

# Efficient Visualization of Crash-Worthiness Simulations

Sven Kuschfeldt and Michael Holzner\*  
Bayerische Motoren Werke AG  
Car Body Design and Engineering

Ove Sommer and Thomas Ertl†  
University of Erlangen  
Computer Graphics Group

## Abstract

Finite element post-processing has been dominated by software that is tightly integrated with simulation packages. Many of these packages have not kept up with the state-of-the-art developments in graphics technology and visualization techniques. Especially the large and time-dependent data sets resulting from crash-worthiness simulations in the automotive development process demand for new visualization tools which allow interactive manipulation of complex geometries and meaningful mapping of physical properties. This article demonstrates that careful design of scene graph structures and extensive use of texture mapping can improve rendering performance and visual appearance for post-processing tasks such as inspecting finite element discretization and analyzing intrusion depth or vector quantities. Furthermore, a new iconic visualization method is introduced which improves the understanding of cross-section forces and bending moments in longitudinal structures of the car body.

## Visualization in crash simulation

One of the main goals in the development of a new car is the achievement of an optimal "crash-worthiness" using as many analytical tools as possible and minimizing hardware-prototype testing. During the last few years, the absolute simulation time for modeling, computing and investigating a complete crash model has been reduced significantly. However, we notice a shift of the proportions between the time required for pre-processing, computation and post-processing respectively. The post-processing stage turned out to become the most time consuming activity performed by the simulation engineers. These changes and the rapid development of computer graphics technology during the last few years has increased the need for new visualization techniques to facilitate the analysis of crash-worthiness simulations.

Considering the progress of scientific visualization in various areas during the last decade, it becomes obvious that the application of 3D visualization techniques to finite element analysis has not been a primary focus [4, 7, 12]. Nevertheless, the use of commercial visualization packages is now well established in the automotive industry. In the case of crash analysis, these traditionally employed post-processors have been designed to manage the enormous amount of simulation data on workstations with limited memory by performing animations of wire-frame meshes and polygonal representations of the simulated crash models. However, associated with these design criteria and with wide platform availability is a trade-off which leads to poor graphics performance in terms of available frame rates on high-end graphics subsystems.

The deficit of many commercial post-processing tools in taking full advantage of the potential of modern graphics workstation was

the starting point for a joint research between the computer graphics group of the University of Erlangen, Germany, and the BMW AG in Munich. This article describes some of the results which we achieved when applying state-of-the-art rendering techniques like scene graph design and extensive use of textures as well as iconic techniques from scientific visualization to time-dependent finite element data sets from structural mechanics.

## Effective scene graph design

Several graphics APIs, such as IRIS Performer or OpenGL Optimizer, have been developed to take advantage of recent progress in compute server and workstation architecture with multiprocessing hardware in mind. Since those APIs are usually scene graph based, we can take advantage of model optimization during scene graph creation and benefit from multiprocessing using frustum culling and occlusion culling while traversing the scene graph to increase frame and interaction rates. Since the time-dependent databases of our FE models are very bulky, an efficient scene graph design is very important in order to handle the complex data interdependencies and to achieve high rendering speed.

Our goal is to visualize meshes of about 250,000 finite elements with nearly the same number of nodes for each one of 60 time steps. Additionally, we have to represent the connectivity of the finite elements. Storing both coordinates and connectivity for each time step would be a waste of memory resources, since the element topology does not change during the crash. Therefore a much better approach is to use an indexed geometry.

In OpenInventor, a widely used object-oriented 3D graphics toolkit, we can store the coordinates of each time step under a `stateSwitch` node and we can place the time-invariant description of the connectivity to the right side of that node once (see top of Figure 1). For each frame the scene graph is traversed by a render action object which holds the traversal state. One member of the traversal state is the actual set of coordinates which are defined by one of the coordinate, nodes and which will be referenced by the `indexedShape` nodes. This approach appears to be a very memory efficient representation of our data in a scene graph structure, but it would not allow local scene graph optimizations or multiprocessing of independent subgraphs. This is because objects on the right hand side of the scene graph may depend on settings of the traversal state which have been made by scene graph nodes on the left hand side.

Hence, we decided to use SGI's Cosmo3D which forms the underlying 3D toolkit for OpenGL Optimizer, an API for large-model visualization supporting features such as multiprocessing, occlusion culling, and accelerated hardware-assisted scene manipulation. Cosmo3D provides a scene graph structure which resembles the semantics of the Virtual Reality Modeling Language [1]. It basically differs from that of the OpenInventor scene graph. There is no information inherited horizontally in the Cosmo3D scene graph which is traversed just downward from top to bottom in each branch. Thus, we have to choose a different scene graph structure (see bottom of Figure 1) which reduces redundant data storage as much as possible by taking advantage of indexed geometries and by shared instancing of scene graph nodes.

\* BMW AG, Entwicklung Karosserie, 80788 München, Germany, email: Sven.Kuschfeldt@bmw.de

† Universität Erlangen, IMMD 9, Am Weichselgarten 9, 91058 Erlangen, Germany, email: Ove.Sommer@informatik.uni-erlangen.de

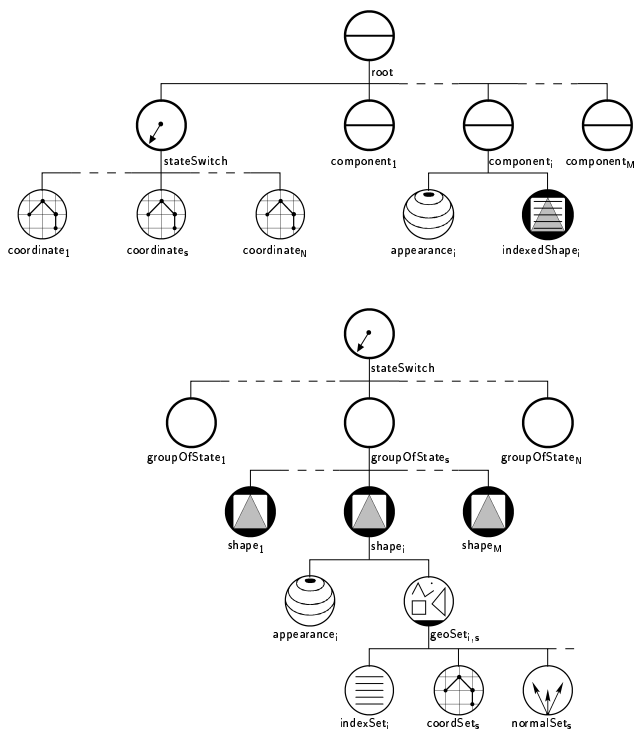


Figure 1: OpenInventor (top) and Cosmo3D (bottom) scene graph

That means we can assign a coordSet node containing all coordinates of time step  $s$  to several geoSet nodes. Each of these geoSet nodes only stores a reference to the data which is resident in main memory just once. Therefore, appearance $_i$  and indexSet $_i$ , which represent one and the same part of a car across all time steps, can also be shared by the shape $_i$  subgraphs. In analogy each geoSet $_{i,s}$  in the subgraph of groupOfState $_s$  has a reference to one and the same coordSet $_i$ . If we want to provide normals for each vertex we can expand indexSet $_i$  and coordSet $_i$  and use the index array also to refer to a normal array held in normalSet $_i$ .

Based on this scene graph design we are now able to visualize an entire crash data set on a modern desktop multiprocessor graphics workstation at interactive frame rates.

## Efficient visualization of physical and structural properties using texture mapping

Using traditional visualization systems, physical properties like plastic strains on FE surfaces are visualized through color coding of the polygons representing the surface elements. Surface areas with nearly equal physical values within predefined ranges are visualized with iso-contouring and color bands. Usually, the intersection points between the contour edges and the polygons representing the elements have to be calculated and additional polygons with different colors have to be created on both sides of the contour line.

In our approach, the physical values serve as entries into a one dimensional texture. Texture mapping is a well established and widely used technique in computer graphics (see the survey by Heckbert [6]) and in scientific visualization [3, 2]. The color of the object, onto which the texture is applied, is modified at each pixel by a corresponding color from the texture image. Hardware

support for texture mapping is now widely available from high-end graphics workstations of various vendors down to PCs.

A texture can be thought of not only as an image, but also as a lookup table [5]. We use a color table texture and map the physical values into floats between 0 and 1 serving as coordinates of the one-dimensional texture which are assigned to the vertices. Utilizing a texture with fewer colors and sharp borders between the colors we create iso-contours on the textured model automatically without any calculation of intersection points and without rendering additional polygons resulting in lower calculation and rendering costs as compared to traditional visualization packages.

Similarly, we use a 1-D texture in order to analyze the intrusion of components into the passenger cell in the case of a frontal or side impact collision. Iso-contours show where in the deformed structure the intrusion of the passenger cell is unacceptable and how far it is away from the acceptable limit.

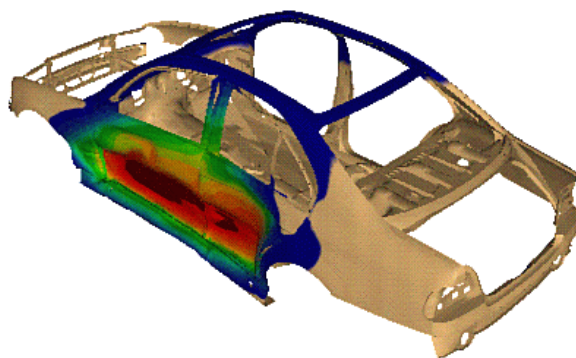


Figure 2: Intrusion of the car body during a side impact collision. By using 1D-texture mapping we can determine without additional rendering cost, whether the acceptable quantitative intrusion limit is exceeded in some areas.

We determine the nature and the magnitude of the intrusion by relating the nodes of the FE structure to a reference plane for each time step of the crash simulation. We define the reference plane within an appropriately chosen coordinate system so that it moves with the car body during the simulated crash. The differences between the distance of the undeformed structure and the distance of the deformed structure from the plane are calculated and scaled to values between 0 and 1, with respect to a predefined range of interest. The scaled values serve as texture coordinates of the vertices of the structure.

If the acceptable limit is changed or the intrusion has to be investigated using a broader or tighter range of interest, no additional computational effort is necessary, only the texture definition has to be adapted. Iso-contour visualization of the intrusion of the car body during a side impact collision is shown in Figure 2. Rendering of a model of 12888 elements on a SGI O<sup>2</sup> can be performed with a frame rate of 6.2 frames per second, in comparison to 1.8 frames per second using the traditional visualization method. A SGI Indigo<sup>2</sup> with Maximum Impact Graphics displays a model of 104768 elements with 2.4 frames per second which corresponds to a speed-up by a factor of 3.

In many situations the model discretization of the FE structure has to be displayed. However, since the quadrilateral FE elements are subdivided into triangles during rendering, the grid is visualized combining wire-frame and shaded polygon representations. Using traditional post-processors, the polygonal model is rendered in a first step, and the lines representing the element borders are drawn in an additional step, resulting in rendering the geometry twice.

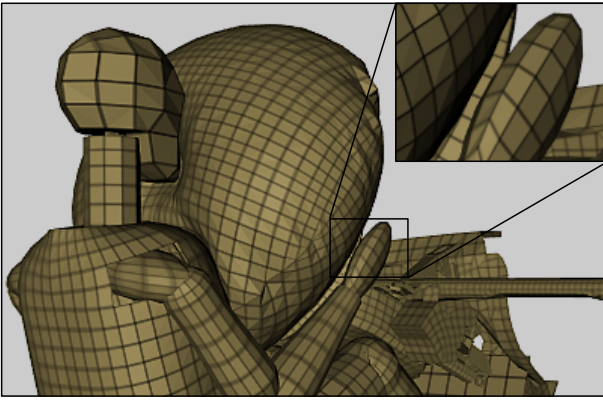


Figure 3: Visualizing FE model discretization using a 2-D texture.

Besides the described property mappings, textures can also be used to improve the understanding of the shape of complex structures [10, 9]. In our case we visualize the grid of the finite element models by mapping a texture, which paints borders onto each element of the FE model. The main goal is to eliminate the rendering cost induced by the additional drawing of wire-frame lines. We employ a two-dimensional texture, which is represented by a white image with a black border. Figure 3 shows the FE mesh of a dummy model visualized with wire-frame mapping.

In the case of crash simulations, usually 90 per cent of the finite elements are four-sided, 10 per cent of the elements are three-sided. The coordinates of the corners of the texture image are assigned to the corresponding vertices of the polygons to be rendered. If the element is three sided, an additional vertex with the same spatial coordinates and the same normal as the third vertex is created. The fourth texture coordinate of the image is assigned to this new vertex. Since an efficient visualization requires the creation of triangle strips from the polygonal model, common vertices of adjacent polygons must have the same texture coordinates. Therefore, the texture coordinates are mirrored along the shared edges of adjacent elements.

Using the same models as in the iso-contouring example we achieved an increase in rendering speed from 4.0 fps to 5.7 fps on the  $O^2$  (small model) and from 1.6 fps to 2.4 fps on the Indigo<sup>2</sup>, because no additional line drawing is necessary.

## Visualization of vector data using animated textures

Traditional post-processors visualize vector data like node velocities with thin and opaque lines and arrow heads. Since one car component usually comprises thousands of nodes and finite elements, the same large number of vector arrows is drawn, covering each other and the underlying structure. This makes the analysis of vector data difficult in many cases. Our goal is the visualization of the vector data in a way that leaves the underlying structure mostly visible. We followed an idea of Yamrom, who visualized flow vector fields using animated textures [11], and adapted and extended this method for the visualization of nodal velocities of structural dynamic non-linear FE models.

In contrast to the traditional way we use lines without arrows, but with segments of changing opacity, which move in the direction of the vector. We achieve this motion by switching 6 different textures with 16 texels each at each vector line.

In the first texture, we start with two semi-transparent texels fol-

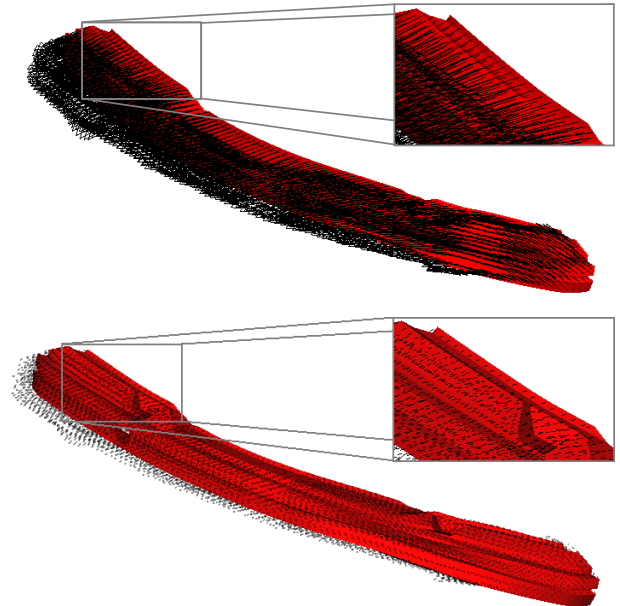


Figure 4: The upper image shows the visualization of the node acceleration vectors of a FE model of the front bumper structure using a traditional post-processing system. The lines and arrows hide parts of the structure. The same bumper structure can be recognized much better in the lower image, where animated opacity changing textures are used.

lowed by four totally transparent texels, again followed by two texels with opacity values greater than the values of the first two texels and so on. The six textures differ in the position of these "opacity fields" within the texture. By switching the textures the fields move with growing opacity towards the top of the vectors allowing us to recognize the direction of the nodal accelerations as well as the structure behind the vectors.

In Figure 4 we show the front bumper structure with additional nodal acceleration vectors. The upper image is visualized using a traditional post-processor, the lower image employs our texture animation technique, which shows a snapshot of the animated vectors revealing the structure behind.

## Force flux visualization with force tubes

During a car collision, each component of the car body is stressed in a different manner. Some parts absorb very high forces, other parts transfer the forces to the passenger cell. The determination of the structural components, which guide the main forces, enables the engineer to design car components with an optimal crash behavior. Since the longitudinal structures within the front part of a car body play an important role for increasing the ability of the body to absorb forces in a frontal crash, it is necessary to detect and to understand the force progression within these components.

In order to calculate the forces that act inside a car component, section force calculations are performed. In existing post-processors, first a section plane must be defined and positioned within the component. Next, the section force is calculated. Finally, a diagram is displayed showing the section force as a sum of the forces of the elements influenced by the section plane at this position of the longitudinal structure and its time progression during the crash. For the investigation of the whole component many

different sections have to be positioned within the component and a large number of diagrams have to be investigated in a very abstract and time consuming task.

Our new approach for the visualization of the force flux is to position an additional tubular element next to the deforming longitudinal structure, whose radius variation visually relates to the local force. The section forces are displayed just like water flow in a flexible tube. Certain parts of the tube are expanding, when the longitudinal force through the corresponding part increases, whereas other parts of the tube are constricting, showing a decrease of force in the structure. Using the tubing method, we can analyze the behavior of longitudinal structures by investigating their deformation and simultaneously their ability to absorb forces.

The section planes for the computation of the section forces are positioned automatically perpendicular to the structure along a trace line, that follows the shape of the longitudinal. During the visualization, we investigate those elements that are intersected by a plane and calculate the forces that affect the element nodes lying on the normal side of the plane. For each section, the forces are accumulated and the vector component of the accumulated force vector that is parallel to the plane's normal is computed.

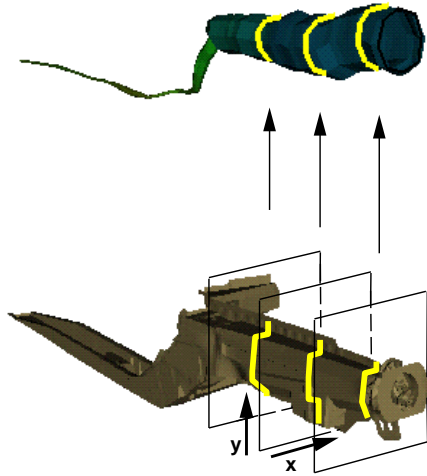


Figure 5: Force tube generation using section force values.

We can position the tube next to the longitudinal structure with a reasonable spacing and parallel to a line through the centers of gravity of the structure. Several rings are positioned around this tubular midpoint-line. Each ring represents one section force. The position of a ring in the tube corresponds to the position of the section in the structure while the diameter is dynamically related to the value of the scaled section force. A number of points are created around the circle of each ring serving as vertices for polygons that connect the rings and form the tube. The creation of a force tube is outlined in Figure 5.

## Visualization of bending moments

The analysis of bending moments in longitudinal structures is very important, because these bending moments can cause high torsion stresses in components, which are connected with the longitudinals. In the following paragraphs, we describe a new method for the visualization of such bending moments based on the force tube approach.

We calculate the bending moments through the longitudinal structures – similar to the section forces – from the nodal forces lying on the normal side of the section planes. For each section, we calculate two bending moments in relation to two different moment axes (defining the local x-axis and the local y-axis of the section plane). Each axis intersects the center of gravity of the longitudinal structure within the section. We compute the two bending moments by accumulating the products of the nodal forces and the lever arms defined by the distances between the nodes and the respective moment axis. Each accumulated moment carries a sign defining a bend that is caused by either a positive rotation or a negative rotation around the moment axis.

Similarly to the force tube, a moment tube is created based on the calculated bending moments per section. Each ring of the tube is cut resulting a half tube which shows the sign of the bending moment in that section.

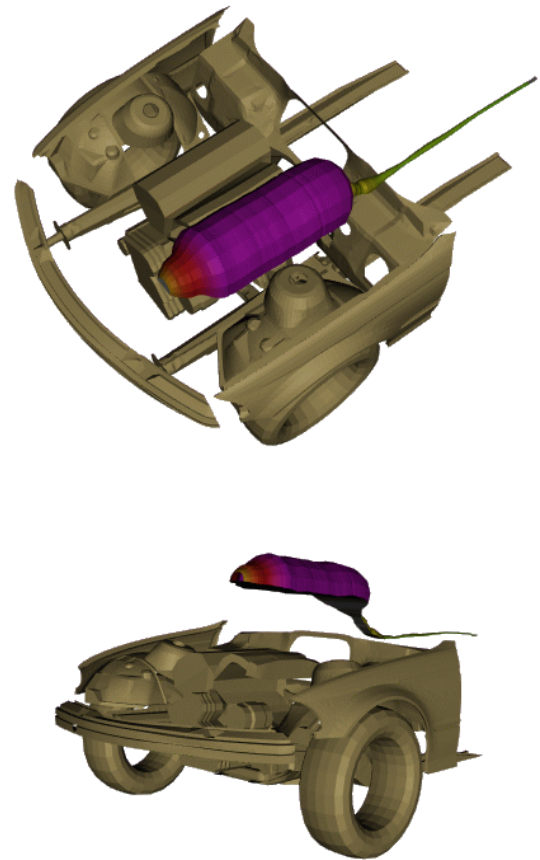


Figure 6: Moment tube over the left longitudinal structure, displaying bending moments that affect the front part of the longitudinal at the time of 12 ms after the crash. After cutting the tube the upper half tube remains and shows a positive bending of the structure "towards the top", whereas the rear part of the longitudinal is not yet affected by bending stresses.

Based on this visualization we can derive information about the moment progression as well as both the magnitude and the sign of the bending moments that affect the longitudinal structure. Figure 6 shows the bending moments in the left longitudinal structure.

## Conclusions

The use of efficient visualization techniques is very important for an effective simulation of the crash-worthiness of a vehicle. Since large amounts of data result from the simulation, we investigated efficient data handling and scene graph design in order to manage both memory requirements as well as high frame and interaction rates.

Furthermore, the presented examples show that the use of texture mapping is a powerful method to improve the quality and the ease of the visual analysis of crash simulations. Using efficient mapping techniques which require a minimum of memory, we ensure that our applications can be run by a great number of engineers working on mid-range workstations with limited texture memory.

Combining these applications with our new techniques for the visualization of force fluxes through a car body structure and bending moments in longitudinal structures we achieve a significant improvement over conventional post-processing tools, because the investigation of these stresses can now be performed more intuitively and much faster than before. Together with polygon reduction algorithms the presented visualization methods are sufficiently interactive on even huge FE models, thus forming a basis for virtual reality applications [8].

## References

- [1] ISO/IEC 14772-1:1997. The virtual reality modeling language. <http://www.vrml.org/Specifications/VRML97/>, 1997.
- [2] B. Cabral, N. Cam, and J. Foran. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In A. Kaufman and W. Krüger, editors, *1994 Symposium on Volume Visualization*, pages 91–98. ACM SIGGRAPH, 1994.
- [3] R. A. Crawfis and N. Max. Texture Splats for 3D Scalar and Vector Field Visualization. In G. M. Nielson and Bergeron D., editors, *Visualization 93*, pages 261–265. IEEE Computer Society, 1993.
- [4] R. Gallagher, R. Haber, G. Ferguson, D. Parker, W. Stillman, and J. Winget. Applying 3D Visualization Techniques to Finite Element Analysis. In *IEEE Visualization 1991*, pages 330–335. IEEE Computer Society Press, 1991.
- [5] P. Haeberli and M. Segal. Texture Mapping as a Fundamental Drawing Primitive. In M. Cohen, C. Puech, and F. Sillion, editors, *Proc. of the Fourth Eurographics Workshop on Rendering, Paris, France, June, 1993*, pages 259–266. Springer, Vienna, New York, 1993.
- [6] P. Heckbert. Survey of Texture Mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.
- [7] G. Kerlick and E. Kirby. Towards Interactive Steering, Visualization and Animation of Unsteady Finite Element Simulations. In *IEEE Visualization 1993*, pages 374–377, 1993.
- [8] S. Kuschfeldt, M. Schulz, T. Ertl, T. Reuding, and M. Holzner. The Use of a Virtual Environment for FE Analysis of Vehicle Crash Worthiness. In *IEEE Virtual Reality Annual International Symposium 1997*, page 209, 10662 Los Vaqueros Circle, P.O.Box 3014, Los Alamitos, CA 90720-1264, March 1997. IEEE Computer Society Press, Los Alamitos, California. ISBN 0-8186-7843-7.
- [9] W. Lorenzen. Geometric Clipping Using Boolean Textures. In *IEEE Visualization 1993*, pages 268–274, 1993.
- [10] P. Rheingans. Opacity-modulating Triangular Textures for Irregular Surfaces. In *IEEE Visualization 96*, pages 219–225, 1996.
- [11] B. Yamrom and K. Martin. Vector Field Animation with Texture Maps. *IEEE Computer Graphics and Applications*, 15(2):22–24, March 1995.
- [12] W. Ye and J. Vance. Visualization of Structural Impact Problems in a Virtual Environment. In A. Tentner, editor, *High Performance Computing 1997*, pages 325–330, P. O. Box 17900, San Diego, CA 92177, U.S.A., April 1997. Society for Computer Simulation International. ISBN: 1-56555-122-2.



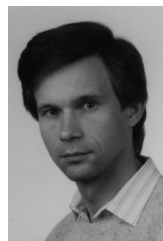
*Sven Kuschfeldt is a simulation engineer in the Car Body Design and Engineering Department at BMW. His research interests include numerical simulations and scientific visualization. He received his MS in mechanical engineering in 1994 from the University of Rostock and finished his PhD thesis in 1998 at the University of Erlangen.*



*Ove Sommer is a PhD student at the computer graphics group of the University of Erlangen, Germany. His research interests include scientific visualization. He received his MS in computer science in 1997 from the University of Erlangen.*



*Thomas Ertl is a professor of computer graphics and visualization in the computer science department of the University of Erlangen where he leads the scientific visualization group. His research interests include volume rendering, flow visualization, multi-resolution analysis, parallel and hardware accelerated graphics, large datasets and interactive steering. He received a MS in computer science from the University of Colorado at Boulder, and a PhD in theoretical astrophysics from the University of Tübingen, Germany.*



*Michael Holzner is head of the crash simulation group in the Car Body Design and Engineering Department at BMW. His research interests include numerical simulations and scientific visualization. He received his MS in mechanical engineering and his PhD from the Technical University of Munich, Germany.*