scFEMod – The New Preprocessor for Efficient Assembly and Model Validation

Authors:

Ove Sommer [1]
Thomas Ertl [1,2]
Norbert Frisch [2]
Dirc Rose [2]
[1] science + computing ag, Tübingen, Germany
[2] Visualization and Interactive Systems Group (VIS),
Department of Computer Science, University of Stuttgart, Germany

Correspondence:

Ove Sommer
science + computing ag
Hagellocher Weg 71-75
D-72070 Tübingen
Germany

Tel: +49-(0)7071-9457 237
Fax: +49-(0)0711-9457 511
e-mail: ove.sommer@science-computing.de
http://www.science-computing.de

Keywords:
preprocessing, independent finite element meshes, assembly,
interactive mesh correction, model validation, interactive constraint definition

ABSTRACT

This paper presents a new preprocessor for the assembly of independently meshed car body parts. The assembly process gains more and more importance in the preprocessing of crash-worthiness simulations. It is desirable to take simulation results into consideration for construction decisions already in the early phase of the car development process. At this stage, CAD data unfortunately does not contain any information about constraints like spotwelds or adhesive bondings between sub-structures. This lack of data has to be resolved by the simulation engineer. We provide an adequate preprocessing tool for this purpose, called scFEMod.
scFEMod supports the simulation engineer in effectively defining missing constraints such as point, surface, edge, or line links. All of these can be specified interactively with the mouse pointer. Hierarchical data structures guarantee quick and automatic flange detection so that the assembly process is significantly accelerated. Furthermore, scFEMod can be used to replace separately meshed car body parts by variants which then need to be adapted to the adjacent mesh structure. Subsequently, initial perforations and penetrations can be detected, visualized, and selectively removed. Sensor points can be positioned and oriented in order to compare simulation results with those of physical crash tests. scFEMod allows to distribute non-structural masses over all car body parts they are connected to. Finally, the proper assembly of the whole car body model can be validated.

INTRODUCTION

The increasing pace at which new industrial products are brought to the market requires appropriate software tools. In order to be competitive, automotive companies aim to place new, enhanced car models on the market within the shortest possible time. Software plays a key role in car development, where a whole range of software tools are involved. Regarding car body design, the passenger's safety must also be considered, so that crash tests have become both necessary and obligatory. Before performing a real crash test, hundreds of crash-worthiness simulations are computed and analyzed. As a result, the number of physical crash tests is reduced to a minimum, saving time and money.

Extensive use of massive parallel solving algorithms in crash-worthiness simulations has enormously reduced the pure computation time. The number of simulation runs with slightly modified models has been increased. Postprocessors, which provide batch control, help to shorten the time needed for the analysis of simulation results. Subsequent to the simulation run a report containing result tables and images from standard viewpoints is generated automatically. This means, more time in proportion to simulation and analysis has to be spent for the preprocessing of the simulation input data sets. Since solvers are now able to map the behaviour of constraining elements like point, line, or surface links there is no need for any longer using homogeneously meshed car body models. Instead, independently meshed car body parts are assembled into fully defined simulation models. As a great advantage single car body parts can be replaced by variants without matching connecting nodes. Nevertheless, assembled substructures often have to be adapted in flange regions where adjacent car body parts converge. One task during the preprocessing of crash simulation input data assembled by independently meshed car body parts is to detect and remove all initial perforations and penetrations, because such model errors lead to initial forces which falsify the simulation results. Furthermore, separate meshes need to be connected to each other by constraining elements. Especially in an early development phase the information about car body part connections is not provided by the construction data in the CAD model. In order to quickly deliver results that influence the car body design with respect to passive safety, the simulation engineer has to overcome the lack of information by defining missing constraints on the basis of the finite element model.

This task requires a preprocessing tool that allows the user to effectively detect potential flanges, to validate mesh correctness and to add link element data, all interactively. Our preprocessor *scFEMod* has been developed in close collaboration with the crash-simulation division of the BMW Group. Its prototype was part of the AutoBench project supported by the BMBF (German Ministry for Education and Research). The visualization module of scFEMod currently is based on SGI's high-level graphics API OpenGL Optimizer / Cosmo3D. The GUI is part of our sc.iViz framework which is also used for PowerVIZ, the standard postprocessor for CFD simulation results computed by Exa PowerFLOW. As presented in the following sections, scFEMod allows to merge partial model data into one data deck. It detects, visualizes and removes initial perforations and penetrations of independently meshed car body parts. scFEMod's hierarchical data structure enables interactive detection of potential flanges, which in turn provides an automatic path generation during the definition of link elements along flanges. Furthermore, non-structural masses can be taken into consideration by distributing their masses to connected finite element structures. Sensor points can be interactively positioned in order to gain results which are comparable to those of physical crash tests. Finally, during model validation scFEMod can be used to highlight erroneous link elements and potential flange areas that are insufficiently connected. All these functions accelerate the development process, avoid wasting computation time and, therefore, reduce

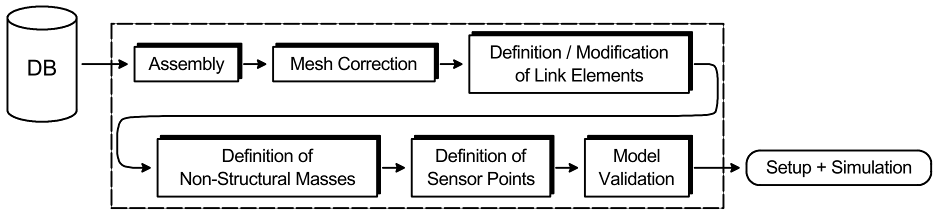

DB
Assembly
Mesh Correction
Definition / Modification of Link Elements
Definition of Non-Structural Masses
Definition of Sensor Points
Model Validation
Setup + Simulation

Figure 1 scFEMod accelerates the preprocessing of independently meshed car body parts. Specialized functionality allows the engineer to interactively detect / resolve mesh errors (perforations / penetrations) and to check / define link elements (spotwelds, line, or surface links). A final model validation helps to increase the plausibility of the simulation results.

development costs.

Mesh error correction

By means of the meshing step, parametric surfaces of the CAD data are transformed into a discretized finite element mesh. Since the whole car body model consists of hundreds of independently meshed car body parts, this process may introduce '*initial perforations*/*penetrations*'. Fig. 2 points out the initial penetrations as points, where one discretized surface is closer to another surface than the specified material thickness (left and right circle). Areas where elements intersect are called perforations (mid circle). Initial penetrations cause initial forces in the simulation task, which falsifies the simulation results. Initial perforations between two or more finite element meshes lead to unpredictable simulation results. Therefore, it is important to detect and remove initial perforations/penetrations during the pre-processing of the simulation input data deck.

Initial Perforation Removal

Initial perforations in the simulation input deck represent an even larger problem then initial penetrations. While the solver is able to remove initial penetrations globally and without any user control there is no such way to eliminate initial perforations. Therefore, we developed an algorithm for detection, visualization and removal of perforations.

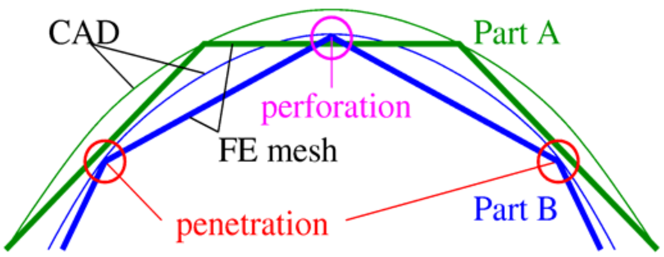


Figure 2 The assembly of independently meshed car body parts may cause 'initial penetrations' which influence the simulation results in an undesirable way. Perforations (mid) can occur in places where the finite element meshes interpenetrate.

In order to efficiently detect perforating elements, scFEMod makes extensive use of hierarchical sub-structuring of the finite element model. Techniques presented by Gottschalk et al. [5] are applied to provide feedback within interactive rates. Regions of perforating elements are emphasized by applying a one-dimensional RGBA texture that by default colors the perforating nodes red (see Fig. 3, left). The environment of the perforating node is also highlighted by the texture, so that the perforating region is visible when looking from either side of the perforated part. In a

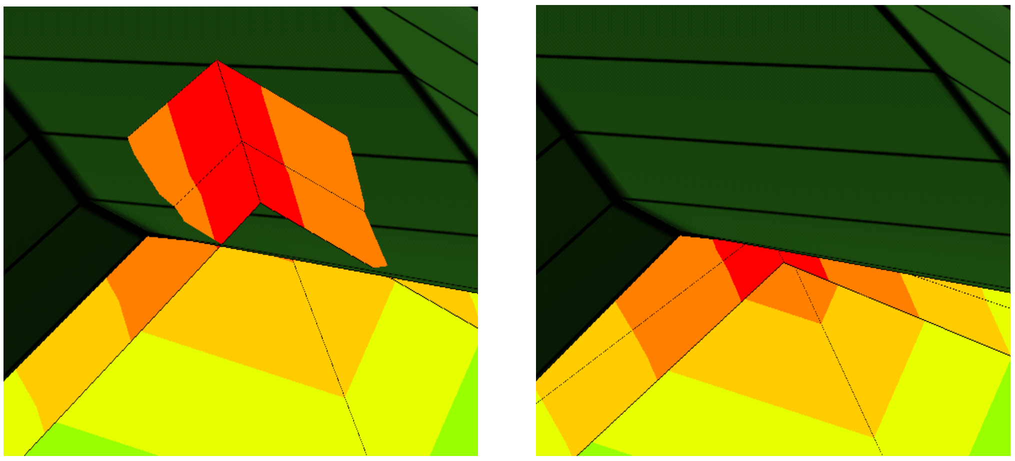


Figure 3 Example of an initial perforation (left). The perforating nodes have been modified by the perforation removal algorithm (right). Here, the color coding was preserved for illustration only.

next step, perforations are removed (see Fig. 3, right). The user is able to define the ‘perforating’ side of the perforating car body part. During an iterative process each node that has been classified as perforating is projected on the perforated surface. Subsequently, it is moved slightly behind that surface. After all perforating nodes have been moved to the other side of the perforated surface, they are still penetrating. In the next step of mesh correction initial penetrations are removed.

Initial Penetration Removal

In order to detect penetrating vertices that are positioned too close to an element of another car component, the minimal distance of each vertex to each finite element of another car body part has to be calculated. Again, this task is accelerated by using the hierarchical sub-structuring of the finite element model.

After detecting all initial penetrations they are highlighted like perforating regions. The engineer can now mark several car parts as '*(un-)modifiable*' before the removal algorithm is started. Thus, the user gains full control over the set of model parts to be modified. This is very important for the replacement of individual components by variants because the node coordinates of the variants should be adapted while the rest of the car body model stays fixed. During initial penetration removal the penetrating nodes of modifiable meshes are moved along the calculated force vector in a number of iterations until the initial force vanishes. Penetrating flange regions remain as smooth as before the removal.

Mesh Relaxation

After the removal of initial perforations the finite element mesh might be distorted. For best numerical results, the finite elements should be mostly equilateral and of similar size. This goal can be achieved through an iterative relaxation algorithm, which is optionally started after the perforation removal. Like in a mass-spring model we consider the nodes being connected by springs instead of edges. Then we move the nodes iteratively by small amounts in the direction of the resulting force with the constraint that each node stays on the initial surface. Corners are not moved, while nodes at the edges or nodes at significant angles are moved only along that edge.

Interactive Definition of Constraints

Spotwelds

Setting

Spotwelds are the prevalent links between independently meshed car body components. Information about link elements is part of the final CAD data, but in an early development stage this kind of information is not (fully) available. Additional link elements need to be defined and some of them eventually need to be modified or deleted even in advanced stages by the simulation engineer. The ability to define spotwelds directly on the finite element data reduces the preprocessing time of the crash

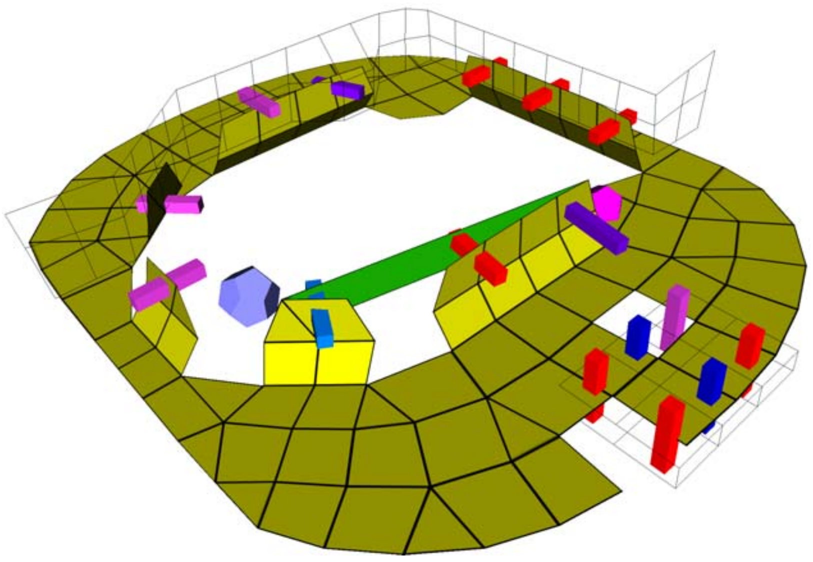


Figure 4 Various valid and erroneous spotwelds, visualized in different colors as cuboids and dodecahedra.

simulation. The detour through the CAD department in order to modify link elements is cut short during the iteration loop.

The visualization of spotwelds by means of small, scalable red cuboids has proven good in practice. Erroneous spotwelds, e.g. spotwelds with missing or inappropriately positioned assembly parts, are visualized in different colors and/or geometries in function of the error type (Fig. 4).

Since a car contains thousands of spotwelds, it can be tedious to edit spotwelds one by one. For this reason, scFEMod provides a facility to define an entire spotweld line at once by specifying the start and end points of the line with the mouse pointer. This generates a set of spotwelds along a straight line. The new spotwelds are equidistantly positioned along this line. If the assembly part is curved, the straight line is projected onto the surface to find proper spotweld positions.

Curved spotweld lines

Obviously, not all spotweld lines in a car will be straight, neither can each curved line be obtained by projection of a straight line onto a car component. Therefore, a feature for generating curved spotweld lines was implemented. A first idea was to define curved lines by means of interactive spline curves. A drawback of this approach is that for the user spline-curves are hard to position exactly on the middle line of the flange. In the time required to position the spline control points the user could also position the spotwelds. Therefore, we opted for a new approach: As spotwelds are usually positioned along flanges, a flange recognition algorithm was developed.

The flange detection algorithm for curved spotweld lines and surface links is realized on a per-element basis. Each finite element either is a flange element or it is not, in function of the distance and the angle of this element with regard to the nearest element of the corresponding component. A distance criterion is also checked. Finally, we observe that flanges have a considerable length but only a limited width. The spotweld positions on the mid-line are computed in function of the desired distance between adjacent spotwelds.

The less trivial steps are finding the path and detecting the mid-line. A special challenge is to find a path even if the flange is interrupted by small gaps and corrugations (Fig. 5). The gaps and corrugations can split the flange area and make it impossible to find a path consisting exclusively of flange elements. In this case, the path should lead over or beneath the obstacles. The algorithm performed well with respect to both result quality and performance. Even large flanges are detected within interactive rates.

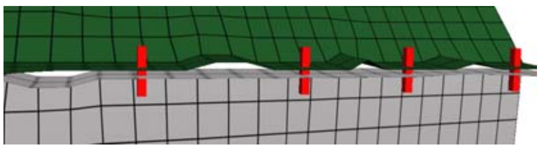


Figure 5 scFEMod's automatic flange detection even manages irregularities like corrugations.

This kind of spotweld creation is performed for a partial flange by specifying start and end point of the element range that should be connected. In addition it is possible to connect the full flange by just two mouse clicks – one for each of the connected car body parts.

Line Links

In addition to spotweld lines, welded joints or line links were introduced to some simulation software. Line links are 1-dimensional linear elements. Three scenarios can be defined for this link element type: edge-edge-connection, edge-surface-connection, and surface-surface-connection along a line. For visualization we use a triangular profile and shift it along the link line. This guarantees good visibility from arbitrary viewpoints. We use the same colors as for spotwelds to clarify the relationship to that link type. An example can be seen in Fig. 6.

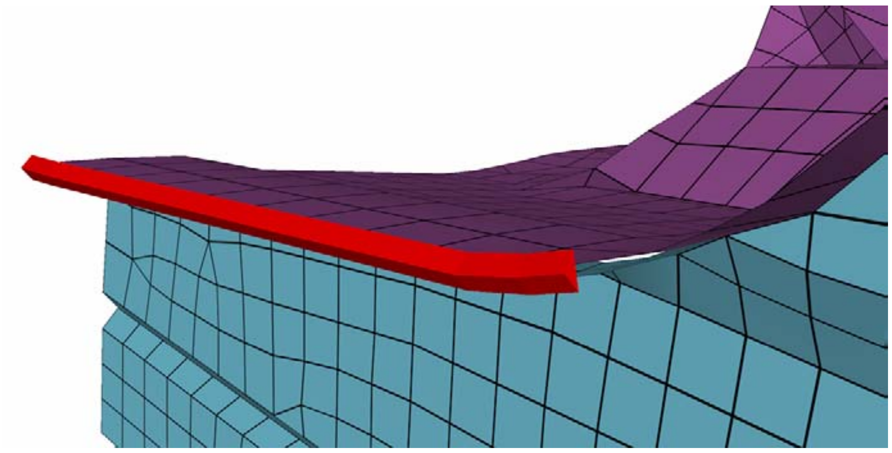


Figure 6 Visualization of line link elements.

When creating a new line link, the user interaction is the same as for spotweld lines. The underlying algorithm, however, is different. The complicated flange detection is unnecessary because line link elements are restricted to the boundary of one or both compound parts. In the vicinity of the selected start and end coordinate the nearest boundary point of both material parts is searched for. The line link elements are gained by projecting the boundary nodes between the start and end point on the second compound material or its boundary line, whatever is nearer. Hereby the shorter of the two possible paths is chosen. As with the other bonding types the actual line link positions are calculated by averaging the corresponding original and projected coordinates. Finally, the elements' distance and normal deviation are verified and illegal links are removed.

Surface Links

Another new bonding technique is the usage of surface links. In contrast to spotweld lines, surface links entail a different way of representation of the bonding agent. However, the user interaction was kept as simple as for the other link types.

Visualization of surface links

The problem of all link types is that they are hardly visible from outside since they naturally are comparatively small and are placed between two or more assembled parts. There are two different ways to solve this problem, each one with its own pros and cons. One solution is to use transparent components. Thereby, the bonding agent and the counterpart can be easily seen. Though, with lots of surface link elements this may be confusing, due to many transparent components, which is a substantial argument against this solution. The other possibility is to illustrate surface links by a thickened representation. The advantage is that the surface link elements can be easily discovered because they stick out of the connected component's surface. However, this representation needs to be semi-transparent to see the connected car body surface.

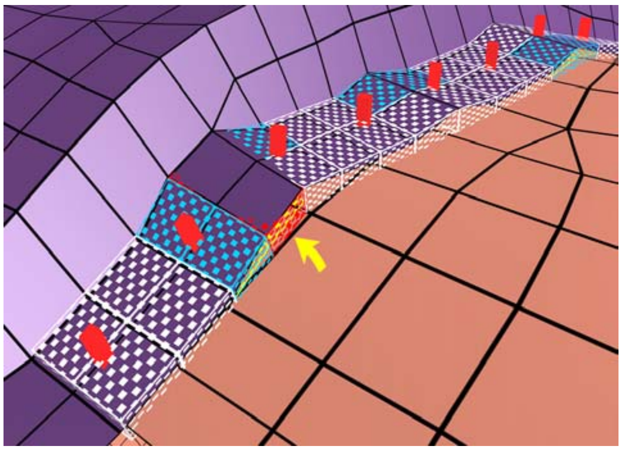


Figure 7 Example of surface links in combination with spotwelds. Erroneous links are color-coded.

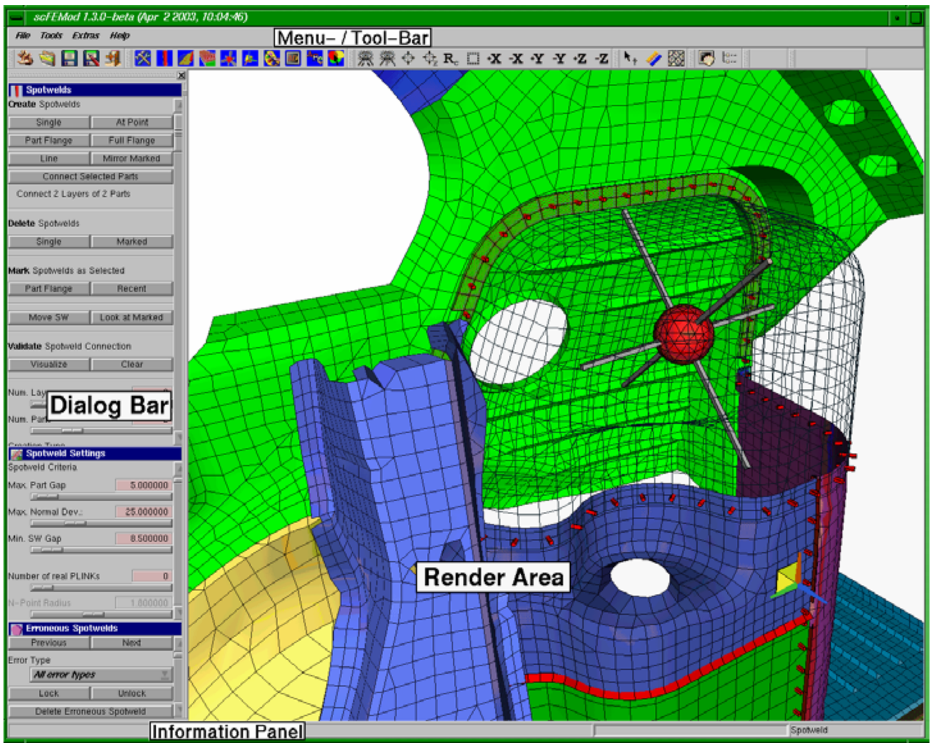
Both solutions have in common that they use transparency in some form. We have chosen the second approach because the first one may conflict with mapping of scalar properties on the finite element structure and the second one yields the best clarity when visualizing many surface links. scFEMod represents surface link elements as three-dimensional hexahedrons. They have a constant thickness in order to represent the maximum allowed distance between assembled parts for this specific link. To improve the conspicuousness of the surface link representation we use a checkerboard look-alike texture alternating opaque white and full transparency. The joined components can easily be seen through the transparent parts. Thus, it is guaranteed that spotwelds are visible when used in combination with surface links. The strips of surface link elements are closed towards the outside (Fig. 7). This provides a better impression of the link boundaries.

Creation of surface link elements

When creating surface link elements we use the same techniques as for spotweld lines. Both user interaction and internal algorithms are similar or inherited. The engineer picks one assembly part and defines the length of the surface link on the second component with only three mouse clicks overall. The initial width of a suface link element is determined by a scalable maximum distance from the detected mid-line of the flange. All elements within that area meeting the flange criteria are used to build up the mid-surface of the actual bonding agent. Therefore, we project the eligible nodes of the first car body component onto the corresponding elements of the second one. The averaged coordinates of original and projected nodes define the mid-surface.

Error detection and validation

When surface link elements are created or read from a simulation input deck they are checked against several error criteria. Creation of erroneous link elements during user interaction is thus prevented. If the input deck already contains surface links elements which do not comply with the user specified requirements their representation is color-coded depending on the error type. Failures like missing projection points, contact parts that are too far apart, a normal deviation that is too large, or the lack of a compound material can also occur for other link types. Therefore, the same color is used for different link types to indicate a corresponding error. Additionally, the actual surface link element is rendered in yellow in the middle of the volumetric bonding representation. This facilitates the engineer's decision on what to do. Generally, he will simply delete the erroneous elements. In Fig. 7 you can find surface link elements with bonding counterparts that are too far apart marked red (naturally, the bonding surface is partially hidden by one of the counterparts in this case) and those with a normal deviation that is too high marked blue. The yellow link elements in the middle are pointed out by an arrow. Elements with the same error type are merged to areas using neighborhood relationships. Then we calculate a camera perspective pointing to the central point and save it into a list. This way the engineer is able to head towards erroneous areas and easily navigate to the next spot.



Max. Part Gap 5.000000
Max. Normal Dev.: 25.000000
Min. SW Gap 8.500000
Number of real PLINKs 0
N-Point Radius 1.800000

Figure 8 The Qt-based scFEMod main window is user configurable. It consists of a menu bar, a tool bar, and a dialog bar. The user obtains feedback through an information panel. The model is visualized inside the render area. All functions can also be used via user configurable key

Usability

scFEMod is embedded into the sc.iViz framework. This framework has been developed especially for high performance visualization applications. Its graphical user interface (Fig. 8) is built on top of Qt — an API that is available on multiple platforms like several Unix systems, Linux and Windows. The structure of the menu and the tool bar is fully configurable by the user. Even the key and mouse button bindings can be specified in a Python script. This allows the user to harmonize the interaction control of scFEMod with other applications he is using. The sc.iViz framework supports scripting and recording of processed interactions. It is possible to track actions and replay them, for example in a demonstration. The framework provides for transmission of events, which can be used for cooperative work between two or more scFEMod users on different machines.

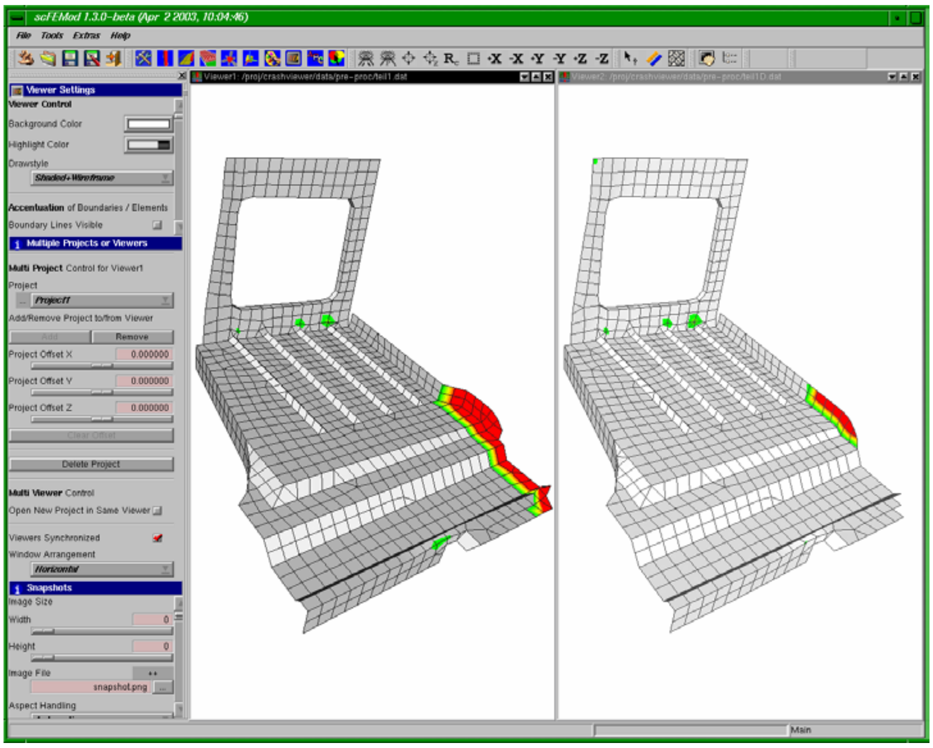


Figure 9 Using multiple synchronized viewers the engineer is able to compare different variants of corresponding car body parts. In addition, differing regions can be emphasized in order to pinpoint model deviations.

The multi-viewer functionality of scFEMod allows the user to directly compare different data sets. Two or more viewers can be placed inside the main window. The viewer cameras can be synchronized in order to provide the same viewpoint for each model. Together with a special mapping functionality that highlights deviating model areas this technique delivers insight into what has changed between two successive model states.

Summary and Conclusions

We presented our new preprocessor scFEMod, which provides specialized techniques to speed-up the preprocessing of independently meshed car body parts for crash-worthiness simulations. It allows the engineer to interactively eliminate mesh errors like perforations and penetrations. Missing link information can be added effectively during the early development phase. Already existing link information can be checked against user specified criteria in order to detect and correct erroneous link elements. Non-structural masses and sensor points can be added using simple mouse control. Applying scFEMod's model validation functionality the user is pointed to potential flanges that seem to be insufficiently connected.
The highly configurable sc.iViz framework provides large flexibility and, therefore, reduces the period of vocational adjustment. Furthermore, the multi-viewer capability helps to manage the ongoing change of the car body model. Finally, the Python interface provides scripting control. Thus, standardized preprocessing steps like model validation can be handled in batch processing.

References

1. BAREQUET, G., KUMAR, S. (1997) "Repairing CAD Models", IEEE Visualization '97 Conference Proceedings, IEEE Computer Society Press, pp. 363-370.
2. CAMPAGNA, S. (1998) „Polygonreduktion zur effizienten Speicherung, Übertragung und Darstellung komplexer polygonaler Modelle", PhD thesis, University of Erlangen-Nuremberg, Germany.
3. CHIN, F., SNOEYINK, J., WANG, C.A. (1995) "Finding the Medial Axis of a Simple Polygon in Linear Time", Proc. 6th Ann. Int. Symp. Algorithms and Computation (ISAAC 95), Lecture Notes in Computer Science 1004, pp. 382-391.
4. FRISCH, N., ERTL, T. (2000) "Embedding Visualization Software into a Simulation Environment", Proc. of the Spring Conference on Computer Graphics, Bratislava, pp. 105-113.
5. GOTTSCHALK, S., LIN, M., MANOCHA, D. (1996) "OBB-Tree: A hierarchical structure for rapid interference detection", SIGGRAPH '96 Conference Proceedings, ACM SIGGRAPH, Addison Wesley, pp. 171-180.
6. SOMMER, O., ERTL, T. (2000) "Geometry and Rendering Optimizations for the Interactive Visualization of Crash-Worthiness Simulations", Proc. Of SPIE: Visual Data Exploration and Analysis VII, vol.3960, pp. 124-134.