

---

# Generation of Mesh Variants via Volumetrical Representation and Subsequent Mesh Optimisation

Katrin Bidmon and Thomas Ertl

Visualization and Interactive Systems Group, University of Stuttgart  
Universitaetsstrasse 38, 70569 Stuttgart, Germany  
(bidmon|ertl)@vis.uni-stuttgart.de  
<http://www.vis.uni-stuttgart.de/>

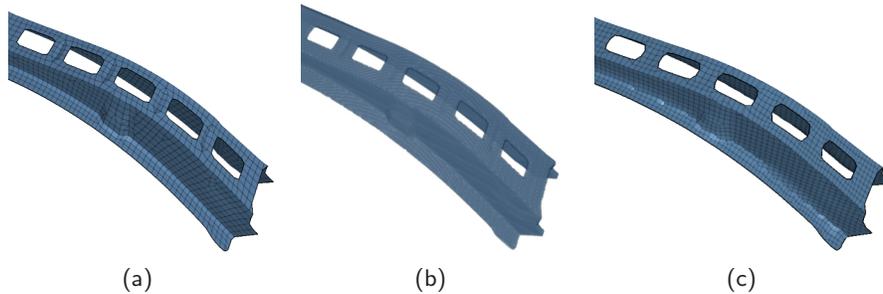
**Summary.** Having reliable finite element (FE) meshes is one of the basics of reliable FE simulations. As development times i.e. in the car industry are expected to decrease, engineers need to edit and optimise FE meshes without access to the underlying CAD geometry. If meshes are not only locally effected by the editing operation, simple mesh optimisations such as mesh relaxation or local remeshing are not sufficient to make the mesh suitable for numerical simulation again and global remeshing is needed. To avoid the traditionally used time-consuming remeshing strategy, we developed a tool to remesh an FE surface model – taking into account the needs for good FE meshes – via volumes. We first voxelise the surface and then generate a new quad mesh via isosurface extraction and subsequent mesh optimisation. This method provides the opportunity to directly couple editing operations on the volumetrical representation with the remeshing procedure.

**Key words:** FE mesh, remeshing, warping, optimization, voxelization, isosurface extraction

## 1 Introduction

Virtual prototyping more and more replaces real mock-ups and experiments in industrial product development such as in automotive industry. The fast increasing processing power of modern computers together with more and more efficient algorithms allows to calculate complex non-linear and highly dynamic processes such as crash worthiness simulations within few days. So, expensive car prototypes are increasingly replaced by virtual simulations based on Finite Element Analysis (FEA).

Usually the car parts are designed as analytical surfaces using computer aided design (CAD) and have to be transformed into an FE mesh for simulations. For this



**Fig. 1.** Surface in different representations: (a) original FE mesh representation, (b) volumetric representation and (c) reconstructed FE mesh representation

purpose several meshing algorithms have been developed (e.g. [BS91, ZZHW91]), but most of them are tailored to fit a specific kind of simulation and to preserve a special surface property. So, a lot of expert knowledge and manual intervention is still needed to adapt those meshes to fit the prerequisites of a reliable numerical simulation. A recent improvement in simulation algorithms made it possible to mesh each car component individually, which was a huge step forward in giving the engineer more flexible tools and to speed up the development cycle. Now single car parts can be exchanged or varied without the necessity of remeshing the whole car model. This fact induced the desire for having tools to directly manipulate and edit the surfaces in the FE mesh representation (e.g. [BRE04]) instead of going back to the CAD department for each change during the development. If the changes in the surface only affect a small region in the mesh, the mesh properties required for simulation can be regained by local optimisation as relaxation or – in more serious cases – local remeshing. But if the mesh gets deformed too much during editing or manipulation, these repairing mechanisms will be insufficient and remeshing is needed. To avoid the frequent and thus time-consuming way back to CAD, the engineers need a fast method to generate new meshes suitable for numerical simulation.

Due to the potential possibility to directly combine this necessary remeshing with coarse-scale modifications of the surface in an intermediate step, we decided for a volume-based approach to remeshing (see Fig. 1): the surface is voxelised as described in Sect. 3, then reconstructed by isosurface extraction taking into account the desired properties of FE meshes (see Sect. 4). As isosurface extraction usually leads to triangle meshes, but quad meshes are needed for our structural mechanics simulations, the triangle mesh is converted to a quad dominated mesh in a subsequent processing step described in Sect. 4.2.

FE models are not very tolerant concerning mesh deformations, whereas generating quad meshes on curved surfaces always introduces warping: the problem that the vertices of a quad are not bound to lie on a common plane as vertices of a triangle inevitably do. But warping, which also can arise in the original mesh resulting from CAD, leads to huge problems during the numerical simulation and must be eliminated from the mesh before the simulation. This process is described in Sect. 5.1 in more detail. Additionally some quads in the mesh might be oriented diagonally to the other quads (see Fig. 9(a)) which may also lead to numerical problems during

simulation. Thus, these quads need to be detected and removed by either splitting them into triangles or merging and re-splitting them together with neighbouring elements. This approach is explained in Sect. 5.2.

The paper presents some previous work in the following section and results and some conclusions in Sect. 6, as well as an outlook to enhancements planned for future work.

## 2 Related Work

There is a lot of work and publications about voxelisation, often bound to special needs of the application they were developed for. An algorithm fitting all interests at the same time does not exist. Especially for closed surfaces and real volumes many sometimes simple but powerful algorithms have been invented – such as binary volumes (e.g. [Kau87, HYFK98]) or distance volumes (e.g. [Gib98, VKK<sup>+</sup>03]) – some also taking into account special requirements like accuracy of surface details [Sra01, HLC<sup>+</sup>01]. A nice overview of the voxelisation literature is given in [COK95].

In our case we do not have closed surfaces but bounded ones, which leads to new problems: how to handle the boundary, how to treat distances measured to the boundary. Using binary volumes leads to problems with thin surfaces and the surface's thickness would have to be increased to get a hole-free voxelisation. Additionally it would lead to double surfaces during reconstruction, as the surface between "inside" and "outside" would be extracted – which is of no use for our purpose. So we decided for signed distance volumes, to be able to extract the zero isosurface as a single surface. Another approach, presented in [KBSS01], is to directly save the points where voxel and surface intersect each other instead of storing a scalar distance value per voxel.

Consequently, there is also a lot of related work in the field of isosurface extraction from volume data. Starting from the fundamental *Marching Cubes* algorithm [LC87], many variants of this method exist: the algorithm has been simplified to *Discretized Marching Cubes* [MSS94] where each intersection of the marching cube with the isosurface is set to the middle of the respective cube's edge. In addition, many extensions and improvements have been applied to the Marching Cubes algorithm in order to solve ambiguous cases and maintain topological properties (e.g. [Che95] and [LLVT03]), geometrical properties like sharp edges (e.g. [KBSS01]), or to get smoother surfaces (e.g. [LB03]).

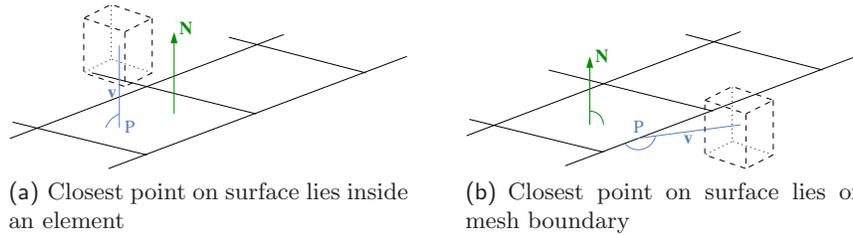
Marching Cubes and its derivatives usually generate triangle meshes. In FE simulations we need quad meshes or at least quad dominated meshes with only few triangles since triangle meshes lead to unstable numerical simulations in FEA. So we have to extract a quad mesh from the volume, similar as in the *Dual Marching Cubes* [Nie04] where the dual grid is used to enhance the triangle mesh. Generating a mesh dual to the one generated by a Marching Cubes variant leads to new problems on bounded surfaces as data might be lost at the boundary (see Sect. 4.2).

In the following sections we describe how we combined the existing body of knowledge to make it available in the context of Finite Elements. By implementing our algorithms into the commercially available preprocessing tool *scFEMod* [sci04] we made this functionality available for productive use in the CAE departments of major German car manufacturers.

### 3 Voxelisation of the FE Mesh

Generating a volume representation of the FE mesh is done by distance calculation. The voxel size (uniform in each direction respectively) depends on the elements' globally minimal edge length to take fine surface features into account. When the volume dimensions are calculated, the shortest Euclidean distance to the FE mesh is computed for each voxel. If the distance to the surface is larger than a specified threshold (e.g. more than the doubled diagonal length of the voxels), the voxel value is set to 255, representing an "invalid" value that should not be taken into account for volume rendering or later isosurface extraction.

The point  $P$  on the surface having shortest distance to the voxel might have different properties depending on its location in respect to the bounded mesh (see Fig. 2): in the ideal case  $P$  lies inside an element (Fig. 2(a)) and the distance vector  $\mathbf{v}$  is perpendicular to the element. Another possibility is, that the point  $P$  lies on an element edge or vertex being part of the surface boundary (see Fig. 2(b)). In this case the distance stored in the voxel would lead to wrong assumptions during volume rendering or surface reconstruction.



**Fig. 2.** Two possible locations of the closest point on the surface in respect to the voxel's position

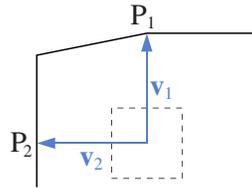
In order to avoid these problems voxels are skipped and their value is set to invalid if the angle enclosed by the corresponding distance vector  $\mathbf{v}$  and the normal of the element the edge belongs to is larger than a specified threshold ( $\mathbf{v} \cdot \mathbf{N} > \varepsilon$ ). In the other cases  $P$  lies on an inner edge or an inner vertex and the distance vector is compared to the average of the neighbouring elements' normals in the same way.

If a shortest distance is measured from the same voxel to an edge as well as to an element, the element gets favoured (see example in Fig. 3).

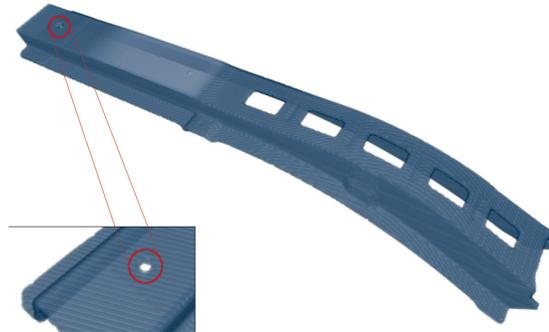
In Fig. 4 an example of a voxelised FE mesh is given, showing that this method preserves surface features such as small holes.

### 4 Surface Reconstruction

In order to retrieve a meshed surface, an isosurface has to be extracted from the volume. The Marching Cubes algorithm computes for each cube the zero points along the edges by linear interpolation between the values at the cell's vertices. If one vertex value is negative and one positive, there has to be a zero point on this edge



**Fig. 3.** In case a shortest distance from one voxel is measured to an edge ( $P_1$ ) and to an element ( $P_2$ ), the element gets favoured.

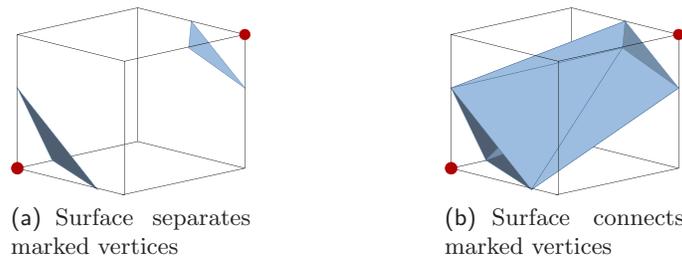


**Fig. 4.** Volumetric representation of the surface – preserving small surface features

which means an intersection of the cube with the surface. As we use signed distance fields to generate the volume, the zero isosurface is the surface we are looking for.

#### 4.1 Isosurface Extraction

The classical Marching Cubes look-up table contains some ambiguous configurations that may lead to topological problems like holes in the mesh. If there are e.g. only two negative values on the vertices connected by an inner diagonal of the cube, the surface inside the cube may separate or connect these two vertices as depicted in Fig. 5.



**Fig. 5.** Two possible triangulations with the same configuration of values on the vertices

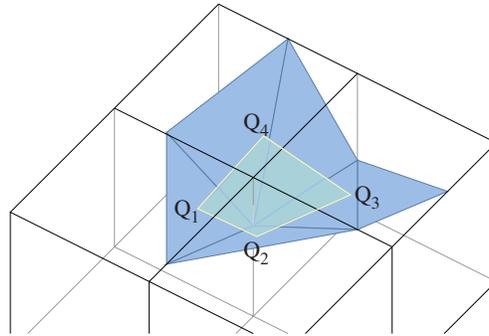
Choosing the wrong configuration would lead to problems when connecting the generated triangles to the ones of the neighbouring cubes, possibly generating cracks in the triangle mesh. The development of enhanced look-up tables taking into account and solving these ambiguities, e.g. in [Che95], allowed algorithms – such as [LLVT03] – preserving the original topology and getting rid of unintentional holes in the mesh. This algorithm, an enhanced version of the *Marching Cubes 33* [Che95], not only takes into account the vertices' values, but also interpolates values at inner points or on surfaces to distinguish between configurations. This is important to ensure topologically correctness. To provide consistent triangulations neighbouring configurations have to be additionally considered.

As avoiding cracks in the mesh is among the most important prerequisites for FEA, we decided to base our meshing method on this algorithm: Using the look-up table of [LLVT03] we determine the connectivity and so the triangulation within each cube. But as a triangle mesh is not suitable for FE applications, we have to convert the obtained mesh into a quad mesh.

## 4.2 Quad Mesh Generation

In [Nie04] a triangle mesh is smoothed by applying the dual operator to the mesh two times. Applying this duality only once, we automatically obtain a quad mesh: the triangles are always constructed within a cube and connected to the triangles inside neighbouring cubes.

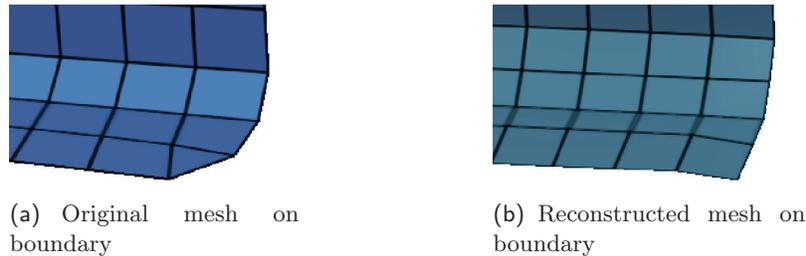
As a first step we consider only cubes of the inner surface, thus not containing triangles of the surface's boundary. Imagine four cubes connected in one conjoint edge. Connecting the centres of these four cubes leads to a square. This method can now be applied to the triangulation obtained as described before. Considering



**Fig. 6.** Construction of a quad element dual to the triangle mesh generated by Marching Cubes

connected triangles within one cube as a single polygon, we can connect the centre of the polygon ( $Q_i$ ) with the ones of the neighbouring cubes, as depicted in Fig. 6, and get quads instead of triangles. Even if there is more than one polygon within a cube, the statement holds for each of these surfaces respectively. The only problems arise on the surface boundary as the quad mesh is slightly smaller than the triangle mesh,

due to the duality construction. To avoid that and to retain the original surface boundary, polygons are added and subsequently split or merged to quads. This method is a trade-off between keeping the shape of boundary lines and exchanging triangles in boundary corners by quads (see Fig. 7).



**Fig. 7.** Boundary reconstruction

### 4.3 Feature Line Preservation

One of the main problems with isosurface extraction is the fact that sharp feature lines are being wiped out. To avoid this we explicitly treat these feature lines. When generating the volume representation, feature lines are detected depending on the angle between two neighbouring elements and the voxels crossed by the lines are marked. Since we use the dual grid described in Sec. 4.2 our new vertices are located in the centre of the polygon extracted by Marching Cubes within each voxel. So, if the voxel is one of those originally crossed by a feature line, we move the new voxel to the closest point on the extracted feature line. If there is a feature point within the voxel, e.g. a corner in the surface boundary, the new vertex is moved to this point. As the vertices are being moved only slightly within the voxel the shape of the elements is only little affected but the feature lines are well preserved (see Fig. 8).

With these methods applied the resulting mesh already looks pretty appropriate for FE simulation. Nonetheless the quad mesh still lacks some enhancements – described in the following section – to fulfil the demands of Finite Element Analysis.

## 5 Quad Mesh Optimisation

Elements in FE meshes have to satisfy special properties to be able to be used for simulation. The mesh should consist of quads shaped as close to squares as possible, oriented all in the same direction (see Fig. 9). If needed few triangles can be included. Extreme angles as well as warping influence the simulation speed and result. Since the mesh obtained by the algorithm described above might lack these prerequisites in some elements, these properties have to be checked.

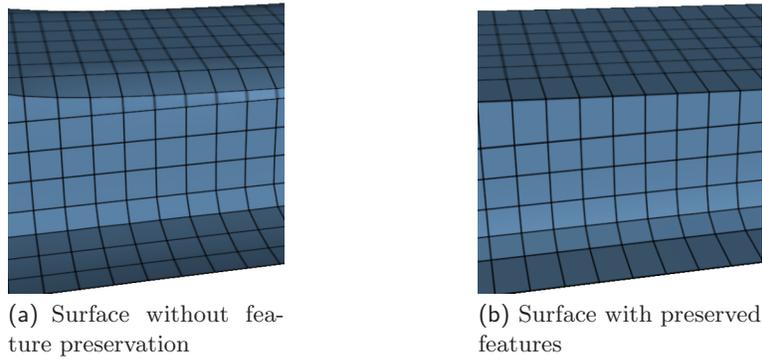


Fig. 8. Preservation of surface feature lines

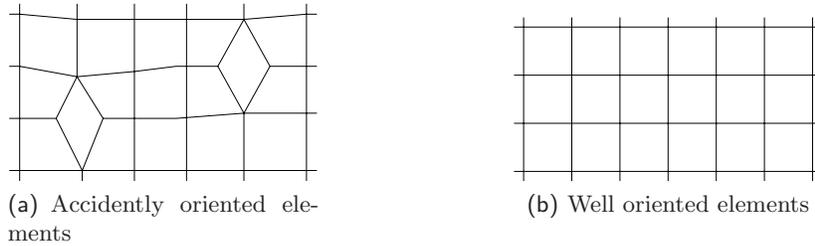


Fig. 9. Orientation of quads in the FE mesh

### 5.1 Warping Removal

Compared to triangle meshes quad meshes bear the risk of warping. This might occur already during construction or as a result of manipulations of the elements.

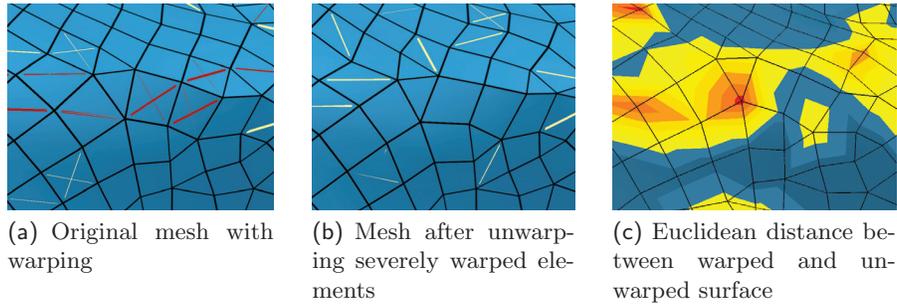
Due to the quad construction described in Sect. 4.2 our mesh usually is warped in some elements. To make these elements suitable for simulation again they have to be detected and unwarped. Our meshing tool allows to mark warped elements by colour as seen in Fig. 10(a). To detect warping each quad is divided into two triangles the normals  $N_1$  and  $N_2$  of which get compared. If the angle is above a specified threshold, the diagonal is marked in dark/red, if it is below the threshold but the element still warped this is marked in light/yellow.

Both possible triangulations of a quad are examined and the direction corresponding to the wider angle between the two normals is the one considered further on during unwarping. To remove warping, all elements are analysed and if the angle exceeds the threshold the element is marked to be unwarped. Using the average of the two normals the linear smoothing plane through the middle point  $P$  of the four vertices  $\mathbf{v}_i$  is calculated:

$$n_1x + n_2y + n_3z - d = 0 \tag{1}$$

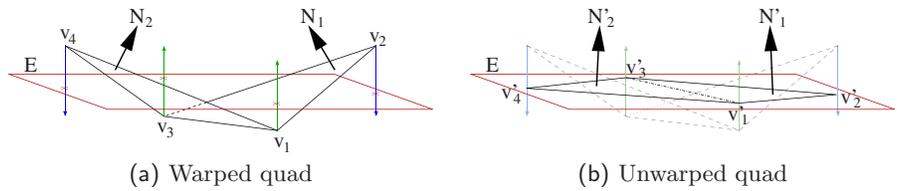
with  $\mathbf{N} = (n_1, n_2, n_3)^T = \mathbf{N}_1 + \mathbf{N}_2$ ,  $d = \sum n_i p_i$  and  $P = (p_1, p_2, p_3) = \sum \mathbf{v}_i$ . Then the vertices' new coordinates are calculated projecting them vertices onto this plane

$$\mathbf{v}'_i = \mathbf{v}_i + t\mathbf{N} \tag{2}$$



**Fig. 10.** (a) Highlighting and (b) unwarping of warped elements: slight warping coloured light/yellow, severe warping marked dark/red. (c) Comparison between warped mesh and unwrapped mesh using distance mapping. Euclidean distance is colour-coded: yellow/light < 0.01 mm to red/dark up to 1 mm

with  $t = (d - \mathbf{N} \cdot \mathbf{v}_i) / \|\mathbf{N}\|$ .



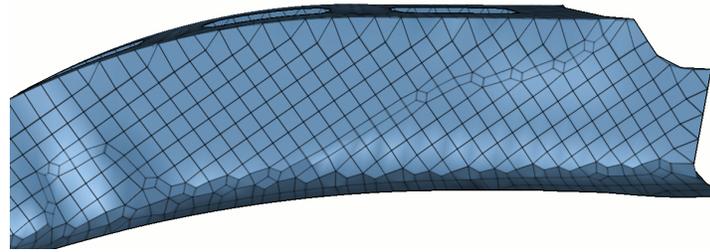
**Fig. 11.** Warping removal: Vertices of warped quads are projected on the linear smoothing plane

As neighbouring elements get affected by this treatment, the new coordinates are set no earlier than all elements are checked and the new coordinates are the average of the calculated coordinates for each vertex respectively. This calculation is repeated iteratively until no element is warped more than allowed by the specified angle threshold.

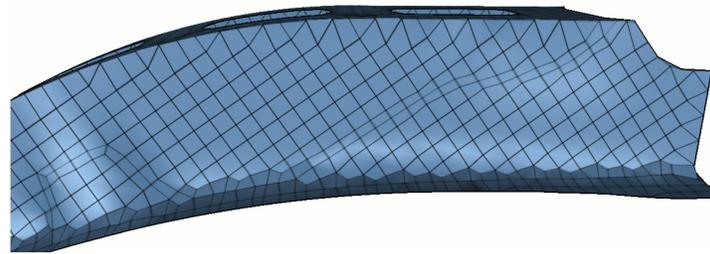
This algorithm is also useful to remove warping in the original mesh as well as after local editing operations (see [BRE04]) as the shape of the surface is only very little affected by unwarping the elements (see Fig. 10(c)). Fixing warped elements may slightly change the feature line’s run but they are not wiped out since they do not run through the elements but along element edges.

### 5.2 Fixing Quad Orientation

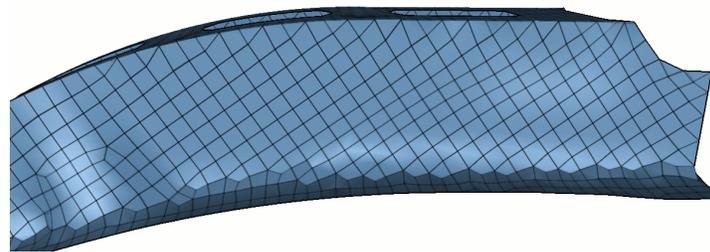
The mesh constructed using the dual grid approach described in Sect. 4.2 usually contains rhombic shaped quads or quads oriented diagonally to its neighbours as depicted in Fig. 12. These artefacts lead to problems during numerical simulation and therefore should be removed.



(a) Reconstructed mesh with rhombic quads



(b) Reconstructed mesh with corrected quads



(c) Reconstructed mesh with corrected quads and mesh relaxation

**Fig. 12.** Removing of rhombic quads

As can be seen in the picture, the vertices of a rhombic quad have a different degree than the ones around it: the vertices within the regular grid have even degree, the ones part of the rhombus have odd degree. Taking that fact as a reference to detect these quads, they can be removed by contracting the diagonal shared by the vertices with lower degree to one single vertex (see Fig. 12(b)). This leads to four equally oriented quads. To further enhance the quads' shape and equalise the elements' size and inner angles, relaxation as described in [BRE04] can be applied to the mesh additionally (see Fig. 12(c)).

## 6 Results and Conclusion

Modifying and editing FE meshes is part of daily engineering work. As the quality of Finite Element Analysis simulations is very sensitive to the FE mesh's quality, one important aim is to provide tools that fit the specific needs of Finite Elements. If the FE mesh is edited locally, it is usually only little deformed and minimal interfering mechanisms like mesh relaxation or local remeshing are sufficient to retain the prerequisites for the simulation. On the other hand, especially in the early design stage sometimes larger scaled modifications of the surfaces are desired and the mechanisms mentioned above will not be sufficient to make the mesh suitable for numerical simulation again. In this case new approaches are needed. In this paper we presented a remeshing method via volumetric representations. We first convert the surface defined by an FE mesh to a signed distance volume. From this volume we reconstruct the surface by isosurface extraction. To retrieve the required quad mesh we apply a duality algorithm to extract a quad mesh out of the modified Marching Cubes algorithm. This resulting mesh is enhanced by removing badly oriented quads, by fixing warped elements and by mesh relaxation (see Fig. 12).

Considering a mesh consisting of some 2300 nodes the calculation of the distance volume as well as the reconstruction takes only a few seconds on a P4 system with 2.8 GHz. The additional mesh enhancement methods work instantly.

The presented method implies the capability to combine the intermediate volumetric representation with the editing operation itself, which is planned for future work. The described procedures are ongoing work. Therefore, the resulting surfaces still show problems we are about to solve, e.g. the reconstruction of small intended holes in the surface, sometimes only of the size of one element in the original mesh.

## Acknowledgements

We would like to thank the *BMB+F* project *AutoOpt*<sup>1</sup> for founding our research as well as the engineers at *BMW AG* for their cooperation and fruitful discussions within this project and for giving insight into today's engineering problems. Additionally we would like to thank Ove Sommer at *science+computing ag* and Alexander Kramer for providing many lines of code being part of this work.

- [BRE04] K. Bidmon, D. Rose, and T. Ertl. Intuitive, Interactive, and Robust Modification and Optimization of Finite Element Models. In *Proceedings 13th International Meshing Roundtable*, pages 59–69, 2004.
- [BS91] 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.
- [Che95] E. V. Chernyaev. Marching cubes 33: Construction of topologically correct iso-surfaces. Technical report, CERN CN 95–17, 1995.
- [COK95] D. Cohen-Or and A. Kaufman. Fundamentals of surface voxelization. *Graph. Models Image Process.*, 57(6):453–461, 1995.

---

<sup>1</sup>[www.auto-opt.de](http://www.auto-opt.de) (only in German)

- [Gib98] S. F. F. Gibson. Using distance maps for accurate surface representation in sampled volumes. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 23–30, New York, NY, USA, 1998. ACM Press.
- [HLC<sup>+</sup>01] J. Huang, Y. Li, R. Crawfis, S. C. Lu, and S. Y. Liou. A complete distance field representation. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 247–254. IEEE Computer Society, 2001.
- [HYFK98] J. Huang, R. Yagel, V. Filippov, and Y. Kurzion. An accurate method for voxelizing polygon meshes. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 119–126. ACM Press, 1998.
- [Kau87] A. Kaufman. Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 171–179, New York, NY, USA, 1987. ACM Press.
- [KBSS01] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 57–66. ACM Press, 2001.
- [LB03] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):16–29, 2003.
- [LC87] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169. ACM Press, 1987.
- [LLVT03] T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. Efficient implementation of Marching Cubes cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.
- [MSS94] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 281–287. IEEE Computer Society Press, 1994.
- [Nie04] G. M. Nielson. Dual marching cubes. In *VIS '04: Proceedings of the IEEE Visualization 2004 (VIS'04)*, pages 489–496. IEEE Computer Society, 2004.
- [sci04] science + computing ag. Efficient preprocessing using scFEMod. <http://www.science-computing.de/en/software/scfemod.html>, 2004.
- [Sra01] M. Sramek. High precision non-binary voxelization of geometric objects. In *SCCG '01: Proceedings of the 17th Spring conference on Computer graphics*, page 220. IEEE Computer Society, 2001.
- [VKK<sup>+</sup>03] G. Varadhan, S. Krishnan, Y. J. Kim, S. Diggavi, and D. Manocha. Efficient max-norm distance computation and reliable voxelization. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 116–126. Eurographics Association, 2003.
- [ZZHW91] 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.