20th AIAA Computational Fluid Dynamics Conference
27 - 30 June 2011, Honolulu, Hawaii

AIAA 2011-3052

Automation of Hole-Cutting for Overset Grids Using the X-rays Approach

Noah Kim* and William M. Chan†

NASA Ames Research Center, M/S 258-2, Moffett Field, CA 94035

Overset grids resolve complicated geometries by creating high quality structured grids that are independently built for each component. This simplifies the process of grid generation, but domain connectivity must be performed so that adjacent grids share information. Shortcomings of the current X-rays method for the hole-cutting step of domain connectivity were explored to automate the X-rays approach to create minimum holecuts around geometric components. Open surfaces of geometric components were automatically closed by Delaunay triangulation on a projected plane. A formfitting membrane surface was created through Laplacian smoothing. Determination of grid points to be cut by each X-ray was automated. Tight-fitting oriented bounding boxes were introduced to enable the above methods.

I. Introduction

VISCOUS flow simulations require building high-resolution grids to perform analysis on complicated geometries. Structured overset grids resolve these geometries by creating high quality structured grids, which follow geometric features to form individual components. This simplifies the process of grid generation, but domain connectivity must be performed so that adjacent grids share information.[1]

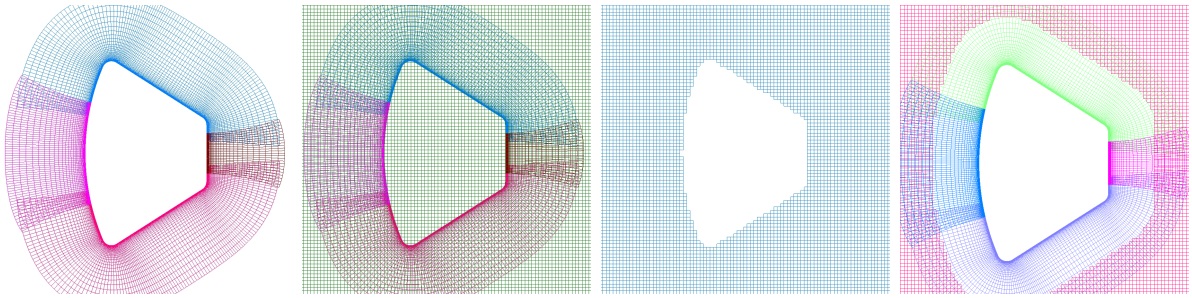
The domain connectivity process is made up of hole-cutting and fringe point interpolation.[1] Hole-cutting can be further divided into minimum and optimal holecuts. Minimal holecuts remove the grid points that end up inside of a solid enclosure of another grid or component. The optimal holecut occurs away from the geometry in the volume grid and tries to adjust the overlap region so that intergrid communication occurs between grid cells of comparable sizes. This is needed for two major reasons. First, the accuracy of the solution degrades if the overlapping cell sizes differ greatly. Cells with lower resolution do not capture the flow features that form in the areas of high resolution. Next, the speed of convergence can be adversely affected by the size difference in the overlap regions. The ability to cut holes allow overset grids to arbitrarily overlap each other, with the focus on best grid distribution for each geometric component. Since neighboring grids need to share information, the overlap between these grids should be sufficient so that donors for the fringe points can be found. If the holecut operation removes too many points, there may not be enough overlap to obtain interpolation stencils. Fringe point interpolation focuses on the boundaries of the grids and looks for donor cells for solution interpolation.

Fig. 1 shows the domain connectivity step for a simple capsule within a Cartesian grid. Fig. 1a shows a central slice of the near body grid of the capsule. This entire system is embedded within a fine Cartesian grid, as seen in Fig. 1b. Fig. 1c shows the minimum holecut that the capsule component cuts into the Cartesian grid. The stair-stepping of the Cartesian grid closely conforms to the surface of the capsule grid and only the grid points inside the capsule component are cut. Fig. 1d shows the optimal holecut that the capsule component cut into the Cartesian grid. This is an optimal hole because the overlapping cell sizes are of comparable sizes. There are two sources of fringe points. The points at the outer boundary of the near body grids, as well as the grid points that are on the boundary of a hole in the Cartesian grid.

The four main criteria for an ideal domain connectivity algorithm are robustness, automation, speed, and low memory requirement, in order of importance. It is desirable to have a robust algorithm that is capable of dealing with a wide set of geometries. In case of a failure, the algorithm should let the user know where and

*PhD Candidate, Stanford University
†Computer Scientist, AIAA Senior Member

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.



(a) (b) (c) (d)

Figure 1. Capsule test case showing hole-cutting. (a) Midplane slice of the capsule. (b) Capsule and Cartesian grid without hole-cutting. (c) Cartesian grid with minimum holecut. (d) Capsule and Cartesian grid with optimal holecut.

how the failure occurred so that it may be mitigated quickly. Automation means performing the task with minimal user input and guidance, and requires no iterative manual steps. In addition, automation relieves the user from repetitive, tedious, and error-prone manual steps, which require knowledge, concentration, and focus. Speed is needed in relative motion problems where domain connectivity is performed in every time step. However, speed cannot take precedence over robustness and automation.

There have been several previous efforts to develop domain connectivity algorithms and software on overset grids.[2,3,4,5] Examples include search-based, direct-cut, and query-cut methods. Search based methods will mark points as a hole if no donors are found. This can erroneously mark points as holes if the volume grids do not have enough overlaps and is far from robust. Direct-cut methods do not use auxiliary approximations of the geometry, and is considered the most accurate method but can be very expensive. However, if the geometry is not watertight, it can lead to a catastrophic failure where it marks all the points in the grid as inside points. This makes hole filling a critical step of the direct cut method. Query-cut based methods use a bounding volume or other geometric approximations to determine if a point is inside a solid boundary.

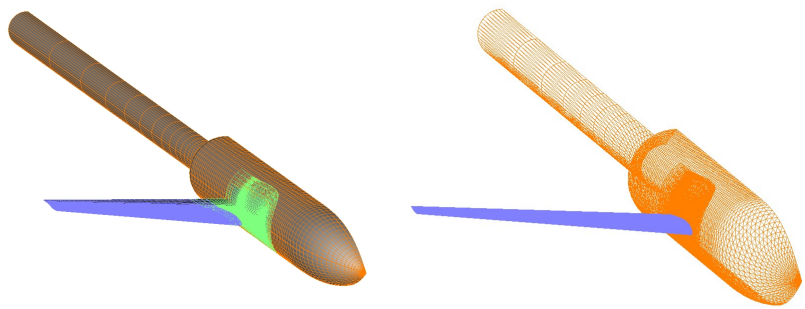
Selected examples of query-cut based method include DCF3D,[6] original X-ray,[7] PEGASUS5[3] and SUGGAR.[4] DCF3D uses analytical shapes, which gives the fastest means of inside outside tests, but it can be quite time intensive for the users because the geometry would have to be visually analyzed before the analytical shapes can be picked and placed manually. PEGASUS5 uses a Cartesian hole map with a line-of-sight algorithm to determine if a point is inside or outside of the given geometry. It is automated and requires minimum input from the user, but it does not have the capability to hole-cut on geometries with relative motion and may have significant memory requirements. Original X-rays use a Cartesian grid to approximate the geometry. By creating an image plane and shooting rays, it is able to find the pierce points on the geometry, storing the pierce point height based on two dimensional indices. It is computationally efficient and has a small memory requirement. This makes the X-rays method an excellent choice for unsteady relative motion problems. However, the user must identify each component and manually close open surfaces which can get immensely complicated for large grid systems. In addition, the user must explicitly set the relationships between the multiple components and grids to label which X-ray cuts which grids. SUGGAR uses an octree based data structure. It identifies cells as being near or far from the surface. For those regions that are close to the surface, it will subdivide the cells and recursively refine the surface representation based on the intersection with the volume cells. This method allows the code to quickly search a small area around a given point, but it can be memory intensive for high resolution cases and inefficient for unsteady cases.

The purpose of this paper is to highlight areas that can be improved in the original X-rays method. By identifying areas with manual steps in the original X-rays algorithm, this paper demonstrates the enhancements that allow the automatic generation of the minimum holecut.

I.A. Original X-rays: User Inputs

For the original X-rays[7] approach, the user is required to perform the following. Initially, all of the components must be manually identified. A component is composed of surface grid subsets around a geometric part that has the quadrilateral surface patches split into triangles. For example, Fig. 2 shows a simple wing and fuselage. The wing would make one component, and the fuselage would make another component. The

collar grid, which is a grid spanning the junction between the wing and the fuselage would have a subset that belongs to the wing component, and another subset that belongs to the fuselage component. Hence, the wing component would have the surface grid points that form the wing, as well as grid points from the collar grid which are near the wing root. The collar needs to be split, half belonging to the wing component and half to the fuselage component.



(a) (b)

Figure 2. Sample wing-fuselage geometry used for ray test comparison. (a) Grid view showing fuselage, wing and collar grids. (b) Component view showing fuselage and wing components.

Continuing the wing fuselage example, it can be seen that the wing component ends abruptly at the junction with the fuselage. This creates an open surface at the wing root, as seen by Fig. 3a. This open boundary in the component must be closed by the user. Closing of the open boundary is a necessary step because the geometry needs to be watertight in order to perform an inside-outside test for any point. This is typically done manually by extracting the open boundary curve points and collapsing them to a single point (Fig. 3a) then concatenating to make a simple surface (Fig. 3b). It can be seen that the triangulation made through this process can be highly skewed.This method is a manual process which typically can take about 5-20 minutes for this example, depending on the expertise of the user.

(a) (b)

Figure 3. Manual closing of the boundary for the wing component (not every grid point is plotted). (a) Open boundary curve and collapsed point. (b) Open surface closed by connecting vertices on open boundary to collapsed point.

Next, an axis aligned bounding box (AABB) is created for each component and the user must select a unique image plane spacing, Δs, for each X-ray. This image plane is always in the X-Y plane, and Δs will determine the distance between the rays that are shot in the Z direction by the algorithm. The best practice Δs is the average surface grid spacing of the component. However, this may not guarantee adequate resolution, and the choice of Δs may require multiple iterations.

Once all the X-rays are created, the user must then manually identify the X-ray to grid interaction, i.e., identifying the list of grids to be cut by each X-ray. For example, the wing X-ray would be allowed to cut grids from the fuselage, and the fuselage X-ray would be permitted to cut grids from the wing. Both of the X-rays would cut any Cartesian grids that are encompassing the geometries. With two components, this step seems trivial. However, this can be a time consuming process once the number of grids and components grow. For example, the Ares-I launch vehicle has hundreds of grids with tens of X-rays which require manual identification. Since there is no easy way to check for errors, the user has to be meticulous, often keeping a spreadsheet of the interactions. The input for this list would take as long as one to two days to generate for this complex configuration, along with iterations and fixing errors.

The following is a summary of the user input in the original X-rays process:

1. Manually identify components by providing the list of grid subset indices.
2. Manually build extra grid surfaces to close all open components.
3. Manually identify which X-ray cuts which grid.
4. Manually specify offset distance for each hole.

I.B. Original X-rays: Algorithm

An axis aligned bounding box (AABB) is created for each component. Using the Δs that the user defined, the equally spaced rays are shot in the Z direction for each component. The image plane is always on the X-Y plane of the AABB, regardless of the orientation of the component. Ray-triangle intersection test[8] is a performed to finds the intersection between each Z-direction ray and triangles. This results in a set of pierce points on the component. Since the X-Y coordinate of each ray is determined from its indicies, only the Z-coordinate of the pierce point needs to be saved into an array. The collection of X-Y image plane and pierce point information makes up the X-ray. An X-ray may be comprised of multiple pierce points for a single ray. For example, a ray intersecting a simple convex component comprised of a single watertight grid has two pierce points, marking the entrance and exit of the ray. For a component that is comprised of multiple grids, more than two pierce points can occur.

The surface normal of the intersected triangle is used to determine the validity of the pierce point. For example, the intersected triangles from two overlapping grids will have the same surface normal whereas the intersected triangles from a thin trailing edge will have opposite normal direction. Thus, multiple pierce points with the similar surface normal within a certain threshold distance ϵ to one another can be considered to be a single pierce point. Alternatively, multiple pierce points with opposing triangle normal direction can be considered to be unique pierce points regardless of ϵ. The inside-outside test is done by comparing the candidate hole point against the pierce points. The pierce points are discretely spaced by Δs. A candidate hole point can be bound by an arbitrary number of pierce points in the image plane. In other words, it is rare of a candidate hole point to be exactly aligned with the pierce points. The interpolated Z-locations of the bounding pierce points are used to determine the upper and lower bound that the candidate hole point is tested against. A more detailed description of this method can be found in Meakin et al.[7] The following is a summary of the algorithmic process in the original X-rays:

1. Generate AABB for each component.
2. Compile X-Y coordinates of each ray.
3. Compile pierce points where each ray pierces the surface triangulation of each component.
4. Test all points in grids cut by the X-ray.
5. Generate minimum hole by testing potential points.

The current best practice for the determination of Δs is to use the average grid spacing of the surface grids that comprise a component. There are two known issues with this practice. Fig. 4 shows the large stair-stepping that occurs when the off-body grid spacing is much finer than the surface grid. Because the Δs determined by the average grid spacing is much larger than the Cartesian grid, the X-ray is not able to capture the details of the holecut. Another example is in rocket stage separation, where there is a need to resolve a tight gap between relatively large geometries. This situation requires a large image plane to have a very fine Δs, causing the X-ray to take a large amount of memory, adversely affecting the performance.

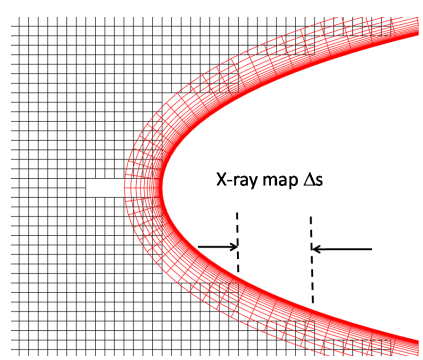


Figure 4. Fine off body grid and the sub-optimal holecut resulting from the best-practice Δs

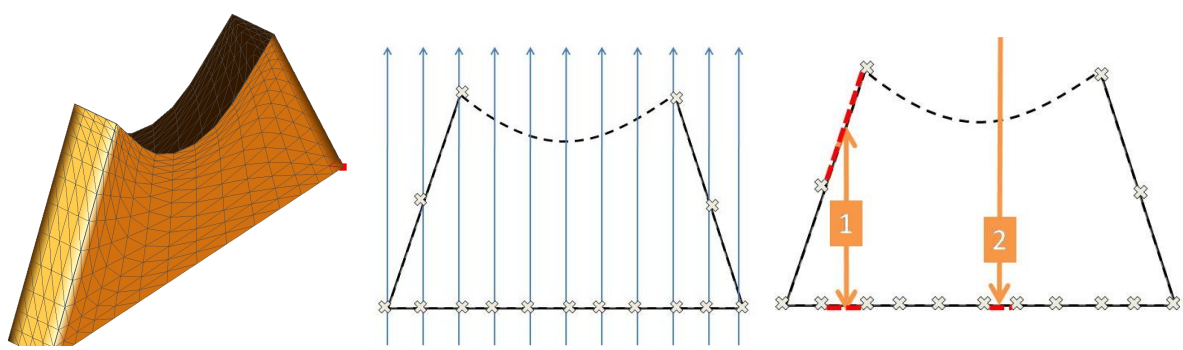
The original X-rays method fulfills some of the criteria for an ideal domain connectivity. It is a robust and fast algorithm, which makes it well suited for moving body unsteady problems where domain connectivity must be performed with every step. The memory requirement for the original X-rays is low. However, there is a large amount of manual and iterative work that must be done by the user in order to successfully run the original X-rays method.

II. Automatic Hole-Cutting Process

The current work seeks to enhance the original X-rays method by automating the manual and iterative effort that are currently involved in the original X-rays method. By identifying areas of difficulty for the user in using original X-rays, priority is set on automating as much of the process as possible. This includes any step that required either user expertise, input and iteration. From the user input perspective, closing the open boundary and determination of grid points to be cut by each X-ray are found to be the top two areas that required automation. The technology behind generating an oriented bounding box is implemented in both of the previously mentioned steps.

II.A. Automatic Hole Closure

In order to cut holes using original X-rays, the surfaces that represent the component must be watertight.[7] The front bipod bracket of the Space Shuttle, seen in Fig 5, demonstrates the need for closing the surface. Fig. 5b shows the central slice of the front bracket. There is an open boundary at the top part of this geometry, marked by dotted lines in the outline. When a ray pierce test occurs over the open surface, only one intersection is marked, and the grid point cannot be determined to be inside or outside. For example, point 1 shown in Fig. 5c is determined to be within the geometry because it is bounded by the interpolation of the pierce points. However, the same figure shows that point 2 resides below an open surface and cannot be determined to be inside or outside the geometry.



(a) (b) (c)

Figure 5. (a) Front bipod bracket of the space shuttle. (b) Central slice with rayshooting in the Z direction. The pierce points are shown as X's. (c) Central slice with pierce points. Grid points to be tested are marked 1 and 2 and the pierce point interpolation is shown as red lines.

Currently, hole closure is done manually. The user must first identify the open boundary through visual inspection of the grid. Once the boundary is found, the curve is extracted and collapsed to a single point at the centroid of the curve. This creates a polar surface mesh. Once the open boundary points are concatenated to the middle point, the interior grid points can be redistributed to generate a simple surface. This is done by either a GUI or a script, both of which requires quite a bit of repetitive work and user expertise. In addition, this method may cause erroneous results for a open boundary with high three dimensional variations or skewed point clustering. For a very large system such as the Ares I, there are over a hundred components and hundreds of grids. Closing the open boundary may take anywhere between five minutes to half an hour each depending on the complexity of the geometry as well as the expertise of the user. This translates to days spent on boundary closure for large systems.

The first step of the proposed automatic algorithm was to close the the open surfaces, as seen in Fig. 6. This was done by first automatically identifying and extracting all the open curves for each component.

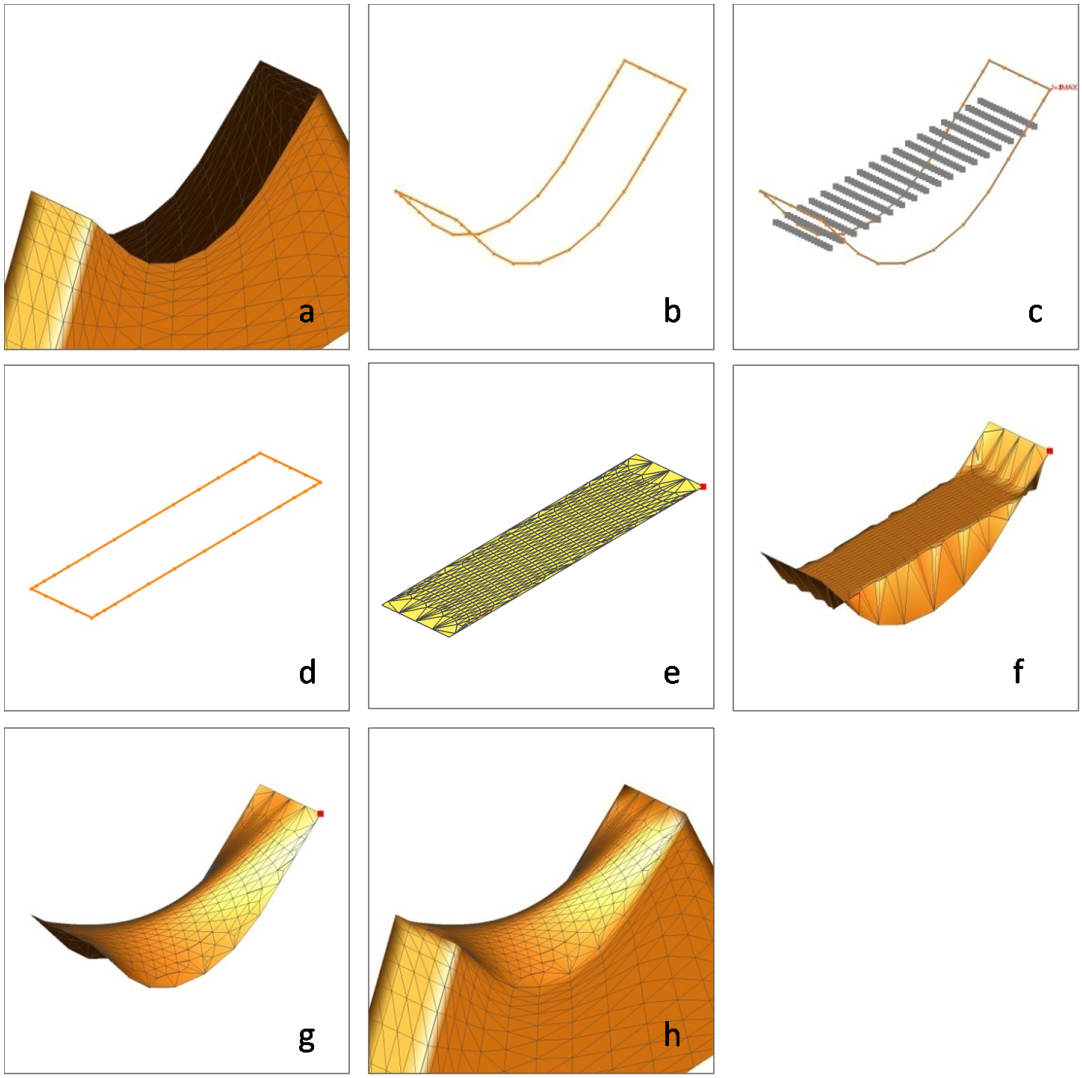


Figure 6. Visual representation of the automatic hole closure process. (a) Open surface on front bipod bracket of the space shuttle. (b) Extracted open boundary curve. (c) The best fit plane. (d) Projected boundary points on the best fit plane. (e) A 2D Delaunay triangulated surface. (f) The open boundary curve at its original location. (g) Membrane surface that tightly wraps the open boundary. (h) Final closed surface triangulation.

This routine is available within the Chimera Component Connectivity library (C3LIB),[9] a software library containing routines utilized in various steps of the domain connectivity process. It is currently able to identify the curve segments that are part of an open boundary. The routine has been developed to identify and match loop sets, individual curves that make up an open boundary. See Ref. 9 for a description of the algorithm. The automatic hole closure algorithm then concatenates the curve segments in each loop into a single curve, so that an open boundary can be defined by a a single curve. This is done by finding and concatenating the nearest neighbor to the end point of each curve segment.

Fig. 6 is a visual representation of the automatic hole closure algorithm. Fig. 6a shows the open surface on a front bipod bracket of the Space Shuttle. This particular component is shown to show the high curvature of the open boundary curve. Fig. 6b shows that the open boundary curve is extracted. The principal directions of the 3D open boundary curve were found through principal component analysis (PCA)[10] as seen in Fig. 6c, and the curve points were projected to the best fit plane(Fig. 6d). The covariance matrix, the generalization of variance to multiple dimensions and a necessary step for PCA seen in Section II.C, was found by using the length of the line segment for its weight. This plane was determined by taking both PCA as well as the direction with the maximum bounded area. A two dimensional Delaunay triangulation routine developed by Shewchuk et al[11] was used to create an interior surface for the curve seen in Fig. 6e. The open boundary

curve and the interior triangulated surfaces were then brought back to its original location in 3D, while keeping the triangle connections (Fig. 6f). This created a sharp jump in the shape of the mesh. Fig. 6g shows the Laplacian smoothing that was performed for all interior points,[12] as shown in Eq. 1.

$$\vec{x}_i = \frac{1}{N_i}\sum_{j=1}^{N_i}\vec{x}_j \tag{1}$$

$\vec{x}_i$, on the left-hand side of Eq. 1, is the new position for this particular node, and $\vec{x}_j$ are the adjacent nodes, both interior and exterior, and N_i is the number of vertices connected to vertex $\vec{x}_i$. Eq. 2 is the expanded form of Eq. 1 applied to a simple triangulation shown in Fig. 7. Points $\vec{a}$ through $\vec{e}$ are the fixed boundary points. Points $\vec{1}$ and $\vec{2}$ are the internal points that have been inserted through Delaunay triangulation. The summation is done for each interior node, as seen by $\vec{x}_1$ and $\vec{x}_2$. Fixed boundary points and internal points are separated, because the initial condition for the interior nodes were arbitrary, based solely on the best fit plane. By arranging the system of equations in a matrix, the new location for all the interior point were solved simultaneously.

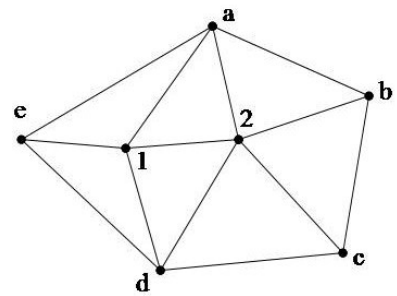


Figure 7. Simple triangular mesh illustrating Laplacian smoothing

$$\begin{aligned} \vec{x}_1 &= \tfrac{1}{4}\left(\vec{x}_2 + \vec{x}_a + \vec{x}_d + \vec{x}_e\right) \\ \vec{x}_2 &= \tfrac{1}{5}\left(\vec{x}_1 + \vec{x}_a + \vec{x}_b + \vec{x}_c + \vec{x}_d\right) \end{aligned} \tag{2}$$

$$\begin{bmatrix} 4 & -1 \\ -1 & 5 \end{bmatrix}\begin{bmatrix} -\vec{x}_1 \\ \vec{x}_2 \end{bmatrix} = \begin{bmatrix} \vec{x}_a + \vec{x}_b + \vec{x}_e \\ \vec{x}_a + \vec{x}_b + \vec{x}_c + \vec{x}_d \end{bmatrix} \tag{3}$$

There were several reasons behind meshing the open boundary in two dimensions. In complexity, Delaunay triangulation in 2D is similar to constructing a Voronoi diagram while the 3D Delaunay triangulation is much more difficult due to the differences in degenerate cases.[13] Often, a component will have an open boundary because it terminates at an intersection with another component. The constructed surface does not need to be high quality because it does not affect the computation or the volume grid directly. Thus, the focus was set on generating a robust surface quickly. It can be noted that closing the open boundary creates a new geometry, which does not guarantee that the newly created surface will be within the original geometry. Visual inspection may be necessary to ensure the quality of the newly generated surface and that it does not protrude into unexpected regions of the original surface geometry.

II.B. Determination of Grid Points Cut by each X-ray

For a complicated geometry with hundreds of components and grids, the manual determination of grid points cut by each X-ray may take on the order of hours to complete. In addition, if changes are made to the grid system, i.e., grids are added or removed, this entire process would have to be done again, a highly error prone step for large number of grids and X-rays.

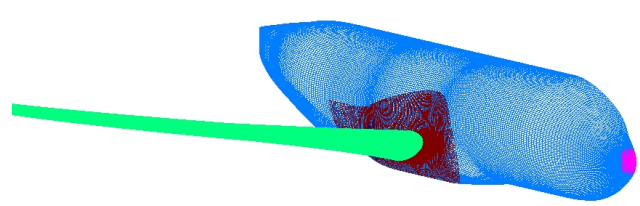


Figure 8. Isometric view of the wing fuselage example.

The proposed method of hole-cutting eliminates the manual step of determining the grid points cut by each X-ray through a set of heuristic rules. Each X-ray has a component associated with it. AABBs for each X-ray and all volume grids are constructed to determine the list of candidate points that are cuttable by each X-ray. A relatively inexpensive bounding

box intersection test narrows down the candidate points for the more expensive ray-triangle intersection tests. The number of eligible grid points are further reduced by eliminating the grid points outside the tighter fitting oriented bounding boxes (OBB). This leaves a smaller set of grid points on the list of candidate hole points. Fig. 9 shows the OBB interaction for a rotor-blade and hub test case. In Fig. 9a, the rotor-blade 1 X-ray has an OBB which intersects only the OBB of the volume grid of the hub. This eliminates grid points from rotor-blades 2 and 3 as possible candidate hole points. Alternatively, Fig. 9b shows the OBB for the hub X-ray. This OBB intersects all three rotor-blades, thus the hub X-ray must include grid points from all three rotor-blades as possible candidate hole points.

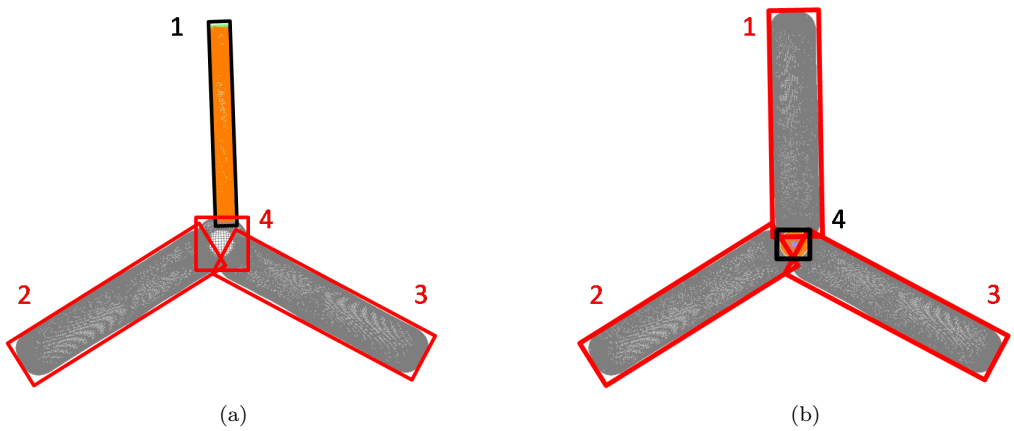


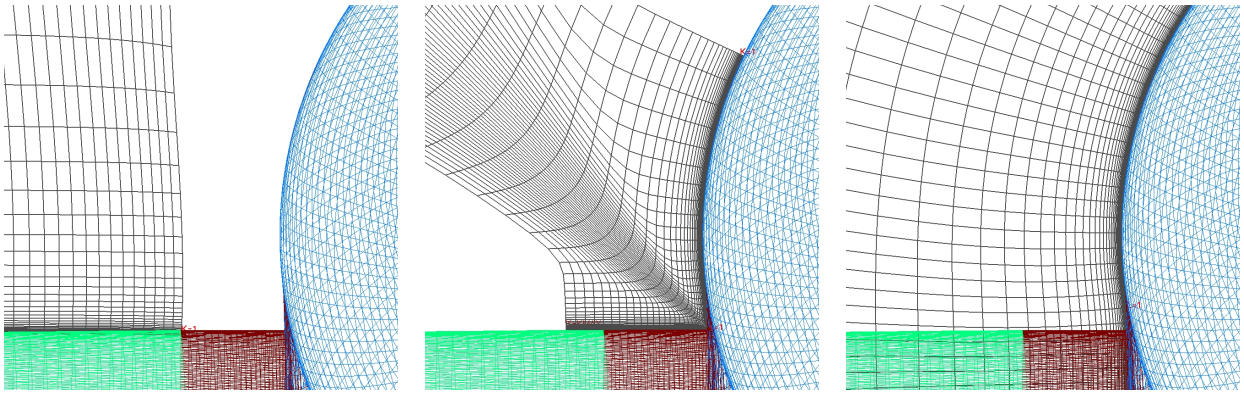
Figure 9. Top view of rotor-blades and hub example. (a) OBB for rotor-blade 1 X-ray is shown in black and the OBB of the hub and rotor-blades 2 and 3 volume grids are shown in red. (b) OBB for hub X-ray is shown in black and the OBB of the rotor-blade volume grids are shown in red.

The next step recalls the subsets of grids that form a component. These grid points, up to a certain grid index threshold in the wall-normal direction, are excluded from the ray-triangle tests because they formed the component surface. This index threshold is an empirical parameter that is initially set as $L_{max}/2$, where L_{max} is the number of grid points in the wall normal direction. This default choice of the threshold appears to work for test cases thus far. If a failure is found, it is a simple parameter that could be changed. At some distance away from the component, there is a possibility of the volume grid intersecting the component surface, so these points are included for the ray-triangle intersection tests. Other grids not part of this component are cuttable, meaning that the grid points are eligible for the ray-triangle intersection tests.

Fig. 8 shows a wing fuselage example used to demonstrate the determination of grid points cut by each X-ray. It can be seen that it is comprised of a wing grid, a collar grid, and a fuselage grid. This system would be decomposed into two components: a wing component and a fuselage component. All of the wing grid and a portion of the collar grid that lies on the surface of the wing would comprise the wing component. Likewise, all of the fuselage grid and a portion of the collar grid that lies on the surface of the fuselage would make up the fuselage component.

Fig. 10 shows the volume grid slices of the wing, collar and the fuselage. The wing X-ray would cut grids from the wing grid shown in Fig. 10a at a certain normal distance or threshold away from the body. Similarly, the volume grid of the collar grid subset that contributes to the wing component would be cuttable only after a certain normal distance. However, all of the collar grid subset that contributes to the fuselage would be eligible to be cut. Fig. 11a shows the eligible collar grid points for the wing component. The subset of the volume grid slice that is darker is not eligible to be cut by the wing. The fuselage grid would be eligible to be cut by the wing X-ray, as seen in Fig. 10c. The fuselage X-ray follows the similar steps to the wing X-ray. The fuselage X-ray would be permitted to cut any grid point from the wing grid and the collar grid subset that resides on the wing component. For the fuselage grid and the collar grid subset that make up the fuselage component, the fuselage X-ray is only allowed to cut after the normal threshold described earlier.

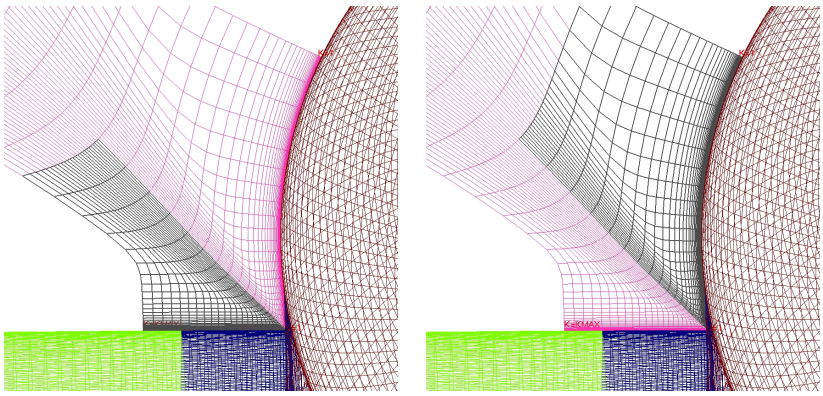
The selected normal threshold is important for two reasons. If the threshold is too small, the boundary layer of the grid points from the volume grids of the component would be captured as candidate hole points. This would result in many unnecessary ray-triangle intersection tests, because the boundary layer of a component would not intersect itself for a best practice overset geometry. However, at a significant distance



(a) (b) (c)

Figure 10. Front zoomed in view of the wing fuselage example. (a) Volume grid slice from the wing grid. (b) Volume grid slice from the collar grid. (c) Volume grid slice from the fuselage grid.

away from the body there is a possibility that the volume grid could wrap around and intersect the surface geometry, as seen in Fig. 12. Fig. 12a shows that the volume grid subset of the collar grid that is a part of the wing component could wrap around and intersect the wing. Thus, a component should be able to cut its own grid subsets beyond a normal threshold.

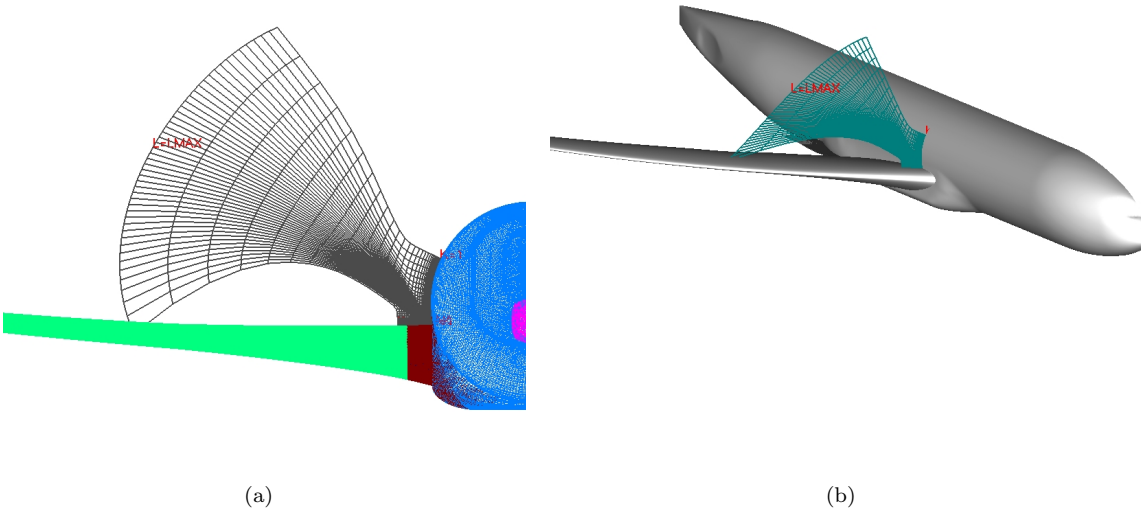


(a) (b)

Figure 11. Front zoomed view of the wing fuselage example. (a) Darker volume grid slice subset of the collar grid, not cuttable by the wing component. (b) Darker volume grid slice subset of the collar grid, not cuttable by the fuselage component.

II.C. Oriented Bounding Box

This technology is used in both automatic hole closure and determination of grid points cut by each X-ray. In the automatic hole closure step, PCA is used to determine the best fit plane, which would be the OBB face with the maximum area. In the determination of grids to be cut by each X-ray, OBBs are generated around each component to quickly identify the grids with close proximity. The original X-rays method uses axis aligned bounding boxes to generate each X-ray. This method is fast, but can be inefficient depending on the shape and orientation of the geometry. A high level trade off between complexity and tight fit against computational cost was explored.[14] As it can be seen in Fig. 13, there can be many different types of bounding volumes to approximate a geometry. Analytical bounding volumes are the fastest in determining inside-outside tests, but may not fit the geometry very well. It also requires user expertise and effort to determine the type of analytical shape to best fit a geometry. An axis aligned bounding box (AABB) requires less user input to generate and can still perform overlap tests quickly. However, depending on the orientation of the geometry, it may be a poor fit. For example, Fig. 14a shows the AABB for each component



(a) (b)

Figure 12. Wing fuselage example. (a) Front view of the volume grid slice of the collar grid intersecting the wing. (b) Isometric view of the volume grid slice of the collar grid intersecting the wing.

of a citation business jet. The AABB around the wing of the jet occupies a much greater volume than the footprint of the wing, leading to numerous ray-triangle intersection tests yielding no intersections. The oriented bounding box approach finds the principal component of the underlying shape, and is able to best fit that geometry with an oriented box (Fig. 14b). Generally, it will be tighter fitting than an AABB, but computationally slower due to the transformation required to get to the local coordinate system. Discrete oriented polytope (DOP) uses a combination of the AABB and the OBB. It is even tighter than oriented bounding boxes, but the inside-outside test is more complicated in that it needs yet another factor to take into account. Convex hulls will generate the tightest possible fitting, with the most expensive overlap test due to its irregular boundaries. OBBs were chosen to approximate the component geometries due to their speed and ability to tightly bound each component with relative ease.

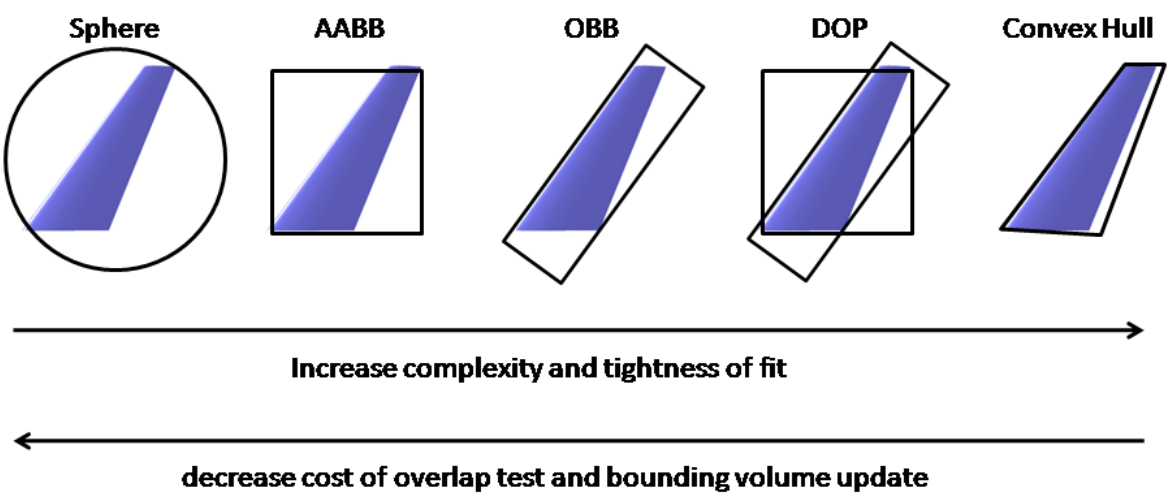


Figure 13. Increasing complexity and tight fit is inversely related to cost of overlap tests and bounding box update.

OBBs were constructed by using PCA.[10] This was done by taking the covariance matrix of the geometry. The equation for the covariance matrix C_{jk} and its subsequent derivation developed by Gottschalk, et al.[15] is shown below.

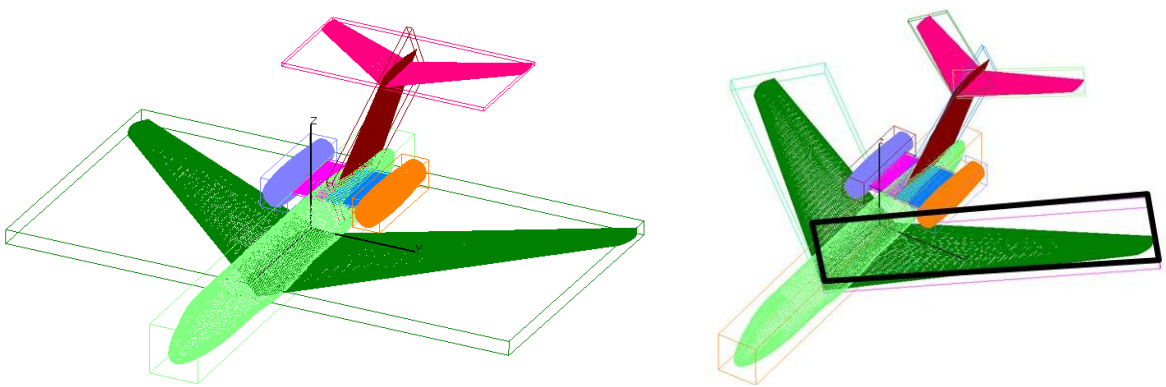
$$C_{jk} = \frac{1}{3n}\sum_{i=1}^{n}({P_j}^i{P_k}^i + {Q_j}^i{Q_k}^i + {R_j}^i{R_k}^i) \qquad 1 \leq j,k \leq 3 \tag{4}$$

where $\vec{P}^i = [{P_1}^i, {P_2}^i, {P_3}^i]$ and $\vec{P}^i$, $\vec{Q}^i$, $\vec{R}^i$ are defined by

$$\begin{aligned} \vec{P}^i &= (\vec{p}^i - \vec{\mu}) \\ \vec{Q}^i &= (\vec{q}^i - \vec{\mu}) \\ \vec{R}^i &= (\vec{r}^i - \vec{\mu}) \end{aligned} \tag{5}$$

$$\vec{\mu} = \frac{1}{3n}\sum_{i=1}^{n}(\vec{p}^i + \vec{q}^i + \vec{r}^i) \tag{6}$$

where $\vec{p^i}$, $\vec{q^i}$, $\vec{r^i}$ are the position vectors of the vertices of the ith triangle, n is the number of triangles, and $\vec{\mu}$ is the centroid of all the vertices over the entire triangulation.



(a) (b)

Figure 14. Citation business jet surface grid, separated by (a) axis aligned bounding boxes, (b) oriented bounding boxes.

The normalized eigenvectors of the covariance matrix were used as the basis vectors for the bounding box. On visual inspection of the OBB generated by the above method, it is apparent that it is not the tightest possible box. Since the above approach weighed all grid points equally, the OBB would come out skewed towards the area of heavy clustering.[15] An example of the skewness due to grid point clustering is shown in Fig 15.

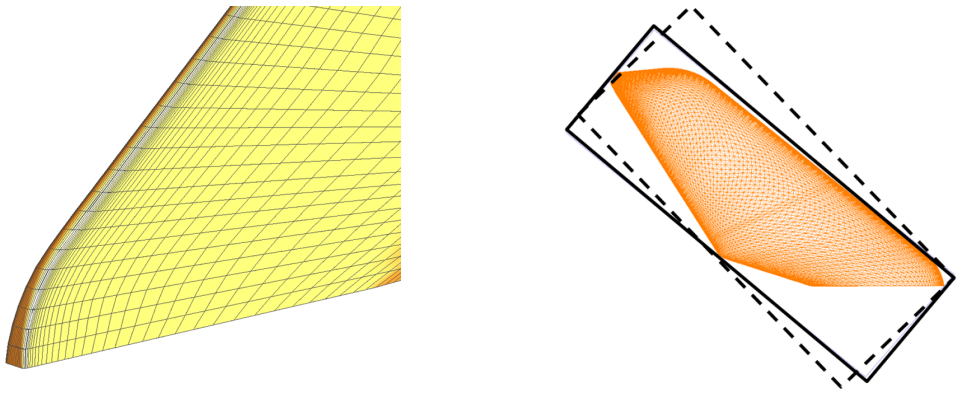
To solve this issue, the centroid of each triangle was identified and the area of each triangle was used as the centroid weights for constructing C_{jk}. The area of the ith triangle is shown in Eq. 7. By summing the area of each triangle, the surface area of the triangulation, A^T is found (Eq. 8). The centroid of the ith triangle is determined by c^i (Eq. 9).

$$A^i = \frac{1}{2}|(\vec{p}^i - \vec{q}^i)\times(\vec{p}^i - \vec{r}^i)| \tag{7}$$

$$A^T = \sum_i A^i \tag{8}$$

$$\vec{c}^i = \frac{1}{3}(\vec{p}^i + \vec{q}^i + \vec{r}^i) \tag{9}$$

The centroid of the surface triangulation $\vec{c}^T$ is found by taking the sum of the weighted averages of triangle centroids (Eq. 10). These elements were used to generate the weighted covariance matrix as seen in $\hat{C}_{jk}$.[15]



(a) (b)

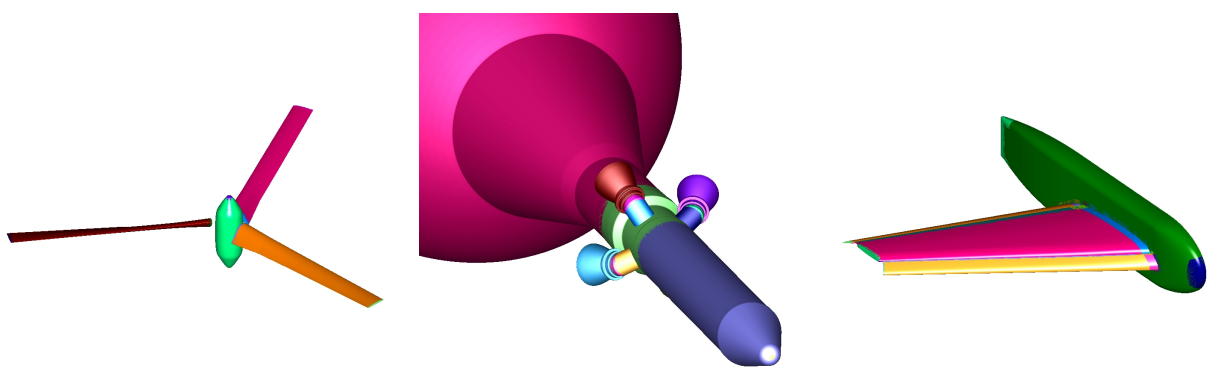
Figure 15. (a) Vertical stabilizer for the space shuttle showing areas of dense clustering, (b) Effect of clustering on bounding box shown in dotted line and the weighted covariance method in the solid line.

$$\bar{c}^T = \frac{\sum_i A^i c^i}{\sum_i A^i} = \frac{\sum_i A^i c^i}{A^T} \tag{10}$$

$$\hat{C}_{jk} = \sum_{i=1}^{n} \frac{A^i}{12A^T}(9c_j^i c_k^i + p_j{}^i p_k{}^i + q_j{}^i q_k{}^i + r_j{}^i r_k{}^i) - c_j^T c_k^T \qquad 1 \leq j,k \leq 3 \tag{11}$$

III. Results

The results described in this section shows the savings from automating the hole-cutter open boundary closure, and the determination of grid points cut by each X-ray. Several complicated geometries were tested, including the High-Lift Prediction Workshop I geometry, a rotor-blades and hub system, and the Orion Launch Abort Vehicle (LAV), shown in Fig. 16. Table 1 shows summary information about the example grids.



(a) (b) (c)

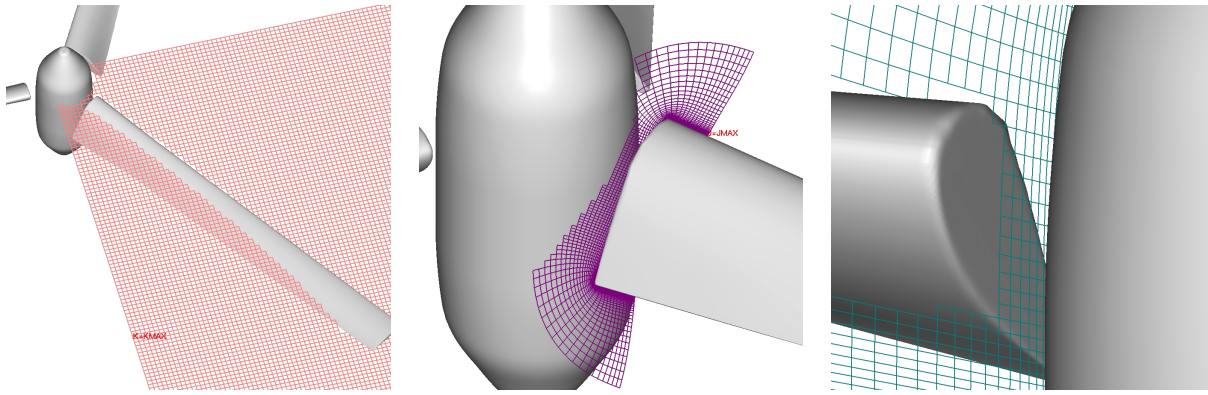
Figure 16. Test geometries used in the proposed X-ray method. (a) Rotors and hub system. (b) Orion Launch Abort Vehicle (LAV). (c) High-lift trapezoidal wing.

Fig. 16a shows the rotors test case, which contains four components: three rotor-blades and a hub. This geometry is simpler than the others in that each component is watertight so that there is no need for open boundary closure. The three geometrically identical blades are spaced apart by 120 degrees, which changes the area of the axis aligned bounding box for each component. Fig. 17 shows the minimum holecut generated through the proposed method for the rotor-blades and hub system. The holecut into the off-body Cartesian

	Number of Components	Number of Grids	Number of Grid Points (million)
Rotor-blade system	4	12	25.6
LAV	7	27	31.1
High-lift	4	28	36.5

Table 1. Test Geometries that were tested for the original X-rays method

grid can be seen in Fig. 17a and Fig. 17c, and the minimum holecut of the hub in the near body grid of the rotor-blade grid can be seen in Fig. 17b.



(a) (b) (c)

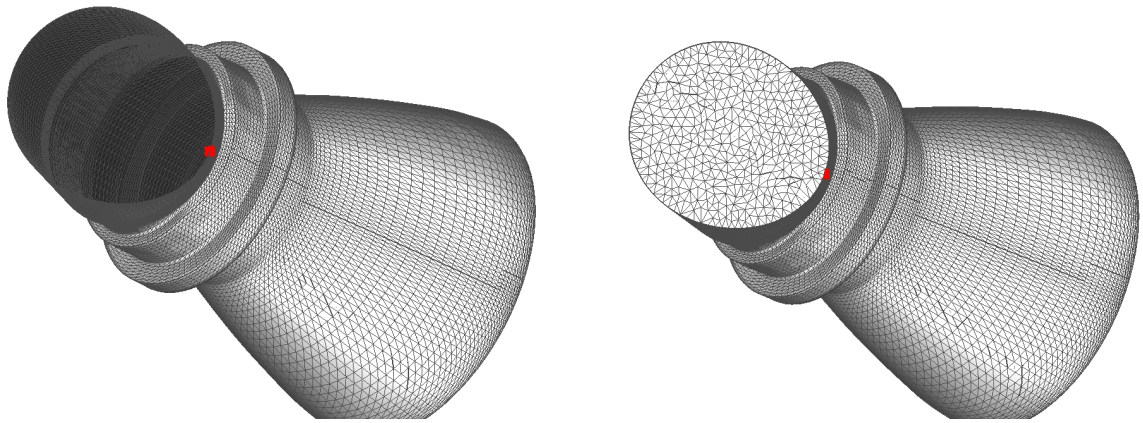
Figure 17. Rotors and hub geometry with minimum holecuts. (a) Top view with holecut into Cartesian grid. (b) Isometric view with near body grid from the blade. (c) Near body grid from the hub cut by the blade.

The Orion LAV geometry shown in Fig. 16b contains seven components with varying length scales. The main component is the axisymmetric main body. There are four identical nozzles spaced apart at the front section, and there are two protuberances that are on the main component. There are open boundaries at each junction of nozzles and two protuberances against the axisymmetric main body. Fig. 18a and Fig. 18b shows the open boundary of the nozzle component and the automatically generated closed surface of the nozzle component, respectively. Fig. 19 shows the minimum holecut of the Orion LAV. Fig. 19a shows the minimum holecut in the coarse, off-body Cartesian grid. Fig. 19b and Fig. 19c show the near-body grids from the axisymmetric main body that are cut by the nozzle. The stair-stepping can be clearly seen in Fig. 19b, showing that the minimum holecut is achieved.

The High-Lift Prediction Workshop I geometry, shown in Fig. 16c, is a wing-fuselage grid with a multi-element trapezoidal wing. The four main components are the fuselage, slat, wing and flap. This geometry has multiple open boundaries at the junction between slat, wing and flap components against the fuselage component. The open boundary closure routine is used to close these open boundaries. Fig. 20 shows the closed surfaces of the slat. The clustering of triangles can be seen near the leading edge of the slat. The algorithm is able to generate a surface even with the differences in the grid resolution. The off-body Cartesian grids cut by the wing and the slat of the high-lift trapezoidal wing is shown in Fig. 21a and Fig. 21b. Again, the tightly conforming stair-stepping can be seen in the off-body Cartesian grid.

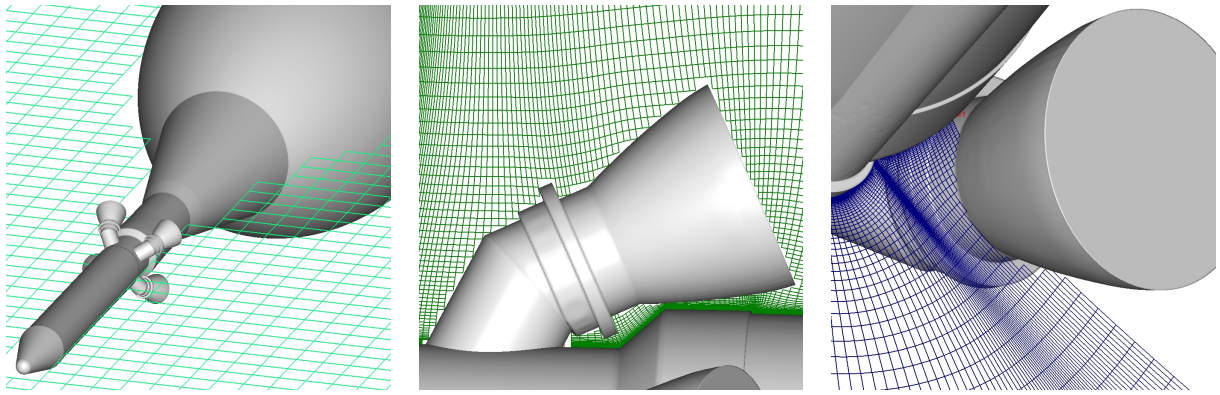
The total time in the hole-cutting process not only involves the CPU time of the respective algorithms, but also the user time required in setting up the problem, and subsequent iterations required. This adds a qualitative layer of complexity with respect to expert user versus novice users of the algorithms. The X-rays method requires specifying the boundary condition for each grid and the surface subsets that make up each component.

The user time for the original X-rays method is outlined in Table 2. The user starts by identifying the grids that make up each component. Once the components are fully defined, the user must identify and close any open surfaces. If the default Δs is inadequate to create an accurate X-ray, iteration on an appropriate Δs is required. It is noted that for the current proposed scheme, the user needs to only do the first step of the original process: identification of the grids that contribute to each component. In this regard, the user



(a) (b)

Figure 18. Orion LAV nozzle. (a) Open boundary at junction between nozzle and main body. (b) Triangulation used to close open boundary.



(a) (b) (c)

Figure 19. Orion LAV section with the minimum holecut. (a) Isometric view. (b) Side view of the nozzle. (c) Rear isometric view of the nozzle.

	Original user time (minutes)	New user time (minutes)
Rotor-blade system	15-30	3-5
Orion LAV	60-120	15-30
High-lift	30-60	10-15

Table 2. Estimate of user time needed to prepare inputs for minimum holecut. The lower and upper estimates correspond to expert to non-expert user times.

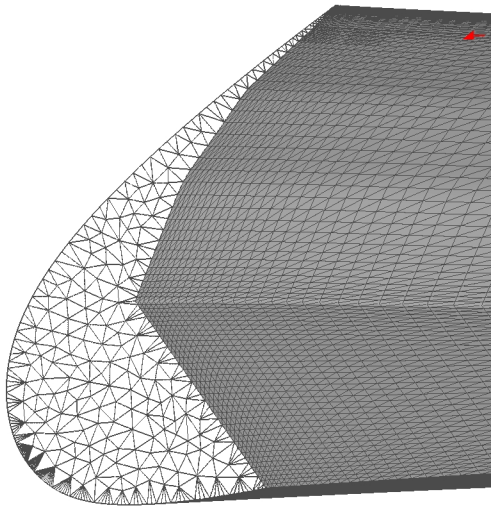
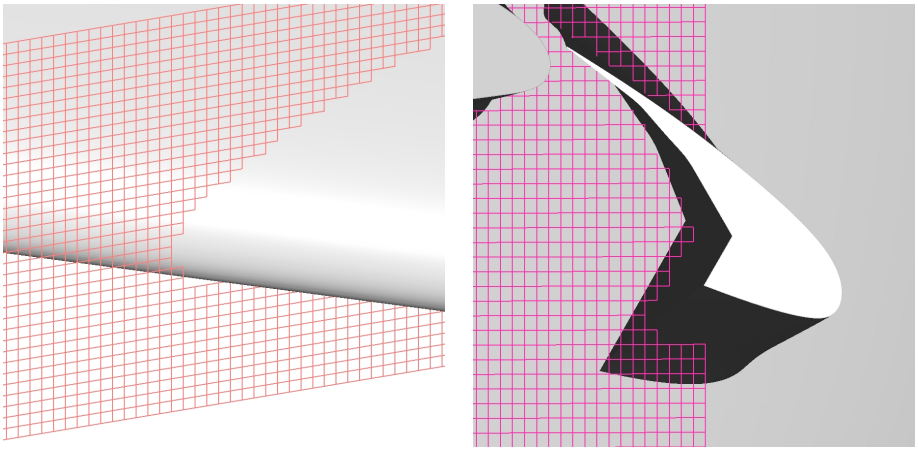


Figure 20. High-lift trapezoidal wing geometry with closed boundaries for the slat.



(a) (b)

Figure 21. High-Lift Prediction Workshop I geometry with minimum hole-cuts. (a) Off-body Cartesian grid slice over the wing. (b) Off-body Cartesian grid slice over the slat.

time for the proposed method will always be shorter compared to the original X-rays method. It can be seen in Table 2 that the user time required varies significantly based on the complexity of the geometry and the user's expertise. For example, it is estimated that the simple capsule (Fig. 1) takes five to ten minutes of user input required for the original X-rays method, while the Orion LAV geometry requires anywhere between one to two hours. The step of defining grids to be cut by each X-ray for the capsule can be summarized as "capsule cuts Cartesian grid." However, the same step in Orion LAV would be "nozzle 1 cuts main body grids, nozzle 2 cuts main body grids, nozzle 3 cuts main body grids, nozzle 1 cuts nozzle 2 grids..." This complexity grows rapidly based on the number of grids and X-ray cutters. With automation of this step, the benefit to the user becomes significant as the geometries become more complex.

	Generate OBB	Auto close	AABB holecut
Rotor-blade system	0.002	0	0.25
Orion LAV	0.003	0.282	2.80
High-lift	0.002	0.025	0.34

Table 3. Individual CPU time breakdown for hole-cutting in minutes

The CPU time used by each step of the proposed processes is listed in Table 3. It can be seen that generating the oriented bounding box and automatically closing the open boundary are relatively fast. The hole-cutting step is the most time consuming step in terms of the algorithm. The original X-ray scheme is implemented within this framework so that the minimum holecut can be generated.

	Original total time (minutes)	New total time (minutes)
Rotor-blade system	16-31	4-6
Orion LAV	63-123	18-33
High-lift	31-61	11-16

Table 4. Estimate of total time to complete minimum holecut. The lower and upper estimates correspond to expert to non-expert user times.

The time required for minimum hole-cutting is the sum total of the user time and the CPU time for steady problems. Table 4 shows the total time to generate the minimum holecut for the test cases. Even with the un-optimized hole-cutting step, the time that dominates is the user time required to close the open boundary and determine the grid points to be cut by each X-ray. Therefore, removing the manual steps of hole closure and determination of grid points to be cut by each X-ray create valuable savings in the minimum holecut generation. A formal qualitative study is planned as part of on-going work to see the effects of time savings on the user's productivity.

IV. Conclusion

A new process of automating the generation of the minimum hole in overset grids was explored with the focus of minimizing user input and effort. Two key areas of the process were identified and developed. Automatic hole closure reduced the time and user interaction required to generate watertight surfaces. Automatic determination of grids cut by each X-ray reduced the input effort required from the user. Oriented bounding boxes were developed for these two steps. The automation of two key steps prior to hole-cutting has the capability to save significant user time in the day-to-day operation of overset grids. The user time and the CPU times were compared between the original process and the new process. Future work will include development and implementation of overlap optimization for the holecut.

V. Acknowledgment

The author would like to thank Mark Potsdam and Dr. Stuart Rogers for insightful discussions into the organization of this paper.

References

[1]Meakin, R. L., "Composite Overset Structured Grids," Ch. 11, *Handbook of Grid Generation*, Eds. Thompson, Soni, Weatherill, CRC Press, 1999.

[2]Henshaw, W. D., "Ogen: An Overlapping Grid Generator for Overture," Research Report UCRL-MA-132237, Lawrence Livermore National Laboratory, 1998.

[3]Rogers, S. E., Suhs, N. E. and Dietz, W. E., "PEGASUS5 : An Automated Pre-Processor for Overset-Grid CFD," *AIAA J.*, Vol. 41, No. 6, pp. 1037-1045, Dec. 2003.

[4]R. Noack, "Direct Cut Approach for Overset Hole Cutting," AIAA Paper 2007–3835, 2007.

[5]Sitaraman, J., Floros, M., Wissink, A. and Potsdam, M., "Parallel Domain Connectivity Algorithm for Unsteady Flow Computations Using Overlapping and Adaptive Grids," *J. Comp. Phys.*, Vol. 229, pp. 4703-4723, March 2010.

[6]Meakin, R. L., "A New Method for Establishing Intergrid Communication Among System of Overset Grids," AIAA Paper 1991-1586, 1991.

[7]Meakin, R. L., "Object X-Rays for Cutting Holes in Composite Overset Structured Grids," AIAA Paper 2001-2537, 2001.

[8]T. Möller, B. T., "Fast, Minimum Storage Ray/Triangle Intersection," Journal of Graphics Tools, Vol 2, Issue 1 pp. 21-28, 1997.

[9]Chan, W. M., "Developments in Strategies and Software Tools for Overset Structured Grid Generation and Connectivity," to be presented in 20th AIAA Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 27-30, 2011.

[10]J. Shlens , "A Tutorial on Principal Component Analysis," Institute for Nonlinear Science, UCSD, 2005.

[11]J. R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," Applied Computational Geometry: Towards Geometric Engineering, Vol. 1148, pp. 203-222., May 1996.

[12]M. Botsch, et al, "Geometric modeling based on polygonal meshes," Tech. rep., 2007.

[13]K. Sugihara, H. Inagaki, "Why is the 3D Delaunay triangulation difficult to construct?" Information Processing Letters pp.54 275-280, 1995.

[14]S. Hadap, e. a., "Collision Detection and Proximity Queries," SIGGRAPH Course, 2004.

[15]S. Gottschalk, M. C. Lin, D. Manchoa, "OBBTree: A Hierarchical Structure for Rapid Interference Detection," SIGGRAPH Paper 1996, 1996.