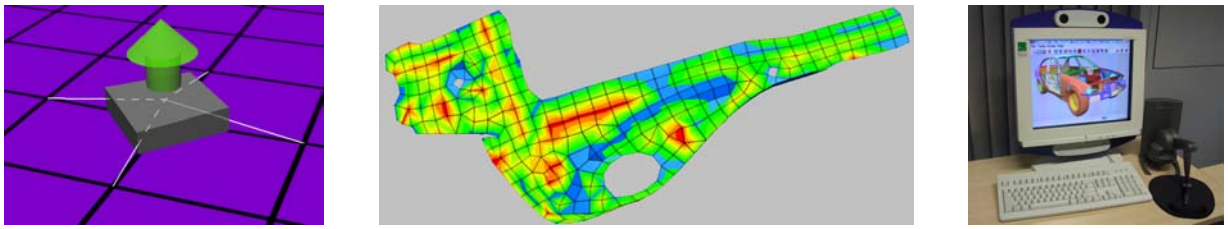
# Intuitive and Interactive Modification of Large Finite Element Models

Dirc Rose Katrin Bidmon Thomas Ertl

Visualization and Interactive Systems Group, University of Stuttgart*



(a) (b) (c)

Figure 1: Tools for interactive editing: (a) widget for intuitive 3D modification; (b) 1D parameter texture revealing node displacement; (c) future engineering workplace: autostereoscopic display and haptic input device.

**ABSTRACT**

Virtual prototyping is increasingly replacing real mock-ups and experiments in industrial product development. Part of this process is the simulation of structural and functional properties, which is in many cases based on Finite Element Analysis (FEA). One prominent example from the automotive industry is the safety improvement resulting from crash worthiness simulations. A simulation model for this purpose usually consists of up to one million finite elements and is assembled from many parts which are individually meshed out of their CAD representation. In order to accelerate the development cycle, simulation engineers want to be able to modify their FE models without going back to the CAD department. Furthermore, valid CAD models might even not be available in preliminary design stages. However, in contrast to CAD, there is a lack of tools that offer the possibility of modification and processing of finite element components while maintaining the properties relevant to the simulation. In this application paper we present interactive algorithms for intuitive and fast editing of FE models and appropriate visualization techniques to support engineers in understanding these models. This includes new kinds of manipulators, feedback mechanisms and facilities for virtual reality and immersion at the workplace, e.g. autostereoscopic displays and haptic devices.

**CR Categories:** I.3.4 [Computer Graphics]: Graphics Utilities—Virtual device interfaces; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.6.3 [Simulation and Modeling]: Applications

**Keywords:** finite element modeling, interaction, manipulators, autostereoscopy

## 1 INTRODUCTION

Almost every new design or model is created with the aid of computers, and this is especially true in the automotive industry, where the computer superseded the drawing board many years ago. Computer aided design (CAD) has evolved to an irreplaceable tool in the daily work routine of a design engineer. This adoption of computer technology in the development cycle matches its introduction into the final production stage, e.g. by steering computerized numerical control (CNC) machines.

However, the development cycle has been slowed down for a long time by the need for experiments to prove the structural and (aero-)dynamic performance of designed models. To carry out such experiments, many expensive prototypes used to be built in a time-consuming process. During the last two decades this procedure has changed and an increasing number of test runs with real prototypes is being replaced by virtual simulations. During its early days, the finite element method helped the engineers to perform relatively simple structural analysis. Thanks to the growing processing power of modern parallel computers and to efficient algorithms it is nowadays possible to calculate complex non-linear and highly dynamic processes like crash worthiness simulations within two or three days. In general, such numerical simulations need a special preparation and simplification of the CAD model: the analytical surfaces must be converted into an FE mesh. For this purpose, many conversion algorithms (e.g. [5, 36]) have been developed, but they are far from being perfect and most of them are tailored to a specific kind of simulation or preservation of a particular model property. Therefore, a lot of expert knowledge and manual work is still required to preprocess these meshes in a manner such that the numerical simulation provides valid results. Some years ago, an improvement in the numerical algorithms was introduced, which alleviated this task by supporting individually meshed components instead of a single consistent mesh for a whole car. This made it possible to exchange only some parts without having to remesh the complete car model. Additionally, the assembly of the different car components can be reproduced more realistically by this approach due to the introduction of spotwelds or adhesive bondings. This change in the computational workflow combined with other efficiency gains results in a huge acceleration of the complete development cycle and even allows for stochastic analysis of model variants.

New challenges arise with this approach. The engineers need appropriate tools to create interconnections between the various car components in a fast and easy way. These interconnections—e.g. spotwelds—are usually placed at flanges, i.e. where the surfaces of two or more components run parallel at a very small distance. Due to meshing inconsistencies, sometimes adjacent materials may penetrate or perforate each other at these flange areas. In general, au-

*Universitaetsstrasse 38, 70569 Stuttgart, Germany
e-mail: (rose|bidmon|ertl)@vis.uni-stuttgart.de
web: http://www.vis.uni-stuttgart.de/

IEEE Visualization 2004
October 10-15, Austin, Texas, USA
0-7803-8788-0/04/$20.00 ©2004 IEEE

tomatic detection, indication, and correction is possible, but, since the adjustment operation can be ambiguous, manual modification is needed. With the conventional approach, this means that the engineer has to ask the CAD department for corrected components, which have to be remeshed from scratch. Again, the engineer has to deal with problems induced by imperfect meshing algorithms. Therefore it can be very profitable to have a comfortable editor that is able to manipulate FE meshes directly. This saves time and it also offers the possibility to create variants of existing components very fast—e.g. the integration of crimpings for structural stiffening. To accomplish such editing operations we need to provide methods that can be used by the engineer to select the three-dimensional surface region he wants to interact with. During the editing operation erroneous elements—i.e. elements that might induce misleading or false results—should be detected and highlighted, and once the modification is completed intelligent algorithms should try to straighten out these errors without destroying the properties and features of the part. In a final step, a restructuring of the affected elements—comparable to a remeshing in situ—can resolve the remaining errors. We will go into detail of the entire procedure in section 3 and 4.

Modern graphics hardware can support these preprocessing methods by using textures to highlight e.g. problematic regions or to compare various modelling states. The latter can be used also to compare the states before and after an automatic or manual modification. Recent developments in display technology like autostereoscopic displays can help the engineers to understand complex or twisted car components, too. Moreover, in CAD it is often easy to estimate the actual position and orientation of a certain part relative to its surrounding components, since the model is very detailed. The FE model on the contrary has been reduced to its important or supporting parts only, and is quite coarsely meshed compared to CAD. Therefore, it is sometimes quite difficult to estimate the correct alignment of a single part that might seem to float in the air. Here, stereo rendering can improve the understanding of the simulation model and we will present a solution for the workplace in section 5.1.

Even though proper visualization of FE models can support the understanding of the structure and quality of a mesh, it is still difficult to carry out editing operations with six degrees of freedom (DOF) using devices like a mouse supporting only two-dimensional input [15]. In section 5.2 we will address hardware that is able to provide 6 DOF input and how to use it for interaction and intuitive modification. We will focus on haptic devices, which additionally supply a feedback mechanism that can be used to prevent the generation of illegal elements or perforation of nearby car components.

This paper starts by providing some previous work in the following section and rounds off with some results and conclusions (section 6).

## 2 Related Work

Many interaction methods, especially the ones implemented for CAD, are geared towards traditional design jobs and many of them try to copy the working methods used at a drawing board, even if they are expanded with high precision and numerical input. Another approach allows the engineers to build their complex constructions from simple geometric primitives. Neither of them is suitable for interaction with FE models. Especially in preliminary design stages there is a need for fast and intuitive methods to adjust the FE mesh instead of accurate and more complicated alignment operations like it is done in CAD.

Therefore, we focused on intuitive interaction techniques, keeping them simple and laconic and requiring only one or two mouse clicks, a behavior which is also suggested by [19]. [11] showed how to remove perforations and penetrations of interfering parts automatically and how to connect individually meshed components by interactively placing spotwelds or adhesive bondings along curved flanges. As mentioned above, a fully automatic solution cannot be always provided for ambiguous tasks. [10] suggested to use directly manipulated free-form deformation to edit such problematic areas manually. However it uses spline volumes for deformation which predefines the final deformation and the interaction is steered by sliders and buttons. A better approach are three-dimensional widgets that can be used to control the modification in situ. SGI's Inventor [33] demonstrates with its manipulators how to use widgets for 3D interaction. A first step in using 3D widgets for modification instead of limiting them to camera and object interaction has been presented by [7]. However, they have only rarely found their way into editing tasks and not at all into the field of FE mesh modification. [8] developed a high-level framework providing 3D widgets for general interaction purposes. Although intuitive, this framework is too over-engineered and extensive for manipulation of surfaces, and its area of application lies within complex interaction with static or animated objects. Surface editing with various 3D widgets was presented by [13], who combined selection of the editing region and manipulation into the same step which results in a somewhat confusing operation. A solution using simple and intuitive 3D glyphs was presented by [27], but it lacked support for direct manipulation.

Our recent approach to simple and at the same time versatile modification of FE meshes uses mouse drags as input and a variety of basis functions to describe the deformation of the surrounding surface. It is similar to the approach presented in [4], which was developed independently and which works on triangulated surfaces. For smoothing the mesh after the modification step, various algorithms have been developed, most of them based on relaxation. The most common method is Laplacian smoothing, a simple recursive method where the nodes are directly displaced depending on the neighbors they are connected to. This method has been improved and extended by several authors to fit some special needs, e.g. [36, 5]. Another technique is the so-called optimization-based smoothing where the nodes are moved depending on the minimization of a special distortion metric. Also combinations of both techniques were presented, e.g. [9]. A more detailed overview is given in [2, 6]. A third approach is physically based smoothing as presented e.g. in [22] where the edges in the mesh are represented by springs and the corresponding force depends on the ratio between the desired and the actual edge length. This approach is quite similar to the one we developed in this paper. As we have to deal with both quadrilaterals and triangles, we also have the condition that the diagonals of the quadrilaterals should be equal, so – in comparison to [22] – springs are added there too, but their desired length is adapted to be $\sqrt{2}$ times longer than the edges of the element. Most of these algorithms have been developed for smooth or even closed surfaces, but in the case of car parts there are boundaries and often holes or edges within the surface. Additionally, in order to preserve the significance of the simulations, it is important that the nodes really lie on the surface originally modeled in CAD. This is why the most important improvement in our relaxation technique is that these features (boundaries and edges on the surface) as well as the surface itself are preserved.

To assist the understanding and modification of three-dimensional models we propose the use of autostereoscopic displays which provide some sort of immersion at the workplace without the need for glasses. Recently, many solutions have been presented, e.g. [25, 24, 29] and starting this year there is an autostereoscopic display for notebooks available on the market [14]. Since stereoscopic representations need the scene to be processed twice, we combine the native support of our application for autostereoscopic displays with hardware-optimized rendering of a simplified car model without any loss in quality as presented earlier [26].

Concerning input for 3D interaction and editing various devices and ideas exist. [34] use two mice as input, but most people find it odd and difficult to handle a mouse with their non-dominant hand, and controlling two mice at the same time will not gain broad acceptance by engineers. [32] introduced 6 DOF input for 3D editing and they also suggest 3D stereo rendering for better perception. However, their system has been designed for character animation and not technical modeling, which has different goals and needs concerning the type of interaction, handling, and visual feedback.

With the growing selection of haptic input devices [21] more direct and intuitive ways to handle and edit surfaces become possible, as well as providing feedback by interactively checking for collisions [17, 18] while performing the modification. An approach for simple interaction with the environment using haptic devices has been presented by [20]. It also uses 3D widgets to give feedback to the user, but does not cover manipulation. The future workplace that is being proposed in this paper (see Fig. 1(c)) has already been partially implemented by [35], who use haptic input and feedback for simple design tasks, and [3], who combine autostereoscopic presentation with a haptic device for interaction with virtual objects, but they focus on planning of space mission operations in virtual environments and not on editing of surfaces.

In the following sections we show how we extended this vast body of knowledge by our own ideas to significantly improve the intuitive modification of FE meshes. By implementing our algorithms into the commercially available preprocessing application *scFEMod* [28] we have made this functionality available for productive use in the CAE departments of major German car manufacturers.

## 3 Selecting and Editing in 3D

Unlike [13] we decided to split the modification process into two consecutive steps. First, the user defines the specific elements or nodes he wants to edit using a simple selection mechanism, and then he performs the actual manipulation itself.

### 3.1 Selection Mechanisms

The easiest way to implement a selection mechanism, from the programmers point of view, is to provide a text box, where the user can enter the unique labels of the nodes. Surprisingly, some engineers know their datasets so well, that they prefer this cryptic method over anything else. Therefore, we also implemented such a dialog, but we extended it with boolean and range selection operations for improved convenience. However, in order to deal with the increasing circulation of new models and changing node IDs, we suggest to use a much more intuitive method, usually utilized for selection in two-dimensional paint programs. There the user can drag the mouse, encircling the pixels he wants to select with a freehand curve and release the button to complete the action. In our application the engineer can draw a freehand line on the screen (see Fig. 2(a)), thereby cutting a sort of pyramidal frustum out of the 3D scene. To guarantee that the polygon is always closed, the start and end points are always connected (thin line in Fig. 2(a)). All nodes of the FE mesh that lie inside the pyramidal frustum are marked as selected and are highlighted via a white octahedron accordingly, as seen in Fig. 2(b). Occluded nodes are rendered transparent, so that at any time all selected nodes can be seen. To decide whether a node is outside or inside the freehand frustum we do not need to perform an expensive calculation in 3D, it suffices to project the nodes into two-dimensional screen space and test the coordinates against the freehand outline. This can be done very fast using the point containment test in [23].

The user can choose whether he wants to select all the nodes inside the selection frustum of a component or if he only wants to select the currently visible nodes. The latter can be implemented by shooting rays into the scene for every potential node candidate and checking for occlusion. This can be accelerated by rendering the car model with a unique color assigned to every element. We can then check the pixels in the neighborhood of the 2D coordinate of a potentially selected node for their colors or rather element labels. If none of these elements is adjacent to the node then it is occluded, otherwise we can continue with the more precise ray intersection test. For better convenience the user can add or subtract new selections to/from the existing ones, e.g. the selection in Fig. 2(c) has been achieved by selecting the node cluster with occlusion culling enabled and then deselecting the nodes in the center via subtraction.

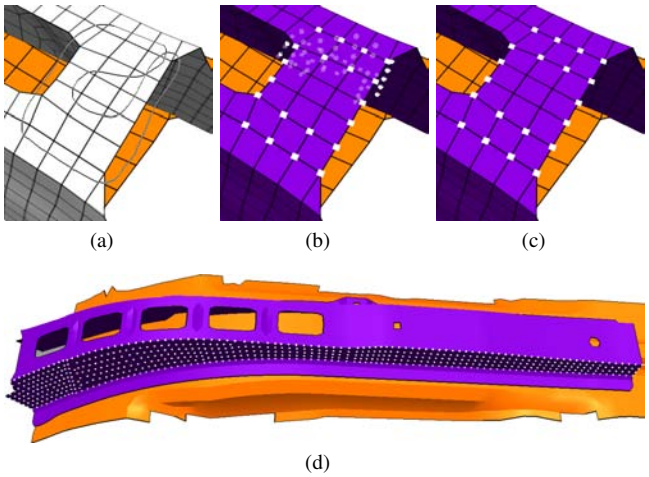


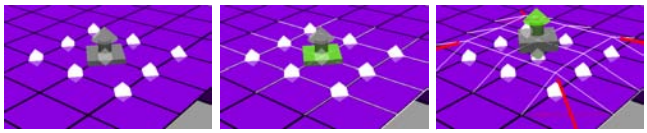

Figure 2: Freehand Selections: (a) encircling nodes; (b) selection without occlusion test, hidden nodes are rendered transparent; (c) selection originated from a subtraction operation; (d) selection of a region delimited by features.

For cases where the engineer wants to edit a complete segment of a component we implemented another selection mode, where the user clicks onto a part and all nodes belonging to the same region are selected at once. Such regions are delimited by features of the part, e.g. sharp edges as seen in Fig. 2(d). For further details on robust feature detection on FE meshes see [26].

### 3.2 Mesh Modification

The simplest editing operation is a collective parallel translation on a group of selected nodes. But in 3D even this cannot be accomplished as in 2D, where mouse movement can be mapped directly to a corresponding translation. Given 2D-only input, e.g. a mouse, 3D interaction with three or more DOF has to be split into multiple 2D movement actions. In our application we use the manipulator widget seen in Fig. 3(a) and Fig. 1(a). It consists of a disk and an arrow. The user can either click on the disk as in Fig. 3(b) and drag it around, whereby the displacements are constrained to the surface of the finite elements, or he can drag the arrow and shift the selection along the local surface normal like in Fig. 3(c). During interaction, the active part of the widget is highlighted in green and the modified surface is rendered as wire frame representation (white lines in Fig. 3(b) and 3(c)). The FE surface then is updated when the user releases the mouse button.

In most cases, simple parallel translations are not flexible enough. We implemented another approach that offers the possibility of much more complex deformations and which contains parallel translations as a special case. The interaction is kept as simple as before, but instead of moving all nodes along the same vector, we introduce a weighting function that describes the position of a selected node relative to the border of the selection and to the position



(a) (b) (c)

Figure 3: Manipulator for 3D movement: (a) neutral appearance when initially locked to node; (b) disk for movement on surface selected; (c) arrow for displacement along local normal selected and lifted by dragging with the mouse.

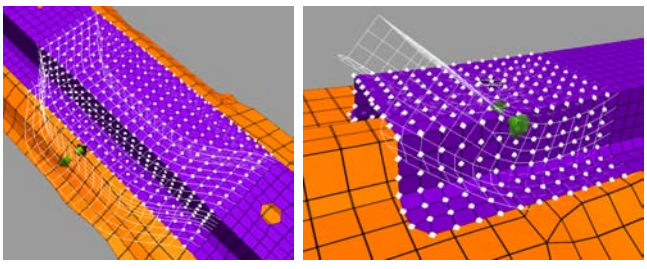
where the user started dragging. If $d_{i,\text{border}}$ denotes the geodesic distance of a certain node $i$ to the border of the selected region and $d_{i,\text{origin}}$ the geodesic distance of the node to the point where the user grabbed the surface, then the weight $w_i$ can be calculated by

$$w_i = \frac{d_{i,\text{border}}}{d_{i,\text{origin}} + d_{\text{i,border}}} . \quad (1)$$

The range of values of all $w_i$ is guaranteed to be within $[0,1]$. If we use these weights directly to scale the displacement vector we derive by the mouse movement along the widget arrow, then we obtain a deformation shaped like a cone or pyramid—depending on the shape of the selection. But the user can also choose from a variety of basis functions, e.g. when the engineer chooses a remapping of the weights similar to a Gaussian then he is able to achieve a result as shown in Fig. 4(a). Here, the Gaussian is flattened in the centerpiece, whereby the amount is configurable, and applied only along one dimension. In the perpendicular direction the weights are ignored and set to constant 1 instead.

Rotations can be achieved by first defining the rotation axis and then describing an arc with the mouse that defines how much the selected region should be twisted (Fig. 4(b)). Furthermore, you can see that the rotation is performed to its whole extent at the free ending without any weighting function applied. Here, free endings are areas between the axis of rotation and border segments where the boundary of the car component and of the selection match exactly, e.g. the left half of the component shown in Fig. 4(b). If the user wants to leave the free ending fixed he can accomplish this by deselecting the outer row of nodes. This behavior in treating free endings applies not only to rotations but also to displacement operations and it is quite intuitive and enables many different kinds of modification. E.g. it is very easy to elongate a certain section of a car component just by defining the section to be streched by selecting its nodes and then dragging at the free border of the selection.

An extension to our modification widget might be dragable shadows as proposed in [16]. There, an object casts shadows onto multiple planes and the user can either interact with the object or with the shadows. The interaction with the objects shadows is easier because the movement is limited to a 2D plane. Applied to our manipulator it might increase its precision, but it is not clear where the shadow



(a) (b)

Figure 4: More complex modifications: (a) creating a bulge; (b) rotation with free ending.

planes should be located for a good and intuitive navigation. The surface of the FE mesh itself can serve as a shadow plane, but on the other hand an additional shadow is more confusing the user than a helpful facility for interaction in some special cases.

## 4 Mesh Optimization

Editing a mesh inevitably leads to displaced nodes, which might result in extreme angles between edges, or edges that vary too much within one element. Because of this the elements may become of poor quality for later computations. For FE methods "good elements" means quadrilaterals and, where triangles cannot be avoided, equilateral triangles.

### 4.1 Visualizing Erroneous Elements

To give the engineer instantaneous feedback about the quality of the current FE mesh, the numerically relevant properties of each element are checked during the modification operation. Erroneous elements are highlighted immediately by using red glyphs which correspond to the error type. Critical elements are marked with yellow glyphs. An example for erroneous elements can be found in Fig. 3(c), where quadrilateral elements have been warped, i.e. where not all nodes of an element lie within the same plane. When warping occurs, the quadrilateral is bent around its two diagonals and we symbolize the larger of both bending angles by a thicker diagonal, respectively. The glyphs and types of other common FE defects are depicted in Fig. 5.

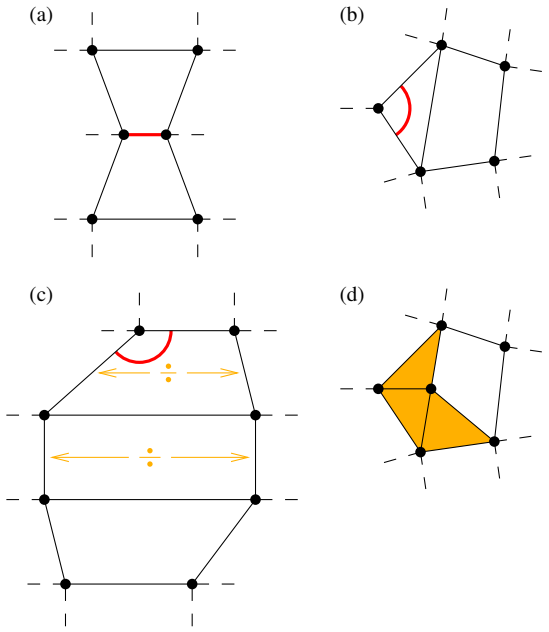
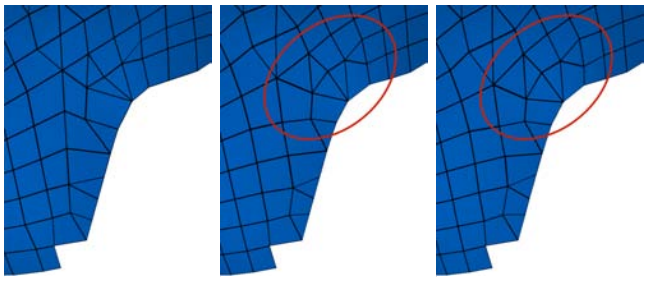



Figure 5: Various finite elements marked as erroneous: (a) edge too short; (b) angle too large (in triangle); (c) angle too large (in quadrilateral) and bad edge length ratios; (d) many adjacent triangles.

### 4.2 Relaxation

After editing the mesh it may be necessary to make the mesh suitable again for numerical simulations. So the positions of the nodes have to be adjusted e.g. by relaxation. In our application we use a relaxation model based on a spring-mass model, where the edges of the mesh as well as the virtual diagonals of the quadrilaterals are represented by springs. Only solving the resulting ordinary dif-



(a) (b) (c)

Figure 6: Comparing mesh relaxation methods: (a) original mesh; (b) hierarchically relaxed mesh; (c) non-hierarchically relaxed mesh.

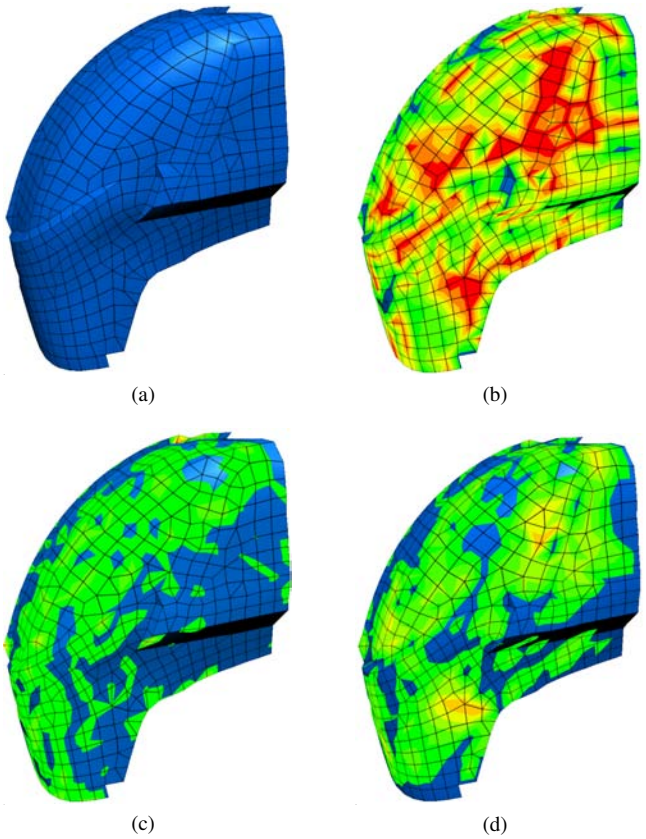
ferential equation (ODE) system would lead to good elements, but the features of the underlying surface would be mostly lost, i.e. the bordering line would change as well as edges in the surface, and the surface itself would also get flattened.

To avoid such artifacts we introduced some control mechanisms: after calculating the new position of each node we compare this position to the original surface. Boundary nodes and nodes on surface edges are only moved along their specific line, nodes that define corners on such lines are not moved at all, and the new coordinates of inner nodes are projected onto the originally modeled surface. As we do not have the parametric representation of the surface but only the mesh, we interpolate the original surface and the surface feature lines on which we project the nodes. Since relaxing the mesh by solving the ODE system with these boundary conditions for one whole part with 500–1000 elements takes up to one or two minutes on a standard PC, we developed an algorithm that produces similar results but with interactive performance (Fig. 6): The mesh is relaxed hierarchically starting at a node selected by the user e.g. in the region of highest irregularity. For each step all neighboring nodes displaced beyond a user-defined threshold are taken into account for the next step, until no new nodes get moved. In the next steps only this region will be relaxed once more. As this relaxation method depends on the starting node it can be applied several times, but the result only changes slightly after the first few times. To give the engineer more control it is also possible to restrict the relaxation to a group of selected nodes.

For evaluation of the editing operations that have been performed, visual feedback is provided: A one level undo function has been implemented, with a toggle for direct comparison. Additionally our application can be started in a multiple view mode, where the user can have two windows with synchronized views. In one the user can keep the original model and in the other he can perform modifications on the corresponding part. In this multiview mode it is also possible to map the differences between the two parts via 1D textures (see Fig. 7). With these textures either the node displacement caused by relaxation (Fig. 7(b) and Fig. 1(b)), or the distance of the relaxed nodes to the original surface can be mapped (Fig. 7(c)). In general, an engineer does not want to cope with the displacement of single nodes and therefore the latter visualization is the more significant one, as it shows how the surface has changed by the relaxation. This kind of texture mapping can also be used to visualize distances between arbitrary parts to detect e.g. penetrating flanges [11]. The distance calculation is based on a bounding volume hierarchy [12] and therefore performs very well.

### 4.3 Mesh Restructuring

Although very powerful, relaxation does not always eliminate all errors in the mesh. Particularly in cases such as when a part gets elongated too much or the user creates a deep buckle, relaxation alone will not be able to produce a valid FE mesh. In these cases



(a) (b) (c) (d)

Figure 7: Comparing mesh relaxation methods: (a) original mesh; (b) node displacement from original to non-hierarchically relaxed mesh ($<$ 1mm transparent, $>$ 5mm red); (c) euclidean distance between non-hierarchically relaxed mesh and original ($<$ 0.05 mm transparent, $>$ 1mm red); (d) difference between hierarchically and non-hierarchically relaxed mesh ($<$ 1mm transparent, $>$ 5mm red).

we find the elements that were the worst ones before the relaxation procedure and try to fix them by a local restructuring. It can be seen as a kind of remeshing, but confined to the direct vicinity of an erroneous element, which guarantees to prevent unnecessary changes in the FE structure. E.g. it is easy to repair warped quadrilaterals—introduced in section 4.1—by splitting each into two triangles along their "thick" bending diagonals. Clean-up operations for various other mesh defects are listed in Fig. 8 according to the errors shown in Fig. 5. The operation presented in Fig. 8(d) usually is one of the last steps and tries to conjoin triangles into quadrilaterals. This decreases the number of triangles that might be produced by previous repair operations. Finally, the neighborhood of erroneous elements is relaxed once again. If the resulting mesh still is not sufficiently well shaped for numerical simulation the algorithm continues the restructuring also on less critical elements. This procedure is repeated until an FE mesh valid for crash worthiness simulations evolves, and experience shows that the algorithm converges very fast.

## 5 Virtual Reality at the Workplace

As discussed before, stereographic projection can support engineers while working on complex FE models and recent display technology allows the engineer to get this 3D illusion without the need for special glasses. Combined with an input device providing more than two degrees of freedom this environment can be considered an immersive workplace.

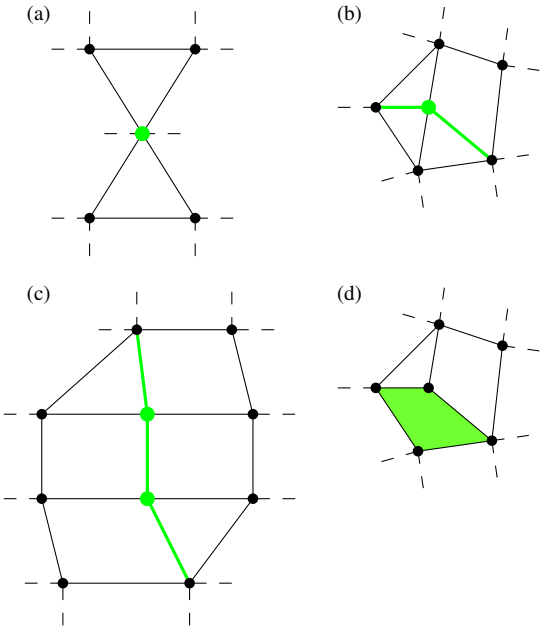



Figure 8: Finite elements from Fig. 5 have been fixed: (a) edge is collapsed to common node; (b) triangle and adjacent quadrilateral is divided; (c) stretched and adjacent elements are divided; (d) adjacent triangles are merged if possible.

### 5.1 Autostereoscopy

In recent years many solutions for autostereoscopic impression have been developed. Most of them use a prism system or thin blades to bend or block the pixel columns of a standard TFT display in a way that alternating columns can be seen either with the left or with the right eye. The display we use (see Fig. 1(c)) contains a prism mask to bend the rays and it also has a dual-camera system which tracks the position of the user's eyes. An embedded processor interprets the eye position and controls the position of the prism mask, which can be displaced horizontally by servo motors so that it is guaranteed that each eye will see the right column. Fig. 9 illustrates the top view of the optical paths.

To drive such a display, a pair of stereographic images has to be rendered vertically interlaced into the frame buffer. Most graphics card manufacturers provide drivers for this purpose, but generally either only for their high-end graphics boards or for WIN32-based platforms only. To circumvent these problems we implemented native support for autostereoscopic displays in *scFEMod*. An alternative to native support is presented in [31], which we extended to support also autostereoscopic displays.

A fast and easy solution to achieve the alternating distribution of the two images is to use a matching stencil mask, render the image for one eye, change the stencil test to the appropriate setting and render the scene again for the other eye (for further details on how to set up the camera parameters for correct stereographic projection refer to [31]). On some architectures the rendering might perform better when the stencil test is disabled for the first pass. This works well as long as one does not want to render thin lines e.g. a wire frame representation of the car model. Especially bevel lines will not be reproduced very well, as the stencil mask will block some pixels (Fig. 10(a)) and as a result the user will see an interrupted line as depicted in Fig. 10(b). Our approach to solve this problem, which additionally produces much better quality when full scene anti-aliasing is enabled, halves the resolution of the viewport and renders narrowed images of the scene. The two images can be stored on the graphics card using texture memory and distributed among the columns by rendering textured lines. Again, on certain architectures it may perform better if for the first image a different strategy is chosen, e.g. distributing the columns via `glCopyPixels`. Fig. 10(c) shows the corresponding result using this method and in Fig. 11 and 1(c) one can see autostereoscopic rendering of a car model.

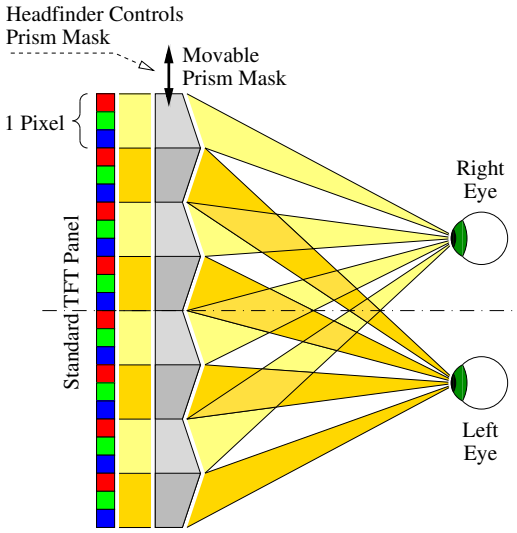



Figure 9: Principle of an autostereoscopic display [29].

The eye position acquired by the two built-in cameras can be read out via a serial port. The position is given with respect to the center of the monitor and can be used for simple tracking and the transformation of the presented model can be updated accordingly.

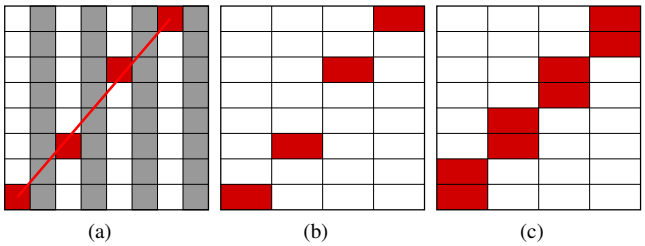



Figure 10: Rendering a line: (a) stencil mask (gray) blocks pixels during rasterization; (b) the prism system of the display widens the pixels and the user sees a dotted line through his left eye; (c) correct result (users view through prisms) by distributing half-sized viewport on odd-numbered columns.

### 5.2 3D and Haptic Editing

As already discussed, 3D editing using 2D input devices is quite problematic. A *Spacemouse* [1] provides input with six degrees of freedom. It has established itself as easy to use interaction device, but it is too inaccurate for editing purposes. Haptic devices, which have been developed over the past years, also provide input of 6 or even more DOF. Specifically, we use a *Phantom Desktop* [30] seen in Fig. 1(c). The user interacts with the model by holding a pen-like device with a button. Currently we are investigating how such an input device can be used for interactive and intuitive editing. The pen can be represented on-screen by a corresponding virtual pen in the 3D scene and its point can be used to interact with the FE surface. Thanks to the *Phantom's* many degrees of freedom it is easy to navigate to the desired node, press the button and perform almost arbitrary translatory and rotatory modifications on the se-

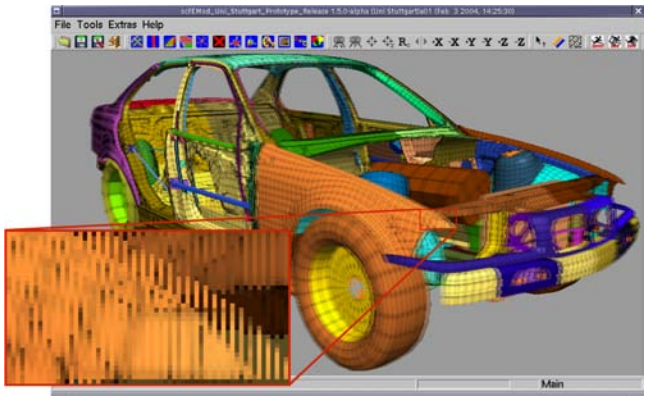



Figure 11: FE model of a car rendered for display on an autostereoscopic monitor (data courtesy of *BMW AG*).

lected set of nodes. For selecting the nodes a 2D mouse is still the best interaction device, but the 3D widget presented in section 3.2 can be replaced completely by the virtual pen making our editing operations even more intuitive to control.

Another advantage of these devices is their ability to provide haptic feedback. This can be useful to prevent the pen from perforating the FE surface by generating appropriate forces in case of a collision. This behavior makes it easier for the user to grab a node lying on that surface and it also aids to avoid perforation of other parts during modification of the mesh. Additionally, force feedback can be employed to let the engineer perceive more and more resistance with increasingly worse element quality. To implement a fast collision detection one can either use the library provided by the manufacturer [30] or develop one of its own [17, 18]. We want to try another approach and use high-resolution distance volumes. This will allow us to have collision detection and corresponding forces very fast and in constant time, i.e. independent from the number of surface elements. Memory consumption will be a challenge, but hierarchical approaches seem to be very promising to handle this task.

## 6 Results and Conclusions

Fig. 12 shows a simple, yet typical example for a problem that an engineer often faces. In the original model, seen in Fig. 12(a), the violet component perforates the orange part. Our application detects the erroneous area and visualizes it by marking the region with an alerting texture (Fig. 12(b)). Since it is a clearly defined perforation, it can be resolved automatically as shown in Fig. 12(c). In Fig. 12(d) it can be seen how the engineer performs a stretching operation by dragging the 3D widget proposed in this paper. Erroneous elements are detected interactively and marked with appropriate glyphs, in this case two elements have a bad aspect ratio—yet not too critical for a valid numerical simulation—and one element contains an angle that is too large. Our preprocessing tool is able to repair these elements by locally restructuring the FE mesh and automatically inserting new elements (Fig. 12(e)) without the need for remeshing the whole part. Then the mesh is smoothed by using our relaxation approach, which guarantees that features—e.g. chamfers and sharp edges—are preserved. To compare the new mesh with the starting mesh one can use distance mapping as mentioned before.

We developed these methods in direct cooperation with engineers at the *BMW AG*. Most of our algorithms are no longer prototypes and have been transfered to the commercially available crash worthiness preprocessing tool *scFEMod*. Therefore, the techniques we presented in this application paper are already being used widely by several German car manufacturers and their subcontractors. Due to their simplicity the presented methods have been quickly accepted and the engineers are now able to solve many mesh-related problems on their own without the need for an additional loop through the CAD department. Generation of new variations of existing FE components—e.g. elongations or creation of stiffening corrugations and folds—is now possible using intuitive 3D widgets. Errors in the mesh that might appear during these editing operations are detected reliably and on-the-fly by our algorithms. Relaxation or local element restructuring can then be used to repair critical mesh regions. Although it is not possible to fix *all* kinds of meshing errors, our methods are nevertheless very powerful and successfully produce valid FE meshes in general. The texturing capabilities of standard graphics cards can be used to visualize the differences between various design stages, or to display erroneous regions of an FE mesh. This highlighting technique can be used also to proof that our methods for mesh repair and smoothing do not impair the surface and do preserve important features.

New hardware approaches for immersive workplaces like autostereoscopic displays and haptic input devices are not yet accepted by the industry. On one hand, this might because of the imperfections of these novelties, e.g. most autostereoscopic displays have a rather low resolution, which is additionally halved due to the display method. On the other hand, these devices are still very expensive. Our application is pushing into the direction of a broad usage of such technology and some engineers at *BMW AG* are already considering to equip some workplaces with these devices. We expect to encourage this trend when our application provides full support for haptic editing of FE models in a 3D environment.

In this application paper we demonstrated that there is a need for new modification methods for FE meshes and we presented solutions that can be operated intuitively. Glyphs and textures can be

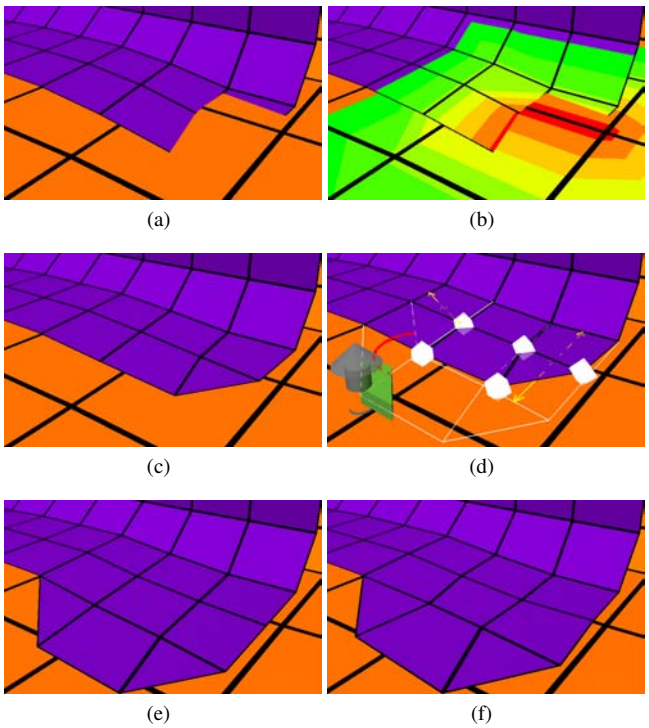



Figure 12: Various stages of repairing and editing a finite element model: (a) original mesh with perforating materials; (b) perforating parts are textured red; (c) perforations and penetrations are removed; (d) user performs a modification and erroneous elements are marked interactively; (e) element errors are fixed automatically; (f) relaxation smooths mesh whilst conserving features.

used to pinpoint erroneous or critical regions. Experience shows that our approach helps to vastly accelerate the development cycle in the automotive industry. It also shows that including modern 3D technology in this process can help to manage the discussed engineering task and that it is the right step towards the future workplace. We will continue our work on this field of research in close cooperation with the engineers, and we will investigate if it is reasonable to transfer our knowledge into other fields of engineering applications.


### Acknowledgements

We would like to thank the *BMB+F* project *AutoOpt* for founding our research and the engineers at *BMW AG* for their cooperation within this project and for delivering insight into today's engineering problems. We would also like to thank Ove Sommer from *science+computing ag* and Reinhold Zöllner for providing many lines of code which are part of this work. Finally, we would like to thank Marcelo Magallón for fruitful discussion and many valuable hints.


### REFERENCES


[1] 3Dconnexion. Spacemouse Datasheet. *http://www.3dconnexion.com/spacemouseplus.htm*, 2004.

[2] N. Amenta, M. Bern, and D. Eppstein. Optimal point placement for mesh smoothing. In *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, pages 528–537. Society for Industrial and Applied Mathematics, 1997.

[3] C. Basdogan, M. Lum, J. Salcedo, E. Chow, S. Kupiec, and A. Kostrzewski. Autostereoscopic and Haptic Visualization for Space Exploration and Mission Design. In *IEEE Virtual Reality Conference 2002*, pages 271–276, 2002.

[4] G. H. Bendels, R. Klein, and A. Schilling. Image and 3d-object editing with precisely specified editing regions. In *Workshop on Vision, Modelling, and Visualization VMV '03*, pages 451–460. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2003.

[5] T. D. Blacker and M. B. Stephenson. Paving: A new approach to automated quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32:811–847, 1991.

[6] S. A. Canann, J. R. Tristano, and M. L. Staten. An approach to combined laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes. In *Proceedings of 7th International Meshing Roundtable*, pages 479–494. Sandia National Laboratories, 1998.

[7] B. D. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, and A. van Dam. Three-dimensional widgets. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 183–188. ACM Press, 1992.

[8] J. Döllner and K. Hinrichs. Object-oriented 3D Modeling, Animation and Interaction. *The Journal of Visualization and Computer Animation*, 8(1):33–64, 1997.

[9] L. Freitag. On combining laplacian and optimization-based mesh smoothing techniques. *AMD Trends in Unstructured Mesh Generation*, 220:37–43, 1997.

[10] N. Frisch and T. Ertl. Deformation Of Finite Element Meshes Using Directly Manipulated Free-Form Deformation. In *Proceedings of Seventh ACM Symposium on Solid Modeling and Applications 2002*, pages 249–256, 2002.

[11] N. Frisch, D. Rose, O. Sommer, and T. Ertl. Visualization and Preprocessing of Independent Finite Element Meshes for Car Crash Simulations. *The Visual Computer*, 18(4):236–249, 2002.

[12] S. Gottschalk, M. C. Lin, and D. Manocha. OBBTree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 171–180. ACM Press, 1996.

[13] C. Grimm and D. Pugmire. Visual interfaces for solids modeling. In *ACM Symposium on User Interface Software and Technology*, pages 51–60, 1995.

[14] J. Harrold, A. Jacobs, G. J. Woodgate, and D. Ezra. Performance of a Convertible 2D and 3D Parallax Barrier Autostereoscopic Display. In *Proceedings of the SID, 20th International Display Research Conference*, 2000.

[15] K. Herndon, A. van Dam, and M. Gleicher. Workshop Report: Challenges of 3D Interaction. *SIGCHI Bulletin*, 26(4), 1994.

[16] K. P. Herndon, R. C. Zeleznik, D. C. Robbins, D. B. Conner, S. S. Snibbe, and A. van Dam. Interactive shadows. In *Proceedings of the 5th annual ACM symposium on User interface software and technology*, pages 1–6. ACM Press, 1992.

[17] D.E. Johnson and P. Willemsen. Six Degree-of-Freedom Haptic Rendering of Complex Polygonal Models. In *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, pages 229–235, 2003.

[18] D.E. Johnson and P. Willemsen. Accelerated Haptic Rendering of Polygonal Models through Local Descent. In *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'04)*, pages 18–23, 2004.

[19] J. Johnson. *GUI Bloopers: Don'ts and Do's for Software Developers and Web Designers*. Morgan Kaufmann Publishers, 2000.

[20] R. Komerska and C. Ware. Haptic Task Constraints for 3D Interaction. In *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, pages 270–277, 2003.

[21] S. D. Laycock and A. M. Day. Recent developments and applications of haptic devices. *Computer Graphics Forum*, 22(2):117–132, 2003.

[22] R. Löhner, K. Morgan, and O. C. Zienkiewicz. Adaptive Grid Refinement for the Compressible Euler Equations. *Accurency Estimates and Adaptive Refinements in Finite Element Computations*, pages 281–297, 1986.

[23] M. E. Mortenson. *Computer Graphics Handbook: Geometry and Mathematics*. Industrial Press, Inc., 1990.

[24] K. Perlin, C. Poultney, J. Kollin, D. Kristjansson, and S. Paxia. Recent Advances in the NYU Autostereoscopic Display. In *Proceedings of the SPIE, Vol. 4297*, 2001.

[25] Perlin, K. AND Paxia, S. AND Kollin J. An autostereoscopic display. In *SIGGRAPH 2000 Conference Proceedings*, 2000.

[26] D. Rose and T. Ertl. Interactive Visualization of Large Finite Element Models. In *Workshop on Vision, Modelling, and Visualization VMV '03*, pages 585–592. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2003.

[27] D. Rose, N. Frisch, T. Ruehr, and T. Ertl. Interaktive Visualisierung neuer Elemente im virtuellen Automobil-Crashversuch. In *Tagungsband SimVis '02, Magdeburg*, 2002.

[28] science + computing ag. Efficient Preprocessing Using scFEMod. *http://www.science-computing.de/en/software/scfemod.html*, 2004.

[29] SeeReal Technologies GmbH. 3D Displays. *http://www.seereal.com/EN/products_cis.en.htm*, 2004.

[30] SensAble Technologies. Phantom Desktop Device. *www.sensable.com/products/phantom_ghost/phantom.asp*, 2004.

[31] S. Stegmaier, D. Rose, and T. Ertl. A Case Study On The Applications Of A Generic Library For Low-Cost Polychromatic Passive Stereo. In *Procceedings of IEEE Visualization '02*. IEEE, 2002.

[32] R. Turner and E. Gobbetti. Interactive Construction and Animation of Layered Elastically Deformable Characters. *Computer Graphics Forum*, 17(2):135–152, 1998.

[33] J. Wernecke. *The Inventor Mentor : Programming Object-Oriented 3D Graphics with Open Inventor, Release 2*. Addison-Wesley Pub Co, 1994.

[34] R. C. Zeleznik, A. S. Forsberg, and P. S. Strauss. Two pointer input for 3d interaction. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 115–120. ACM Press, 1997.

[35] J. Zhang, S. Payandeh, and J. Dill. Haptic Aided Design: A Case Study. In *11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03)*, pages 254–261, 2003.

[36] J. Z. Zhu, O. C. Zienkiewicz, E. Hinton, and J. Wu. A new approach to the development of automatic quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32:849–866, 1991.