Visualization of High Resolution, Three-Dimensional, Nonlinear Finite Element Analyses *

Mark A. Christon, Thomas Spelce
Methods Development Group, L-122
Lawrence Livermore National Laboratory, P.O. Box 808
Livermore, California 94550

Abstract
A two-pass surface extraction algorithm for adaptive finite element meshes is presented in the context of a visualization study for a particle impact and a turbine blade containment problem. The direct use of finite element data structures for the computation of external surfaces, surface normals, and derived physical quantities is discussed. An overview of the in-betweening which accounts for rigid body dynamics effects is presented with a brief discussion of a “direct-to-videodisk” animation strategy.

1 Introduction
Scientific Visualization is the process whereby numerical data is transformed into a visual form to augment the process of discovery and understanding [1]. Visualizing the data generated by finite element analyses presents many challenges due to the geometric complexity and the presence of multiple materials and element types. The data produced by a finite element analysis code can consist of time varying geometry, temperature, stress, strain and possibly mesh topology. Further, the mesh topology may be heterogeneous in two aspects: first, the topology may encompass beam, shell, and solid elements (1-D, 2-D, and 3-D elements), and second, the topology is generally unstructured. The finite element model may include special features such as sliding interfaces or material boundaries which must be tracked with time. Additionally, treating both Lagrangian and Eulerian grid based data in a single graphics application is complicated by the volume of data produced when high resolution meshes are employed for analyses with many time steps. This is especially true considering the fact that the data is usually of a vector or tensor form (e.g., stress tensor).

This paper focuses upon the application of surface rendering techniques for finite element data primarily from large scale Lagrangian simulations of the transient dynamic response of solids and structures. One of the key components to visualizing this type of data is the ability to rapidly perform surface extraction while maintaining both topological adaptivity in the finite element mesh and volumetric information. Further, it is necessary to perform some type of data abstraction thereby reducing the order of the data from a second order tensor to a physically meaningful scalar field. Other key aspects of the visualization application include the ability to selectively display material components of the mesh, provide palette manipulation, map derived physical quantities into surface color, and perform in-betweening for animations. This must be done without inhibiting the ability to move randomly through large datasets viewing individual states in simulation time. The codes used to generate data for the visualization application discussed herein are DYN3D [2] and NIKE3D [3] developed by the Methods Development Group at Lawrence Livermore National Laboratory (LLNL).

This paper presents an algorithm for surface extraction which provides the ability to incrementally adapt to topological changes during the finite element simulation. The algorithm is presented in the context of visualizing several large scale finite element analyses which involve both rigid body dynamic motion and localized plastic deformation. In the demonstration calculations, elements are eliminated from the simulation based upon a maximum strain failure criterion making it necessary to handle the arbitrary adaptive topology of the mesh. The adaptive aspect of the meshes considered makes it impossible to adopt the well known practice of generating a single list of external surface polygons thereby discarding all interior volumetric information. Instead, all volumetric data is preserved in a way which enables the continuous re-definition of external surfaces accounting for the topological evolution. The process of data abstraction and in-betweening for this application are briefly presented, with results in the form of a video tape, and still frames for the simulation of the impact of a high velocity particle on idealized turbine stator blades, and failure containment in a turbine engine shroud.

2 Surface Extraction with Volumetric Adaptivity
In the finite element method, the physical domain of interest is approximated by a finite number of smaller sub-domains called elements. The elements

---

*Work performed under the auspices of the U. S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48.

U.S. Government Work Not Protected by U.S. Copyright
typically used consist of eight node hexahedra (bricks), four node quadrilaterals, and two node beams. For the purposes of surface extraction and rendering, both beams and quadrilateral elements may be ignored for obvious reasons. However, the brick elements require special attention because of the need to adapt incrementally and then render only external surfaces with the associated nodal point data. As mentioned above, the solid and shell elements in DYNASD can be subjected to a strain failure criteria whereby the mesh topology will change with time. This requires the ability to preserve volumetric data since permitting incremental surface extraction both during the interactive interrogation process for data exploration and the in-betweening process for animation.

Unlike finite difference models which rely on logically regular grids, finite element models are topology based. Figure 1 illustrates the three element types and their associated local canonical topological node numbering schemes. Here, the local nodal numbering simply implies the use of pointers for indirect addressing using the global node numbers. Thus, the global node numbering for the mesh provides a means for relating nodes to elements. The nodal connectivity data for a given element may be represented by $C_i$ where $i = 1$ to $N_{ele}$ and $N_{ele}$ is the number of nodes per element. Typically, finite element models group like elements together to simplify the connectivity (e.g., the connectivity is blocked into solid, shell and beam elements). In the case of the solid elements, $C_i$ may be considered to be made up of six reduced sets of connectivity data which associates local nodal numbers with the faces of the solid elements. The reduced connectivity is referred to as $S_i$ where $i = 1$ to $N_{Hex}$ and $N_{Hex}$ is the number of nodes per surface element. The nodal connectivity for $S_i$ for the canonical local node numbering scheme in Figure 1 is shown in Table 1.

<table>
<thead>
<tr>
<th>Face No.</th>
<th>Node-1</th>
<th>Node-2</th>
<th>Node-3</th>
<th>Node-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$n_1$</td>
<td>$n_2$</td>
<td>$n_6$</td>
<td>$n_5$</td>
</tr>
<tr>
<td>2</td>
<td>$n_2$</td>
<td>$n_3$</td>
<td>$n_7$</td>
<td>$n_6$</td>
</tr>
<tr>
<td>3</td>
<td>$n_4$</td>
<td>$n_8$</td>
<td>$n_3$</td>
<td>$n_7$</td>
</tr>
<tr>
<td>4</td>
<td>$n_1$</td>
<td>$n_5$</td>
<td>$n_6$</td>
<td>$n_4$</td>
</tr>
<tr>
<td>5</td>
<td>$n_1$</td>
<td>$n_4$</td>
<td>$n_3$</td>
<td>$n_2$</td>
</tr>
<tr>
<td>6</td>
<td>$n_5$</td>
<td>$n_6$</td>
<td>$n_7$</td>
<td>$n_8$</td>
</tr>
</tbody>
</table>

$S_i$ is a vector of unit normals, the (i) superscript indicates the current element number, $N_{Hex}$ is the number of solid elements, and $j$ ranges over the number of nodes per surface of the solid element and is contained in $S_i$. In Eq. 1, it is understood that the normal vector for each nodal point must be computed for each face to which it belongs in the solid element. After the global assembly has been performed, all nodal points which are interior to the body have a global unit normal whose components are zero. While providing a robust surface extraction mechanism, the ASN algorithm fails to retain both volumetric and topological information which is central to any adaptive scheme. Further, this approach proves far too computationally intensive requiring $24N_{Hex}$ unit normal computations plus the global assembly operations for a given topology.

Another frequently used algorithm is the polygon-search/cancel (PSC) algorithm [5] which relies upon the nodal connectivity and avoids some of the problems associated with ASN. One difficulty with PSC is that it relies on a linear search which results in $O(6N_{Hex})^2$ operations. This can be improved by using a hash table lookup, but requires chaining to resolve collisions in the hash table. Neither the ASN nor the PSC algorithm inherently deals with volumetric adaptivity which is an important form of adaptivity found in finite element applications (albeit, only one form of adaptivity). It is from this point which we begin our description of the two pass surface extraction algorithm implemented herein. The first pass of the algorithm is only performed once because the resulting data structures contain all the necessary information for volumetric adaptivity.

Whereas the ASN and PSC algorithms essentially discard all volumetric data, it is convenient and computationally efficient to retain volumetric information for the adaptive nature of the problems considered herein. It is possible to apply the ASN or PSC algorithm after each adaptive step to extract the current external surface, but this approach involves considerable overhead which can be avoided. By viewing the internal surfaces as paired between adjacent elements as in the ASN algorithm (Figure 2), it is possible to combine features of the ASN and PSC algorithms to efficiently construct the tables necessary to perform an incremental surface extraction while maintaining volumetric adaptivity. Note that the incorporation of adaptivity for beam and shell elements is essentially automatic given the fact that the analysis code produces the necessary data for the determination of element activity at each time step in the simulation.

In finite element node numbering there can be no redundant nodes (i.e., two node numbers with identical spatial coordinates) except where there are special
Figure 1: Canonical Local Node Numbering in Computational Space for Brick, Shell, and Beam Elements.

Figure 2: Internal Paired Surfaces between Solid Elements.
model features such as sliding interfaces. Therefore, the key to incremental surface extraction with volumetric adaptivity is the identification of paired internal surfaces. The identification is considered to be the first pass in the two pass extraction algorithm discussed below. We define a hash function which will be used in a table of length $6N_{tet}$ for the solid elements, i.e., the worst case possible is that all faces of the brick elements are exterior faces (a real possibility in the most general case).

$$H = \sum_{i=1}^{N_{tet}} w_i \bar{S}_i \quad (2)$$

Here $\bar{S}_i$ represents the surface connectivity in a sorted ascending order, and $w_i$ is a nodal weight. The nodal numbering scheme in the finite element method ensures that there are no repeated nodal numbers, that is, $n_i \in [1, N_{np}]$ where $N_{np}$ is the total number of nodal points in the mesh. Because of the locally ordered nodal connectivity used in Eq. 2, it is possible to define the nodal weighting as: $w_i = 10^j - 1$. This choice in weights helps to maintain a relatively high degree of dispersion in the hash keys, but by no means ensures that each hash value will be unique [6]. However, the goal here is not to use the hash key for direct addressing, but to identify paired surfaces in the volume. Therefore, the hash keys are entered into the hash table in sequential order as they are computed.

As the hash values are computed and entered in the hash table, the associated element number and face number are also entered in the table. An indexed heap sort [7], (an $O(N \ln(N))$ algorithm), is used to generate a list of hash values in ascending order. Traversing this list of hash values, it is possible to identify any paired surfaces in the volume because such surfaces will yield identical hash function values which are ultimately adjacent to each other in the ordered hash list. During the traversal of the hash list, a master surface list which contains a count of surfaces which may be paired (i.e., surfaces which are back-to-back) for each hash value is generated to enable the detection of invalid pairing in the ordered hash list. A maximum of two element and face numbers are included in the master surface list for later use in the second-pass volume adaptive surface extraction.

In the second pass of the surface extraction algorithm, the master surface list is used to generate a polygon vertex pointer list which includes the global node numbers for the polygon vertices, and an element material number. At this point, the active shell and beam vertices and materials are concatenated to the polygon vertex pointer list. Initially, only unpaired surfaces from the elements are added to the master surface list.

As finite elements in the simulation fail (according to the maximum strain at failure criteria) inactive elements are removed by setting an activity flag to zero. This activity flag is available in the graphics database, and it is used to remove inactive surfaces from the polygon pointer list, as well as, to split paired surfaces where failed elements have been removed. Figure 3 illustrates the elimination of a single element revealing the previously hidden interior surfaces. The element number which has been identified for failure is used to locate and decrement the face count in the master surface list. Any faces in the master surface list with a face count of 1 are considered to be valid surfaces for display. The canonical local node numbering scheme for the solid elements ensures that exposed polygon normals face outwards in the animation scene at all times.

Vertex normals for the active polygons are computed in a two-step process where the first step involves the computation of vertex normals and the assembly of these normals into a global normal vector as:

$$\mathbf{e}_n = \sum_{i=1}^{N_e} \sum_{j=1}^{N_{a}} \mathbf{e}_{n_j}^{(i)} \quad (3)$$

where $N_e$ is the number of active surfaces for rendering. During the assembly of the global normals, a count of polygons contributing to each nodal normal is accumulated. This allows an averaged normal at each polygon vertex to be computed. To account for "sharp" corners, a simple heuristic based upon the dot product of a vertex normal and its corresponding global normal is employed to decide when to use the averaged normal, and when to use the local normal. This provides a fast mechanism for the detection of edges which exploits the existing finite element data structures already in place.

3 Data Abstraction

The implementation of the finite element visualization code is based upon Silicon Graphics’ GL library. The surface rendering techniques allow the specification of user defined material properties such as specular, diffusivity, and color, as well as the color and position of up to eight light sources in a simplified scene description. Both vector (wire frame) and solid images (i.e., surface rendering) can be generated interactively, with field variables such as time dependent effective stress or plastic strain optionally mapped onto the surface of the time varying geometry. Multiple selectable color palettes are available for mapping the variation of field variables to surface color.

Mapping field variables onto surface color implies that a representative physically meaningful scalar field is available. For the analysis codes considered here, a stress tensor ($\sigma_{ij}$) and plastic strain ($\epsilon_p$) are readily available. While the plastic strain is a scalar valued function, the stress is a symmetric tensor meaning that there are six unique values of stress available at each node in the mesh, or alternatively at each polygon vertex. The stress tensor may be reduced to an effective stress which is a scalar valued function defined as:

$$\sigma_{eff} = \left( \frac{3 \sigma_{ij} \sigma_{ij}}{2} \right)^{\frac{1}{2}} \quad (4)$$

where

$$\sigma_{ij} = \sigma_{ij} - \frac{1}{3} \delta_{ij} \sigma_{kk} \quad (5)$$
In Eq. 4 and Eq. 5, the indicial notation implies a summation on repeated indices, and $\delta_{ij}$ denotes the Kronecker delta [8].

To perform a table lookup or function evaluation for color specification, the scalar valued function must be appropriately scaled. Here, the scalar valued lookup function, $\phi$, is computed as:

$$
\phi = \begin{cases} 
\frac{\sigma_{eff} - \min(\sigma_{eff})}{\max(\sigma_{eff}) - \min(\sigma_{eff})} & \text{Effective Stress} \\
\frac{\varepsilon_{eff} - \min(\varepsilon_{eff})}{\max(\varepsilon_{eff}) - \min(\varepsilon_{eff})} & \text{Eff. Plastic Strain}
\end{cases}
$$

(6)

In Eq. 6, the min/max values are global minimums and maximums over the entire simulation. The scalar valued function, $\phi$, is used as the entry into either a color lookup table, or is passed to a color map function evaluation routine. The individual vertex colors are set based upon a user specified field cutoff parameter. Vertex colors inherit the specified material colors everywhere the field value ($\phi$) is below the cutoff.

4 In-Betweening/Animation Strategy

The type of animation technique used for the finite element data considered herein is referred to as key-frame because it relies on “key frames” of data computed at discrete times levels by the analysis codes. In order to produce enough animation frames for video (30 frames per second of play time), it is necessary to interpolate between the key-frames from the analysis code. This process is often referred to as “in-betweening”. This section briefly discusses the type of interpolation used for animating the key-frame data, and the use of a “direct-to-videodisk” animation strategy.

For Lagrangian finite element calculations, the presence of rigid body dynamics can cause difficulties in the interpolation process. When rigid body dynamics are not important, a linear in-betweening is used where the time rate of change of spatial coordinates and field variables is computed based on key-frame data. As is typical with grid based explicit transient solutions to systems of partial differential equations, the physical time step size varies during the simulation due to stability constraints.

Inactive solid and shell elements are removed from the list of candidate elements for rendering when the interpolated time spans a simulated time level accounting for the topological adaptivity in the simulation. When rigid body dynamics is important (e.g., flight of an object before impact), it is necessary to modify the in-betweening algorithm. In this case, user input is required to identify objects by material in the mesh for material specific conditional interpolation. The conditional interpolation must be activated and de-activated at the appropriate times in the animation to capture the correct impact event adequately. If this approach is not taken, artifacts can be produced when key-frames span an important event such as the impact of two bodies. In the case of the particle - turbine stator blade simulation, the plastic deformation of the turbine blades and the associated plastic strain field would appear prematurely in Figures 5 and 6 if the conditional interpolation were not applied correctly.

Typical “disk based” animation systems require the animator to render and store images on rotating disk, only to later re-display, scan convert and record the images on video tape or videodisk. The scan conver-
sion step is also referred to as digital encoding, and is the process whereby the RGB color signals are converted to composite signals such as NTSC (National Television Standards Committee) [9].

At best, disk based animations are batch processes, often requiring batch rendering, and consuming large amounts of disk resources. The approach adopted here was to avoid the use of rotating disk for animation frames by recording the frames directly to a laser videodisk. The desired interactivity of the visualization application dictated the use of a hardware rendering platform, and the ability to perform scan conversion of images to a format appropriate for recording (e.g., Beta, NTSC, PAL, etc.) "on the fly".

The Silicon Graphics Video Framer provides a secondary low-resolution frame buffer for use during digital encoding and recording. Additionally, the Video Framer provides a V-LAN transmitter enabling the code for control of the video transport device to be essentially independent of the actual video transport device. For example, the same coded instructions can be used with a frame accurate video tape deck instead of a laser videodisk. The trade-off with the "direct-to-disk" animation system is that the hardware rendering process can produce somewhat lower image quality in comparison to other rendering techniques (e.g., ray tracing). However, it offers the ability for the analyst or researcher to immediately preview dynamic processes both interactively on the workstation, and via playback from the videodisk.

The "direct-to-disk" approach enables the user to preview an animation, then directly record the animation from a window on the workstation to the videodisk. When compared to disk based animation, this approach reduces the total time required to produce an animation segment, and reduces the impact on computing resources. Key frames may be previewed individually, and optionally written to a 24-bit HDF file [10] for later use such as generation of 35mm slides. Alternatively, key frames may be viewed in an NTSC mode in order to allow the user to compensate for color changes and image degradation characteristic of NTSC video.

5 Results and Conclusions

The results presented in this section are in the form of a visualization study performed on several "proof-of-concept" problems using the newly implemented SAND (Slide-surfaces with Adaptive New Definitions) technology in DYNA3D. DYNA3D is a nonlinear, time explicit, 3-D finite element code for solid and structural mechanics. SAND technology extends the capability of DYNA3D to include penetration problems not typically analyzed with a Lagrangian code. The images in Figures 4, 5, 6 and 7, 8, 9 illustrate the SAND technology as applied to a particle - turbine stator blade impact, and a turbine blade fragment containment problem. Each simulation required approximately 10,000 solid elements (O(10,000) nodes). The simulated times are on the order of milliseconds for both problems. In each analysis, plastic strain was chosen as the field variable to be mapped onto the geometry.

(See color plates, p. CP-35.)
In the images where effective plastic strain is mapped onto the geometry, red indicates the maximum while blue signifies the minimum value of the plastic strain. The first simulation demonstrates a particle traveling at high velocity and impacting idealized turbine stator blades. The sequence of images in Figures 4, 5 and 6 clearly illustrates the local, short time hydrodynamic response near the particle coupled with the global, long time structural response of the blades. A symmetry plane is used to simplify the computation, and to illustrate the variation of plastic strain through the volume of the plates along the symmetry plane.

The second simulation shown in Figures 7, 8 and 9 demonstrates a turbine blade fragment striking a containment ring. Elapsed physical time in the simulation is approximately 1 millisecond. At the midpoint in the simulation, the blade fragment has not yet pierced the containment ring, but significant localized plastic deformation is apparent. At the end of the simulation, the turbine blade tip has nearly passed completely through the containment ring. A "shear plug" has been punched from the containment ring and is falling just below the turbine blade tip. The extreme plastic deformation (shown as red) remains highly localized in the containment ring.

The successful application of the two-pass surface extraction algorithm to this class of problems has been demonstrated for a simplified rendering/animation application. This algorithm permits the complex topology of finite element meshes with arbitrary geometries to be handled efficiently without discarding volumetric data which is crucial to adaptive problems. Further, the implementation of the "direct-to-disk" animation process has proved invaluable in placing the interrogation and animation tool in the hands of the analysts performing the finite element simulations. This tool enables analysts to gain new insight into the voluminous data generated by large scale simulations which ultimately increases the value of such simulations.

Acknowledgements
The SAND technology was developed and implemented in DYNA3D by Robert G. Whirley and Bruce Englemann of the Methods Development Group at LLNL.

References


