ISOSURFACE EXTRACTION FROM HYBRID UNSTRUCTURED GRIDS CONTAINING PENTAHEDRAL ELEMENTS

Akshay Narayan[1], Jaya Sreevalsan-Nair[1], Kelly Gaither[2] and Bernd Hamann[3]
[1]*International Institute of Information Technology - Bangalore, 26/C, Electronics City Hosur Road, Bangalore 560100, India*
[2]*Texas Advanced Computing Center, University of Texas at Austin, 10100 Burnet Road, Austin, TX 78758, U.S.A.*
[3]*Institute for Data Analysis and Visualization, Department of Computer Science, University of California Davis, One Shields Avenue, Davis, CA95616, U.S.A.*

Keywords: Volume Visualization, Isosurface Extraction, Hybrid Meshes, Marching Methods.

Abstract: Grid-based computational simulations often use hybrid unstructured grids consisting of various types of elements. Most commonly used elements are tetrahedral, pentahedral (namely, square pyramids and right triangular prisms) and hexahedral elements. Extracting isosurfaces of scalar fields defined on such hybrid unstructured grids is often done using indirect methods, such as, (a) subdividing all non-tetrahedral cells into tetrahedra and computing the triangulated isosurfaces using the marching tetrahedra algorithm, or (b) triangulating intersection points of edges of cells and computing the isosurface using a standard triangulation algorithm. Using the basic ideas underlying the well-established marching cubes and marching tetrahedra algorithms, which are applied to hexahedral and tetrahedral elements, respectively, we generate look-up tables for extracting isosurfaces directly from pentahedral elements. By using appropriate look-up tables, it is possible to process nearly all types of hybrid unstructured grids used in practical applications without the need for indirect methods. We construct look-up tables for square pyramidal and triangular prismatic cells with accurate topological considerations.

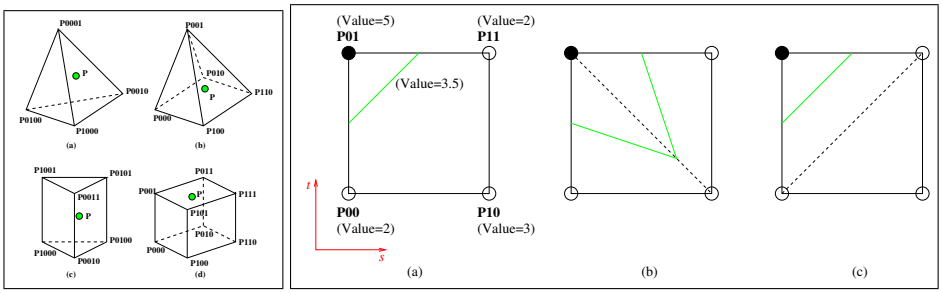
1 INTRODUCTION

Computer simulations of physical phenomena often use hybrid unstructured grids with complex geometrical structure. For example, these grids are routinely used in computational fluid dynamics (CFD) and finite element analysis (FEA). Performing correct direct interpolation on such hybrid grids, e.g., in the context of direct volume visualization or isosurface extraction, is central to its analysis. The complex geometry and topology, i.e. element connectivity, of such grids makes its analysis more challenging.

A great deal of progress has been made in recent years in the field of volume rendering using ray-casting. However, research related to isosurface extraction from hybrid grids has not been fully addressed for the entire spectrum of element types, which are commonly used, apart from hexahedra and tetrahedra. Some of the commonly used elements, which we will be focussing on in this work, are shown in Figure 1(A). We have used the following interpolation models with three linearly independent variables for these elements:

- **Tetrahedron:** Linear interpolation, often using barycentric coordinates, α, β, γ, and δ, is used, where the following rule applies: $\alpha+\beta+\gamma+\delta=1$.
- **Square Pyramid (Pentahedron):** Bilinear interpolation is used for the base quadrilateral combined with linear interpolation from the base quadrilateral to the apex, i.e. the opposing single vertex. (s,t) are used for bilinear interpolation, and u is the parameter used for linear interpolation along axis.
- **Right Triangular Prism (Pentahedron):** Linear interpolation, often performed using barycentric coordinates, is applied to the two opposite triangular faces (bases) combined with linear interpolation along the axis of the prism, i.e., in the direction from one triangular base to the other. u is the parameter used for linear interpolation along the axis and (α,β,γ) are used as barycentric coordinates for the interpolation within the triangular

Narayan A., Sreevalsan-Nair J., Gaither K. and Hamann B..
ISOSURFACE EXTRACTION FROM HYBRID UNSTRUCTURED GRIDS CONTAINING PENTAHEDRAL ELEMENTS.
DOI: 10.5220/0003852506600669
In *Proceedings of the International Conference on Computer Graphics Theory and Applications* (IVAPP-2012), pages 660-669
ISBN: 978-989-8565-02-0



(a) (b)

Figure 1: (a) Different element/cell types: (a) Tetrahedron, (b) Pentahedron - Square Pyramid, (c) Pentahedron - Right Triangular Prism, and (d) Hexahedron. Using parametric representation based on the interpolation models, a point P in space with respect to each cell is represented as (a) $P(\alpha,\beta,\gamma,\delta)$, (b) $P(s,t,u)$, (c) $P(\alpha,\beta,\gamma,u)$, (d) $P(s,t,u)$. (b) Difference of isolines without and with triangulation: For extracting isoline for function value 3.5, (a) shows the approximation of isoline using bilinear interpolation-based contouring, (s,t) being the parametric representation; (b) and (c) show the two different isolines obtained when considering different triangulations of the quadrilateral. Note that the isolines in (b) and (c) are still topological homotopes.

bases. The barycentric coordinates follow the following condition: $\alpha+\beta+\gamma=1$.

- **Hexahedron:** Trilinear interpolation is used, with parametric representation s,t,u, which represent x-, y-, and z-coordinates respectively.

All parameters of points interior to the cell satisfy the property of being in the real interval $[0,1]$. The models are popularly used owing to their lower computational complexity. In addition to these cell types, there exist several other types of finite elements all having their own associated interpolation functions. Nevertheless, those other types do not occur in our specific applications.

The primary reason for lack of efforts has been that hybrid grids consist of elements or cells whose basis functions are no longer purely linear or trilinear. This leads to more complex interpolating functions which are more challenging to analyze. However, recent increase in the availability and accessibility of powerful computational resources has increased use of hybrid grids for several CFD and finite element method (FEM) applications, thus demanding more in-depth analysis of such grids.

Existing isosurface extraction methods, such as the commonly used marching cubes or marching tetrahedra algorithms work well on hybrid grids, subject to converting a given hybrid grid first to a purely hexahedral grid or a purely tetrahedral grid, respectively. Converting the hybrid grid, either by resampling the grid to a rectilinear grid, or decomposing the grid elements into tetrahedral elements, introduces additional approximation errors and is not a unique solution. Additionally, the conversion approaches scale linearly with problem size, which may slow down performance for larger datasets.

We attempted to use open source visualization tools like ParaView (Moreland, 2011), and GMV (Ortega, 2011), on a large hybrid grid dataset, whose size is of the order of 1 GB. However, loading such a large dataset into the system memory and generating isosurfaces required approximately 10 minutes in the serial implementation of ParaView. ParaView's parallel implementation showed slightly better performance than the serial version. It was not possible to obtain isosurfaces in real time for our specific dataset, which motivated our research to perform direct extraction of isosurfaces from hybrid grids. Though our results do not show any significant improvement in performance in the case of the large dataset, our work conclusively analyzes the various topological orientations of the pentahedral cells so that we obtain a unique isosurface for any given value.

The key contribution of our work is to provide look-up tables to extract isosurfaces directly from five-node (square pyramidal) and six-node (right triangular prismatic) pentahedral elements, which includes all topological configurations, and the computations involved in determining body saddle points for specifically, the triangular prismatic element. These look-up tables enable us to obtain a unique solution for isosurface for a given function value. Additionally, we have used a combination of look-up tables from the traditional marching tetrahedra and extended marching cubes, and our proposed tables for pentahedral cells, to use the native elements directly to compute isosurfaces.

2 RELATED WORK

The foundation of our work has been the marching

cubes algorithm, which has been extensively researched in the scientific visualization community, for which (Lorensen and Cline, 1987), (Nielson and Hamann, 1991), (Nielson, 2003), (Lopes and Brodlie, 2003) are some of the representative papers, (Newman and Yi, 2006) is a comprehensive survey of the variants of the algorithm. Additionally, a great deal of work has been done for both software- and hardware-based approaches in volume rendering, for example, (Williams et al., 1998), (Muigg et al., 2007), etc.

(Gallagher and Nagtegaal, 1989) extended the marching cubes algorithm to hybrid grids containing tetrahedral, prismatic and hexahedral elements. (Takahashi et al., 2004) elaborated the look-up table approach for octahedral elements.

In the visualization software GMV, isosurface extraction from hybrid grids is implemented by determining intersection points of isosurfaces on edges of cells and triangulating these points to generate a 2-manifold surface (Ortega, 2008). However, this method discards the information provided by the grid during the triangulation stage, which can lead to a different solution from the one obtained when using an interpolation function for each cell.

(Bhaniramka et al., 2004) presented an algorithm for automatically generating case tables for isosurfaces in cells containing hypercubes, cells with 2^k vertices in k-dimensional space. Their algorithm creates look-up tables similar to that of (Montani et al., 1994). This algorithm can be extended to pentahedral cells, as it is applicable to all topological homotopes of hypercubes in the k-dimensional space.

(Weber et al., 2003) described a crack-free isosurface extraction algorithm using "stitch cells", specifically for grids subjected to adaptive mesh refinement (AMR). These stitch cells could be tetrahedra, pentahedra, or hexahedra. We have used the interpolation functions that (Weber et al., 2001) use.

While methods by (Gallagher and Nagtegaal, 1989), (Weber et al., 2003) and (Bhaniramka et al., 2004) work for our application, we are going a step further towards resolving the topological configurations for pentahedral cells by using similar patterns that are found in hexahedra. We follow the indexing for the configurations used by (Nielson, 2003) for hexahedra. In relation to our argument against subdivision of elements, (Carr et al., 2006) have discussed various artifacts that can be introduced while performing simplicial subdivision of a hexahedral element.

3 DISADVANTAGES OF APPROACHES BASED ON TETRAHEDRALIZATION OF HYBRID GRIDS

The marching tetrahedra algorithm is one of the most convenient isosurface extraction algorithms, devoid of the ambiguous cases which occur in the case of the marching cubes algorithm. In the case of hybrid unstructured grids, extracting isosurfaces using the marching tetrahedra algorithm requires an extra processing step of subdividing the non-tetrahedral elements into tetrahedral ones ensuring continuity of isosurface across faces. Since tetrahedra are basic building blocks, and all complex geometric shapes can be broken down into tetrahedra, it is one of most commonly used finite elements. However, one has to be aware of the differences in interpolants that occur when decomposing a hybrid grid to a tetrahedral grid.

A straightforward tetrahedralization can be done without inserting new vertices in the grid. However, this is not possible in certain cases. The first step in tetrahedralization of cells is the subdivision of its polygonal faces to triangles. In a standard Lagrange finite element, a bilinear interpolation function is used in the quadrilateral face. Considering the parametric representation of a function F on a bilinear surface, at any point $P(x,y,z)$, we get, $F(x,y,z) = f(s,t) = (1-s)(1-t)F_{00} + s(1-t)F_{10} + stF_{11} + (1-s)tF_{01}$, where (s,t) is the parametric representation of the point $P(x,y,z)$ in the quadrilateral $P_{00}P_{10}P_{11}P_{01}$, as shown in Figure 1(B)(a); and F_{ij} is the function value at vertex P_{ij} for $\{i,j\} = \{0,1\}$. On the diagonals (where $s = t$ or $s + t = 1$), the interpolating function is quadratic in either s or t, different from the linear interpolation function used on triangulating the face. The different interpolation models used for computing and visualizing the solution can lead to artifacts in the isosurface as shown in (Carr et al., 2006).

Subdividing a quadrilateral face can lead to two solutions as either of its two diagonals can be used to triangulate the surface. Thus, different possible tetrahedralizations can lead to different results for isosurfaces. There will be differences in the isolines generated for a quadrilateral face, depending on the choice of triangulation, as shown in Figure 1(B).

Additionally, in the case of large datasets, the computational and storage overhead induced by generating and using the additional elements may cancel the gain of eliminating ambiguities and using linear elements. Minimally, a pyramid can be decomposed to two tetrahedra, a prism to three and a hexahedron to five. For example, in the missile dataset we have used,

for each time-step we have around 24,800,000 tetrahedra, 17,700 pyramids and 4,207,000 prisms; which on tetrahedralization would result in 37,456,400 tetrahedra, a 30% increase in the number of cells. These additional tetrahedra also introduce additional memory overhead which can slow down performance to a certain extent.

(Carr et al., 2006) discussed on how minimal subdivision of hexahedral cells causes cracks in the isosurface and hence using a parity rule while subdividing is essential for a crack-free isosurface. The parity rule ensures that a quadrilateral face shared between two cells uses the same diagonal for triangulation to ensure C_0 continuity of the isosurface across the face. In (Lasser, 1985), continuity conditions for Bernstein-Bézier functions defined over the types of volumetric grid elements, which we are concerned with here, were used for construction of gradient-continuous spline approximations.

4 PENTAHEDRAL CELLS

Pentahedral cells are frequently used in conformal grids for "stitching" pure tetrahedral and/or pure hexahedral grids. The five-node (square pyramidal) and the six-node (right triangular prismatic) pentahedral cells are very commonly used as "stitch cells" or filler cells in hybrid unstructured grids.

4.1 Interpolating Functions

The interpolation functions for a square pyramid and right triangular prism depend on their respective orientation. For sake of simplicity, we assume that the axis of the cell is along z-axis in its local coordinates. This section shows that the interpolation functions in the case of the pentahedral cells are not symmetric with respect to the basis vectors, as are the cases with the tetrahedron and the hexahedron. The interpolation functions that we use for the pentahedral cells, reduce to a linear function at the edges and triangular faces, and to a bilinear function at the quadrilateral faces. Thus, our interpolation models ensure that the resulting isosurface is C_0-continuous across elements.

4.1.1 Square Pyramids

The interpolation function for a square pyramid is given by the following algebraic expression with real coefficients:

$F(x,y,z) = C_0 + C_1x + C_2y + C_3xy + C_4z$

The simplest parametric representation of a point in the cell using three linearly independent variables is by using: (a) parameter u representing position in the z direction, and (b) (s,t) for the parametric representation of the point on a bilinearly interpolated surface in a quadrilateral slice containing the point as shown in Figure 2(A). $(u=0)$ and $(u=1)$ represent the quadrilateral base and the apex, respectively; $(s=0)$, $(s=1)$, $(t=0)$, and $(t=1)$ represent the four triangular faces of the pyramid, respectively.

Let the cell be defined with vertices P_{001} at the apex and P_{000}, P_{100}, P_{110}, and P_{010} at the base. The function value at P_{stu} is given by F_{stu} and coordinates are given by $(x_{stu}, y_{stu}, z_{stu})$. Every point $P(x,y,z)$ in space can be represented as $p(s,t,u)$ with respect to this cell, and the function value at P, $F(x,y,z)$ is interpolated using: $F(x,y,z) = f(s,t,u) = uF_{001} + (1-u)F'_{001}$

where, $F'_{001} = (1-s)(1-t)F_{000} + s(1-t)F_{100} + stF_{110} + (1-s)tF_{010}$

For any point in the interior or on the boundary of the cell, $0.0 \leq s,t,u \leq 1.0$.

Any permissible value of u defines a quadrilateral slice formed by vertices at a ratio of $u:(1-u)$ along the edges from apex to base. To determine (s,t,u) for a given point, $P(x,y,z)$, we perform the following steps:

1. For a planar base, we use the following values to determine u:

   (a) For the base $P_{000}P_{100}P_{110}P_{010}$, the normal vec-

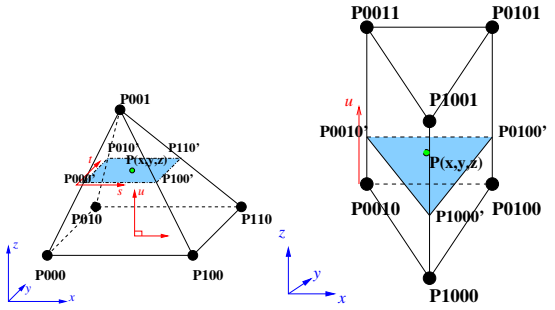


Figure 2: Determining parametric representation of an interior point $P(x,y,z)$ in: (a) a square pyramid: (s,t,u) three-tuple parametric representation is used for points. With respect to the quadrilateral slice $P'_{000}P'_{100}P'_{110}P'_{010}$ containing P being parallel to base face $P_{000}P_{100}P_{110}P_{010}$, u is the parameter in the z direction; (s,t) is the two-tuple parametric representation of the point in the slice, which degenerates to a single point at the apex P_{001}. (b) a right triangular prism: (α,β,γ,u) four-tuple parametric representation is used for points. With respect to the triangular slice, $P'_{1000}P'_{0100}P'_{0010}$, containing P and the *axial* edges, namely, $P_{1000}P_{1001}$, $P_{0100}P_{0101}$, and $P_{0010}P_{0011}$, u is the linear parameter along one of the three axial edges; (α,β,γ) gives the barycentric coordinates of P in the slice.

tor $\hat{n}$ and its plane equation, $Ax+By+Cz+D_0=0$, where $\hat{n}=\{A,B,C\}^T$.

(b) At the apex P_{001}, $D_1=-(Ax_0+By_0+Cz_0)$.

(c) $u(x,y,z)=-\frac{Ax+By+Cz+D_0}{D_1-D_0}$.

2. Using u, we determine the quadrilateral slice $P'_{000}P'_{100}P'_{110}P'_{010}$ containing P, and represent P using bilinear interpolation on the slice using two-tuple (s,t), determined using the three equations implied by the three coordinates of a point, i.e., $P=(1-s)(1-t)P'_{000}+$ $s(1-t)P'_{100}+stP'_{110}+(1-s)tP'_{010}$.

4.1.2 Right Triangular Prisms

The interpolation function for a triangular prism is given by the following algebraic expression with real coefficients:
$F(x,y,z)=C_0+C_1x+C_2y+C_3z+C_4xz+C_5yz$

Any point in the cell can be represented parametrically by using four parameters: (a) a parameter u representing the position of a triangular slice containing the point, with respect to any of the three edges not belonging to the triangular bases (referred to as *axial* edges), and (b) three barycentric coordinates, (α,β,γ), of the point in the triangular slice, as shown in Figure 2(B). $(u=0)$ and $(u=1)$ represent the two triangular bases, respectively, and the three quadrilateral faces are represented by $(\alpha=1)$, $(\beta=1)$, and $(\gamma=1)$, respectively.

Let the cell contain vertices, P_{1000}, P_{0100}, P_{0010} on one triangular face and P_{1001}, P_{0101}, P_{0011} at the other end. The function values at $P_{\alpha\beta\gamma u}$, are given by $F_{\alpha\beta\gamma u}$ and coordinates are $(x_{\alpha\beta\gamma u},y_{\alpha\beta\gamma u},z_{\alpha\beta\gamma u})$. Every point $P(x,y,z)$ in space can be represented as $p(\alpha,\beta,\gamma,u)$ with respect to the cell. Suppose the triangular slice $P'_{1000}P'_{0100}P'_{0010}$ containing P has its vertices with parametric representation u with respect to the axial edges. The function value at P, $F(x,y,z)$ is interpolated using the formula:

$$\begin{aligned}F(x,y,z)=&f(\alpha,\beta,\gamma,u)\\=&(1-u)(F_{1000}\alpha+F_{0100}\beta+F_{0010}\gamma)\\&+u(F_{1001}\alpha+F_{0101}\beta+F_{0011}\gamma)\end{aligned}\quad(1)$$

For any point in the interior to or on the boundary of the cell, $0.0\le u,\alpha,\beta,\gamma\le 1.0$ and $\alpha+\beta+\gamma=1$. Due to linear dependence of α, β, and γ, we can reduce the parametric representation to a three-tuple, (α,β,u) to represent the set of linearly independent variables. However, to maintain the ease of representation, we continue to refer to the four-tuple parametric representation (α,β,γ,u), in the interpolation function.

To determine (α,β,γ,u) for a given point $P(x,y,z)$, we perform the following steps:

1. The parameter u defines the triangular slice $(P'_{1000}P'_{0100}P'_{0010})$ containing P.
$P'_{1000}=P_{1000}+u(P_{1001}-P_{1000})$
$P'_{0100}=P_{0100}+u(P_{0101}-P_{0100})$
$P'_{0010}=P_{0010}+u(P_{0011}-P_{0010})$
If the two triangular bases are parallel, we obtain u the same way as in the case of square pyramids. We determine the following:

(a) For the base $P_{1000}P_{0100}P_{0010}$, the normal vector $\hat{n}$ and its plane equation, $Ax+By+Cz+D_0=0$, where $\hat{n}=\{A,B,C\}^T$.

(b) For the opposite triangular face $P_{1001}P_{0101}P_{0011}$, the corresponding value of D_1 using one of the three vertices in the face as (x,y,z): $D_1=-(Ax+By+Cz)$.

(c) We determine the parameter $u(x,y,z)=-\frac{Ax+By+Cz+D_0}{D_1-D_0}$.

However, more generally, the value for u is the solution of the cubic equation in u given by:
$(P'_{1000}-P)\cdot((P'_{0100}-P)\times(P'_{0010}-P))=0$.
In the case of a right triangular prism, we get a unique real value for u, which satisfies the condition $0.0\le u\le 1.0$.

2. Using u, we determine the triangular slice $(P'_{1000}P'_{0100}P'_{0010})$ containing P, and determine the barycentric coordinates (α,β,γ) of the point in the triangle.

4.2 Look-up Tables

Just as in the marching cubes algorithm, we represent the cases of pentahedral cells using a bit-string

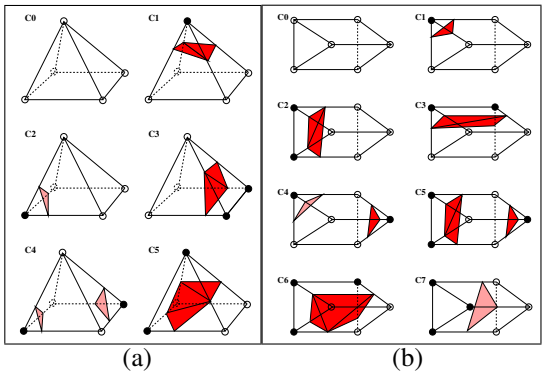


Figure 3: Base configurations of a pentahedral element: (a) Six of a square pyramid; (b) eight of a right triangular prism. Black points and circles indicate vertices whose function values are larger or smaller than the isosurface value, respectively.

Table 1: Classification of all cases into base configurations of a pentahedral element: six for square pyramid and eight for triangular prism cells.

Configuration	Pyramid Cases	Prism Cases
0	0, 31	0, 63
1	1, 30	1, 2, 4, 8, 16, 31, 32, 47, 55, 59, 61, 62
2	2, 4, 8, 15, 16, 23, 27, 29	3, 5, 6, 15, 23, 24, 39, 40, 48, 57, 58, 60
3	6, 7, 12, 13, 18, 19, 24, 25	9, 18, 27, 36, 45, 54
4	10, 11, 20, 21	10, 12, 17, 20, 29, 30, 33, 34, 43, 46, 51, 53
5	3, 5, 9, 14, 17, 22, 26, 28	14, 21, 28, 35, 42, 49
6	-	11, 13, 19, 22, 25, 26, 37, 38, 41, 44, 50, 52
7	-	7, 56

where each bit corresponds to a specific vertex of the cell. For a k-node cell, thus, there can be 2^k cases. However, these cases can be reduced to unique base configurations using mapping based on mirroring, rotation and complementation. Thus, 32 cases of square pyramid reduce to six unique configurations; and 64 cases of right triangular prisms reduce to eight. Figure 3 shows the unique base configurations of both cell types; and Table 1 classifies all cases according to their respective base configurations.

4.2.1 Resolving Ambiguities

Quadrilateral faces in the pentahedral cells can lead to ambiguities which are resolved using the asymptotic decider (Nielson and Hamann, 1991), as done in the marching cubes algorithm. As shown in Figure 4(A), configuration 4 of the square pyramidal cell contains a single ambiguous face, ABCD. Subconfigurations 4.0 and 4.1 occur when ABCD is "separated" and "connected", respectively. As shown in Figure 4(B), configurations 4 and 5 of the right triangular prismatic cell have been resolved. Configuration 4 contains a single ambiguous face, ABDC; while configuration 5 contains two, namely, ABDC and CDFE. Subconfigurations 4.0 and 4.1 occur when ABDC is separated and connected, respectively. Subconfigurations of 5 occur with possibilities of ABDC and CDFE being separated and connected. Thus, we have subconfigurations 5.0 and 5.3 with both faces being separated and connected, respectively. 5.1 and 5.2 are complementary: in 5.1, ABDC is separated and CDFE is connected; and in 5.2, vice versa.

In the prismatic cell, the configurations 5.1 and 5.2 require *tangent points* to obtain accurate topological representation of the isosurface, as the isosurface assumes the behavior of topological type A.2 (Lopes and Brodlie, 2003). Generally, for configuration 5, the tangent at the body saddle point of the isosurface will be parallel to the single non-ambiguous face, hence, the tangent point coincides with the body saddle point. The computation of body saddle points for right triangular prisms is discussed in the Appendix.

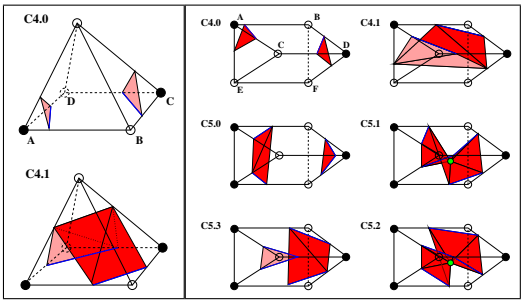
The surfaces computed from our interpolation models for all the configurations for both cell types are shown in Figure 5.

5 DIRECT ISOSURFACE EXTRACTION FROM HYBRID MESHES

As explained in Section 3, for direct isosurface extraction from hybrid grids, we use the respective look-up tables for each element type.

5.1 Gradient Approximation

Gradient interpolation is required for computing normal vectors which are used for lighting purposes during rendering the isosurface. We implemented per-

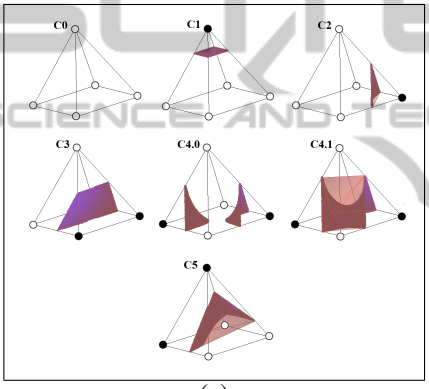


(a) (b)

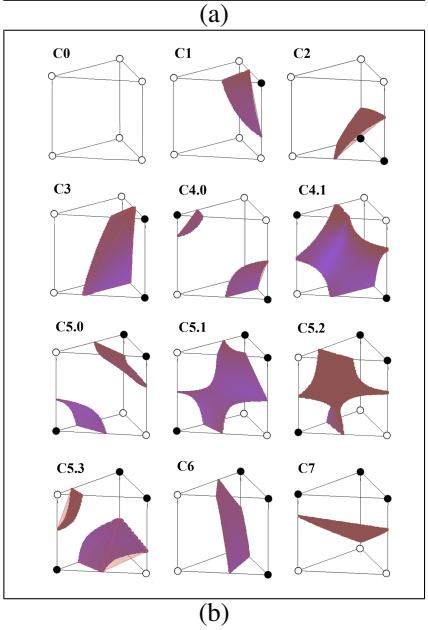
Figure 4: Subconfigurations of pentahedral cells resolving ambiguities: Blue lines help to show if the ambiguous face is connected or separated. (a) Square pyramidal cells: In 4.0 and 4.1, ABCD is separated and connected, respectively. (b) Right triangular prismatic cells: In 4.0 and 4.1, ABDC is separated and connected, respectively. In 5.0 and 5.3, ABDC and CDFE are both separated and connected, respectively; in 5.1 and 5.2, which are complementary, the isosurfaces require additional vertices, namely the body saddle points, indicated by green vertices.

vertex gradient approximation using a preprocessing step. We use a least squares procedure (Anderson and Bonhaus, 1994), which computes unweighted gradients in two-dimensional space by solving an over-determined system of equation, which we extended to the three-dimensional case using the equation: $f_i = f_0 + f_{x_0}(x_i - x_0) + f_{y_0}(y_i - y_0) + f_{z_0}(z_i - z_0)$, where f_i and f_0 are the values of the function f at points P_i and P_0, and P_i, for $i = 1, 2, \ldots, N$ have edges with P_0. The gradient at N_0 is $\vec{f}_0' = (f_{x_0}, f_{y_0}, f_{z_0})$. This leads to an $N \times 3$ system of equations:

$$\begin{bmatrix} \Delta x_1 & \Delta y_1 & \Delta z_1 \\ \Delta x_2 & \Delta y_2 & \Delta z_2 \\ \vdots & \vdots & \vdots \\ \Delta x_N & \Delta y_N & \Delta z_N \end{bmatrix} \begin{Bmatrix} f_{x_0} \\ f_{y_0} \\ f_{z_0} \end{Bmatrix} = \begin{bmatrix} f_1 - f_0 \\ f_2 - f_0 \\ \vdots \\ f_N - f_0 \end{bmatrix}$$



(a)



(b)

Figure 5: Isosurfaces, with topological considerations for: (a) all seven configurations of the square pyramidal cells; (b) all twelve configurations of the right triangular prismatic cells.

For solving this over-determined system of equations, we computed

$$r_{ab} = \sum_{i=1}^{N} (a_0 - a_i)(b_0 - b_i), \text{ where } a, b \in \{x, y, z\}$$

$$\begin{bmatrix} r_{xx} & r_{xy} & r_{xz} \\ r_{xy} & r_{yy} & r_{yz} \\ r_{xz} & r_{yz} & r_{zz} \end{bmatrix} \begin{Bmatrix} W_i^x \\ W_i^y \\ W_i^z \end{Bmatrix} = \begin{bmatrix} x_i - x_0 \\ y_i - y_0 \\ z_i - z_0 \end{bmatrix}$$

We solved for the weights W_i^x, W_i^y and W_i^z using Cramer's rule, and approximated the gradient using:

$$f_{a_0} = \sum_{i=1}^{N} W_i^a (f_i - f_0), \text{ where } a \in \{x, y, z\}.$$

6 RESULTS

We implemented our algorithm on a visualization cluster, Colt, at TACC, with the following specifications: 2 Intel Xeon quad core E5440 processors (8 cores total), 16 GB of RAM, nVidia Quadro FX5800 graphics card with 4 GB memory.

We tested our method for a synthetic dataset consisting of 1,331 nodes, containing all prisms; and decomposed it to a purely tetrahedral grid. We extracted isosurfaces from the original prism and the tetrahedralized grids, as shown in Figure 6. We used a blue-to-red colormap to map the quality measure of the triangles from 0 to 1, for which we used the incircle to circumcircle radius ratio (Pébay and Baker, 2003). The time taken to render the isosurfaces in the prism grid and tetrahedral grid are 0.01 and 0.03 seconds, respectively, for the value 0.305.

We applied our method to hybrid grids defining (a) a wind-tunnel model from NASA, shown in Figure 7, with 442,368 hexahedral, 721,413 tetrahedral, and 13,824 pyramidal elements; (b) a missile, shown in Figure 8, with 6,378,506 nodes, 2,479,668 tetrahedra, 17,691 pyramids and 4,207,433 prisms. The results of our direct isosurface extraction method are tabulated in Table 2. Our timing measurements for reading the file and preprocessing gradients did not show any improvement compared to our experiments using ParaView, and hence, we have not included them here.

7 CONCLUSIONS

Figures 6, 7 and 8 show that our method generated results comparable to those from standard methods. As expected, our method generated fewer triangles compared to applying the marching tetrahedra algorithm

Table 2: Results of our isosurface extraction algorithm: #Tri-tetra represents the number of triangles rendered using look-up table for tetrahedra, #Tri-pyram that for square pyramids, #Tri-prism that for triangular prisms, and #Tri-hexa that for hexahedra.

Dataset	Isosurface value	Extraction time (in s)	#Tri-tetra	#Tri-pyram	#Tri-prism	#Tri-hexa
Wind-tunnel	13.0464	0.95	29,241	586	0	30,226
dataset	12.1772	0.77	25,117	390	0	5,198
Missile	1.167	180.26	1,446,544	4,316	352,177	0
dataset	0.62	93.6	236,384	4,570	700,971	0

on a tetrahedralized grid owing to the lower number of elements. However, our method being a marching method does not always produce good quality triangles in the mesh, as can be seen in Figure 6. Owing to lower triangle count, our method performs better in regions with lower surface curvature and worse in regions with higher surface curvature, in comparison to the marching tetrahedra method on the tetrahedralized grid. Our method is comparably computationally efficient, especially for larger datasets. The gradient computation requires $O(n)$ time for n being the number of nodes in the dataset. The gradient estimation step needs to be performed just once as a preprocessing step, which makes real-time generation of smooth isosurfaces efficient. However, in case of the missile dataset, we did not see a significant improvement in performance owing to the fact that dataset consists of 85% tetrahedra.

Furthermore, we have covered all the ambiguous configurations possible that can occur for pentahedral elements, i.e., our look-up tables are complete. We have found that computation of body saddle points in a triangular prismatic element is similar to that of the hexahedron.

Our method can be further enhanced with the optimization strategies which are used in the marching cubes method.

ACKNOWLEDGEMENTS

We are grateful to: (a) Dr. David Marcum of Mississippi State University, and to Dr. Benjamin Kirk of NASA/JSC for sharing the missile and wind-tunnel datasets, respectively; (b) Dr. John W. Peterson of TACC for helping with using GMV and for providing experimental FEM datasets; (c) TeraGrid Resource Provider Grant as well as IIIT-Bangalore for support; (d) members of the Visualization Group in TACC and IIIT-Bangalore, especially, Prof. K. V. Dinesha, and Mr. Pratap J. Rao.

REFERENCES

Anderson, W. K. and Bonhaus, D. L. (1994). An Implicit Upwind Algorithm for Computing Turbulent Flows on Unstructured Grids. *Comput. Fluids*, 23:1–21.

Bhaniramka, P., Wenger, R., and Crawfis, R. (2004). Isosurface Construction in Any Dimension Using Convex Hulls. *IEEE Transactions on Visualization and Computer Graphics*, 10:130–141.

Carr, H., Moller, T., and Snoeyink, J. (2006). Artifacts Caused by Simplicial Subdivision. *IEEE Transactions on Visualization and Computer Graphics*, 12:231–242.

Gallagher, R. S. and Nagtegaal, J. C. (1989). An Efficient 3-D Visualization Technique for Finite Element Models and Other Coarse Volumes. *SIGGRAPH Comput. Graph.*, 23:185–194.

Lasser, D. (1985). Bernstein-Bézier Representation of Volumes. *Computer Aided Geometric Design*, 2(1-3):145–149.

Lopes, A. and Brodlie, K. (2003). Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, 9:16–29.

Lorensen, W. E. and Cline, H. E. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proceedings of ACM conference on Computer graphics and interactive techniques - SIGGRAPH 1987*, pages 163–169. ACM Press.

Montani, C., Scateni, R., and Scopigno, R. (1994). A Modified Look-up Table for Implicit Disambiguation of Marching Cubes. *The Visual Computer*, 10(6):353–355.

Moreland, K. (2011). *ParaView User's Guide v3.12*. Kitware Inc., Sandia National Laboratory, CSimSoft, http://www.itk.org/Wiki/ParaView/Users_Guide/Table_Of_Contents.

Muigg, P., Hadwiger, M., Doleisch, H., and Hauser, H. (2007). Scalable Hybrid Unstructured and Structured Grid Raycasting. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1592–1599.

Newman, T. S. and Yi, H. (2006). A Survey of the Marching Cubes Algorithm. *Computers & Graphics*, 30(5):854–879.

Nielson, G. M. (2003). On Marching Cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):283–297.

Nielson, G. M. and Hamann, B. (1991). The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes. In Nielson, G. M. and Rosenblum, L. J., editors, *Proceedings of IEEE Visualization 1991*, pages 83–91. IEEE Computer Society Press.

Ortega, F. A. (2008). Personal correspondence via e-mail with chief developer of GMV. http://www-xdiv.lanl.gov/XCM/gmv/GMVHome.html.

Ortega, F. A. (2011). *GMV 4.5 General Mesh Viewer User's Guide*. Los Alamos National Laboratory, CPFD Software, http://www.generalmeshviewer.com/.

Pébay, P. P. and Baker, T. J. (2003). Analysis of Triangle Quality Measures. *Mathematics of Computation*, 72:1817–1839.

Takahashi, T., Mekada, Y., Murase, H., and Yonekura, T. (2004). High Quality Isosurface Generation from Volumetric Data and Its Application to Visualization of Medical CT Data. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3 - Volume 03*, ICPR '04, pages 734–737, Washington, DC, USA. IEEE Computer Society.

Weber, G. H., Kreylos, O., Ligocki, T. J., Shalf, J., Hagen, H., Hamann, B., Joy, K. I., and Ma, K.-L. (2001). High-quality Volume Rendering of Adaptive Mesh Refinement Data. In *Proceedings of the Vision Modeling and Visualization Conference 2001*, VMV '01, pages 121–128. Aka GmbH.

Weber, G. H., Kreylos, O., Ligocki, T. J., Shalf, J. M., Hagen, H., Hamann, B., and Joy, K. (2003). *Extraction of Crack-Free Isosurfaces from Adaptive Mesh Refinement Data*, pages 19–40. Springer-Verlag, Heidelberg, Germany.

Williams, P. L., Max, N. L., and Stein, C. M. (1998). A High Accuracy Volume Renderer For Unstructured Data. *IEEE Transactions on Visualization and Computer Graphics*, 4:37–54.

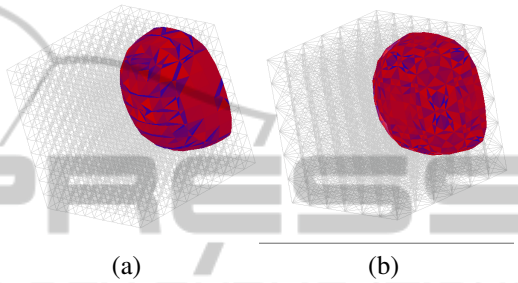
APPENDIX

Computation of Body Saddle Points in Triangular Prisms for Configurations 5

As explained in Section 4.1.2, the combined interpolation model uses three linearly independent variables, (α, β, u). The quadrilateral faces of the prism can be represented as either ($\alpha = 1$), ($\beta = 1$), or ($\gamma = 1$), respectively. In Figure 2(B), the face $P_{0100}P_{0101}P_{0011}P_{0010}$, corresponds to ($\alpha = 1$). Here, we describe the computations for the case where ($\alpha = 1$) is the non-ambiguous face, which can be appropriately extended to the cases when ($\beta = 1$) or ($\gamma = 1$) is the non-ambiguous case. The body saddle point occurs at a point where, for an isosurface of value I, the following conditions are satisfied:

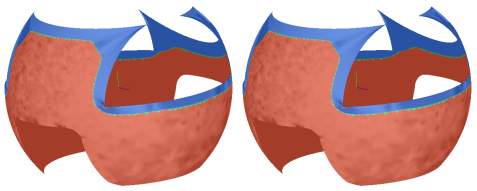
1. $F = F(x,y,z) - I = f(\alpha_\tau, \beta_\tau, \gamma_\tau, u_\tau) - I = 0$
2. $\frac{\partial F}{\partial u} = F_u = 0$
3. $\frac{\partial F}{\partial \alpha} = F_\alpha = 0$

We use the following notations for simplifying algebraic expressions:
$D_{(\alpha,0)} = F_{1000} - F_{0010}; D_{(\alpha,1)} = F_{1001} - F_{0011};$
$D_\alpha = D_{(\alpha,0)} - D_{(\alpha,1)}$
We use similar notations in the case of β: $D_{(\beta,0)}$,



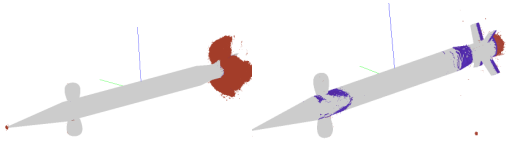
(a) (b)

Figure 6: Isosurface for same value extracted from a grid consisting of 1331 nodes, with: (a) 2000 triangular prismatic elements; and (b) 5220 tetrahedral cells. The isosurface consists of: (a) 478 triangles, and (b) 888 triangles. The color of triangles shows its quality mapped from worse to better, corresponding to blue-to-red colormap.



(a) (b)

Figure 7: Isosurfaces of function $\ln(c \parallel \vec{x} - \vec{x_0} \parallel_2)$, for constants c and $\vec{x_0}$, in hybrid grid of a wind-tunnel model for values: (a) 13.0464 and (b) 12.1772. The red surface is generated by using look-up table for tetrahedra, the blue surface by that for hexahedra and green surface by that for pyramids.



(a) (b)

Figure 8: Isosurface from the missile dataset for density in compressible flow for values (a) 1.167 and (b) 0.62. The blue surface is generated using our new look-up tables for prisms and pyramids, and the red one is generated by that for tetrahedra.

$D_{(\beta,1)}$, and D_β.

Differentiating F with respect to α,

$$F_\alpha = (1-u_\tau)D_{(\alpha,0)} + u_\tau D_{(\alpha,1)} \tag{2}$$

After applying the condition $F_\alpha = 0$ in Equation 2 and simplifying, one obtains:

$$\frac{1-u_\tau}{u_\tau} = \frac{D_{(\alpha,1)}}{D_{(\alpha,0)}} \quad \Rightarrow \quad u_\tau = \frac{D_{(\alpha,0)}}{D_\alpha} \tag{3}$$

On substituting for $\gamma_\tau = (1-\alpha_\tau-\beta_\tau)$ and u_τ from Equation 3, in Equation 1, the coefficient of α reduces to 0. Hence, on applying the condition $F = 0$, one obtains:

$$\beta_\tau = \frac{(D_\alpha \cdot I + F_{1001}\cdot F_{0010} - F_{1000}\cdot F_{0011})}{(D_{(\alpha,0)}\cdot D_{(\beta,1)} - D_{(\alpha,1)}\cdot D_{(\beta,0)})} \tag{4}$$

Differentiating F with respect to u,

$$\begin{aligned} F_u = &-(F_{1000}\alpha_\tau + F_{0100}\beta_\tau + F_{0010}\gamma_\tau) \\ &+ (F_{1001}\alpha_\tau + F_{0101}\beta_\tau + F_{0011}\gamma_\tau) \end{aligned} \tag{5}$$

After applying the condition $F_u = 0$ in Equation 5 and $\gamma_\tau = (1-\alpha_\tau-\beta_\tau)$; and simplifying using β_τ from Equation 4, one obtains:

$$\alpha_\tau = \frac{(F_{0011}-F_{0010})}{D_\alpha} - \frac{D_\beta}{D_\alpha}\cdot\beta_\tau \tag{6}$$

Using the interpolation model from Equation 1, the values for parameters from Equations 3, 4, and 6, and substituting $\gamma_\tau = (1-\alpha_\tau-\beta_\tau)$, the tangent point is computed as:

$$\begin{aligned} P_\tau(x,y,z) &= p(\alpha_\tau,\beta_\tau,\gamma_\tau,u_\tau) \\ &= (1-u_\tau)(\alpha_\tau P_{1000} + \beta_\tau P_{0100} + \gamma_\tau P_{0010}) \\ &\quad + u_\tau(\alpha_\tau P_{1001} + \beta_\tau P_{0101} + \gamma_\tau P_{0011}) \end{aligned}$$