

# Crashing in Cyberspace - Evaluating Structural Behaviour of Car Bodies in a Virtual Environment

Martin Schulz <sup>†‡</sup>, Thomas Reuding <sup>‡</sup>, Thomas Ertl <sup>†</sup>

<sup>†</sup> University of Erlangen, IMMD IX, Computer Graphics Group  
Am Weichselgarten 9, D-91058 Erlangen, Germany

<sup>‡</sup>BMW AG, Entwicklung Karosserie,  
80788 Munich, Germany

Email: schulz@informatik.uni-erlangen.de, thomas.reuding@bmw.de

## Abstract

*The use of virtual prototypes generated from engineering simulations can be crucial to the efficient development of innovative products. Performance predictions and functional evaluations of a design are possible long before results of real prototype tests are available. With the rise in model complexity, data quantity, computing performance and accuracy, we increasingly find ourselves lacking the tools, methods and metaphors to deal with the information that is being generated. Here we present new results of on-going research at the University of Erlangen and BMW in the development of a virtual environment for car-body engineering applications as illustrated by examples from acoustics, vibration and impact dynamics.*

## 1 Introduction

Numerical analysis in car-body engineering extends to a variety of different fields such as structural mechanics, aerodynamics, acoustics and climatization, to name just a few.

In each of these fields, the task at hand is twofold: first to predict the physical behaviour of a complex technical system both as a whole or as individual components; second to use the analysis results to effectively introduce the necessary engineering changes to compensate for detected design shortcomings.

Previous research has shown that a virtual environment can offer a wide variety of analytical postprocessing tools. For example, the *Virtual Windtunnel* project ([BF95], [BL92] and [Bry94]) is one of the first applications based on virtual

reality techniques that clearly shows the advantages of these techniques over traditional finite element analysis postprocessing methods. The work of Ye [YV97] and Yeh [YhV97] also uses a virtual environment for the visualization of finite element models and further illustrates the usefulness of this technique.

In this paper we present new results of on-going research at the University of Erlangen and BMW in the development of a virtual environment for car-body engineering applications. In an early prototype of our system *VtCrash* we focused on the analysis of results from a finite element solver and performed a time-dependent real-time visualization of the crash performance of a vehicle (Figure 1). This proved the value of providing novel computer-human interface techniques for intuitive and interactive analysis of crash-test simulation data [KSR<sup>+</sup>96, Sch97]. However, several weaknesses were identified: deficiencies in interactive performance, lack of versatility in the configuration of the virtual environment and limited extensibility to new capabilities.

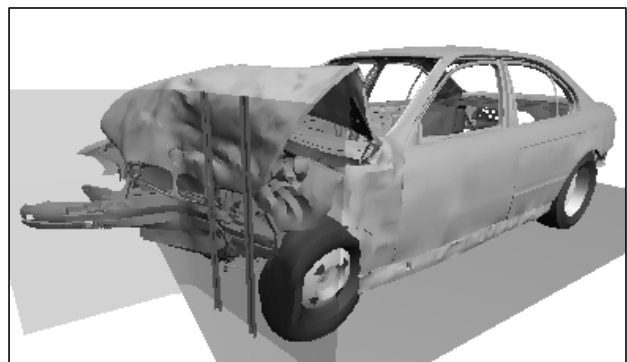


Figure 1: Visualization of a simulation of a frontal offset crash

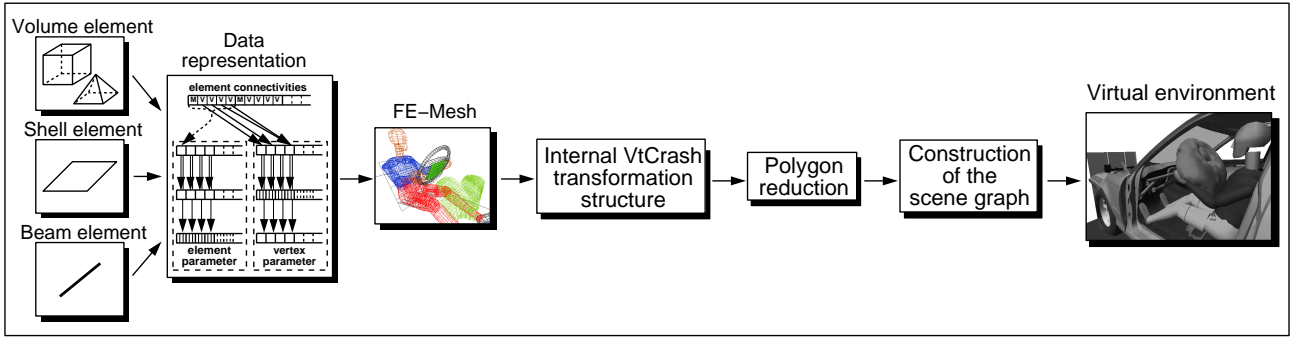


Figure 2: Transfer of the finite element model into the virtual environment

The current version of *VtCrash* is written in C++ and is implemented on *Silicon Graphics* workstations (*Octane* and *Onyx*). It supports a variety of interface hardware, interactive functions and addresses new fields of application. We will demonstrate its improved capabilities for interactive “what-if” studies - giving multidisciplinary engineering teams an intuitive sense of the performance of a design as well as facilitating the assessment of alternatives and helping to achieve optimal solutions to conceptual design problems. Examples from the areas of noise, vibration and impact dynamics will be discussed.

## 2 Motivation for a Virtual Environment

The concept phase of product development in automotive design is characterized by the need to evaluate complex engineering scenarios under conditions in which the relevant information is either only partially available or the correctness of underlying assumptions regarding the viability of certain technical solutions is not yet proven.

Numerical simulation has always played a key role in this context. Performance predictions and functional evaluations are possible long before results of real prototype tests are available. However, with the rise in model complexity, data quantity, computing performance and accuracy we increasingly find ourselves lacking the tools, methods and metaphors to deal with the information that is being generated. This is supported by the observation that currently about 30 percent of the efforts involved in a typical simulation go into the preprocessing phase and about

10 percent is taken up by the actual computation, while approximately 60 percent goes into the analysis and communication of the results. Clearly there is a strong incentive to reduce the last percentage through the implementation of insightful, intuitive visualization tools which allow effective communication between engineers.

The study of realistically simulated scenarios in structural and fluid mechanics involves very large, transient data sets. In the past these have proved too large for the available hardware capabilities to display in real-time. However, due to advances in hardware technologies and the efforts of a number of researchers - Nishioka et al. [NN95], Renze et al. [RO96], Schroeder et al. [SZL92] - in the area of data reduction techniques, the visualization of these data sets is now possible.

## 3 System Design Issues

The finite element models consist of various types of elements (solids, shells and beams), which are collected into groups to define the vehicle’s components. All of the data is time-dependent and comprises, for each timestep of the simulation, nodes in global coordinates and elements which reference the groups they belong to as well as their constituent nodes.

*VtCrash* is of an object-oriented design with the data structured into a class hierarchy (Car, Group, Poly, Vertex) which is partly derived from the element structure of the finite element models themselves. Furthermore, it employs data sorting methods to generate new local polygon lists and creates a data structure suitable for the animation of all timesteps (Figure 2).

### 3.1 Polygon Generation and Reduction

Shell and beam elements are converted into polygons in a straightforward manner. In the case of solids, an algorithm has been designed that first builds polygons from the convex hull of each solid, then eliminates redundant polygons between adjacent solids. For all vehicle components originally modelled as solids, only those polygons which define the outer surface are finally retained.

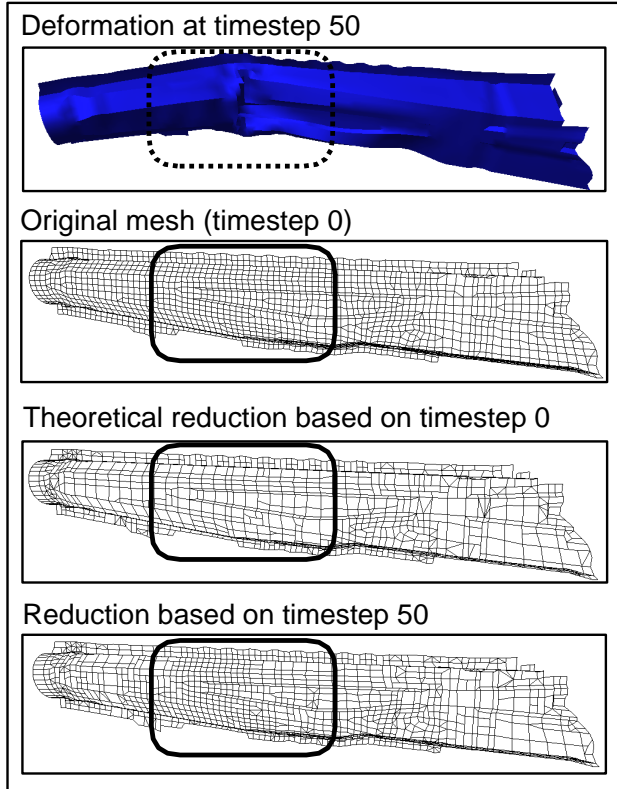


Figure 3: Polygon decimation of the engine mount

To meet memory requirements and to maintain frame rates of at least ten frames per second, the polygon mesh of the model needs to be simplified. It is equally important to keep the shape of the model consistent over time, therefore the simplification algorithm is applied to all timesteps, identifying and preserving those vertices relevant to the simulation and eliminating the rest. Figure 3 shows how this technique reduces the mesh of an engine mount in unimportant regions while maintaining a fine mesh in the area of deformation (highlighted). Our method is based on the work by Schroeder [SZL92] and does not create new vertices. This is essential since we need to preserve scalar and vector-valued data computed

in the finite element analysis for each vertex of the original set.

The chosen polygon decimation criteria is geometric in nature. We compare the angle between tangent plane normals of polygons sharing a vertex. For differences of five degrees or less, this vertex is deleted and a new polygon is created, removing the other polygons formerly belonging to this vertex. Next, adjacent vertices and their polygons are checked and if the normal vector criterion applies again and another new polygon has been created, the node lying at the edge between both new polygons can then be eliminated. The goal is to create larger polygons with fewer vertices and to obtain as few triangles as possible from these multi-edged polygons (Figure 4). The

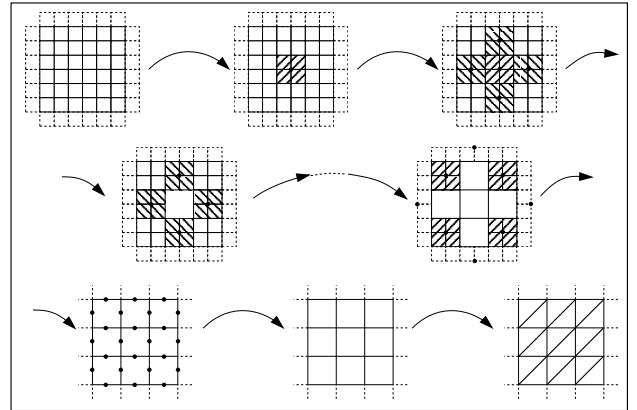


Figure 4: Strategy of the vertex decimation

time required to reduce the mesh added to the time required to build the scene graph of multiple timesteps of the reduced geometry is still less than the time required to construct the scene graph of the same number of timesteps of the unreduced geometry. Typically reductions in the number of polygons in the order of 50 percent are achieved.

### 3.2 Scene Graph and Interaction

A hierarchically-built scene graph encapsulates the graphics and visual simulation features. The object hierarchy consists of a root-node and environment-control nodes, which control the animation and interaction, and geometry nodes that contain the topological information of the vertices as well as graphical attributes of the polygons. The scene graph nodes can be manipulated interactively at runtime (Figure 5)

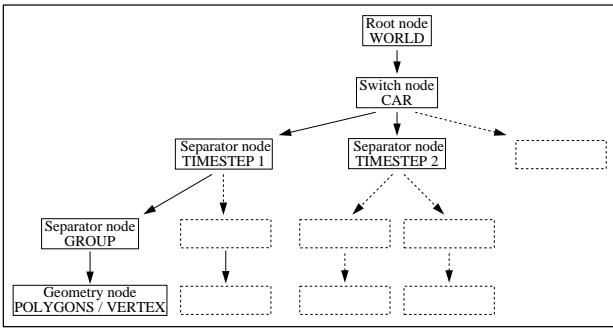


Figure 5: *VtCrash* scene graph

The data varies with time, therefore the visualization will, in general, be in motion. Because moving objects can be difficult to analyse, users must have the opportunity to lock on what they want to see. We provide the ability to attach oneself to arbitrary objects in the scene, thereby observing only relative motion. This can provide the user with, for instance, the crash-test dummy's perspective of driving the vehicle. In addition, individual objects can be selected and freely moved during the animation. Furthermore scalar values calculated at computation-time are displayed either as color maps on the picked geometry or as a function of time and location drawn on a virtual sketch pad appearing in the user's peripheral field of view (Figure 6).

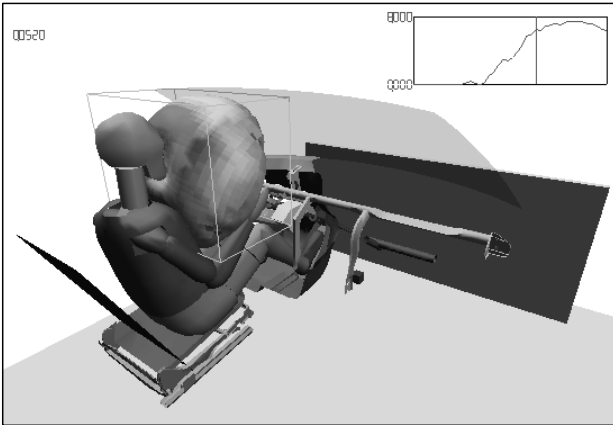


Figure 6: Display of a scalar value as a function of time on the sketch pad

Another interactive feature is the use of a cutting plane. It can also be moved freely in virtual space and slice through virtual objects. Engineers are accustomed to viewing 2D sections of a 3D model, so with the virtual cutting-plane they can enhance a useful tool from their analysis toolbox. They primarily use two methods of calculat-

ing a cut: position-oriented (Euler) or geometry-oriented (Lagrange). The computation of the Euler cut involves performing polygon intersection tests for the geometry for every timestep. The Lagrange cut is computationally the better solution because the desired object is cut in only one timestep and the resulting intersection vertices are then transferred to the other steps, giving the cut geometry the appearance of truly deforming during the simulation. Both methods compute a time-dependent cutting-plane, so the user can see it throughout the animation.

The devices that are used in a particular user session are specified at start-up. Several interface hardware devices are currently supported, including a *FakeSpace BOOM*, stereoscopic large-screen projection as well as the conventional workstation display. In addition, the user can choose between a *CyberGlove*, a 3D mouse or a conventional 2D mouse and a keyboard for input (Figure 7).

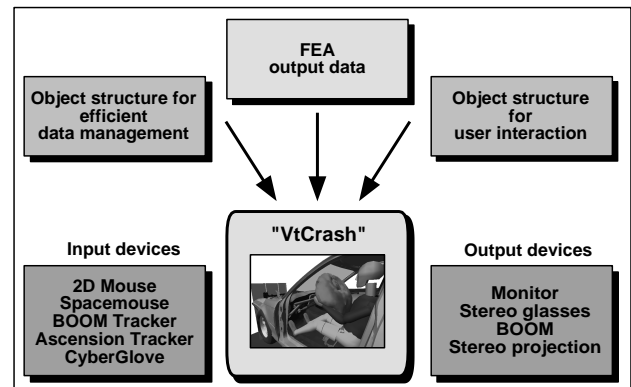


Figure 7: System design

This flexibility supports different usage scenarios - from two or three engineers discussing details to design-build teams of ten people engaged in a project meeting. A simple menu is used to control the visualization. The animation may be stopped and run forward or backward at a user-defined rate in incremental steps.

We learned from the comments of users that stereoscopic viewing is a valuable feature. The best results were achieved with two-channel, large screen projection, where the viewers wore simple polarized glasses that do not require a transmitter and allow individuals to move freely around the space in front of the screen. However, we found test-users somewhat apprehensive of the 3D mouse as a navigational tool and work is continuing in this area to provide the end-user with

an intuitive, comfortable means of traversing the virtual space.

## 4 Application Issues

Our virtual environment for car-body engineering applications supports visualization methods that allow the exploration of different phenomena. There are three different types of simulation towards which this work is directed. The crash-test and vibration simulations are based on shell, beam and solid finite elements and the acoustic simulation primarily makes use of volume elements.

### 4.1 Impact Dynamics

The crash-test simulation results are calculated by the non-linear, transient, dynamic finite element solver *PAM-CRASH* in sixty discrete timesteps. A single crash-test data set consists of a geometry for every timestep and a set of scalar values for each finite element of that geometry. Typically the geometry of a single timestep contains about 400,000 finite elements.

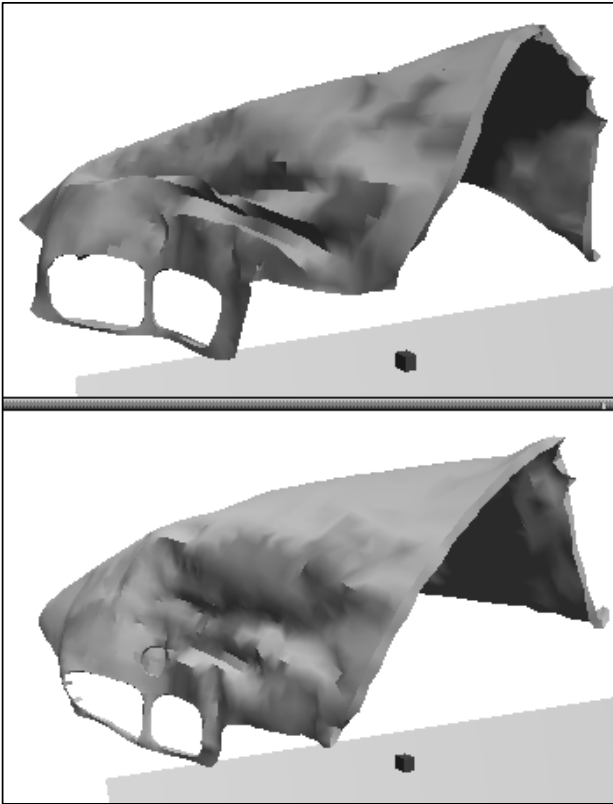


Figure 8: Comparison of deformational behaviour of two variants of the same component

Geometry data and physical properties data like stress, strain, acceleration or velocity are displayed to illustrate the performance of the whole vehicle or any subset of components. It also proved to be particularly useful to run two visualizations in parallel, each with a different design alternative (Figure 8).

### 4.2 Vibration

Vibrational analysis is computed as a linear modal eigenvalue simulation of the car-body structure. The car structure consists of about 200,000 finite elements. As opposed to the crash-test simulation there is only one complete geometry description which contains the nodes and the element connectivities. Every vertex  $\vec{v}_0$  in the data set has a complex displacement vector  $\vec{d}$  associated with it. The following equation is used to calculate the vertices  $\vec{v}_i$  corresponding to the timestep  $i$ ,

$$\vec{v}_i = \vec{v}_0 + \vec{d}e^{i\tau_i}$$

$\tau_i$  corresponds to the timestep  $i$  and is defined by,

$$\tau_i = i \frac{2\pi}{N}$$

where  $i = \text{number of the current timestep}$   
 $N = \text{total number of timesteps}$ .

In this way we can create the geometry for every desired timestep and visualize it in the virtual environment. Over the course of the animation the vehicle vibrates with a user-defined amplification factor and the engineer can analyse the simulation more clearly than by interpreting just the complex displacement vectors.

### 4.3 Acoustics

The noise level (or preferably the absence thereof) in the passenger compartment is a feature of relatively high customer value because it is easily noticeable. This noise occurs when the volume of air that occupies the passenger compartment is subjected to a change in pressure caused by the excitation of resonating components such as the floor panel and the fore and aft firewalls. The simulation software calculates one complex scalar value of the air pressure,  $p$ , for every element. We calculate the pressure for every timestep using the equation,

$$p_i = pe^{i\tau_i} ,$$

where  $\tau_i$  is defined previously.

Typically the volume of air consists of about 20,000 voxels and as a component begins to vibrate a pressure wave originates from it. An intuitive means of showing this noise level behaviour consists of calculating an isosurface for the air pressure for every timestep. Lorensen et al. [LC87] propose a marching cube algorithm for isosurface calculation by interpolating the scalar values at the edges of the voxels. The input volume consists of a variety of voxel types (tetrahedron, pyramid, prism and hexahedron) which are broken down into tetrahedrons to avoid ambiguities in the surfaces.

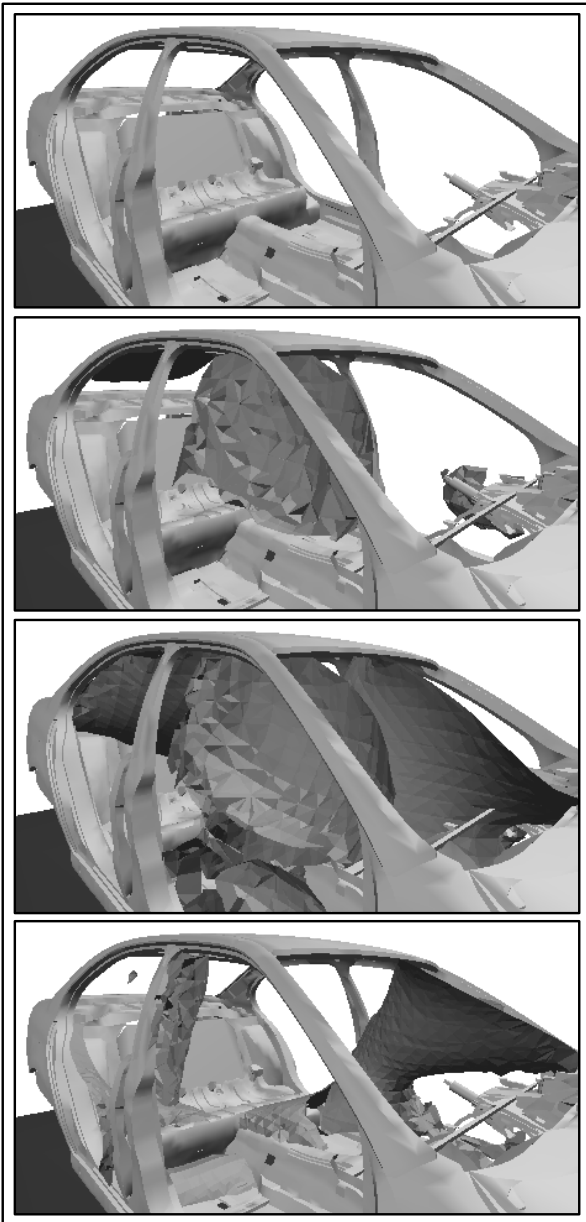


Figure 9: The noise level wavefront moving through the passenger compartment

The source of the change in air pressure and the resulting noise level can be localized and the propagation of the wavefront through the passenger compartment visualized (Figure 9).

## 5 Conclusions

We presented results of on-going research in the development of a virtual environment for car-body engineering applications.

*VtCrash* is able to load simulation files directly and efficiently without any special preparation and provide insightful, intuitive visualization which allows effective communication between users.

This visualization tool is being used at *BMW* in a large-screen stereoscopic projection format for design reviews between engineers and other design professionals.

Work on the integration of different simulation types into the system is proceeding well. Further research areas include creation of an isosurface at a picked point in space, improved isosurface calculation performance as suggested by Bajaj et al. [BPS96] and a cutting-plane through the wavefront to display isolines, Meyer et al. [MG93].

Furthermore, texture mapping and other visualization techniques such as “force tubes” , Kuschfeldt et al. [KEH97], are also being explored as improved means to visualise scalar and vector quantities.

## 6 Acknowledgements

Many of the initial ideas elaborated on in this paper were inspired by many discussions with Sven Kuschfeldt of the University of Erlangen and Michael Holzner of *BMW*. We would like to thank Stephen Smyth for his contribution to software development and preparation of this manuscript. Conceptual inspiration came from the work of Steve Bryson et al. on the *Virtual Windtunnel* project at *NASA Ames Research Center*. Finally, we gratefully acknowledge the continuing support of the University of Erlangen and *BMW*.

## References

- [BF95] S. Bryson and S. Feiner. Virtual Environments in Scientific Visualization. In *Virtual Reality for Visualization, Course Notes of Tutorial 5 at Visualization 95*, 1995.
- [BL92] S. Bryson and C. Lewit. The Virtual Windtunnel. In *IEEE Computer Graphics and Applications*, 12(4):25-34, 1992.
- [BPS96] C.L. Bajaj, V. Pascucci, and D.R. Schikore. Fast Isocontouring for Improved Interactivity. In *Proc of IEEE Visualization 96*, 1996.
- [Bry94] S. Bryson. Approaches to the Successful Design and Implementation of VR Applications. In *Proc. SIGGRAPH'94*, 1994.
- [KEH97] S. Kuschfeldt, T. Ertl, and M. Holzner. Hardwareunterstuetzte Visualisierung von Struktureigenschaften und physikalischen Belastungsgroessen in Crash-Simulationen. In *Proc. VisEng'97, Rechenzentrum Universitaet Stuttgart, Stuttgart, Germany*, 1997.
- [KSR<sup>+</sup>96] S. Kuschfeldt, M. Schulz, T. Reuding, T. Ertl, and M. Holzner. Visualization of Crashworthiness Simulations using Virtual Reality Techniques. In *Proc. International Conference on High Performance Computing in Automotive Design, Engineering and Manufacturing, Paris*, 1996.
- [LC87] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Comput. Graph.* 21,4(1987),163-169, 1987.
- [MG93] T. Meyer and A. Globus. Direct Manipulation of Iso-surfaces and Cutting Planes in Virtual Environments. In *Technical Report RNR-93-019, NASA Ames Research Center*, 1993.
- [NN95] D. Nishioka and M. Nagasawa. Reducing Polygonal Data by Structural Grouping Algorithm. In *Lecture Notes in Computer Science 1024, ICSC 95*, 1995.
- [RO96] K.J. Renze and J.H. Oliver. Generalized Surface and Volume Decimation for Unstructured Tesselated Domains. In *Proc. IEEE Conference VRAIS'96, Santa Clara, CA.*, 1996.
- [Sch97] M. Schulz. Einsatz von Virtual Reality Techniken zur Darstellung von Ergebnissen der Karosserieberechnung. In *Proc. VisEng'97, Rechenzentrum Universitaet Stuttgart, Stuttgart, Germany*, 1997.
- [SZL92] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of Triangle Meshes. In *Proc. of SIGGRAPH 92*, 1992.
- [YV97] W. Ye and J.M. Vance. Visualization of Structural Impact Problems in a Virtual Environment. In *Proc. SCS Simulation MultiConference, Atlanta, GA.*, 1997.
- [YhV97] T.P. Yeh and J.M. Vance. Combining Sensitivity Methods, Finite Element Analysis, and Free-Form Deformation to Facilitate Structural Shape Design in a Virtual Environment. In *Proc. 23rd ASME Design Automation Conference, Sacramento, CA.*, 1997.