4.3 Geometric Algorithms for 3D Interface Reconstruction

Conference Paper · January 2007

DOI: 10.1007/978-3-540-75103-8_23 · Source: DBLP

2 authors, including:

Mikhail Shashkov
Los Alamos National Laboratory
330 PUBLICATIONS **8,738** CITATIONS

4.3

Geometric Algorithms for 3D Interface Reconstruction

Hyung Taek Ahn[1] and Mikhail Shashkov[2]

[1] Los Alamos National Laboratory htahn@lanl.gov
[2] Los Alamos National Laboratory shashkov@lanl.gov

Summary. We describe geometrical algorithms for interface reconstructions for 3D generalized polyhedral meshes. Three representative piece-wise linear interface calculation methods are considered, namely gradient based method, least squares volume-of-fluid interface reconstruction algorithm, and moment-of-fluid method. Geometric algorithms for the 3D interface reconstructions are described. Algorithm for the intersection of a convex polyhedron with half-space is presented with degenerate cases. Fast iterative methods for volume matching interface computation are introduced. The numerical optimization method for interface normal computation is presented, and its super-linearly convergence is demonstrated. Finally, actual reconstruction of complex geometry is demonstrated.

Key words: Interface reconstruction, polyhedral mesh, volume of fluid, moment of fluid

1 Introduction and Background

There are several well established methods for dealing with interfaces in fluid flow: the volume of fluid (VoF) method, [8, 16, 2]; the front tracking, [22, 6, 21]; and level set method,[20, 19, 13]. For modeling three-dimensional, high-speed compressible, multi-material flows, on general meshes with the interface topology changing in time, and when exact conservation is critical, VoF seems to be method of choice, [2]. The typical VoF method consists of two steps: interface reconstruction (using volume fractions) and updating the volume fractions in time. Excellent reviews of VoF methods and, in particular, general interface reconstruction methods can be found in the following papers [16, 2, 17]. In this paper we are only interested in the *geometric algorithms* for interface reconstruction.

The most common interface representation used by interface reconstruction methods consists of a single linear interface (a line in 2D and a plane in

3D) per cell composed of multi-materials. This class of interface representation is commonly called Piecewise-Linear Interface Calculation (PLIC). The location of the linear interface, for a given volume at a cell, is uniquely defined by the interface normal. There are a number of ways to define the direction of the normal. We will describe three representative methods in Section 3.

Most of the papers related to interface reconstruction deal with the two dimensional case. There are only several papers which describe interface reconstruction in 3D; almost all of them deal with a cartesian mesh and the case of two materials, e.g., [11], and references therein. Descriptions of 3D interface reconstruction methods on distorted and unstructured meshes are very rare and are usually in unpublished reports related to ALE methods, e.g., [10, 4].

The goal of this paper is to explain geometric algorithms for 3D interface reconstructions on generalized polyhedral meshes. We consider three representative PLIC methods: (i) gradient based method (GRAD), (ii) the least squares volume-of-fluid interface reconstruction algorithm (LVIRA) [15, 14], and (iii) most recent moment of fluid (MoF) method [3, 1].

The outline of the rest of this paper is as follows. In Section 2, we describe the concept of generalized polyhedral mesh. In Section 3, we describe the three representative interface reconstruction methods. In Section 4, we introduce the intersection algorithm of convex polyhedron with half-space with degenerate cases. In section 5, we introduce two iterative volume matching interface computation algorithms. In section 6, we describe optimization method for interface normal computation. In section 7, we give a demonstration of interface reconstruction. Finally, we conclude in Section 8.

2 Multi-material interface representation in generalized polyhedral meshes

In this paper, we are interested in a mesh that consists of generalized polyhedra - *generalized polyhedral mesh - (GPM)*. A generalized polyhedron can be thought as a 3D solid obtained from a polyhedron by perturbing positions of its vertices, which makes its faces non-planar as well as the polyhedron itself non-convex.

The geometry of a face whose vertices are not all in a single plane, however, is not unique. Therefore, we adopt a faceted representation to obtain a consistent definition of its geometry (see [5] for more detail). As illustrated in Fig. 1, first we define the "face center" by averaging vertex coordinates associated with the face. Next, the faces of the generalized polyhedral cell are triangulated by using the face center and two vertices of each edge. In the second step, the triangulated generalized polyhedral cell is decomposed into sub-tetrahedra by using triangulated surfaces and one additional vertex inside the cell, called the "cell center", defined by averaging coordinates of all vertices of original polyhedral cell.

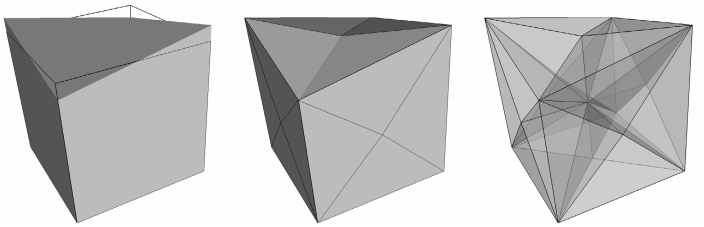

Fig. 1. Generalized polyhedral representation of a hexahedron with non-planar faces. The left figure shows the initial hexahedral cell with a non-planar top face (the wire frame of a cube is overlapped to emphasize the non-planar top face), the middle figure shows the surface triangulation of the hexahedron, and the right figure shows a sub-cell decomposition of the hexahedron.

Hence, an m-vertexed polygonal face is divided into m-triangles, and an n-faced polyhedron is further decomposed into $n \times m$ sub-tetrahedra provided that each face is composed of m vertices. For example, the generalized polyhedral representation of a hexahedral cell results in 6×4 sub-tetrahedra as displayed in Fig. 1.

This generalization of polyhedral cells has two advantages. First, the planar face restriction of a polyhedral cell is relaxed so that it can have vertices not always in a plane. Second, it allows us to deal with non-convex cells, as long as the cell can be decomposed into valid sub-cells (that is, the cell center can "see" all the vertices). These features are advantageous for dealing with meshes in ALE methods.

Three different types of generalized polyhedral mixed cells are represented in Fig. 2. Each cell includes three materials (colored red, green, and blue). Each interface is reconstructed by the intersection of the polyhedral cell with half-spaces. The sub-cells, initially tetrahedra, evolve to convex polyhedra as they intersect with their corresponding half-spaces. A wire frame view of these sub-cells, shown in the bottom row of Fig. 2, reveals these sub-cell structures.

3 Interface reconstruction methods

Three representative piece-wise linear interface calculation (PLIC) methods are discussed: the gradient based method, the least squares volume-of-fluid interface reconstruction algorithm, and the moment-of-fluid method.

3.1 Review of representative PLIC methods

In PLIC methods, each mixed cell interface between two materials is represented by plane. It is convenient to specify this plane in *Hessian normal form*

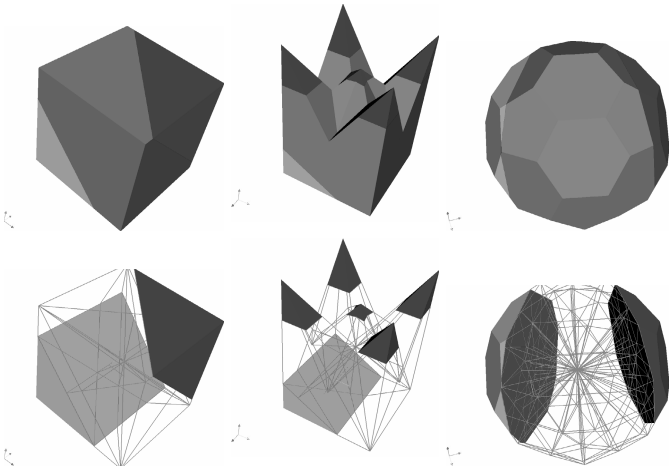

Fig. 2. Generalized polyhedral cells with multi-materials (red, green, and blue). The left column shows a hexahedral cell, the middle column shows a *non-convex* enneahedron (obtained by subdividing the top face of a hexahedron and disturbing the vertices on the faces), and the right column represents a truncated icosahedron. The top row shows the solid view, and the bottom row shows the wire-frame view of the solid which reveals the sub-cell structure.

$$\mathbf{n} \cdot \mathbf{r} + d = 0\,, \tag{1}$$

where $\mathbf{r} = (x, y, z)$ is a point in the plane, $\mathbf{n} = (n_x, n_y, n_z)$ are components of the unit normal to the plane, and d is the signed distance from the origin to the plane. The principal reconstruction constraint is local volume conservation, i.e. the reconstructed interface must truncate the cell, c, with a volume equal to the reference volume $\mathcal{V}_c^{ref}$ of the material (or equivalently, the volume fraction $f_c^{ref} = \mathcal{V}_c^{ref}/\mathcal{V}_c$, where $\mathcal{V}_c$ is the volume of the entire cell c).

Since a unique interface configuration does not exist, the interface geometry must be inferred based on local data and the assumptions of the particular algorithm. PLIC methods differ in how the normal $\mathbf{n}$ is computed. For a given normal, d is uniquely defined from the reference volume $\mathcal{V}_c^{ref}$.

In the remainder of this section, we briefly describe the main ideas of GRAD, LVIRA and MoF methods, give some details relevant to 3D extensions of these methods, and present a summary of algorithms for their implementation.

3.2 Gradient based interface reconstruction (GRAD)

In the gradient based method, the interface normal, $\mathbf{n}$, is computed by approximating the gradient of the volume fraction function f as

$$\mathbf{n} \sim -\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)\,. \tag{2}$$

In the case of a 3D unstructured mesh consisting of generalized polyhedra, it is convenient to use a least squares procedure (see, for example, [5]) to estimate the gradient of the volume fraction.

To define the interface, one needs to find the distance d in Eq. (1) such that intersection of the corresponding half-space and cell has volume $\mathcal{V}^{ref}$. To find d we need to solve the equation

$$\mathcal{V}(d) = \mathcal{V}^{ref} . \tag{3}$$

The volume $\mathcal{V}(d)$ is a continuous and monotone function of d, which guarantees that Eq. (3) always has a unique solution. Let us note that all PLIC methods require solving Eq. (3) many times. The geometrical algorithms for the intersection of a half-space and a convex polyhedron, and computation of the volume of a polyhedron are presented in the following sections.

3.3 Least squares volume-of-fluid interface reconstruction algorithm (LVIRA)

In the LVIRA interface reconstruction method introduced by Puckett [15, 14], the interface normal is computed by minimizing the following error functional:

$$E_c^{LVIRA}(\mathbf{n}) = \sum_{c' \in \mathcal{C}(c)} (f_{c'}^{ref} - f_{c'}(\mathbf{n}))^2 \tag{4}$$

where $f_{c'}^{ref}$ is the reference volume fraction of neighbor c', and $f_{c'}(\mathbf{n})$ is the actual (reconstructed) volume fraction of neighbor c' taken by extending the interface of central cell-c, under the constraint that the corresponding plane exactly reproduces the volume fraction in the cell under consideration.

The stencil for the error computation in LVIRA is illustrated in Fig. 3, where 2D meshes are employed for simplicity. The neighboring cells around a central cell-c are referenced with index j. The stencil is composed of immediate vertex neighbors. The picture on the left of Fig. 3 represents a structured quadrilateral mesh, and picture on the right shows a stencil on an unstructured polygonal mesh.

Like the GRAD method, LVIRA also requires information about the volume fractions from all immediate neighboring cells. In contrast with the GRAD method, LVIRA requires minimization of the non-linear objective function, as shown in Eq. (4). In 3D, the normal can be described by polar angles, and therefore implementation of LVIRA requires an algorithm for the minimization of a non-linear function of two variables.

3.4 Moment-of-fluid interface reconstruction (MoF)

The *moment-of-fluid* method was introduced by Dyadechko and Shashkov [3, 1] for interface reconstruction in 2D. The MoF method uses information about

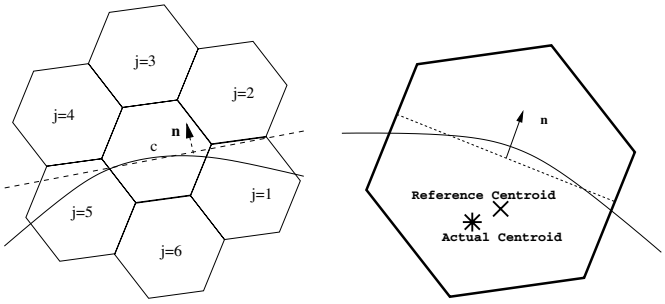

Fig. 3. Left – stencil for LVIRA error computation. The solid curved line represents the true interface, and the dashed straight line represents the extension of a piecewise linear volume fraction matching interface at the central cell-c. Right – stencil for MoF error computation. The stencil is composed of only the cell under consideration.

the volume fraction, f_c^{ref} and centroid, $\mathbf{x}_c^{ref}$ of the material, but *only* from the cell c under consideration. No information from neighboring cells is used, as illustrated in Fig. 3.

In the MoF method, the computed interface is chosen to match the reference volume exactly and to provide the best possible approximation to the reference centroid of the material. That is, in MoF, the interface normal, $\mathbf{n}$, is computed by minimizing (under the constraint that the corresponding pure sub-cell has exactly the reference volume fraction in the cell) the following functional:

$$E_c^{MoF}(\mathbf{n}) = \| \mathbf{x}_c^{ref} - \mathbf{x}_c(\mathbf{n}) \|^2 \tag{5}$$

where $\mathbf{x}_c^{ref}$ is the reference material centroid and $\mathbf{x}_c(\mathbf{n})$ is the actual (reconstructed) material centroid with given interface normal $\mathbf{n}$.

Similar to the LVIRA, the implementation of MoF method requires the minimization of the non-linear function of two variables. The computation of $E_c^{MoF}(\mathbf{n})$ requires the following steps. The first step is to find the parameter d of the plane such that the volume fraction in cell c exactly matches f_c^{ref}; this is also performed in the GRAD and LVIRA algorithms. Secondly we compute the centroid of the resulting polyhedron. This is a simple calculation, described in [12]. Finally, one computes the distance between actual and reference centroids.

4 Intersection of convex polyhedron with half-space

Convex polyhedron intersection with a half-space is the base operation for interface reconstruction in 3D. First the algorithm of intersection is presented for regular case (no vertices on cutting plane), and later issues and strategies for degenerate cases (vertices on cutting plane) will be addressed.

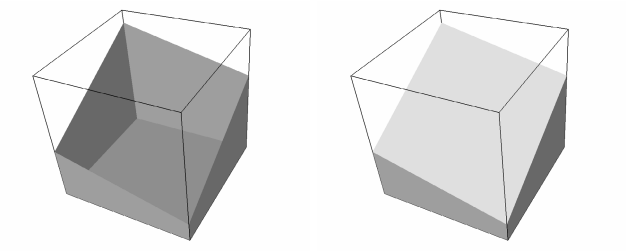

Fig. 4. Convex polyhedron intersection by clipping and capping. The left shows an open hexahedron by clipping the hexahedron with a given plane, and the right shows the closed polyhedron by clipping as well as capping.

4.1 Regular case

Algorithm for intersection of convex polyhedron with a half-space is composed of two different conceptual stages; clipping and capping [18]. The idea of clipping and capping is delineated in Fig. 4 with an example of hexahedron and plane intersection. In clipping stage, each face of polyhedron is visited and polygonal intersection is performed if the cutting plane passes through it. Depending on the distance and orientation of the given plane, it may be no intersection and the face is considered as a *pure* face, *i.e.* the face is completely above or below with respect to the given plane. In the clipping, stage, no specific order is necessary for visiting polyhedron faces, and each face visit can be considered as a polygon and plane intersection in 3D.

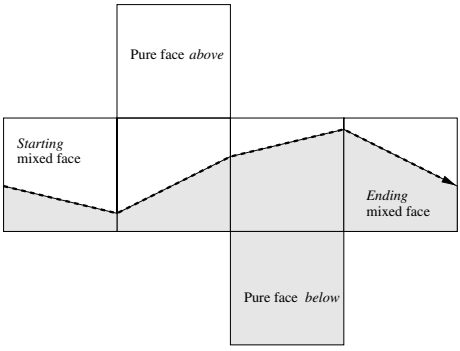

Fig. 5. Unfolded faces of hexahedron. Clipped hexahedron faces are displayed in gray color, and continuation of slice curve (polylines on unfolded plane) for capping is illustrated with dashed arrow lines.

In the latter, capping stage, the polygonal slice face has to be constructed. Without capping, the merely clipped polyhedron will result in an *open* poly-

hedron as shown in Fig. 4. The boundary (edges with only single neighbor) of the open polyhedron represents the slice curve generated by the given cutting plane and original polyhedron. To make the open polyhedron be a closed polyhedron having all edges two neighbors, the slice face is identified by capping stage. In contrast to the clipping stage, the capping stage needs a proper orders of face-visits for slice face construction. The slice face is constructed by continuation of the slice curve as delineated in Fig. 5. The slice curve can get started with any given mixed face. The curve is continued by looping the adjacent mixed faces until it returns back to the initial mixed face and completing closed slice curve.

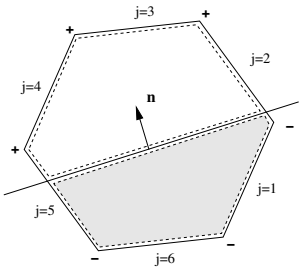

Fig. 6. Convex polygon intersection with a plane of interface normal (**n**) in 3D. Polygon intersection routine returns two sub-polygons indicated by dashed lines: first sub-polygon which is below to the cutting plane (gray part) and second sub-polygon above (void part). New vertices are generated by intersection of the plane and edges with different signs ($j = 2, 5$).

The convex polyhedron intersection algorithm incorporating both clipping and capping is illustrated in Algorithm 1. The inputs of the polyhedron intersection routine is initial *polyhedron* and cutting *plane*, and the outputs are two closed sub-polyhedra; *phed*1 below the given cutting plane and *phed*2 above the plane. Inside of the loop of mixed faces, polygon intersection is performed. Intersection of convex polygon with a plane in 3D is illustrated in Fig. 6. In the convex polygon intersection subroutine as described in Algorithm 2, like polyhedron intersection routine, the inputs are a *polygon* (a face of *polyhedron*) and cutting *plane*, and the outputs are two closed polygons; *pgon*1 which is below the plane and *pgon*2 which is above the plane. Polygon intersection algorithm is similar to that of polyhedron intersection as presented in Algorithm 1. The main difference of polygon intersection is edge-wise loop is carried out instead of face-wise loop in polyhedron intersection.

4.2 Degenerate cases

Cutting plane does not always intersect edges (break the edge into two parts), and either or both of the vertices can be exactly on the cutting plane. For

```
Input: polyhedron, plane
Output: phed1, phed2
foreach face of polyhedron do
    if unvisited face then
        if face below to plane then                    /* pure face */
            add this face to phed1 ;
            mark this face visited ;
        else if face above to plane then               /* pure face */
            add this face to phed2 ;
            mark this face visited ;
        else if face gets intersection then            /* mixed face */
            face_start ← this face ;
            repeat
                perform polygon/plane intersection ;
                add face below to phed1 ;
                add face above to phed2 ;
                mark this face visited ;
                face_next ← next face ;
            until face_start = face_next ;
        end
    end
end
```

Algorithm 1: Convex polyhedron intersection

```
Input: polygon, plane
Output: pgon1, pgon2
foreach edge of polygon do
    if edge below to plane then                /* both vertices (-) */
        add this edge to pgon1 ;
    else if edge above to plane then           /* both vertices (+) */
        add this edge to pgon2 ;
    else if edge gets intersection then        /* vert's sign mixed */
        perform segment/plane intersection ;
        add edge below to pgon1 ;
        add edge above to pgon2 ;
    end
end
```

Algorithm 2: Convex polygon intersection

example, the vertices of polyhedron can be exactly (or within some tolerance) on the given plane. This results in degenerate cases, as delineated in Fig. 7. Degenerate cases requires two additional considerations. First, in polygon intersection routine, additional vertex may not be generated by intersection (of plane and edge in 3D), instead an existing vertex is used for it. Second, for the continuation of the slice curve as delineated in Fig. 8, next adjacent face should be found carefully because not only the mixed faces but also the pure

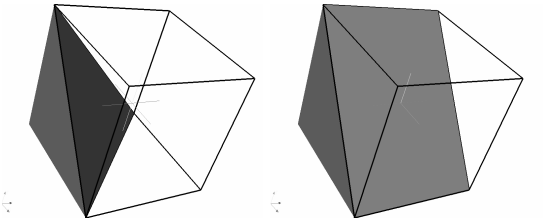

Fig. 7. Degenerate cases in polyhedron intersection: vertices on the intersecting plane (left), and vertices as well as edges on the intersection plane (right).

faces (if an edge is on the cutting plane) can be the candidate for the next face.

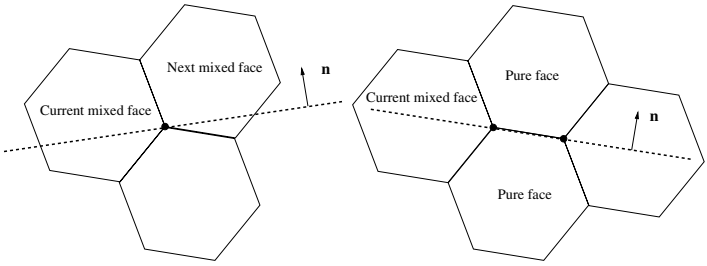

Fig. 8. Degenerate cases in polygon intersection: one vertex on the intersecting plane (left), and two vertices (edge) on the intersection plane (right). Cutting plane is delineated with dashed line, and vertices on the plane is marked with •.

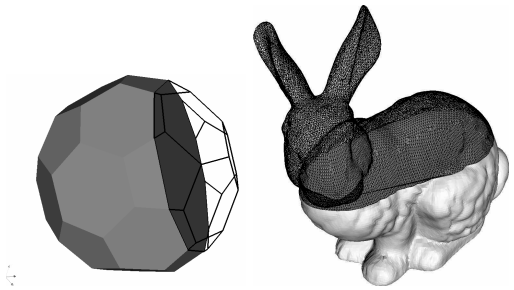

Fig. 9. Intersection of complex polyhedra. Left shows 32 faced truncated icosahedron, and right shows 725,000 faced bunny mesh. Both surface mesh represents a polyhedron.

4.3 Test cases

The convex polyhedron intersection algorithm is applied for more general cases in Fig. 9. Two polyhedra intersected by the present algorithms are displayed. First the intersection of truncated icosahedron (*a.k.a* soccer ball geometry with 12 pentagons and 20 hexagons) is presented, and next the polyhedral representation of bunny (725,000 triangular surface mesh) is intersected. The bunny geometry is not an example of convex polyhedron, but as long as the slice face is *simply connected* and the polyhedron is convex-faced (triangulated surface here) the current algorithm can be applied.

4.4 Volume and centroid computation

For each polyhedron intersection, volume and centroids of the intersected subcell have to be computed for measuring the error of interface reconstructed. For this purpose, fast and accurate computation of moment data of general polyhedron is indispensable, and our implementation is based on [12]. The algorithm is based on multi-step reduction of the volume integral to successively lower dimensions by using Divergence and Green's theorems.

5 Volume matching interface computation

In this section, the target volume fraction matching interface calculation, with *given normal*, methods are presented. The primary mechanism of volume preserving interface reconstruction requires cutting appropriate volume fraction of the cell, as expressed in Eq. (3). The equation can also be expressed as follows

$$f(d) = f^{ref}$$

where $f(d) = \mathcal{V}(d)/\mathcal{V}_{cell}$ is volume fraction defined by d and $f_{ref} = \mathcal{V}^{red}/\mathcal{V}_{cell}$ is reference (target) volume fraction, which are both normalized by cell volume $\mathcal{V}_{cell}$. Since the normal (orientation) of cutting plane is given, the volume of intersection is purely function of distance, $d \in [d_{min}, d_{max}]$. For example, $f\,(d_{min}) = 0$ and $f\,(d_{max}) = 1$.

Several approaches are proposed, but they are mainly described in 2D. These methods, *e.g.* analytical method [3] and semi-iterative method [16], require two pre-processing: first vertex-wise volume fraction evaluation ($O(n)$ volume fraction evaluation) and then another vertex-wise volume fraction sorting ($O(n \log n)$ operations in sorting), where n is number of vertices.

The analytical approach can be extended for tetrahedral cell in 3D [23]. For cells with small number of vertices, such as triangles in 2D and tetrahedra in 3D, this pre-processing and analytical approach could save CPU time. As the cells contains more vertices, typical for 3D polyhedral cells, these preprocessing demand considerable amount of CPU time as well as extra memory space besides the implementation efforts.

In order to cut target volume fraction accurately as well as efficiently two fully iterative schemes are employed, namely secant method and bisection method. The algorithm for the iterative methods is described in Algorithm 3.

Input: f^{ref}, $\mathbf{n}$, d_{min}, d_{max}
Output: d
$d_1 = d_{min}$;
$d_2 = d_{max}$;
$f_1 = 0$;
$f_2 = 1$;
repeat
  if *Secant method used* **then**
    $secant = (f_2 - f_1)/(d_2 - d_1)$;
  end
  update d by Secant or Bisection method;
  intersect cell with defined interface $(\mathbf{n}, d)$;
  compute $f(d)$;
  $\Delta f = |f^{ref} - f(d)|$;
  update d_1, d_2, f_1, f_2;
until *($\Delta f < tol$)* ;

Algorithm 3: **Iterative volume fraction matching interface computation**

These iterative schemes have too distinctive advantages:

1. no pre-processing: vertex-wise volume fraction evaluation or sorting
2. fixed number of iteration regardless of number of vertices

First, no vertex-wise volume fraction evaluation or sorting is needed. For the start of the iteration, only the minimum and maximum distances with respect the the given interface normal are required. This is because of the monotonically increasing behavior (actually C^1 if the cell is convex and C^0 if not) of volume fraction with respect to the distance. Second, both iterative schemes are converging in almost fixed number of iterations regardless of cell size. In bisection method, with unit interval of distance [0,1] the number of iterations required to achieve distance error tolerance of $tol = 10^{-10}$ is

$$\log_2 \frac{1}{10^{-10}} = 33.2193$$

regardless of function behavior [7]. Due to monotonicity of the function, the volume fraction error tolerance of $tol = 10^{-10}$ is also achieved with approximately same number of bisection iterations as shown in Fig. 10.

Fig. 10 shows the volume fraction convergence history of the two iterative methods applied to three polyhedral cells shown in Fig. 2. First, the secant method shows super linear convergence in volume fraction error. Less than 10

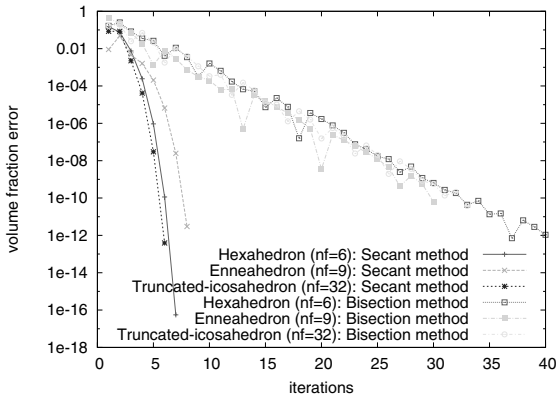

Fig. 10. Volume fraction convergence of secant and bisection methods with three polyhedral cells shown in Fig. 2.

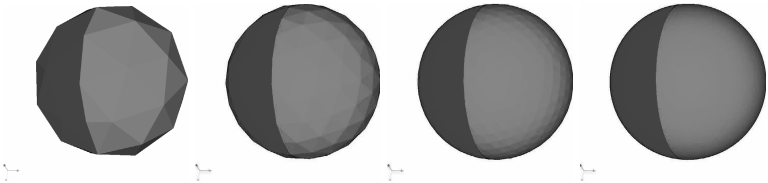

Fig. 11. Spherical cell containing two materials (blue and red) with successive refinement. From the coarsest (left) mesh to the finest (right), the numbers of faces are 80, 320, 1280, and 5120.

iterations are required to achieve the volume fraction error $< 10^{-10}$. Bisection method shows linear convergence, but it guarantees that only fixed number of iteration is required regardless of the number of vertices, n, for the cell.

The efficiency of iterative methods are further demonstrated with large size spherical cells as displayed in Fig. 11. Four levels of successively refined spherical surface meshes are used as a single polyhedral cell representation. In Fig. 12, using the four levels of spherical cells, the number of secant iteration to achieve $err(f) < 1.e-10$ is measured with target volume fraction between [0,1]. The number of iteration required is irrespective to the size of polyhedral cell, and the target volume fraction is achieved almost less than 10 iterations.

6 Numerical optimization

The second-order accurate interface reconstruction methods, LVIRA and MoF, require additional optimization process minimizing the error function-

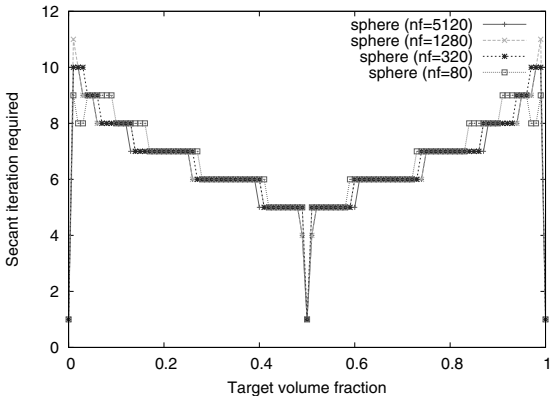

Fig. 12. Number of secant iterations required for appropriate volume fraction cutting measured with four levels of spherical cells displayed in Fig. 11

als. In both case optimization has to be performed with respect to the normal $\mathbf{n}$ in the Eq. (1). In 3D normal $\mathbf{n}$ is defined by two polar angles (ϕ, θ).

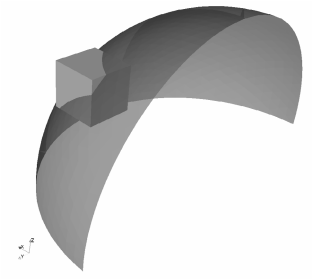

Fig. 13. Interface configuration by a sphere, centered at $(-0.1, -0.2, -0.3)$ with radius $r = 1.3$, and equispaced 3^3 hexahedral mesh covering the domain of $[0, 1]^3$. For visualization purpose, a fraction of transparent sphere and the central cell of the mesh are displayed.

By using the spherical interface configuration delineated in Fig. 13, typical behavior of the objective functions are displayed in Fig. 14. For the interface configuration in Fig. 13, sphere centered at $(-0.1, -0.2, -0.3)$ with radius $r = 1.3$ is used and equispaced $3 \times 3 \times 3$ hexahedral mesh covering the domain of $[0, 1] \times [0, 1] \times [0, 1]$ is used. For visualization purpose, a transparent fraction of the sphere and the cell centered at $(0.5, 0.5, 0.5)$ are displayed.

The behavior of objective functions for the center cell shown in Fig. 13 are displayed in the top row of Fig. 14. General trends of these objective functions

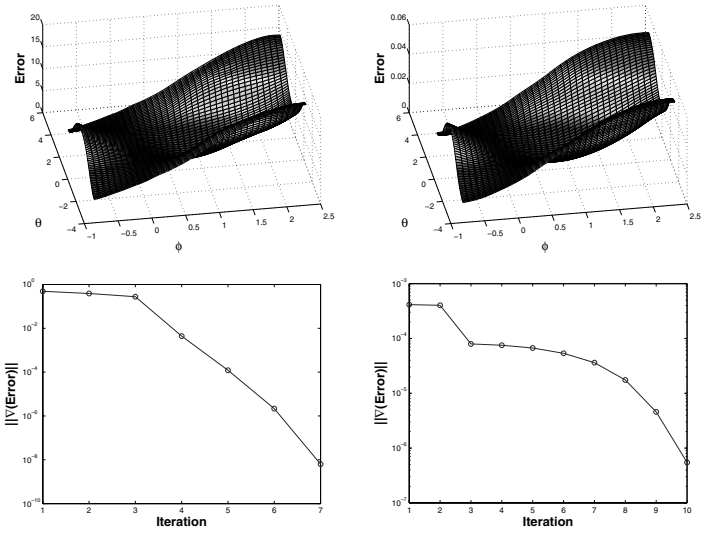

Fig. 14. Behavior of objective function and optimization of it: Left — LVIRA, and right — MoF. Top row is 3D view of the objective functions, and the bottom row shows the convergence history of optimization process.

are similar for both LVIRA and MoF. However, the scale of absolute values of the functions are different. This is because LVIRA uses accumulated volume fraction difference from neighbors and MoF uses normalized distances between centroids as the objective function.

For the above multi-dimensional minimization, Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [9] is used. It is a quasi Newton method, approximating Hessian matrix with a set previous of gradients. The gradients of the objective function are computed by finite differences. For each search direction, a quadratic or cubic polynomial line search is performed for sufficient decrease in the error with the Armijo rule for step size control. Detailed discussion of the BFGS method can be found in [9]

For the initial guess of the optimization, the gradient of volume fraction computed as in GRAD is utilized for LVIRA.

$$\mathbf{n}_0(\phi_0, \theta_0)_{LVIRA} = -\mathbf{GRAD}(f). \tag{6}$$

For MoF, the unit vector along the given material centroid to the cell centroid is used as follows

$$\mathbf{n}_0(\phi_0, \theta_0)_{MoF} = \frac{\mathbf{x}_c^{cell} - \mathbf{x}_c^{ref}}{\| \mathbf{x}_c^{cell} - \mathbf{x}_c^{ref} \|} \tag{7}$$

where $\mathbf{x}_c^{cell}$ is cell centroid and $\mathbf{x}_c^{ref}$ is the reference material centroid. Once the gradient of the objective function becomes less than a given tolerance, it is considered that a local minimum is found and the optimization process terminates.

The convergence history of the LVIRA and MoF are displayed in the bottom row of Fig. 14. Both LVIRA and MoF show super-linear convergence rate, and $\| \nabla(Error) \| < 10^{-6}$ are achieved with 10 iterations.

The final reconstructed interface for the mixed cell configuration shown in Fig. 13 is displayed in Fig. 15. Fraction of original spherical interface is overlapped with transparency. Depending on the reconstruction methods, the interface normal is different and this results different interface reconstruction as shown in the figure. The volumes of symmetric difference, between the original and reconstruction, at the cell are measured as follows: 6.1616e-04 (GRAD), 5.9999e-04 (LVIRA), 5.5676e-04(MoF). This result strengthens that MoF gives the best accuracy.

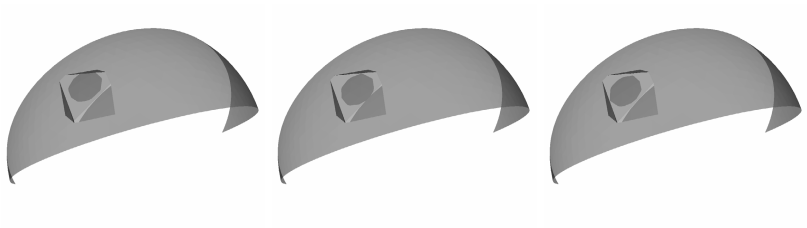

Fig. 15. Interface reconstruction of the configuration shown Fig. 13. Left – GRAD, middle – LVIRA, and right – MoF.

7 Reconstruction of complex interfaces

The effectiveness of the geometric algorithms is demonstrated by reconstructing complex interfaces, as shown in Fig. 16. The initial geometry of the object is given by a surface mesh, and then a tetrahedral volume mesh is generated based on it. Tetrahedron-tetrahedron intersection is performed to compute the volume fraction and moment data exactly. The reconstructed object by MoF method is displayed in Fig. 16.

8 Conclusion

We have described geometric algorithms for 3D interface reconstructions with 3D generalized polyhedral meshes. Three different reconstruction methods are

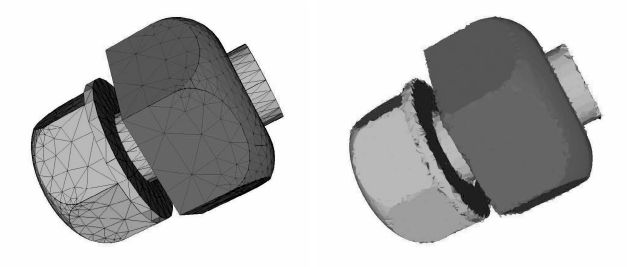

Fig. 16. Multi-material ($n_{mat} = 3$, i.e. bolt, nut, and background) interface reconstruction with MoF. The left represents original material region represented by tetrahedral volume meshes, and the right shows its reconstruction with the MoF method on an unstructured tetrahedral mesh.

considered, namely GRAD, LVIRA, and MoF. Intersection algorithm for convex polyhedron with a half-space is presented including degenerate cases. Fast iterative methods for volume fraction matching interface calculation are presented. Optimization algorithm for second order accurate interface reconstruction methods (LVIRA and MoF) is explained and super-linear convergence of it is demonstrated. Performance of the methods is demonstrated with actual reconstruction of complex geometry.

Acknowledgement. This work was supported by the Advanced Simulation and Computing (ASC) program at the Los Alamos National Laboratory.

References

1. H. T. Ahn and M. Shashkov. Multi-material interface reconstruction on generalized polyhedral meshes. *Journal of Computational Physics*, In press, 2007.
2. D. J. Benson. Volume of fluid interface reconstruction methods for multi-material problems. *Applied Mechanics Reviews*, 55:151–165, 2002.
3. V. Dyadechko and M. Shashkov. Moment-of-fluid interface reconstruction. Technical Report LA-UR-05-7571, Los Alamos National Laboratory.
4. D. M. Gao. A three-dimensional hybrid finite element-volume tracking model for mould filling in casting processes. *International Journal for Numerical Methods in Fluids*, 29:877–895, 1999.
5. R. Garimella, M. Kucharik, and M. Shashkov. An efficient linearity and bound preserving conservative interpolation (remapping) on polyhedral meshes. *Computers and Fluids*, 36:224–237, 2007.
6. J. Glimm, J. W. Grove, X. L. Li, K. Shyue, Y. Zeng, and Q. Zhang. Three-dimensional front tracking. *SIAM Journal on Scientific Computing*, 19:703–727, 1998.
7. Michael T. Heath. *Scientific Computing: An Introductory Survey, Second Edition.* McGraw-Hill, New York, 2002.

8. C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39:201–225, 1981.
9. C.T. Kelly. *Iterative methods for optimization*. Society for Industrial and Applied Mathematics, 1999.
10. D. B. Kothe, M. W. Williams, K. L. Lam, D. R. Korzekwa, P. K. Tubesing, and E. G. Puckett. A second-order accurate, linearity-preserving volume tracking algorithm for free surface flows on 3-d unstructured meshes. In *Proceedings of the 3rd ASME/JSME joint fluid engineering conference, San Francisco, CA, USA, fEDSM99-7109*, 1999.
11. P. Liovic, M. Rudman, J.-L. Liow, D. Lakehal, and D. Kothe. A 3d unsplit-advection volume tracking algorithm with planarity-preserving interface reconstruction. *Computers & Fluids*, 35:1011–1032, 2006.
12. B. Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1:31–50, 1996.
13. S. Osher and R. P. Fedkiw. Level set methods: An overview and some recent results. *Journal of Computational Physics*, 169:463–502, 2001.
14. J. E. Pilliod and E. G. Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199:465–502, 2004.
15. E.G. Puckett. A volume-of-fluid interface tracking algorithm with applications to computing shock wave refraction. In H. Dwyer, editor, *Proceedings of the Fourth International Symposium on Computational Fluid Dynamics*, pages 933–938, 1991.
16. W. J. Rider and D. B. Kothe. Reconstructing volume tracking. *Journal of Computational Physics*, 121:112–152, 1998.
17. R. Scardovelli and S. Zaleski. Direct numerical simulation of free-surface and interfacial flow. *Annual Review of Fluid Mechanics*, 31:567–603, 1999.
18. Michael B. Stephenson and Henry N. Christiansen. A polyhedron clipping and capping algorithm and a display system for three dimensional finite element models. *ACM SIGGRAPH Computer Graphics*, 9:1–16, 1975.
19. M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148:81–124, 1999.
20. M. Sussman, E. Fatemi, P. Smereka, and S. Osher. Improved level set method for incompressible two-phase flows. *Computers & Fluids*, 27:663–680, 1998.
21. G. Tryggvasona, B. Bunnerb, A. Esmaeelic, D. Juricd, N. Al-Rawahic, W. Tauberc, J. Hanc, S. Nase, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, 169:708–759, 2001.
22. S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100:25–37, 1992.
23. Xiaofeng Yang and Ashley J. James. Analytic relations for reconstructing piecewise linear interfaces in triangular and tetrahedral grids. *Journal of Computational Physics*, 214:41–54, 2006.