Isosurface Extraction from Hybrid Unstructured Grids Containing Pentahedral Elements

Akshay Narayan¹, Jaya Sreevalsan-Nair¹, Kelly Gaither² and Bernd Hamann³

¹International Institute of Information Technology - Bangalore,
26/C, Electronics City, Hosa Road, Bangalore 560100, India
²Texas Advanced Computing Center, University of Texas at Austin,
10100 Burnet Road, Austin, TX 78758, U.S.A.
³Institute for Data Analysis and Visualization, Department of Computer Science,
University of California, Davis,
One Shields Avenue, Davis, CA95616, U.S.A.
akshay.narayan@iiitb.org, jnair@iiitb.ac.in, kelly@tacc.utexas.edu, hamann@cs.ucdavis.edu

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 an 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. For example, these grids are routinely used in computational fluid dynamics (CFD) and finite element analysis (FEA). Hybrid unstructured grids can have very complex geometrical structure, compared to, for example, rectilinear grids. Performing correct direct interpolation on such hybrid meshes, 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 meshes makes its analysis more challenging. Throughout this paper, we have used the terms “grids” and “meshes” interchangeably.

A great deal of progress has been made in recent years in the field of volume rendering using ray-casting. However, research related to simpler and more traditional visualization techniques like isosurface extraction from hybrid meshes has not been fully addressed for the entire spectrum of element types, which are commonly used, in addition to hexahedra and tetrahedra. Some of the commonly used elements are shown in Figure 1.

In our work, we have used the following interpolation models for elements of our interest:

- Tetrahedron: Linear interpolation, often using barycentric coordinates, is used. Using barycentric coordinates, α, β, γ, and δ, this model uses three linearly independent variables and a condition for the barycentric coordinates, i.e., α + β + γ + δ = 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 parameters 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 \( \alpha, \beta, \gamma \) are used as barycentric coordinates for the interpolation within a triangular slice considered along the axis. This model uses three linearly independent variables in total, after considering the condition that \( \alpha + \beta + \gamma = 1 \).

- **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 interval \([0,1]\).

The interpolation models explained above use three linearly independent variables. These models are generally used for these elements owing to their low computational complexity. Interpolation models for pyramidal and prismatic elements are discussed in detail in Section 4.1.

The primary reason for lack of efforts has been that although these grids are commonly used for modeling physics occurring in complex geometries, hybrid meshes consist of elements or cells other than tetrahedra and hexahedra, and the basis functions for such cells are no longer purely linear or trilinear. This leads to more complex interpolating functions. Recent increase in the availability and accessibility of powerful computational resources has increased use of hybrid meshes for several CFD and finite element (FEM) applications. The commonly used marching cubes or marching tetrahedra algorithm is then, respectively, implemented on the refined mesh to extract isosurfaces.

Existing isosurface extraction methods for hexahedra and tetrahedra work well on hybrid meshes, subject to converting a given hybrid grid first to a purely hexahedral grid or a purely tetrahedral grid, respectively. The preprocessing stage for converting the hybrid mesh entails either resampling the grid to a rectilinear grid, or decomposing the hybrid grid into tetrahedral elements. The commonly used marching cubes or marching tetrahedra algorithm is then, respectively, implemented on the refined mesh to extract isosurfaces.

The conversion of a given hybrid mesh to a mesh consisting only of hexahedra and tetrahedra introduces additional approximation error, if introducing new vertices is required. Since the conversion to a simpler grid type is not uniquely defined, it can lead to multiple analysis results, i.e., results that depend on the choice made among several combinatorially possible geometries for the conversion to another grid. This in turn can lead to multiple topological orientations for an isosurface.
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 mesh dataset, whose size is of the order of 1 GB. However, loading such a large dataset into system memory and generating isosurfaces requires approximately 10 minutes in the serial implementation of ParaView. ParaView’s parallel implementation shows 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 meshes. 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.

Our datasets consists of tetrahedral, pentahedral, and hexahedral elements. The contributions of our work are 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. The look-up tables enable us to obtain a unique solution for isosurface for a given function value. Specifically we have done the following:

1. We have implemented and tested new look-up tables for all possible cell configurations for direct isosurface extraction from the square pyramidal and triangular prismatic cells.

2. We have used our derived equations to compute the body saddle points for the triangular prismatic cell, which need to be included in certain configurations to complete the isosurface tessellation within the cell.

3. We have used a combination of look-up tables from the traditional marching tetrahedra and extended marching cubes, and our proposed ones 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), (Nielsen and Hamann, 1991), (Nielsen, 2003), (Lopes and Brodlie, 2003) are some of the representative papers. In particular, we point the readers to a very comprehensive survey of the marching cubes algorithm in (Newman and Yi, 2006).

Concerning isosurface extraction from hybrid unstructured grids, work has been done towards extending the marching cubes algorithm to hybrid meshes containing tetrahedral, prismatic and hexahedral elements in (Gallagher and Nagtegaal, 1989). Though our work is similar, we discuss the look-up tables more exhaustively, for both five- and six-node pentahedral elements. In a related paper (Takahashi et al., 2004), look-up table approach for octahedral elements has been discussed.

In the visualization software GMV, isosurface extraction from hybrid meshes 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 would obtain by 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). It uses the convex hull of an intersection set of vertices, $V_h^+$ and $U_h$, where $V_h^+$ are vertices of the hypercube with scalar values greater than the isosurface value and $U_h$ is the set of the intersection points of isosurface and the hypercube. It further removes $(k − 1)$ dimensional facets that lie on the faces of the hypercube from the convex hulls, and the remnant of the convex hulls are the isosurfaces. This algorithm can be extended to pentahedral cells, as it is applicable to all topological homotopes in the k-dimensional space. Tetrahedral, pentahedral, hexahedral and octahedral cells, which are the typical cell types used in hybrid meshes, are convex polyhedra with Euler characteristic $\chi = 2$, and hence are topological homotopes of a three-
dimensional hypercube. (Weber et al., 2003) described a crack-free isosurface extraction algorithm, specifically for meshes subjected to adaptive mesh refinement (AMR). This scheme uses “stitch cells” to bridge the gap between meshes of different levels of hierarchy. These stitch cells could be tetrahedra, pentahedra, or hexahedra. They have further suggested extending the marching cubes algorithm appropriately for each of the cell types and handling ambiguities, as has been done in the case of successors of the original marching cubes algorithm. We have used the interpolation functions that (Weber et al., 2001) use.

While methods by (Weber et al., 2003) and (Bhaniramka et al., 2004) work for our application, we are going a step further to resolve the topological configurations for pentahedral cells by using similar patterns that are found in hexahedra. Our work closely follows 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.

In the area of visualization of hybrid unstructured grids, a great deal of research has been done for both software- and hardware-based approaches in volume rendering. The direct volume rendering methods for unstructured grids have been done for both software- and hardware-based approaches in volume rendering. (Williams et al., 2007) 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 since it does not have to deal with any of the ambiguous cases which occur in the case of the marching cubes algorithm. However, 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 in the mesh into tetrahedral ones while ensuring continuity of isosurface across faces. A tetrahedral element inherently uses a linear interpolation function, therefore making its processing simple and straightforward. 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 mesh to a tetrahedral mesh. We discuss some of these differences in the following.

- A straightforward tetrahedralization can be done without inserting new vertices in the mesh. However, in certain cases, it is not possible without the introduction of additional vertices in the interior of a complicated element bounded by triangular and quadrilateral faces. Nevertheless, the first step in subdivision of cells to tetrahedra is subdivision of its quadrilateral faces to triangles. In a standard Lagrange finite element, a bilinear interpolation function is used in the quadrilateral face, and piecewise linear interpolation functions are used after subdivision of the face.

Considering the parametric representation of a function $F$ on a bilinear surface, at any point $P(x, y, z)$, one obtains:

$$F(x, y, z) = f(s, t) = (1 - s)(1 - t)F_{00} + s(1 - t)F_{10} + stF_{11} + (1 - s)tF_{01}$$

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 2(a); and $F_{ij}$ is the function value at $P_{ij}$ for $\{i, j\} = \{0, 1\}$. On the diagonals (where $s = t$ or $s + t = 1$), the interpolating function is quadratic in either $s$ or $t$. However, when we triangulate the quadrilateral and use one of the diagonals as an edge to compute points on the isosurface, we use a linear interpolation model. 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).

- Tetrahedralization of a hybrid mesh is an approach that leads to non-unique solutions. 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. The differences in the isolines generated for a quadrilateral face, depending on the choice of triangulation is shown in Figure 2.
Figure 2: 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, (b) and (c) show the two different isolines one obtains when considering the two possible triangulations of the quadrilateral. Note that the isolines in (b) and (c) are still topological homotopes. (a) also shows the parametric representation of the vertices in \((s, t)\) in a bilinear interpolation model.

- Though we can obtain a good approximation of the isosurface using the marching tetrahedra algorithm on a purely tetrahedral mesh, 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. In one of our datasets, the missile dataset, for each time-step we have around 24,800,000 tetrahedra, 17,700 pyramids and 4,207,000 prisms; which on tetrahedralization results in 37,456,400 tetrahedra, which is a 30% increase in the number of cells. The additional tetrahedra not only require extra computations to generate, but 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 minimal subdivision discussed there is equivalent to the decomposition we are discussing here, and the parity rule implies a rule that 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 the absence of indexed cells in a grid, one has to build the spatial neighborhood of each cell to ensure this parity rule. This is a huge computational overhead, in the case of large hybrid meshes and it negates the benefits of “embarrassingly parallel” nature of the marching algorithms.

For the missile dataset which consists of 85% tetrahedra, tetrahedralization of the grid does not lead to any significant benefits with respect to spatial or time complexity, or performance. Alternatively, we explore computing isosurfaces in the non-tetrahedral cells by implementing separate look-up tables for each of the unique geometric element types. In the following, we describe in detail how these look-up tables are constructed for five- and six-node pentahedral elements, and how they are used in a marching method to extract isosurfaces directly. We have found that our method, using our new look-up tables, shows comparable performance with respect to existing visualization tools.

4 PENTAHEDRAL CELLS

Pentahedral cells are frequently used in conformal grids in CFD and FEM applications, for “stitching” pure tetrahedral or hexahedral grids. The five-node (square pyramidal) and the six-node (triangular prismatic) pentahedral cells are very commonly used as “stitch cells” or filler cells in hybrid unstructured grids. The marching algorithms can be extended for the pentahedral cells by building appropriate look-up tables. Just as in the marching tetrahedra and the marching cubes algorithms, these look-up tables ensure piecewise-
continuous isosurfaces for pentahedral cells. In the marching algorithms, we use linear interpolation on the edges of the cells, and the isosurfaces on the faces of the cells reduce to isolines, thus ensuring piecewise continuous surfaces.

4.1 INTERPOLATING FUNCTIONS

The interpolation functions for a square pyramid and triangular prism depends on their respective orientation. For sake of simplicity while deriving the interpolation functions, 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.

4.1.1 Square Pyramids

The interpolation function for a square pyramid, with its z-axis defined by the direction implied by the base quadrilateral and the apex, is given by:

\[
F(x, y, z) = C_0 + C_1 x + C_2 y + C_3 xy + C_4 z
\]

for real values of \( C_i \), for \( 0 \leq i \leq 4 \), \( i \) being a non-negative integer.

There are several parametric representations possible. The simplest representation using three linearly independent variables is by using a parameter \( u \) representing position in the direction of z-axis, and two parameters \((s, t)\) for the parametric representation of a bilinearly interpolated surface in a quadrilateral slice at a particular height, as shown in Figure 3 (\( u = 0 \)) 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 \( P_{stu} \), for 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 \( P \), \( F(x, y, z) = f(s, t, u) \) is interpolated using:

\[
F_{001} = (1-s)(1-t)F_{000} + s(1-t)F_{100} + stF_{110} + (1-s)tF_{010}
\]

\[
F(x, y, z) = f(s, t, u) = uF_{001} + (1-u)F_{001}'
\]

where, \( u \) is the parameter along the z-axis, and \( s \) and \( t \) are parameters for the bilinear representation of the quadrilateral slice containing the point. For any point in the interior or on the boundary of the cell, \( 0.0 \leq s, t, u \leq 1.0 \). To determine \((s, t, u)\) for a given point, \( P(x, y, z) \), we perform the following steps:

1. 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. For a planar base, we perform the following steps to determine \( u \):
   (a) We determine the normal vector \( \hat{n} \) of the base \( P_{000}P_{100}P_{110}P_{010} \), and derive the plane equation of the base, \( Ax + By + Cz + D_0 = 0 \), where \( \hat{n} = \{A, B, C\}^T \).
   (b) At the apex \( P_{001} \), we determine \( D_1 = -(Ax_0 + By_0 + Cz_0) \).
   (c) We determine the parameter \( u(x, y, z) = -\frac{Ax + By + Cz + D_0}{A^2 + B^2 + C^2} \).

2. Further, we determine the quadrilateral slice \( P_{000}'P_{100}'P_{110}'P_{010}' \) containing \( P \), and we represent \( P \) using bilinear interpolation on the slice. The parameter two-tuple \((s, t)\), required for bilinear interpolation in a slice, is defined by the three equations implied by the three coordinates of a point, i.e.,

\[
P = (1-s)(1-t)P_{000} + s(1-t)P_{100}' + stP_{110}' + (1-s)tP_{010}'
\]
This interpolation function reduces to a bilinear function on the quadrilateral face, and a linear function on the triangular faces and the edges, thus ensuring \( C_0 \) continuity at the edges and faces of the cells.

### 4.1.2 Right Triangular Prisms

The interpolation function for a triangular prism, with its \( z \)-axis defined by its opposite triangular faces, is given by

\[
F(x, y, z) = C_0 + C_1 x + C_2 y + C_3 z + C_4 x y + C_5 x z + C_6 y z
\]

The cell can be represented parametrically by using a parameter \( u \) representing the position with respect to any of the three edges not belonging to the triangular bases (referred to as \textit{axial} edges), and three barycentric coordinates, \((\alpha, \beta, \gamma)\), in the triangular slice obtained from the points at the parameter \( u \) on the three axial edges, as shown in Figure 4 (\( u = 0 \)) 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, \( P_{0000}, P_{0100}, P_{0010} \) on one triangular face and \( P_{0101}, P_{0011}, P_{0011} \) 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. The function value at \( P \), \( F(x, y, z) = f(\alpha, \beta, \gamma, u) \) is interpolated using the formula:

\[
F(x, y, z) = f(\alpha, \beta, \gamma, u) = (1 - u)(F_{0100\alpha} + F_{0100\beta} + F_{0100\gamma}) + u(F_{0011\alpha} + F_{0101\beta} + F_{0011\gamma})
\]

where, \( u \) is the parameter along \textit{axial} edges and \( \alpha, \beta, \gamma \) are barycentric coordinates of the point in the triangular slice \( P'_{0100,0100,0010} \). For any point in the interior to or on the boundary of the cell, \( 0.0 \leq u, \alpha, \beta, \gamma \leq 1 \) and \( \alpha + \beta + \gamma = 1 \). Due to linear dependence of \( \alpha, \beta, \gamma \), we can reduce the parametric representation to a three-tuple, \((\alpha, \beta, u)\) to represent the set of linearly independent variables. However, to maintain the ease of representing the interpolation function, we continue to refer to the four-tuple parametric representation, \((\alpha, \beta, \gamma, u)\).

To determine \((\alpha, \beta, \gamma, u)\) for a given point \( P(x, y, z) \) we perform the following steps:

1. The parameter \( u \) defines the triangular slice \( (P'_{0100,0101,0010}) \) containing \( P \).
   \[
   P'_{0100} = P_{0100} + u(P_{0101} - P_{0100})
   \]
   \[
   P'_{0101} = P_{0100} + u(P_{0101} - P_{0100})
   \]
   \[
   P'_{0010} = P_{0010} + u(P_{0011} - P_{0010})
   \]

If the two triangular bases are parallel, we obtain \( u \) the same way as in the case of square pyramids.

(a) We determine the normal vector \( \hat{n} \) to the base \( P_{0100,0101,0010} \) and formulate the plane equation of the base, \( Ax + By + Cz + D_0 = 0 \), where \( \hat{n} = [A, B, C]^T \).

(b) For the opposite triangular face \( P_{0101,0100,0011} \), we determine the corresponding value of \( D_1 \) using one of the three vertices in the face: \( D_1 = - (Ax + By + Cz) \).

(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:

\[
(P'_{0100} - P) \cdot ((P'_{0101} - P) \times (P'_{0010} - P)) = 0
\]

In the case of a right triangular prism, we get a unique real value for \( u \), which satisfies the condition \( 0.0 \leq u \leq 1 \).

2. Further, we determine the triangular slice \( (P'_{0100,0101,0010}) \) containing \( P \), and determine the barycentric coordinates \((\alpha, \beta, \gamma)\) of
the point in the triangle, which is given by the ratios of areas subtended by the point to the vertices of the specific triangular slice, to the total area of the triangle.

This interpolation function reduces to a linear function at the triangular faces, and to a bilinear function at the quadrilateral faces and the edges. Thus, our interpolation model ensures that the resulting isosurface is continuous across elements.

4.2 LOOK-UP TABLES

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0, 31</td>
</tr>
<tr>
<td>1</td>
<td>1, 30</td>
</tr>
<tr>
<td>2</td>
<td>2, 4, 8, 15, 16, 23, 27, 29</td>
</tr>
<tr>
<td>3</td>
<td>6, 7, 12, 13, 18, 19, 24, 25</td>
</tr>
<tr>
<td>4</td>
<td>10, 11, 20, 21</td>
</tr>
<tr>
<td>5</td>
<td>3, 5, 9, 14, 17, 22, 26, 28</td>
</tr>
</tbody>
</table>

Table 1: Classification of all cases into six base configurations of a square pyramid element.

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

Table 2: Classification of all cases into eight base configurations of a right triangular prismatic element.

Just as in the marching cubes algorithm, we represent the cases of pentahedral cells using a bit-string where each bit corresponds to a specific vertex of the cell and the value of the bit is the boolean value indicating if the data value at the vertex is greater than or equal to the isosurface value. For a $k$-node cell, thus, there can be $2^k$ cases. However, these cases can be reduced to unique 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. Figures 5 and 6 show the
unique base configurations of both cell types. Tables 1 and 2 classify all cases according to their respective base configurations.

### 4.2.1 Resolving Ambiguities

Quadrilateral faces in the pentahedral cells can lead to ambiguities. Just as in the marching cubes algorithm, the ambiguities are resolved using the asymptotic decider [Nielsen and Hamann, 1991].

As shown in Figure 7, configuration 4 contains a single ambiguous face, ABDC. Subconfigurations 4.0 and 4.1 occur when ABCD is “separated” and “connected” [Nielsen and Hamann, 1991], respectively.
ibilities of ABDC and CDFE being separated and connected, respectively. 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.

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, which is discussed in (Lopes and Brodlie, 2003). The computation of tangent points for right triangular prisms is discussed in Appendix.

The surfaces computed from our interpolation models for all the configurations for both cell types are shown in Figures 9 and 10.

5 DIRECT ISOSURFACE EXTRACTION FROM HYBRID MESHES

As explained in Section 3 for direct isosurface extraction from hybrid meshes, we use the respective look-up tables for each element type, namely, tetrahedra, square pyramids, triangular prisms and hexahedra.

In addition to these cell types which are commonly used finite elements, there exist several other types of finite elements all having their own associated complicated interpolation functions. Nevertheless, those other types do not occur in our specific applications. Often, these other types of cells, which might be bounded by an arbitrary number of $n$-gons, require a combination of appropriate linear interpolation functions.

5.1 Gradient Approximation

Gradient interpolation is required for computing normal vectors which are used for lighting purposes during rendering the isosurface. We implemented gradient approximation using a preprocessing step implemented at all the grid nodes. 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 equations. We extend the algorithm in (Anderson and Bonhaus, 1994) 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 $f_0' = (f_{x_0}, f_{y_0}, f_{z_0})$. This leads to an $N \times 3$ system of equations:

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

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

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

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

We solve for $W^x$, $W^y$ and $W^z$ using Cramer’s rule, and approximate the gradient using these weights:

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

$$\vec{f}'_0 = (f_{x_0}, f_{y_0}, f_{z_0})^T$$

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 1,315 nodes, containing all prisms. For comparison purposes, we further tessellated this dataset into a purely tetrahedral mesh. We extracted isosurfaces from the original prism dataset and the purely tetrahedral dataset, and provide our comparative results in Figure 11. The time taken to render the isosurfaces in the prism grid and tetrahedral grid are 0.01s and 0.03s seconds, respectively, for an isosurface value of 0.2019.

We applied our method to hybrid meshes defining (a) a wind-tunnel model from NASA, shown in Figure 12, and (b) a missile, shown in Figure 13. The wind-tunnel dataset consists of 442,368 hexahedral, 721,413 tetrahedral, and
13,824 pyramidal elements. The missile dataset consists of 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 3. 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

Our implementations in Figures 11-13 show that our method generated results comparable to those from standard methods. Our method generated fewer triangles compared to applying the marching tetrahedra algorithm on a tessellated grid. This is mainly due to the fact that the number of elements (tetrahedral, pentahedral, and others) being handled is smaller than in the case of a hybrid mesh that has been tetrahedralized.

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. However, the gradient estimation step needs to be performed just once as a preprocessing step, which makes real-time generation of smooth isosurfaces efficient. Furthermore, we have covered all the ambiguous configurations possible that can occur for pentahedral elements, i.e., our look-up tables are complete. Computing 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. Specific methods which are not influenced by the shape of the elements can be used directly without any modifications, as we use the same algorithm with the exception of different look-up tables.

ACKNOWLEDGEMENTS

We are grateful to Dr. David Marcum of Mississippi State University, and to Dr. Benjamin Kirk of NASA/JSC for sharing the missile and wind-tunnel datasets, respectively. We are also grateful to Dr. John W. Peterson of TACC for helping with using GMV and for providing experimental FEM datasets. This work has been supported by a TeraGrid Resource Provider Grant as well as IIT-Bangalore. We would like to thank all members of the Visualization Group in TACC and IIIT-Bangalore for supporting our work. Our special gratitude to Prof. K. V. Dinesha and Pratap J. Rao for helping with discussions and resources.

REFERENCES


### Table 3: Results of our isosurface extraction algorithms from (top) the wind-tunnel dataset and (bottom) the missile dataset. #Tri-tetra represents the number of triangles rendered using look-up table for tetrahedra, #Tri-pyram represents the number of triangles for square pyramids, #Tri-prism represents the number of triangles for triangular prisms, and #Tri-hexa represents the number of triangle for hexahedra.

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


### APPENDIX

**Computation of tangent points in triangular prisms for configurations 5.1 and 5.2:**

As explained in Section 4.1.2, the combined interpolation model of linear (along axis) and linear (in the triangular slice) uses three linearly independent variables, namely, $\alpha$, $\beta$, and $u$. Since barycentric coordinates are used for the triangular bases as well as for intermediate slices, the quadrilateral faces of the prism can be represented as either $(\alpha = 1)$, $(\beta = 1)$, or $(\gamma = 1)$, respectively. For e.g., $(\alpha = 1)$ refers to the face $F_{0100} F_{0101} F_{0011} F_{0010}$, as shown in Figure [4].
Differentiating $F$ with respect to $u$,
\[ F_u = -\left( F_{1000}\alpha + F_{0100}\beta + F_{0010}(1 - \alpha - \beta) \right) + \left( F_{1001}\alpha + F_{0101}\beta + F_{0011}(1 - \alpha - \beta) \right) \]
\[ \text{(4)} \]

After applying the condition $F_u = 0$ on Equation 4 and simplifying, one obtains:
\[ D_\alpha \alpha + D_\beta \beta = F_{0011} - F_{0010} \]
\[ \text{(5)} \]

Substituting for $\beta_\tau$ from Equation 3 in Equation 5 one obtains:
\[ \alpha_\tau = \frac{(F_{0011} - F_{0010})}{D_\alpha} - \frac{D_\beta}{D_\alpha} \left( (D_\alpha \cdot I + F_{1001} \cdot F_{0010} - F_{1000} \cdot F_{0011}) \right) \]
\[ \text{(6)} \]

Differentiating $F$ with respect to $\alpha$,
\[ F_\alpha = (1 - u_\tau)(F_{1000} - F_{0010}) + u_\tau(F_{1001} - F_{0011}) = (1 - u_\tau)D_{(\alpha,0)} + u_\tau D_{(\alpha,1)} \]
\[ \text{(7)} \]

After applying the condition $F_\alpha = 0$ on Equation 7 and simplifying, one obtains:
\[ \frac{1 - u_\tau}{u_\tau} = \frac{D_{(\alpha,1)}}{D_{(\alpha,0)}} \]
\[ \text{(8)} \]

Using the interpolation model from Equation 2 the values for parameters from Equations 3, 6 and 8 and substituting $\gamma_\tau = (1 - \alpha_\tau - \beta_\tau)$, the tangent point is computed as:
\[ P_\tau(x,y,z) = p(\alpha_\tau, \beta_\tau, \gamma_\tau, u_\tau) \]
\[ = (1 - u_\tau)(\alpha_\tau P_{1000} + \beta_\tau P_{0100} + \gamma_\tau P_{0010}) + u_\tau(\alpha_\tau P_{1001} + \beta_\tau P_{0101} + \gamma_\tau P_{0011}) \]
\[ \text{(9)} \]

Similar computations can be implemented for configurations 5.1 and 5.2 of the triangular prismatic element, in the case of ($\beta = 1$) and ($\gamma = 1$) being the non-ambiguous face.