. Size function display of the transition layer.

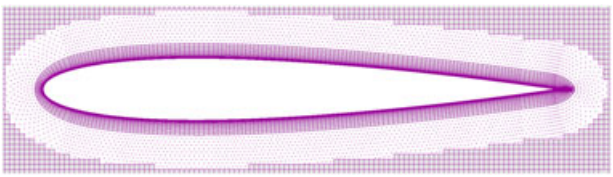


Figure 17. Point sampling in the transition layer.

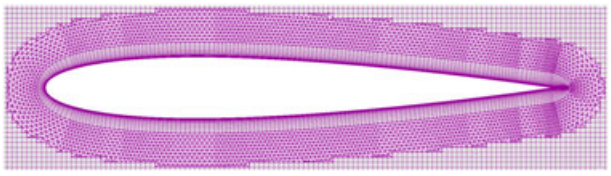


Figure 18. Initial constrained Delaunay triangulation.

DOI: 10.1002/fld

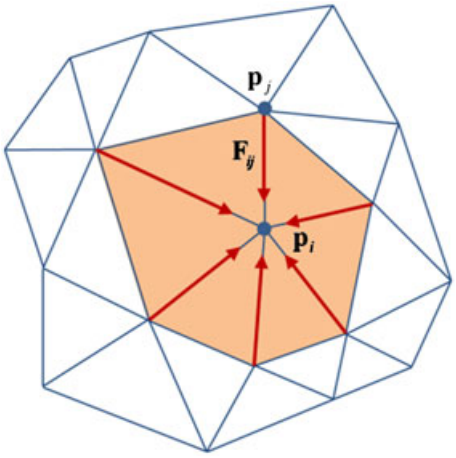


Figure 19. Inner forces.

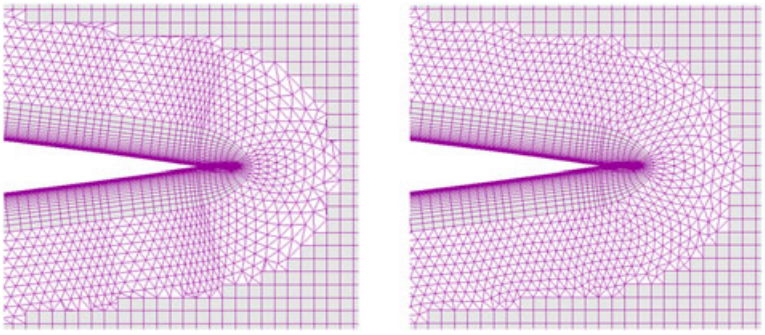


Figure 20. Initial triangular mesh (left) and optimization (right).

interior node $\mathbf{p}_i$ and its one ring neighbor nodes $\{\mathbf{p}_j\}$, we want to find node positions satisfying,

$$\sum\nolimits_j \mathbf{F}_{ij} = \mathbf{0} \tag{8}$$

$\mathbf{F}_{ij}$ is the repulsive force to the node $\mathbf{p}_i$ from $\mathbf{p}_j$ as shown in Figure 19.

$$|\mathbf{F}_{ij}| = \begin{cases} k(l_{i0} - l_{ij}) & \text{if } l_{ij} < l_{i0} \\ 0 & \text{if } l_{ij} \geqslant l_{i0} \end{cases} \tag{9}$$

where l_{i0} means the initial size value of the $\mathbf{p}_i$. The goal is to find point positions, which make the total net force zero.

$$\frac{d\,\mathbf{p}_i}{dt} = \mathbf{F}(\mathbf{p}_i), \quad t \geqslant 0 \tag{10}$$

If a stationary solution is found, it satisfies the system $\mathbf{F}(\mathbf{p}) = 0$. Same as the work of Persson *et al.* [31], we approximate Equation (1) using the forward Euler method $\mathbf{p}_i^{n+1} = \mathbf{p}_i^n + \Delta t \mathbf{F}(\mathbf{p}_i^n)$ Figure 20 shows the result of the optimization.

6. EXAMPLES

We have tested our algorithm with several 2D and 3D models. All 2D models are piecewise liner curves and 3D models are polygonal surfaces (triangular net or quadrangular net) as input.

Figure 21 shows the resulting hybrid mesh of NACA0012 model (left) and its closed-up view (right). In Figure 22, the propagation of body fitted mesh near the concave region in the blue

DOI: 10.1002/fld

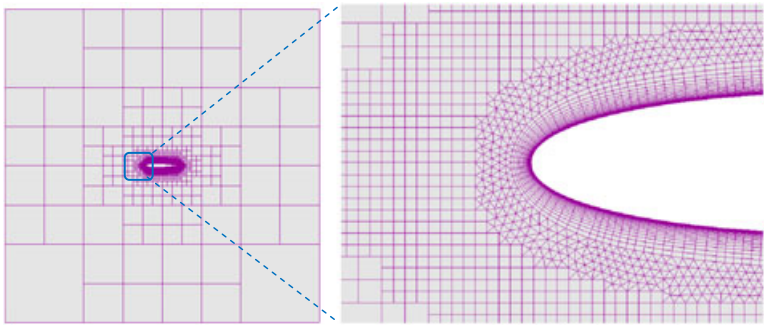


Figure 21. NACA0012 model (2D); (left) total grid; and (right) expanded view of the front region.

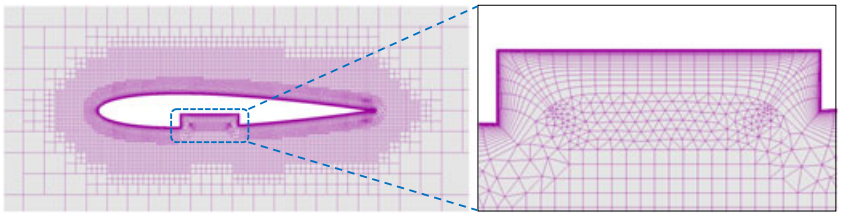


Figure 22. Propagation in concave region.

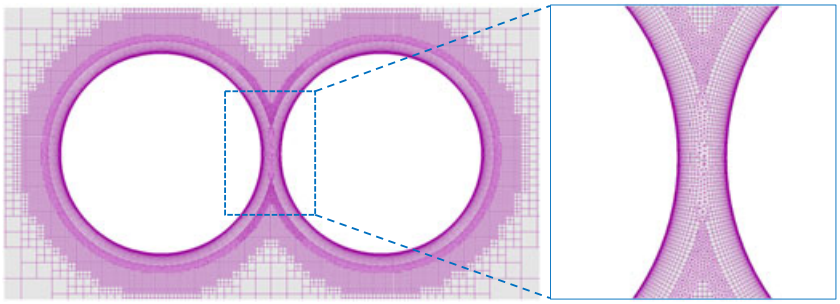


Figure 23. Two spheres; multi-body case.

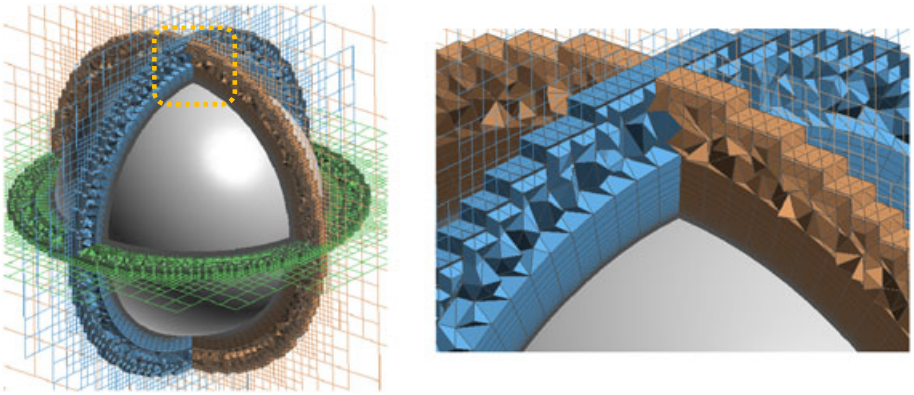


Figure 24. Sphere example; the right figure shows the details in the yellow rectangle in the left one.

rectangle of the upper one can be found. Note that the contour of the body-fitted layer gets smoother in both cases as the mesh piled up due to the different propagation speed determined by the curvature of the contour.

DOI: 10.1002/fld

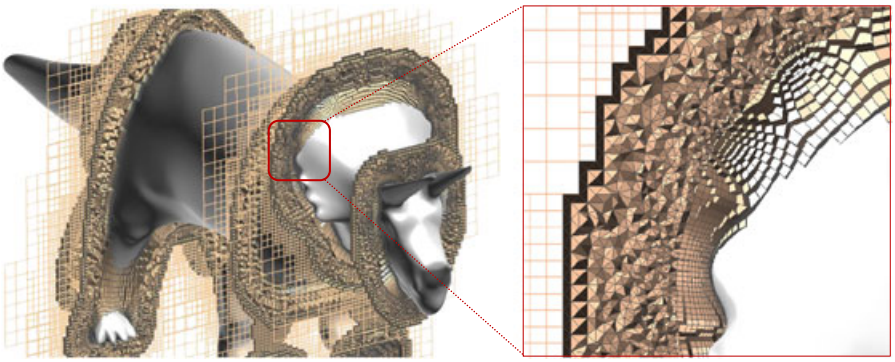
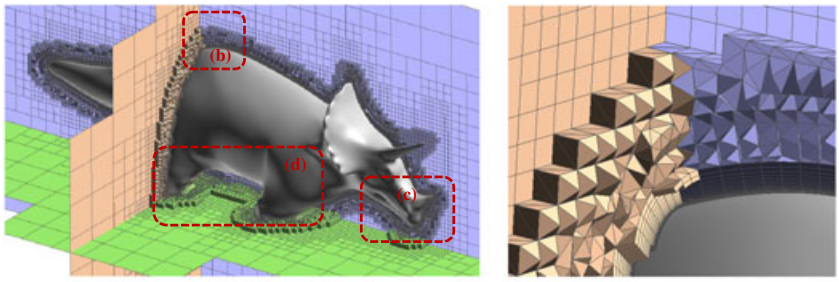
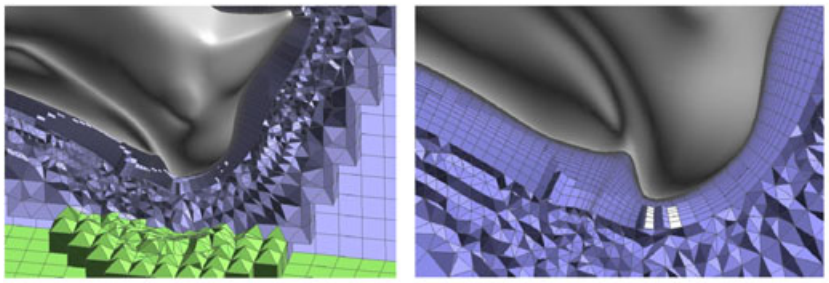


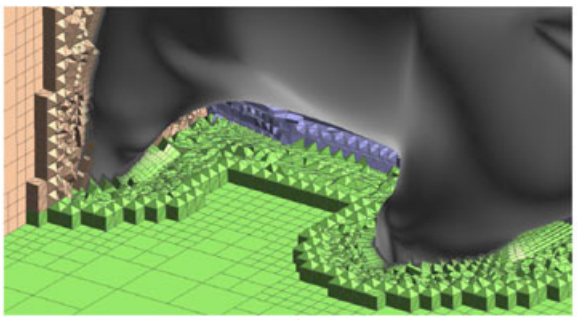
Figure 25. The dinosaur example with x-axis sectional planes.



(a) (b)



(c)



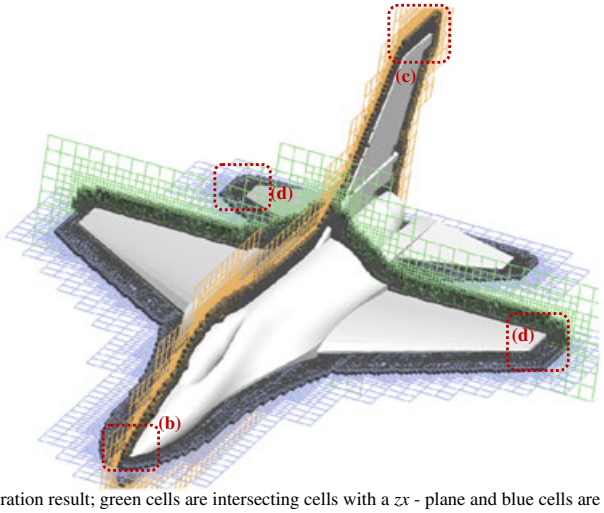
(d)

Figure 26. The dinosaur example with different sectional planes.

A multi-body case is shown in Figure 23. A hybrid mesh that fills the space surrounding two spherical bodies is generated. The right figure shows the close-up view of the in-between region.

A 3D grid was generated from the sphere model in Figure 24. Body fitted-hexahedrons are layered from the solid body, and tetrahedron cells fill the space in between the body-fitted and Cartesian hexahedron grids.

DOI: 10.1002/fld



(a) Mesh generation result; green cells are intersecting cells with a *zx* - plane and blue cells are intersecting cells with *xy*-axis plane and green

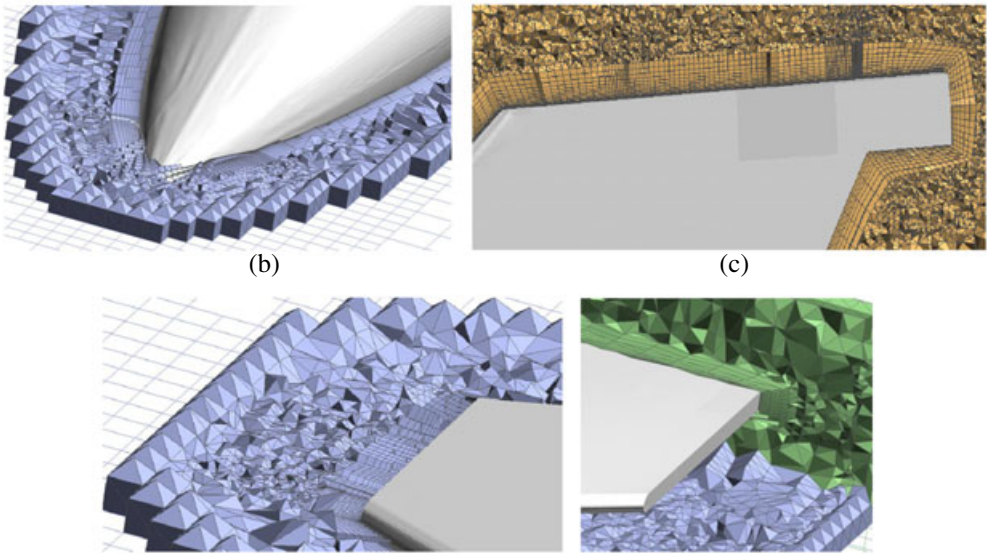


Figure 27. Fighter aircraft example.

In order to see how it works on a more complex model, a 3D dinosaur model, composed of 90,642 surface meshes, was tested in Figures 25 and 26. Figure 26(c) shows that the resulting hybrid grid from the gradual region is quite similar to the result of a simple spherical model shown in Figure 24. For more complicated regions, it still generates proper body-fitted layers and evenly distributed transition grids (Figure 26(c) and (d)).

Figure 27 shows the mesh generation result of a fighter airplane model. The intersecting grid cells with axis parallel planes at the certain positions are displayed. There is no local interference, and the body fitted layer is transformed to the Cartesian layer along tetrahedral cells.

We have tested the mentioned objects using a desktop PC with an Intel Core i5 CPU (2.66 GHz), 4 GB RAM, and the program developed with Microsoft Visual C++ (ver. 10.0 in Visual Studio 2010). Following two tables show the detailed experimental results from the three 3D models; computation time analysis is represented in Table III, and Table IV shows quality analysis with several quantitative geometric measures for tetrahedral cells.

DOI: 10.1002/fld

Table III. Computation time/memory table.

		Sphere	Dinosaur	Fighter aircraft
Input	# of faces in B	14,000	90,642	303,160
	Size of S	10 times of object bounding box		
	Initial offset length	10^{-5}		
	Extension ratio	1.2		
Output	# of total cells	1,413,150	3,746,254	11,139,264
	# of prism cells	150,000	1,359,360	4,547,400
	# of octree cells	173,288	233,936	156,084
	# of tetrahedron cells	1,089,862	2,152,958	6,435,780
Computation time (s)	Body-fitted layer	39.771	387.818	1647.893
	Octree	3.182	7.752	6.334
	Transition layer	415.710	663.022	1730.645
	-Optimization	225.294	399.475	876.915
	Total time	458.663	1058.592	3384.872

Table IV. Geometric quality of a grid.

Input		Sphere	Dinosaur	Fighter aircraft
Tetrahedron cells	Minimum dihedral angle (°)	2.00	3.00	1.15
	Average dihedral angle (°)	69.41	69.47	69.95
	% < 15. dihedral angle	0.712 %	0.737 %	0.973 %
	Maximum area ratio	1.99	1.99	2.00
	Average area ratio	1.22	1.18	1.36
	% > 1.5 area ratio	less than 0.001 %	less than 0.001 %	less than 0.001 %

As shown in Table III, it takes about 18 min to generate 3.7M cells for a dinosaur model, and about 56 min for 11M cells. It is reasonable to wait, but there still remains to be improved in the computational time aspect. For geometric quality analysis, we tested each model with two values, a dihedral angle and an area ratio whose meanings are the following:

- Dihedral angle: the internal dihedral angle value between two adjacent faces in a tetrahedron;
- % < 15° dihedral angle: the percentage of the number of cells including less than 15° dihedral angle among total tetra cells;
- Area ratio: the ratio of the maximum area of a face to the average face area of a tetrahedron cell; and
- % > 1.5: the percentage of the number of cells with bigger than 1.5 area ratio.

As shown in Table IV, our approach gives good quality of mesh for the unstructured cells excepting very small number of cells, which can be removed in the post-processing step. Note that the average area ratio is bigger than 1 because we do not generate uniform cells, but generate gradually growing cells in size to connect two structured layers.

7. CONCLUSIONS

In this work, we have developed a unified framework to create hybrid mesh for general objects when analyzing the viscous flow, which requires very high accuracy near the surface boundary of the models. We combined the structured body-fitted mesh, Cartesian, and unstructured grid in order to satisfy the accuracy near the surface boundary for the first layer, to make grids robustly and efficiently for the Cartesian, and to connect the two layers. To satisfy different purposes such as accuracy, robustness, and efficiency, different layers were used in this work, and for each layer, first, we analyzed detailed requirements, and proposed moderate approaches. As a result, we adopted the level set approach for the body-fitted layer, and scan conversion for the adaptive Cartesian layer. For the transition layer connecting two structured layers with different orientations, we optimized

DOI: 10.1002/fld

unstructured grids from the CDT result with initial point samples. We can refine unstructured cells simply by adjusting the sampling density where there is need to split into smaller cells. Our approach shows robust results about the geometries of the input model.

We have tested our framework with several complicated geometric objects as well as basic models in 2D and 3D, and it gives robust results in a reasonable time even though there is possibility to improve computation time at several steps by introducing parallel processing using GPU [35].

ACKNOWLEDGEMENTS

This work was partially supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD090020CD and UD080042AD.

REFERENCES

1. Shaw JA. Hybrid Grids. In *Handbook of Grid Generation*, Thompson JF, Soni BK, Weatherill NP (eds). CRC Press: Boca Raton, FL, 1999.
2. Baker TJ. Mesh generation : art or science? *Progress in Aerospace Sciences* 2005; **41**:29–63.
3. Ebeida MS, Davis RL, Freund RW. A new fast hybrid adaptive grid generation technique for arbitrary two-dimensional domains. *International Journal for Numerical Methods in Engineering* 2010; **84**:305–329.
4. Chand KK. Component-based hybrid mesh generation. *International Journal for Numerical Methods in Engineering* 2005; **62**:747–773.
5. Zhang LP, Wang ZJ. A block LU-SGS implicit dual time-stepping algorithm implicit dual time-stepping algorithm for hybrid dynamic meshes. *Computers & Fluids* 2004; **33**:891–916.
6. Wang ZJ, Chen RF. Anisotropic solution-adaptive viscous Cartesian grid method for turbulent flow simulation. *AIAA Journal* 2002; **40**(10):1969–1978.
7. Fujimoto K, Fujii K, Wang Z. Improvements in the reliability and efficiency of body-fitted Cartesian grid method. *47th AIAA Aerospace Sciences Meeting AIAA Paper* 2009-1173, Orlando, Florida, USA, 2009.
8. Dawes WN, Harvey SA, Fellows S, Favaretto CF, Velivelli A. Viscous layer meshes from level sets on Cartesian meshes. *45th AIAA Aerospace Sciences Meeting & Exhibit*, Reno, Nevada, USA, 8-11 January, 2007.
9. Dawes WN, Kellar WP, Harvey SA. Using level sets as the basis for a scalable, parallel, geometry engine and mesh generation system. *47th AIAA Aerospace Sciences Meeting AIAA Paper 2009-372*, Orlando, Florida, USA, January 2009.
10. Khawaja A, McMorris H, Kallinderis Y. Hybrid grids for viscous flows around complex 3D geometries including multiple bodies. *AIAA 12th Computational Fluid Dynamics Conference, AIAA-95-1685*, San Diego, California, USA, 1995.
11. Khawaja A, Kallinderis Y. Hybrid grid generation for turbomachinary and aerospace applications. *International Journal for Numerical Methods in Engineering* 2000; **49**:145–166.
12. Gloth O, Vilsmeier R. Level set as input for hybrid mesh generation. *9th International Meshing Roundtable*, New Orleans, Louisiana USA, 2000; 137–146.
13. Pattinson J, Malan AG, Meyer JP. A cut-cell non-conforming Cartesian mesh for compressible and incompressible flow. *International Journal for Numerical Methods in Engineering* 2007; **72**:1332–1354.
14. Sethian JA. *Level Set Methods*. Cambridge University Press: New York, 1996.
15. Sethian JA. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press: New York, 1999.
16. Xia H, Tucker PG, Dawes WN. Level sets for CFD in aerospace engineering. *Progress in Aerospace Sciences* 2010; **46**:274–283.
17. Osher S, Fedkiw RP. Level set methods: an overview and some recent results. *Journal of Computational Physics* 2001; **169**(2):463–502.
18. Xia H, Tucker PG. Finite volume distance field and its application to medial axis transforms. *International Journal for Numerical Methods in Engineering* 2010; **80**(1):114–134.
19. Peng D, Merriman B, Osher S, Zhao H, Kang M. A PDE-based fast local level set method. *Journal of Computational Physics* 1999; **155**:410–438.
20. Hamann B. Curvature approximation for triangulated surfaces. *Computing* 1993; **8**:139–153.
21. Farin G. *Curves and Surfaces for CAGD*, (5th edn). Morgan Kaufmann: San Diego, 2002.
22. Leatham M, Stokes S, Shaw JA, Cooper J, Appa J, Blaylock TA. Automatic mesh generation for rapid response Navier–Stokes calculations. *FLUIDS 2000 Conference and Exhibit, AIAA Paper* 2000-2247, Denver, Colorado, USA, 2000.
23. Cary A, Michal T. Generalized prisms for improved grid quality. *15th AIAA Computational Fluid Dynamics Conference, AIAA Paper 2001-2552*, Anaheim, California, June 2001.
24. Chalasani S, Thompson D. Quality improvements in extruded meshes using topologically adaptive generalized elements. *International Journal for Numerical Methods in Engineering* 2004; **60**:1139–1159.

DOI: 10.1002/fld

25. Ito Y, Nakahashi K. Unstructured mesh generation for viscous flow computations. *11th International Meshing Roundtable*, Ithaca, New York, USA, 2002; 367–377.
26. Aubry R, Lohner R. Generation of viscous grids at ridges and corners. *International Journal for Numerical Methods in Engineering* 2009; **77**(9):1247–1289.
27. Kim SJ, Lee DY, Yang MY. Offset triangular mesh using the multiple normal vectors of a vertex. *Computer-Aided Design and Applications* 2004; **1**:285–291.
28. Foley JD, van Dam A, Feiner SK, Hughes HF. *Computer Graphics: Principles and Practice*. Addison Wesley: Boston, 1996.
29. Kaufman A, Shimony E. 3D scan conversion algorithms for voxel-based graphics. *Proceedings of the Workshop on Interactive 3D Graphics*, 1986; 45–75.
30. Peraire J, Peiro J, Morgan K. Adaptive remeshing for three-dimensional compressible flow computations. *Journal of Computational Physics* 1992; **103**:269–285.
31. Persson PO, Strang G. A simple mesh generator in MATLAB. *SIAM Review* 2004; **46**(2):329–345.
32. Persson PO, Strang G. Adaptive unstructured mesh generation using distance functions. *Conference on Adaptive Methods for Partial Differential Equations and Large-scale Computation*, Troy, New York, October 2003.
33. CGAL Library : www.cgal.org (accessed on Febuary 15, 2012).
34. Cavalcanti PR, Mello UT. Three-dimensional constrained Delaunay triangulation: a minimalist approach. *Proceedings of 8th International Meshing Roundtable, Sandia National Laboratories*, 1999; 119–129.
35. Park S, Shin H. Efficient generation of adaptive Cartesian mesh for computational fluid dynamics using GPU. *International Journal for Numerical Methods in Fluids* 2012. DOI: 10.1002/fld.2750.