Composite Finite Elements for 3D Image Based Computing

Florian Liehr, Tobias Preusser, Martin Rumpf, Stefan Sauter, Lars Ole Schwen

no. 372

Diese Arbeit ist mit Unterstützung des von der Deutschen Forschungsgemeinschaft getragenen Sonderforschungsbereichs 611 an der Universität Bonn entstanden und als Manuskript vervielfältigt worden.

Bonn, Januar 2008

Composite Finite Elements for 3D Image Based Computing

Florian Liehr, Tobias Preusser[1], Martin Rumpf[2], Stefan Sauter[3], and Lars Ole Schwen[2]

[1] CeVis, Center for Complex Systems and Visualization, University of Bremen, Germany, tp@cevis.uni-bremen.de
[2] INS, Institute for Numerical Simulation, University of Bonn, Germany, {martin.rumpf, ole.schwen}@ins.uni-bonn.de
[3] Institute for Mathematics, University of Zurich, Switzerland, stas@math.unizh.ch

Abstract. We present an algorithmical concept for modeling and simulation with partial differential equations (PDEs) in image based computing where the computational geometry is defined through previously segmented image data. Such problems occur in applications from biology and medicine where the underlying image data has been acquired through e. g. computed tomography (CT), magnetic resonance imaging (MRI) or electron microscopy (EM). Based on a level-set description of the computational domain, our approach is capable of automatically providing suitable composite finite element functions that resolve the complicated shapes in the medical/biological data set. It is efficient in the sense that the traversal of the grid (and thus assembling matrices for finite element computations) inherits the efficiency of uniform grids away from complicated structures. The method's efficiency heavily depends on precomputed lookup tables in the vicinity of the domain boundary or interface. A suitable multigrid method is used for an efficient solution of the systems of equations resulting from the composite finite element discretization. The paper focuses on both algorithmical and implementational details. Scalar and vector valued model problems as well as real applications underline the usability of our approach.

AMS Subject Classifications: 65N30, 65N55, 65N50.

1 Introduction

Frequently, physical modeling and simulation rely on the solution of partial differential equations (PDEs). In particular for applications in the field of biological and medical simulation, these PDEs have to be solved on domains with complicated boundaries or internal geometric structures. The coefficients of the PDEs represent a material property of the physical process or an inhomogeneous physical object, which consists of different homogeneous parts. Consequently the coefficients are often discontinuous across the boundaries of internal object structures.

On one hand, resolving the microscale in a simulation results in more reliable results for physical or biological phenomena. On the other hand, macroscopic medical simulations provide radiologists and surgeons with information for treatment planning. Such application scenarios have become very popular during the last decade. Here, we mention only a few examples: Simulations of electric fields and heat diffusion, as e. g. in hyperthermia [15] and RF-ablation [25], yield therapy parameters. The computation of elastic stresses in the femur and vertebra estimates the risk of fractures [18]. The computation of brain-shift helps to improve the results of neuro-surgeries [51].

Finite element discretizations are very flexible with respect to the computational domain because they allow the use of various geometrical primitives for a discretization of the computational domain. To apply standard finite elements, a certain adaptivity has to be employed to treat problems with complicated geometrical structures. For this purpose, it is popular to use unstructured grids and to refine the computational grid in the vicinity of the interfaces or boundaries until a sufficient approximation quality is reached. While it is well known and understood in two dimensions (2D), the automatic generation of proper grids and the resolution of complicated structures still means a challenge for three-dimensional (3D) geometries [49].

Several authors have investigated numerical methods on interfaced domains. The "Immersed Interface Method" (IIM) [5,27,50,45] uses adaptive finite difference stencils near the interface. Efficient solvers are presented in [28,1]. A combination with finite volume methods is discussed in [8,7] and an extension to "Immersed Finite Elements" in 1D and 2D is proposed in [29,30]. The "Partition of Unity Method" (PUM) [32,3] uses a priori knowledge about the solution (discontinuities at interfaces) to obtain special PUM finite element spaces. In the "Generalized Finite Element Method" (GFEM) [48,46], the PUM and classical FEM basis functions are used together to improve the approximation. Starting from classical FEM and "enriching" the FE spaces by additional basis functions to incorporate discontinuities has been exploited with the "Extended Finite Element

Methods" (XFEM) [4]. The "Finite Cell Method" [35] is based on the idea of extending the PDE outside the actual object domain such that a domain-independent mesh for FEM can be used.

In this paper, we present an approach for the automatic generation of 3D composite finite elements for problems of the class mentioned above. Our approach is inspired by [20] for composite finite elements (CFE) in 2D. It combines the idea of CFE with the efficiency of structured grids as they are used in image processing [36].

Composite finite elements do not require an adaptivity of the computational grid. Instead, they build the necessary adaption into basis functions, which are then used in a standard Galerkin method [20]. Far from the complicated structures (domain boundaries or interfaces between internal object structures), the basis functions coincide with the standard basis functions on the structured grid. In the vicinity of the interface or the boundary, the standard basis is modified to resolve either the shape of the domain boundary or to meet the jump condition imposed by the material coefficient. In contrast to the web-spline approach [22], the CFE method can be used both for problems with discontinuous coefficients and for problems with a complicated domain boundary.

The particular focus of this paper lies on an efficient implementation, which substantially borrows methodology from the area of scientific visualization. To generate a virtual grid that resolves complicated structures, we proceed as in the well known marching cubes/marching tetrahedra algorithms [31,33]: We create a lookup table that contains all possible topological variants of a surface intersecting hexahedral elements. When assembling matrices, only a proper scaling of the pre-computed topological variants is needed.

The remainder of the paper is organized as follows: In Section 2 we review segmentation methods from the area of mathematical image processing. They provide level-set functions describing the domain and internal structures of the domain and thus are needed for the generation of a virtual grid. The virtual grid is needed locally for the construction of a composite finite element space only. In Section 3, we describe a suitable multigrid hierarchy for the composite finite element approach. The algorithmical and computational aspects of the method are discussed in detail in Section 4. In particular, an efficient algorithm for the assembly of system matrices is discussed. There, we also present algorithms for the construction of lookup tables. In Section 5 applications are presented for scalar as well as for vector valued problems in medical simulation. Finally, we draw conclusions in Section 6.

2 Construction of the composite finite element space

2.1 Extracting geometries from images

On the domain $\Omega \subset \mathbb{R}^3$, we assume an image intensity $u_0 : \Omega \to \mathbb{R}$ to be given. This may be the result of computed tomography (CT) or magnetic resonance imaging (MRI) of a part of the human body, or it might come from electron miscroscope (EM) imaging of biological phenomena. The basis for a further quantification or physical simulation is given by the partition of the image into segments that represent the original objects of interest. Examples range from the segmentation and successive volumetry of liquor spaces in the brain [21], or lung nodules [26] over the identification of liver segments and vascular structures draining these segments [42] to the segmentation and computation of mechanical loads in human joints [24].

To achieve the segmentation of the image data, various approaches have been discussed in the past. Simple thresholding works well in very few cases and in the absence of noise only. Other approaches are based on edge indicators of the image. A discrete hierarchical segmentation algorithm that combines several boundary indicators has been presented in [16]. The most prominent approach of discrete segmentation is the morphological watershed transform which creates a tessellation of the domain Ω into segments by simulating rainfall. To remedy an oversegmentation, various modifications of the watershed transform and hybrid approaches have been proposed [44].

More flexible with respect to inhomogeneities in the image data are active contour models and snakes [9,10, 14,23,52]. These are based on the evolution of curves and surfaces, respectively, which are driven by certain forces towards the boundaries of the corresponding segments. The driving forces are designed such that in the limit of the evolution, the curves/surfaces yield a proper approximation of the segment boundary. Active contour models incorporate a wide range of driving forces which result from the minimization of energy functionals. Those energies balance the smoothness and curvature of the resulting geometric objects with the force attracting towards the object boundaries of interest.

In explicit implementations of snake approaches, a major problem results from the fact that topological changes of the initial curves/surfaces are hard to realize. Implicit approaches through level-set functions [34] do not suffer from this limitation. In the level-set approach, the corresponding object boundary is given by the zero level-set of a function $\phi(t,\cdot) : \Omega \to \mathbb{R}$. The evolution of the level-set function with time is then controlled by a partial differential equation of the type

$$\partial_t \phi - f(t,x)|\nabla \phi| = 0, \tag{1}$$

where $f(t,x)$ is the driving force. Using the level-set approach corresponds to an embedding of the snakes into a higher dimensional space in which necessary topology changes can be achieved easily.

A diffuse segment boundary is achieved by using a phase-field approach: the sharp object boundary is approximated by a smooth phase-field function that is zero on object boundaries only and attains the value one away from these boundaries. The level-set and phase-field approaches have been used to approximate the Mumford-Shah functional for the joint image-denoising and segmentation [2,11].

2.2 A virtual grid

Let us now describe how a computational grid can be efficiently generated in the case of a geometry determined by previously acquired image data $u_0 : \Omega \to \mathbb{R}$. In the following, we assume that $\phi : \Omega \to \mathbb{R}$ is a level-set function resulting from a segmentation process based on the image u_0. Here, ϕ is usually considered as the asymptotic limit of the solution of some level-set propagation (1). We emphasize that this is not a restriction, because starting from a variety of other segmentation results (parametric surfaces, characteristic functions, phase-field etc.), a level-set function can be generated e. g. by computing signed distance functions [6, 40] or simple algebraic operations like in the case of solid modeling. Furthermore, a combination of different segments can be easily achieved by min and max operations on the respective level-set functions.

Let us refer to

- the set $\Omega_- := \{x \in \Omega \,|\, \phi(x) < 0\}$ as the *interior* of the segment(s),
- the set $\Omega_+ := \{x \in \Omega \,|\, \phi(x) > 0\}$ as the *exterior* of the segment(s),
- $\gamma := \Omega \cap \partial\Omega_- = \{x \in \Omega \,|\, \phi(x) = 0\}$ as the *interface*, i. e. the interior boundary portion of Ω_- which lies inside Ω.

In the following, the set Ω_- is the physical domain of interest.

In the remainder of this section, we work with discretized data and thus identify images with their nodal values (voxels) and a corresponding set of basis functions or interpolation rules. Since images are typically given on cuboid domains, we assume the computational domain $\Omega \subset [0,1]^3 \subset \mathbb{R}^3$ to be discretized by a hexahedral grid $\mathcal{G}^\square$ of dimension $\{0,\dots,N_x\}\times\{0,\dots,N_y\}\times\{0,\dots,N_z\}$, with $N_x, N_y, N_z \in \mathbb{N}$. We define the grid-width h as

$$h = \min\left\{\frac{1}{N_x}, \frac{1}{N_y}, \frac{1}{N_z}\right\}.$$

The image voxels of u_0 and the nodal values of the level-set function ϕ are defined on the vertices of this grid, which we order lexicographically. Thus, u_0 and ϕ are assumed to be piecewise trilinear. For the later use of hierarchical approaches or multigrid solvers it is convenient to assume that $N_x = N_y = N_z = 2^L + 1$ for some $L \in \mathbb{N}$, such that the resulting grid comes along with an octree structure.

Since we are going to define the geometry for our PDE problem (cf. Section 2.3) based on the given image data u_0, the accuracy is limited by the resolution N_x, N_y, N_z of the image. However, if it is possible to take further information into account, one might as well start from a finer initial resolution or incorporate adaptive mesh refinement below the initial resolution.

Let us divide each hexahedral element of $\mathcal{G}^\square$ into 6 tetrahedra (cf. Figure 1) such that a regular tetrahedral grid $\mathcal{G}^\boxtimes$ of the domain Ω is obtained. We denote the elements of this grid by $T \in \mathcal{G}^\boxtimes$, and the set of nodes of $\mathcal{G}^\boxtimes$ by $\mathcal{N}^\boxtimes$. Note that the hexahedral grid $\mathcal{G}^\square$ and the tetrahedral grid $\mathcal{G}^\boxtimes$ have the same set of nodes.

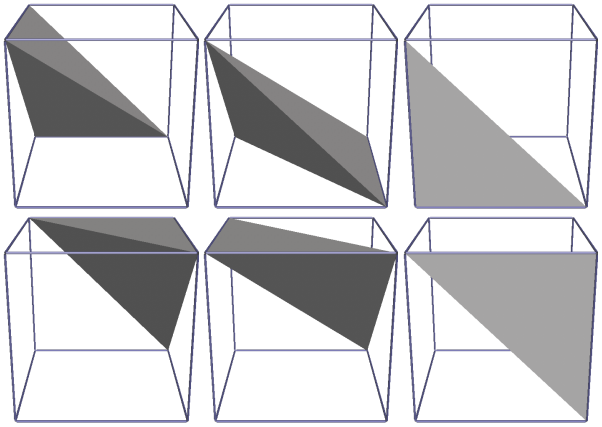

Fig. 1. The subdivision of a hexahedron into 6 tetrahedra. The diagonals on the left and right, top and bottom, and front and back faces are pairwise consistent. Hence the resulting tetrahedral grid $\mathcal{G}^\boxtimes$ is admissible in the usual sense (cf. [13]).

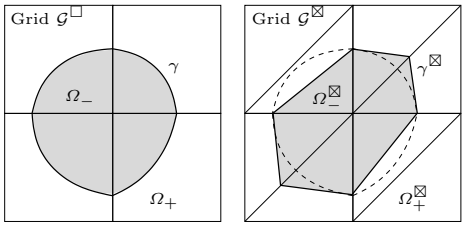

Fig. 2. The approximation of the interface γ by a reinterpretation of the level-set function is sketched in 2D. On the left, the level-set γ is evaluated using tri-linear interpolation of the function ϕ on the hexahedral grid $\mathcal{G}^\square$. On the right, the level-set function ϕ is re-interpreted on a regular tetrahedral grid and the approximated interface $\gamma^\triangle$ results from the linear interpolation on the tetrahedral grid $\mathcal{G}^\boxtimes$.

If we re-interpret the image function u_0 prescribing the original voxel values on $\mathcal{G}^\square$ to the tetrahedral grid $\mathcal{G}^\boxtimes$, we get a piecewise linear representation $u_0^\boxtimes$, whereas, on the hexahedral grid, the original image u_0 was interpreted as piecewise trilinear. Analogously, the re-interpretation of the level-set function ϕ yields approximations $\gamma^\triangle$, $\Omega^\boxtimes_+$ and $\Omega^\boxtimes_-$ of the interface as well as the interior and exterior regions. In particular, $\gamma^\triangle$ is a piecewise planar approximation of the original interface γ (cf. Figure 2). Let us assume that each hexahedron in $\mathcal{G}^\square$ is intersected by the interface $\gamma^\triangle$ at most once. Finally, we define $\mathcal{N}^\boxtimes_\pm := \Omega^\boxtimes_\pm \cap \mathcal{N}^\boxtimes$ such that $\mathcal{N}^\boxtimes = \mathcal{N}^\boxtimes_+ \cup \mathcal{N}^\boxtimes_-$.

Obviously, the re-interpretation of the trilinear data as linear data on the tetrahedra leads to an incorrect position of the interface. But we emphasize that these errors are of sub-voxel size and thus dominated by the inaccuracy of the image-data acquisition process and the segmentation (cf. Section 2.1).

If a tetrahedral element $T \in \mathcal{G}^\boxtimes$ is intersected by the interface approximation $\gamma^\triangle$, it is split (cf. Figure 3)

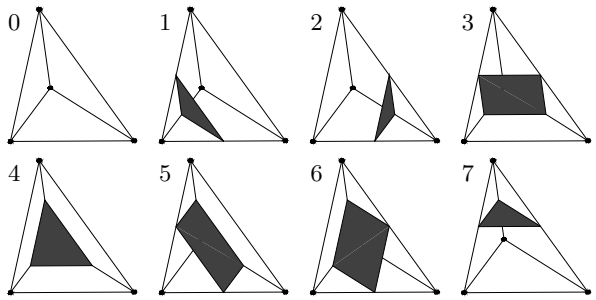

Fig. 3. We show the eight different cases of how the approximate interface $\gamma^\triangle$ can cut a tetrahedron. A tetrahedron is cut by $\gamma^\triangle$ either into a smaller tetrahedron and a prism (cases $1, 2, 4, 7$) or into two prisms (cases $3, 5, 6$).

– either into one prism and one tetrahedron, or
– into two prisms.

Our goal is to split the resulting prisms further to obtain a *virtual grid* $\mathcal{G}^\triangle$ that contains only tetrahedra T. To achieve this, we split each quadrilateral face of the prisms into two triangles. Consistency at faces of the tetrahedra is obtained by requiring that the tetrahedral vertex with the smallest global index does not belong to both triangles. Repeating this procedure for all interfaced tetrahedra yields the desired grid $\mathcal{G}^\triangle$ whose elements resolve the approximated interface $\gamma^\triangle$ due to the construction.

A better aspect ratio of the resulting tetrahedra could be obtained by always using the shortest diagonal. This splitting strategy, however, depends on the values of the level set function and not merely on their signs, so precomputing the splitting as described in Section 4.1 would become more complicated.

To avoid extremely small tetrahedra in the resulting fine grid, we shift the level set function away from zero by $\phi(x) \pm \delta$, $\delta \ll 1$ if $|\phi(x)| < \epsilon \ll 1$. In our computations, we use $\delta = \epsilon = 2 \cdot 10^{-4}$ times the grid spacing h. Thus, we avoid configurations in which the interface $\gamma^\triangle$ intersects grid nodes. In real world applications, the errors introduced by this modification can be neglected compared to errors of the segmentation process.

We refer to this grid $\mathcal{G}^\triangle$ as the virtual grid, since for a suitable CFE space, it is only needed in the initial phase for the construction of basis functions (cf. Section 2.3).

The intersection of $\gamma^\triangle$ with the tetrahedra of $\mathcal{G}^\boxtimes$ generates *virtual nodes* at the intersections of $\gamma^\triangle$ with the edges of $\mathcal{G}^\boxtimes$. For later use we denote the set of the virtual nodes by $\mathcal{N}_v$ and the set of all fine grid nodes by $\mathcal{N}^\triangle := \mathcal{N}^\boxtimes \cup \mathcal{N}_v$. In the following we furthermore refer to the nodes $\mathcal{N}^\boxtimes$ of the regular grid $\mathcal{G}^\boxtimes$ as the *regular nodes*. In Figure 4 we depict the virtual nodes of a domain which has been segmented from CT-data.

In summary, we construct three intertwined grids:

– the original hexahedral voxel grid $\mathcal{G}^\square$ of the given image data u_0,
– the tetrahedral grid $\mathcal{G}^\boxtimes$ that defines the approximated interface $\gamma^\triangle$ as well as $\Omega^\boxtimes_\pm$, and

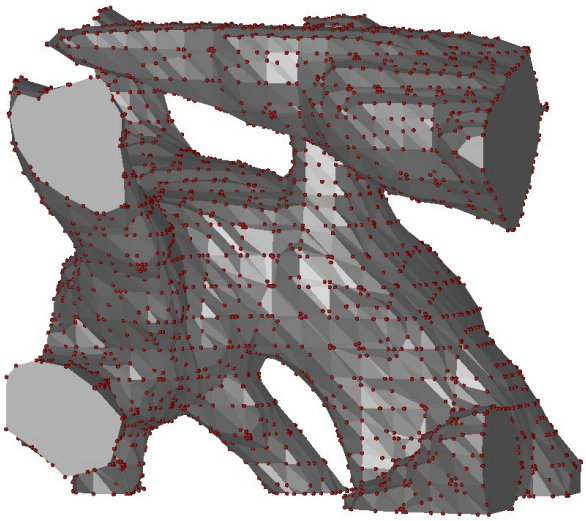

Fig. 4. The virtual nodes for an interface γ are depicted as red dots. By the construction of the virtual grid the interface γ has been approximated by $\gamma^\triangle$ and thus appears non-smooth in the image. Intentionally, we do not use a smooth shading of the interface to enhance the visibility of the underlying hexahedral grid. The interface shown here represents a detail of a vascular tree extracted from CT-data of the human liver (cf. Section 5.3 and Figures 12, 14 and 15).

– the virtual grid $\mathcal{G}^\triangle$ that resolves the approximated interface $\gamma^\triangle$.

As described above, $\mathcal{G}^\triangle$ is a refinement of $\mathcal{G}^\boxtimes$ and $\mathcal{G}^\boxtimes$ is a refinement of $\mathcal{G}^\square$. In the following section we use the virtual grid $\mathcal{G}^\triangle$ for the construction of a CFE space for problems on domains with complicated boundaries (cf. [20]). The multigrid method presented in Section 3 leads to convergence rates which are independent of the grid width h. Later, in Section 5 we present an outlook for the use of the virtual grid to define a CFE space for problems with discontinuous coefficients.

2.3 Composite finite element basis functions for problems on domains with complicated boundary

For an elliptic operator of second order

$$L = -\sum_{\alpha,\beta=1}^{3} \partial_\beta(a_{\alpha\beta}\partial_\alpha \cdot)$$

let us consider the scalar PDE

$$Lu = f \tag{2}$$

on the domain Ω_- described by the level-set function ϕ and for boundary data to be specified later. For the numerical solution of this problem, we want to construct a finite element space whose degrees of freedom (DOF) reside on the coarse grid nodes $\mathcal{N}^\boxtimes$ but which resolves the complicated structure $\gamma^\triangle$.

In a straightforward and well-known fashion the grids $\mathcal{G}^{\triangle}$ and $\mathcal{G}^{\boxtimes}$ define finite element spaces of piecewise linear functions

$$\mathcal{V}^{\triangle\,h} := \Big\{ v \in C^0(\Omega) \,\Big|\, v|_T \text{ is affine } \forall T \in \mathcal{G}^{\triangle} \Big\},$$
$$\mathcal{V}^{\boxtimes\,h} := \Big\{ v \in C^0(\Omega) \,\Big|\, v|_T \text{ is affine } \forall T \in \mathcal{G}^{\boxtimes} \Big\}.$$

Bases for these spaces are given by the standard "hat-functions" on $\mathcal{G}^{\triangle}$ and $\mathcal{G}^{\boxtimes}$ respectively:

$$\begin{aligned}\mathcal{V}^{\triangle\,h} = \operatorname{span}\Big\{ \psi_i^{\triangle} \in C^0(\Omega) \,\Big|\, \psi_i^{\triangle}|_T \text{ is affine } \forall T \in \mathcal{G}^{\triangle} \\ \text{and } \psi_i^{\triangle}(x_j) = \delta_{ij}\ \forall x_j \in \mathcal{N}^{\triangle} \Big\}, \\ \mathcal{V}^{\boxtimes\,h} = \operatorname{span}\Big\{ \psi_i^{\boxtimes} \in C^0(\Omega) \,\Big|\, \psi_i^{\boxtimes}|_T \text{ is affine } \forall T \in \mathcal{G}^{\boxtimes} \\ \text{and } \psi_i^{\boxtimes}(x_j) = \delta_{ij}\ \forall x_j \in \mathcal{N}^{\boxtimes} \Big\},\end{aligned} \tag{3}$$

Obviously $\mathcal{V}^{\boxtimes\,h}$ has its DOF on the nodes $\mathcal{N}^{\boxtimes}$, but it does not resolve $\gamma^{\triangle}$. Conversely, $\mathcal{V}^{\triangle\,h}$ does resolve $\gamma^{\triangle}$ but it has DOF on virtual nodes which are not nodes of the coarse grid. The construction of the composite finite element space uses both $\mathcal{V}^{\triangle\,h}$ and $\mathcal{V}^{\boxtimes\,h}$. In fact, we define the space that resolves the interface $\gamma^{\triangle}$ by

$$\mathcal{V}^h_{\mathrm{CFE}} := \operatorname{span}\{\psi_i^{\mathrm{CFE}}\} \qquad \text{with } \psi_i^{\mathrm{CFE}} := \psi_i^{\boxtimes} \chi_{\Omega_-^{\boxtimes}}, \tag{4}$$

where $\chi_{\Omega_-^{\boxtimes}}$ is the characteristic function of the subdomain. Obviously, this construction removes all basis functions from $\mathcal{V}^{\boxtimes\,h}$ whose support lies completely outside $\Omega_-^{\boxtimes}$. Basis functions $\psi^{\boxtimes}$ that are completely supported inside the subdomain are not modified, and basis functions whose support crosses the interface $\gamma^{\triangle}$ are cut-off. Let us remark that there are also DOF at nodes outside the domain of interest but whose corresponding basis functions are supported inside the domain only.

The construction of the finite element space (4) can be simplified by using the virtual grid $\mathcal{G}^{\triangle}$ and the space $\mathcal{V}^{\triangle\,h}$. Since the grids $\mathcal{G}^{\triangle} \subset \mathcal{G}^{\boxtimes}$ are nested and since both bases are partitions of unity, every basis function $\psi_i^{\boxtimes}$ can be expressed as a linear combination of virtual grid basis functions $\psi_j^{\triangle}$. So, for each $x_i \in \mathcal{N}^{\boxtimes}$ there exists a set of fine grid nodes $C_i := \{x_{i_1}, \ldots, x_{i_{c_i}}\} \subset \mathcal{N}^{\triangle}$ and weights $\mu_{i,l} \in R$ such that

$$\psi_i^{\boxtimes} = \sum_{l=1}^{c_i} \mu_{i,l} \psi_{i_l}^{\triangle}. \tag{5}$$

Obviously the inclusion $\mathcal{N}_v \subset \bigcup_i C_i$ holds. This means, that in the composite finite element space the virtual nodes do not appear as degrees of freedom because a virtual node $x_v \in C_j$ is constrained by the node $x_j \in \mathcal{N}$. We emphasize that the construction (5) is only needed for basis functions whose support crosses $\gamma^{\triangle}$. The basis functions of $\mathcal{V}^h_{\mathrm{CFE}}$ which are completely supported in the interior of the domain $\Omega_-^{\boxtimes}$ coincide with the corresponding standard basis functions of $\mathcal{V}^{\boxtimes\,h}$. For later use let us introduce the set $D_j \subset \mathcal{N}$ which is complementary to C_i: For a fine grid node $x_j \in \mathcal{N}^{\triangle}$ the set D_j contains the regular nodes in $\mathcal{N}$ which constrain x_j.

To derive a standard Galerkin approach (multiplying with a test function $v \in \mathcal{V}^h_{\mathrm{CFE}}$ and integrating over $\Omega_-^{\boxtimes}$), we consider the weak form

$$a(u,v) = l(v) \tag{6}$$

of the boundary value problem (2), where

$$a(u,v) := \int_{\Omega_-^{\boxtimes}} \sum_{\alpha,\beta=1}^{3} a_{\alpha\beta} \partial_\alpha u\, \partial_\beta v \, dx \quad \text{and} \quad l(v) := \int_{\Omega_-^{\boxtimes}} f v \, dx. \tag{7}$$

For basis functions in (4) which are not affected by the cut-off by $\chi_{\Omega_-^{\boxtimes}}$, the computation of the linear and bilinear form in (7) is standard. For basis functions in (4) whose support is intersected by the interface $\gamma^{\triangle}$, the integration makes use of (5) so that

$$\int_{\Omega_-^{\boxtimes}} \psi_i \, dx = \sum_{l=1}^{c_i} \mu_{i,l} \sum_{T \in \mathcal{G}^{\triangle} \cap \Omega_-^{\boxtimes}} \int_T \psi_{i_l}^{\triangle} \, dx$$

and for products of derivatives appearing in $a(\cdot,\cdot)$

$$\begin{aligned}&\int_{\Omega_-^{\boxtimes}} \partial_\alpha \psi_i \, \partial_\beta \psi_j \, dx \\ &\quad = \sum_{l=1}^{c_i} \mu_{i,l} \sum_{k=1}^{c_j} \mu_{j,k} \sum_{T \in \mathcal{G}^{\triangle} \cap \Omega_-^{\boxtimes}} \int_T \partial_\alpha \psi_{i_l}^{\triangle} \, \partial_\beta \psi_{j_k}^{\triangle} \, dx.\end{aligned} \tag{8}$$

The CFE approach benefits from the fact that we do not need to assign global indices to the virtual nodes and we can work with the very simple and efficient lexicographical ordering induced by the underlying image data. Nodes outside the subdomain that are no DOF can easily be masked out in the iterative solver. In Section 4, we will have a closer look at the algorithmic construction of the composite finite element space and the assembly of mass and stiffness matrices.

2.4 Boundary Conditions

On the interior boundary $\gamma = \partial\Omega_- \cap \Omega$ (see Section 2.2), we allow homogeneous Neumann boundary conditions only. On the exterior boundary $\partial\Omega \cap \partial\Omega_- =: \Gamma_D \dot\cup \Gamma_N$, we impose Dirichlet boundary conditions on Γ_D and Neumann boundary conditions on Γ_N. This means in particular that Dirichlet boundary conditions can only be applied to the exterior boundary of our physical object. As for our meshes, we assume that the set $\overline{\Gamma_N} \cap \overline{\Gamma_D}$ is contained in the set of regular nodes of $\mathcal{G}^{\square}$. For the treatment of complicated Dirichlet boundaries, i. e. $\gamma \cap \Gamma_D \neq \emptyset$, we refer to [39].

2.5 Quality of the tetrahedral meshes

Due to the construction of the virtual grid $\mathcal{G}^{\triangle}$, tetrahedra can have very small diameter and furthermore very small angles between edges. It is well known that this can pose problems for the solution of linear systems resulting from Galerkin FE discretizations of PDE [47]. This will be relaxed by the multigrid method presented in the next section. See [38] for an example where tetrahedra of bad aspect ratio and/or small size lead to linear systems with large condition numbers which do not affect the efficiency of our multigrid solver.

A basic property of the construction presented in the last paragraphs is that regular and virtual nodes can only lie on edges of the six tetrahedra forming one cube. Consequently we observe that, among tetrahedra with all vertices close to a line, spires and splinters can occur whereas spears, spikes and spindles cannot, if we follow the classification of badly shaped tetrahedra in [12]. Moreover, among tetrahedra with all vertices close to a plane but no line, wedges and slivers can occur but spades and caps cannot.

3 A CFE multigrid solver

The geometric structure of the grids defined in Section 2.2 comes along with a natural octree structure if we assume that $N_x = N_y = N_z = 2^L + 1$ for some grid-depth $L \in \mathbb{N}$. This octree structure defines a natural hierarchy of hexahedral grids $\mathcal{F}^l$ and corresponding sets of nodes $\mathcal{M}^l$ for $l \in \{0, \dots, L\}$. Obviously the inclusion $\mathcal{M}^0 \subset \cdots \subset \mathcal{M}^L$ holds, moreover, $\mathcal{F}^L = \mathcal{G}^{\square}$ and $\mathcal{M}^L = \mathcal{N}^{\boxtimes}$. As in Section 2.2, the sets of nodes also describe a regular tetrahedral grid if we divide each hexahedron into 6 tetrahedra as shown in Figure 1.

For a node $x_i^l \in \mathcal{M}^l$, we denote the set of neighboring fine-grid nodes by $\mathcal{A}_i^l \subset \mathcal{M}^{l+1}$. It is the set of at most 14 nodes which are connected to x_i^l by the edge of one of these tetrahedra including the node itself.

In this section we use the hierarchical octree structure to define a geometric CFE multigrid method. I. e. the basis functions on coarse grids are supposed to resolve the boundary. First we construct a two-grid method starting on the finest grid and explain how to obtain a multigrid solver by recursion. We are inspired by [17] for 1D composite finite elements.

3.1 Structure Coarsening

In the following, we denote a CFE-grid on level $l \in \{0, \dots, L\}$ by $\mathcal{G}^l$ and the corresponding set of nodes by $\mathcal{N}^l$. Recall that we constructed the CFE basis functions on the fine grid by selecting DOF from the set of nodes $\mathcal{N}^{\boxtimes} = \mathcal{M}^L$ and cutting off the standard basis functions such that they resolve $\gamma^{\triangle}$. So we have

$$\mathcal{N}^L := \{x_j \in \mathcal{N}^{\boxtimes} \,|\, \operatorname{supp} \psi_j^{\boxtimes} \cap \Omega_-^{\boxtimes} \neq \emptyset\} \subset \mathcal{M}^L,$$
$$\mathcal{G}^L := \{T \in \mathcal{G}^{\boxtimes} \,|\, T \cap \Omega_-^{\boxtimes} \neq \emptyset\}$$

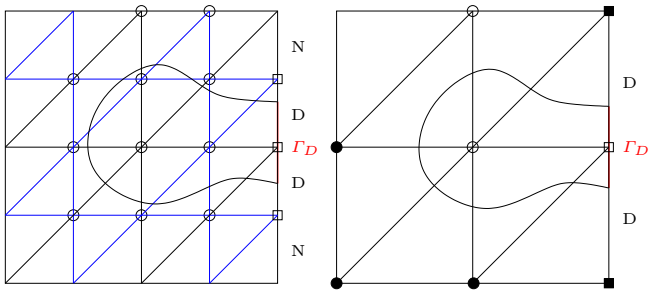

Fig. 5. When coarsening the CFE grid for the domain shown here, we obtain newly introduced DOF at positions marked by • and Dirichlet nodes at positions marked by ■. DOF already present in the fine grid are marked with ○ and ▫, respectively. The fine grid is depicted on the left, and the coarse grid is shown on the right. Faces on the right boundary portion of the domain are marked as "Dirichlet" (D) or "Neumann" (N).

for the finest level of the hierarchy.

To define the CFE multigrid-structure, we proceed similarly by choosing corresponding nodes (DOF) from the octree nodes $\mathcal{M}^l$ and by defining the corresponding basis functions. For a fine grid $\mathcal{G}^l$ we define the nodes of $\mathcal{G}^{l-1}$ by

$$\mathcal{N}^{l-1} := \{x_i \in \mathcal{M}^{l-1} \,|\, x_i \in \mathcal{N}^l \text{ or } \mathcal{A}_i^l \cap \mathcal{N}^l \neq \emptyset\}. \quad (9)$$

So a node in $\mathcal{M}^{l-1}$ is a DOF in $\mathcal{N}^{l-1}$ if it is a fine grid node as well, or if one of its 14 fine-grid neighbors is a DOF in $\mathcal{N}^{l-1}$. The last part of this definition is the peculiarity of the definition of the CFE-multigrid hierarchy: It leads to non-nestedness of the grids. One layer of nodes outside the domain of computation may be assigned DOF, so nodes in the coarse grid may have DOF whereas the geometrically corresponding node in the fine grid does not (cf. Figure 5).

Let us remark that for grid cells of the coarse grid that are intersected by a half plane interface, the coarsened basis functions are exactly the same as those we would obtain if we constructed them from the interface resolved on the coarse grid.

If we are dealing with a Dirichlet boundary on the finest grid, we require the assumptions on the mesh from Section 2.4 to be satisfied for all grids in our hierarchy. For newly introduced nodes on coarsened grids, we need to decide whether to make them Dirichlet nodes or not. For this purpose, consider the hierarchy of boundary faces (objects of codimension 1): a face on the finest grid is labeled "Dirichlet" if at least one of its vertices is a Dirichlet node, "Neumann" otherwise. When coarsening, a coarse boundary face is labeled "Dirichlet" if at least one of the face's children is "Dirichlet"; it is labeled "Neumann" otherwise. Dirichlet nodes are those nodes that are vertices of Dirichlet boundary faces (cf. Figure 5).

3.2 Operator Coarsening

We define the coarse grid basis functions and the prolongation and restriction operators respectively by local

Galerkin products [19] on regular tetrahedral meshes. This involves the weights 1, 1/2, and 0 in the prolongation matrices $P_{l\to l+1}$ for the interpolation from grid-level l to grid-level $l+1$. As usual, we set $R_{l+1\to l} := P_{l\to l+1}^T$ for the corresponding restriction.

Coarse grid basis functions $\psi_i^{\mathrm{CFE},l}$ are obtained by the corresponding weighted sums of the fine grid basis $\psi_j^{\mathrm{CFE},l-1}$, where $\psi_j^{\mathrm{CFE},L} := \psi_j^{\mathrm{CFE}}$. This standard approach leads to piecewise affine basis functions $\psi_i^{\mathrm{CFE},l}$ on the coarse tetrahedral grids (induced by the $\mathcal{M}^l$) which are cut-off through the multiplication with a characteristic function $\chi_{\Omega^{\boxtimes}}$ of the computational domain.

Note that in general the coarse grids are not able to resolve the computational domain $\Omega_-^{\boxtimes}$. Consequently, the support of a coarse grid basis function may consist of several disconnected components. However, it is easy to see that the basis functions $\psi_i^{\mathrm{CFE},l}$ still form a partition of unity on the computational domain $\Omega_-^{\boxtimes}$.

Finally, if A_l denotes the matrix representation of the discrete bilinear form (cf. (7)) on grid level l, the standard Galerkin coarsening leads us to the corresponding coarse grid discrete forms by

$$A_l = R_{l+1\to l}\, A_{l+1}\, P_{l\to l+1}. \tag{10}$$

3.3 Multigrid-Cycles

In our multigrid solver, we first construct a hierarchy of grids and coarsened operators (starting from the finest grid) up to some coarse level by recursively applying the procedure described above.

We then use a multigrid method with symmetric Gauß-Seidel iterations as a smoother. For scalar problems, the Gauß-Seidel smoother is a standard one, for vector valued problems, we use a Block-Gauß-Seidel method. The unknowns are (implicitly reordered and) indexed in such a way that we treat the spatial components of our solution simultaneously. I. e. in the case of a three-dimensional problem, we use Gauß-Seidel iterations on 3×3 blocks.

In our computations in Section 5, V-cycles with no more than 3 pre- and post-smoothing steps turned out to be a reasonable choice.

4 Algorithmical aspects

The intention of this paper is in particular to present the effective and efficient algorithmic setup to use CFE methods in image based computing. Thus, let us now focus on the algorithmical aspects behind the general concept described in the last section. This section discusses how to efficiently compute the intersection of the approximated interface $\gamma^{\triangle}$ with the tetrahedral grid $\mathcal{G}^{\boxtimes}$ and how to assembly finite element matrices. For this purpose, we adopt methodology that has been used earlier in the field of scientific visualization in form of the marching cubes/marching tetrahedra algorithms for isosurface extraction [31,33].

4.1 Hashing topology

Creating the grid $\mathcal{G}^{\triangle}$ can be simplified enormously by observing that only few topologically different configurations of the interface $\gamma^{\triangle}$ cutting an element $E \in \mathcal{G}$ can occur (cf. Figure 3). The topological type of a local configuration is determined by the values of the level-set function ϕ at the vertices of the hexahedron E. Based on this observation, a lookup table can be created that contains the topological information (cf. Algorithm 1).

To make this more precise, let us introduce the *signature* $\sigma(E)$ of a hexahedral element $E \in \mathcal{G}$

$$\sigma(E) = (\sigma_0, \ldots, \sigma_7), \text{ with } \sigma_i = \operatorname{sign}(\phi(x_i)) \in \{-1, +1\},$$

where the x_i are the nodes of E. Since each hexahedron has 8 vertices, obviously at most $2^8 = 256$ different signatures can occur. In an implementation the signature can be easily stored in one byte in which the bits are set if e. g. the level-set function ϕ has negative sign at the corresponding vertex. Furthermore, the topology of sub-tetrahedra resolving the interface can be determined from the signature (i. e. the signs of the level-set function ϕ) only. In fact, as shown previously in Figure 3, only 8 distinct cases have to be considered. All other configurations can be transformed into one of those 8 by a rotation or an inversion of the sign of ϕ.

The splitting of elements E into tetrahedra T and moreover local reference matrices and derivatives of basis functions $\psi_i^{\triangle}$ can be pre-computed and stored. For the later assembly of finite element mass or stiffness matrices, the actual contributions can be obtained by a simple scaling of the pre-computed data.

Algorithm 1 describes the generation of a topology lookup table. In the main loop (lines 2-6), we run a splitting algorithm for every signature $\sigma = 0, \ldots, 255$. For reasons of simplicity, we do not take into account the symmetries mentioned above in this part of the code. The splitting itself checks the signature of the tetrahedron (lines 11+12) and creates a local set of nodes $\mathcal{N}_{\pm}^{\mathrm{LOC}}$ on either side of the interface $\gamma^{\triangle}$. To create the topology, we assume that $\gamma^{\triangle}$ splits edges in the middle and correspondingly a set of local virtual nodes is constructed (lines 13-18). We refer to the local virtual nodes by the local indices of their constraining parent nodes. A virtual node located at $\frac{1}{2}(x_{j_k} + x_{j_l})$ is referred to as (j_k, j_l). This implicitly gives us the set of constraining parent nodes D_j from Section 2.3.

To split either configuration (tetrahedron/tetrahedron or tetrahedron/prism) we now take symmetries into account: The local setting is transformed by reordering the vertices of the tetrahedron and by swapping the (+) and (−) sides of the local configuration (lines 19-33). Since we are interested in the tetrahedra resolving the computational domain $\Omega_-^{\boxtimes}$, we store the splitting of the (−) side in the lookup table only.

In our algorithms, we do not save coordinates of the (virtual) sub-tetrahedra, but local index pairs of their constraining vertices only. Thus, a tetrahedron

$$T = \{a_0a_1, b_0b_1, c_0c_1, d_0d_1\}$$

Algorithm 1 Create topology lookup table

1: **procedure** CREATELOOKUPTABLE
2: **for** $\sigma \leftarrow 0, \dots, 255$ **do** ▷ For each topological type σ
3: **for** $j \leftarrow 0, \dots, 5$ **do** ▷ For each standard tetrahedron of E with $\sigma(E) = \sigma$
4: **if** Tetrahedron T_j is interfaced **then**
5: SplitTetrahedron($\sigma|_{T_j}, T_j$)
6: **end if**
7: **end for**
8: **end for**
9: **end procedure**

10: **procedure** SPLITTETRAHEDRON(signature $\sigma = \{\sigma_0, \dots, \sigma_3\}$, tetrahedron $T = \{x_{j_0}, \dots, x_{j_3}\}$)
11: $\mathcal{N}_+^{\text{LOC}} \leftarrow \{x_{j_i} \,|\, \sigma_i = +1, i = 0, \dots, 3\}$ ▷ Positive vertices
12: $\mathcal{N}_-^{\text{LOC}} \leftarrow \{x_{j_i} \,|\, \sigma_i = -1, i = 0, \dots, 3\}$ ▷ Negative vertices

13: $\mathcal{N}_v^{\text{LOC}} \leftarrow \emptyset$ ▷ Create the set of local virtual nodes
14: **for** each edge (x_{j_k}, x_{j_l}) of T **do**
15: **if** $\sigma_k \neq \sigma_l$ **then** ▷ If the edge is interfaced
16: $\mathcal{N}_v^{\text{LOC}} \leftarrow \mathcal{N}_v^{\text{LOC}} \cup \{(j_k, j_l)\}$
17: **end if**
18: **end for**

19: **if** $|\mathcal{N}_-^{\text{LOC}}| = 1$ **then** ▷ T splits into a tetrahedron inside $\Omega_-^{\boxtimes}$ and a prism outside
20: Reorder vertices such that
• $\mathcal{N}_-^{\text{LOC}} = \{x_{j_0}\}$ and $\mathcal{N}_+^{\text{LOC}} = \{x_{j_1}, x_{j_2}, x_{j_3}\}$
• $j_1 < j_2 < j_3$
21: $T_0 \leftarrow \{j_0j_0, j_0j_1, j_0j_2, j_0j_3\}$ ▷ T_0 has vertices $\{x_{j_0}, \overline{x_{j_0}x_{j_1}}, \overline{x_{j_0}x_{j_2}}, \overline{x_{j_0}x_{j_3}}\}$
22: **else**
23: **if** $|\mathcal{N}_-^{\text{LOC}}| = 2$ **then** ▷ T splits into two prisms
24: Reorder vertices such that
• $\mathcal{N}_-^{\text{LOC}} = \{x_{j_0}, x_{j_1}\}$ and $\mathcal{N}_+^{\text{LOC}} = \{x_{j_2}, x_{j_3}\}$
• $j_0 = \min\{j_i\}$ and $j_2 < j_3$
25: $T_0 \leftarrow \{j_0j_0, j_1j_1, j_1j_3, j_1j_2\}$ ▷ T_0 has vertices $\{x_{j_0}, x_{j_1}, \overline{x_{j_1}x_{j_3}}, \overline{x_{j_1}x_{j_2}}\}$
26: $T_1 \leftarrow \{j_0j_0, j_0j_2, j_0j_3, j_1j_2\}$ ▷ T_1 has vertices $\{x_{j_0}, \overline{x_{j_0}x_{j_2}}, \overline{x_{j_0}x_{j_3}}, \overline{x_{j_1}x_{j_2}}\}$
27: $T_2 \leftarrow \{j_0j_0, j_1j_3, j_0j_3, j_1j_2\}$ ▷ T_2 has vertices $\{x_{j_0}, \overline{x_{j_1}x_{j_3}}, \overline{x_{j_0}x_{j_3}}, \overline{x_{j_1}x_{j_2}}\}$
28: **else** ▷ T splits into a prism inside $\Omega_-^{\boxtimes}$ and a tetrahedron outside
29: Reorder vertices such that
• $\mathcal{N}_-^{\text{LOC}} = \{x_{j_1}, x_{j_2}, x_{j_3}\}$ and $\mathcal{N}_+^{\text{LOC}} = \{x_{j_0}\}$
• $j_1 < j_2 < j_3$
30: $T_0 \leftarrow \{j_1j_1, j_1j_0, j_2j_0, j_3j_0\}$ ▷ T_0 has vertices $\{x_{j_1}, \overline{x_{j_1}x_{j_0}}, \overline{x_{j_2}x_{j_0}}, \overline{x_{j_3}x_{j_0}}\}$
31: $T_1 \leftarrow \{j_1j_1, j_2j_2, j_2j_0, j_3j_0\}$ ▷ T_1 has vertices $\{x_{j_1}, x_{j_2}, \overline{x_{j_2}x_{j_0}}, \overline{x_{j_3}x_{j_0}}\}$
32: $T_2 \leftarrow \{j_2j_2, j_3j_3, j_1j_1, j_3j_0\}$ ▷ T_2 has vertices $\{x_{j_2}, x_{j_3}, x_{j_1}, \overline{x_{j_3}x_{j_0}}\}$
33: **end if**
34: **end if**
35: **end procedure**

has the vertices

$$\overline{x_{a_0}x_{a_1}}, \overline{x_{b_0}x_{b_1}}, \overline{x_{c_0}x_{c_1}}, \overline{x_{d_0}x_{d_1}}$$

where we use the abbreviation $\overline{x_ix_j} = \frac{1}{2}(x_i + x_j)$. Using this notation, we can easily refer to virtual nodes if two different constraining indices exist. Furthermore we fix the notational convention that the first index always refers to the regular node inside the domain $\Omega_-^{\boxtimes}$. E. g. the tetrahedron T_0 in line 21 has the indices $\{j_0j_0, j_0j_1, j_0j_2, j_0j_3\}$ which means that it contains one regular node $x_{j_0} \in \mathcal{N}_-$ and three virtual nodes $\frac{1}{2}(x_{j_0} + x_{j_1}), \frac{1}{2}(x_{j_0} + x_{j_2}), \frac{1}{2}(x_{j_0} + x_{j_3})$.

After the lookup table has been created, the set of virtual nodes $\mathcal{N}_v$ can be easily generated from the sets of local virtual nodes $\mathcal{N}_v^{\text{LOC}}$. Let us mention that, in a final sweep over the lookup table, the tetrahedra should be checked for positive orientation in the usual mathematical sense.

For each tetrahedron T obtained by the splitting algorithm we compute a reference mass matrix $M^{\text{REF}} = (\,(\psi_i, \psi_j)\,)_{i,j=1,\dots,4}$ and a Jacobian $J^{\text{REF}} = (\,\partial_\alpha \psi_i\,)_{i=1,\dots,4,\,\alpha=1,2,3}$ of the vector of standard linear basis functions on T. Both M^{REF} and J^{REF}, which are stored in the lookup table, are needed later for an efficient assembly of global matrices.

4.2 Hashing virtual nodes

For the CFE approach presented in Section 2.3, it is necessary to store additional geometric information associated with individual virtual nodes. Therefore an efficient

storage method for data associated with virtual nodes is needed. It is not possible to store virtual nodes per element since the same virtual node may be present in adjacent elements as well, being used in an earlier or later stage of a grid traversal. Instead, we choose a hashing strategy in which we build the hash key from the global indices of the DOF that geometrically constrain a virtual node. The hash key for a virtual node $z = (x_i, x_j)$ is created from $\text{IDX}(i)$ and $\text{IDX}(j)$, where $\text{IDX}(k)$ gives the global index of node x_k. As a simple hash-function is given by the concatenation of the bits of $\text{IDX}(i)$ and $\text{IDX}(j)$ where $\text{IDX}(i) < \text{IDX}(j)$.

4.3 Matrix assembly

The key algorithmic building block of any finite element scheme is the assembly of matrices and vectors used in the resulting linear systems of equations to be solved. We deal here with the matrix assembly. The case of a vector assembly is then straightforward.

Making use of the above introduced hashing concept, a global finite element matrix is assembled from pre-computed local reference matrices. These reference matrices correspond to the set of all (virtual) reference tetrahedra resulting from the subdivision (as obtained by Algorithm 1) of the 6 tetrahedra in a reference cube E according to all possible signatures $\sigma(E)$.

Algorithm 2 describes this assembly: For each hexahedral element of the octree $\mathcal{G}^{\square}$, which is intersected by the computational domain $\Omega^{\boxtimes}_{-}$, we run over all sub-tetrahedra (line 5). The set of sub-tetrahedra can easily be taken from the lookup table corresponding to the signature $\sigma(E)$.

Next, the weights $\lambda_{i,0}, \lambda_{i,1}$ represent the actual intersection point with the interface on an edge and are computed correspondingly (lines 34-42). A local matrix $(M_T^{\text{LOC}}[i][j])_{ij}$ for the sub-tetrahedron T is retrieved from a lookup table and scaled according to the actual geometry of T (line 7). Below we describe how this scaling is performed in the case of mass and stiffness matrices. The entries of the local matrix are weighted and accumulated into the global one (lines 8-23).

The local matrices refer to piecewise affine basis functions which attain values 0 or 1 on the virtual nodes. In the case of the composite finite element basis functions the nodal value 1 is replaced by the properly interpolated value. These interpolation values are given by the weights $\mu_{\cdot,\cdot}$ in (5). However, locally for a virtual node $i = (i_0 i_1)$ which resides on the edge between two nodes $x_{\text{IDX}(i_0)}, x_{\text{IDX}(i_1)} \in \mathcal{N}$ only two weights are relevant. Those relevant weights correspond to the intersection points with the interface $\lambda_{i,0}, \lambda_{i,1}$.

To distribute the entries of the local matrices, we loop over all local vertices of the sub-tetrahedron and check whether those are virtual nodes. If a virtual node occurs, its contribution from the local matrix has to be weighted by products of the $\lambda_{i,\cdot}$ and distributed on the constraining nodes. Here, we use the fact that we refer to vertices of tetrahedra by pairs of indices as described in Section 4.1.

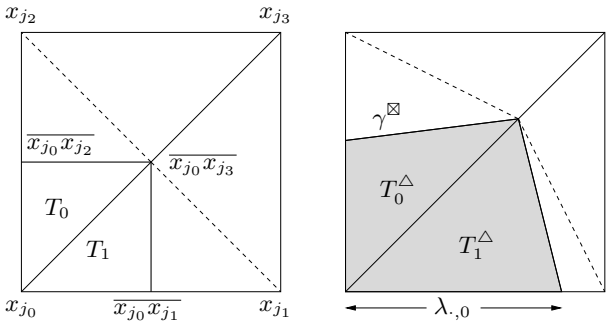

Fig. 6. The transformation from the reference configuration stored in the lookup table to the actual local geometric setting is sketched in two dimensions. On the left we depict a reference configuration for an element E with (2D) signature $\{-1, 1, 1, 1\}$ having three virtual nodes and two elements $T_{1,2}$ inside the computational domain. In the reference configuration the virtual nodes are assumed to reside in the middle of the edges of E. On the right we show an actual geometrical configuration. For the assembly of matrices the virtual nodes are moved according to the weights $\lambda_{\cdot,0}$ and thus the virtual elements $T^{\triangle}_{1,2}$ are obtained. Pre-computed local mass matrices M^{REF} and Jacobians J^{REF} are scaled to reflect the actual geometrical setting.

We achieve a speedup by not running over the sub-tetrahedra of elements not intersected by the interface, but by providing again a lookup table for such hexahedra (lines 26-29). In the applications considered here, a simple mid-point quadrature rule is sufficient for the coefficients of the bilinear form. These local matrices are also pre-computed in a set-up phase. Let us finally mention that the weights $\lambda_{i,\cdot}$ are computed in advance as well and stored in a hash table.

Algorithm 3 describes how to obtain the scaled local matrices based on information stored in the lookup tables. The procedure is very simple and straightforward in the case of mass matrices: For each tetrahedron obtained through the splitting Algorithm 1 we compute and store the mass matrix in advance. In Algorithm 3 this matrix is retrieved (line 3). Then we construct up a scaling matrix B built from the weights $\lambda_{\cdot,0}$ (lines 4-9) which reflects the transformation from the stored topological tetrahedron to the actual geometry. Here the splitting weights $\lambda_{\cdot,0}$ come into play to move the virtual nodes (so far assumed to be in the middle of edges) to the right position (cf. Figure 6). The volume of a virtual tetrahedron T is given by $\text{vol}(T) = \det(B)\text{vol}(T)$, where T is the reference tetrahedron from the lookup table. Thus, the scaled mass matrix M is obtained by scaling the stored version M^{REF} with the determinant of B (line 10).

For the assembly of stiffness matrices we proceed similarly. For the local stiffness matrix $L = (L_{ij})_{i,j=1,\cdots,4}$ on a virtual tetrahedron T we obtain

$$L = \left(\int_T \sum_{\alpha,\beta=1}^{3} a_{\alpha\beta} \partial_\alpha \psi_i^{\triangle} \partial_\beta \psi_j^{\triangle} \right)_{i,j=1,\ldots,4} = \text{vol}(T) J^{\triangle} A (J^{\triangle})^T$$

Algorithm 2 Assembly of system matrix for composite finite elements

1: **procedure** ASSEMBLE
2: **for** each hexahedral element E of the grid $\mathcal{G}$ **do**
3: **if** $E \cap \Omega_-^{\boxtimes} \neq \emptyset$ **then** ▷ If element intersects domain
4: **if** $E \cap \gamma^{\triangle} \neq \emptyset$ **then** ▷ If element is interfaced
5: **for** $T = (a_0a_1, b_0b_1, c_0c_1, d_0d_1) \in \text{LookupTable}(\sigma(E))$ **do** ▷ For each sub-tetrahedron $T \in \mathcal{G}^{\triangle} \cap E$
6: ComputeWeights(λ, ϕ, E, T)
7: ComputeLocalTetraMatrix(E, T, λ, M_T^{LOC})
8: **for** $i, j \leftarrow \{a, b, c, d\}$ **do** ▷ For each pair of vertices
9: **if** i, j are regular nodes **then**
10: $M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_0)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_0)] + M_T^{\text{LOC}}[i][j]$
11: **else if** i is virtual node, j is regular node **then**
12: $M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_0)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_0)] + \lambda_{i,0} M_T^{\text{LOC}}[i][j]$
13: $M^{\text{GLOBAL}}[\text{IDX}(i_1)][\text{IDX}(j_0)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i_1)][\text{IDX}(j_0)] + \lambda_{i,1} M_T^{\text{LOC}}[i][j]$
14: **else if** i is regular node, j is virtual node **then**
15: $M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_0)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_0)] + \lambda_{j,0} M_T^{\text{LOC}}[i][j]$
16: $M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_1)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_1)] + \lambda_{j,1} M_T^{\text{LOC}}[i][j]$
17: **else if** i, j are virtual nodes **then**
18: $M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_0)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_0)] + \lambda_{i,0}\lambda_{j,0} M_T^{\text{LOC}}[i][j]$
19: $M^{\text{GLOBAL}}[\text{IDX}(i_1)][\text{IDX}(j_0)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i_1)][\text{IDX}(j_0)] + \lambda_{i,1}\lambda_{j,0} M_T^{\text{LOC}}[i][j]$
20: $M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_1)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i_0)][\text{IDX}(j_1)] + \lambda_{i,0}\lambda_{j,1} M_T^{\text{LOC}}[i][j]$
21: $M^{\text{GLOBAL}}[\text{IDX}(i_1)][\text{IDX}(j_1)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i_1)][\text{IDX}(j_1)] + \lambda_{i,1}\lambda_{j,1} M_T^{\text{LOC}}[i][j]$
22: **end if**
23: **end for**
24: **end for**
25: **else** ▷ If E is not interfaced
26: LookUpLocalHexaMatrix(E,M_E^{LOC})
27: **for** $i, j \leftarrow 0, \ldots, 7$ **do** ▷ For each pair of local DOF
28: $M^{\text{GLOBAL}}[\text{IDX}(i)][\text{IDX}(j)] \leftarrow M^{\text{GLOBAL}}[\text{IDX}(i)][\text{IDX}(j)] + M_E^{\text{LOC}}[i][j]$
29: **end for**
30: **end if**
31: **end if**
32: **end for**
33: **end procedure**

34: **procedure** COMPUTEWEIGHTS(weights λ, level-set-function ϕ, hexahedron E, tetrahedron T)
35: Assume $E = \{x_0, \ldots, x_7\}$ and $T = \{a_0a_1, \ldots, d_0d_1\}$
36: **for** each vertex $(j_k j_l) \in T$ **do**
37: **if** $j_k \neq j_l$ **then** ▷ Vertex is virtual node
38: $\lambda_{j,0} \leftarrow \dfrac{\phi(x_{j_k})}{\phi(x_{j_k}) - \phi(x_{j_l})}$ ▷ Intersection with interface = zero of ϕ
39: $\lambda_{j,1} \leftarrow 1 - \lambda_{j,0}$
40: **end if**
41: **end for**
42: **end procedure**

where $A = (a_{\alpha\beta})_{\alpha,\beta=1,2,3}$ is the coefficient matrix and $J^{\triangle} = (\partial_\alpha \psi_i^{\triangle})_{i=1,\ldots,4,\,\alpha=1,2,3}$ the Jacobian of the vector of local basis functions. This Jacobian on the virtual element T can be obtained via scaling from pre-computed Jacobians J^{REF} stored again in a lookup table (line 14). In fact, we obtain $J^{\triangle} = J^{\text{REF}} B^{-1}$ for the above scaling matrix B (lines 15-20). Thus, we end up with (line 21)

$$L = \text{vol}(T)\det(B) J^{\text{REF}} B^{-1} A B^{-1} (J^{\text{REF}})^T .$$

Let us emphasize that B is invertible since in Section 2.2 we have assumed that the virtual nodes never coincide with coarse grid nodes, thus $\lambda_{\cdot,0} \neq 0$. Moreover, the computation of $\det(B)$ and B^{-1} is trivial since B is diagonal.

4.4 Multigrid

To use the standard coarsening, the basis for the fine grid is extended by hat functions of height zero centered at all nodes in $\mathcal{M}^l \setminus \mathcal{N}^l$, i. e. nodes outside the domain $\Omega_-^{\boxtimes}$ whose standard basis function do not cross the interface. After standard coarsening, all hat functions with empty support (i. e. being zero everywhere) are removed to obtain a basis for the coarse grid CFE functions.

For the construction of the coarse grid operators of the multigrid we have to compute various matrix products as in (10). On the finest grid we use vectors of length $n_L = (2^L + 1)^3$ such that we have entries for all regular nodes (and mask out those entries corresponding to non-DOF). Then the prolongation operator $P_{l-1\to l}$ can be represented by a $n_{l-1} \times n_l$ matrix. Due to the spar-

Algorithm 3 Computation of local matrices from lookup table

procedure ComputeLocalTetraMassMatrix(hexahedron E, tetrahedron T, weights λ, local matrix M)
  Assume $T = \{a_0a_1, \ldots, d_0d_1\}$
  RetrieveReferenceMassMatrixFromLookupTable(M^{REF}, E, T)
  $B \leftarrow \mathrm{Id}_{3\times 3}$
  for each vertex $(j_k j_l) \in T$ **do**
    if $j_k \neq j_l$ **then** ▷ Vertex is virtual node
      $B_{jj} \leftarrow 2\lambda_{j,0}$ ▷ Store scaling factor in matrix
    end if
  end for
  $M \leftarrow \det(B)\, M^{\text{REF}}$ ▷ Scale pre-integrated mass matrix
end procedure

procedure ComputeLocalTetraStiffnessMatrix(hexahedron E, tetrahedron T, weights λ, local matrix L)
  Assume $T = \{a_0a_1, \ldots, d_0d_1\}$
  RetrieveReferenceJacobianFromLookupTable(J^{REF}, E, T)
  $B \leftarrow \mathrm{Id}_{3\times 3}$
  for each vertex $(j_k j_l) \in T$ **do**
    if $j_k \neq j_l$ **then** ▷ Vertex is virtual node
      $B_{jj} \leftarrow 2\lambda_{j,0}$ ▷ Store scaling factor in matrix
    end if
  end for
  $L \leftarrow \mathrm{vol}(T)\det(B) J^{\text{REF}} B^{-1} A B^{-1} (J^{\text{REF}})^T$ ▷ Scale Jacobian and compute stiffness matrix
end procedure

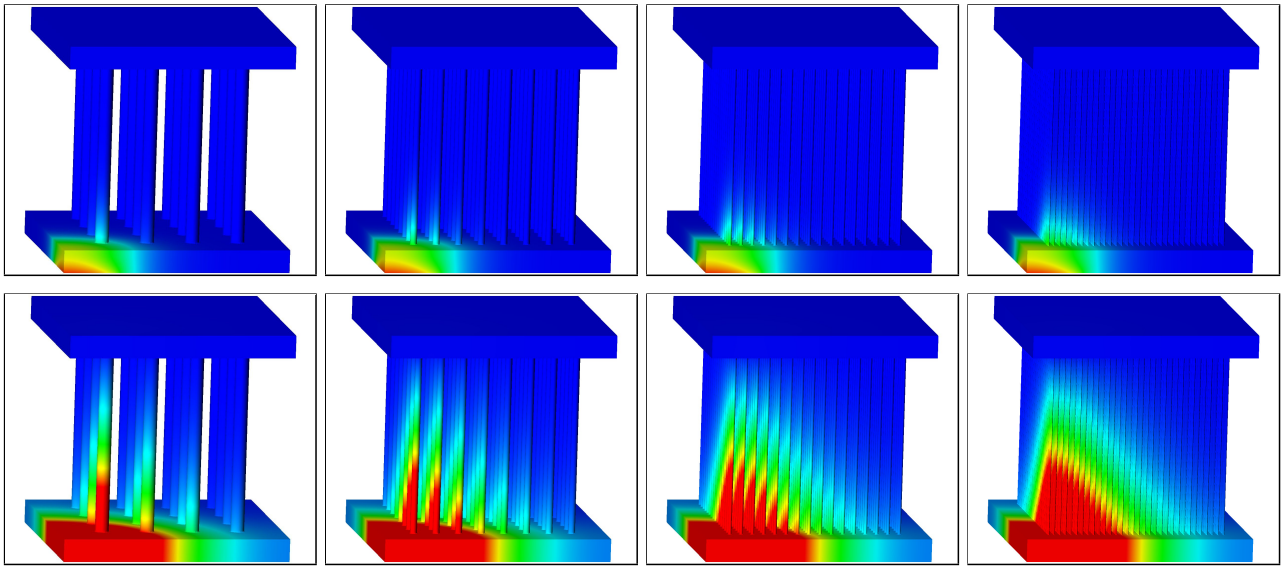

Fig. 7. On a sample geometry consisting of parallel pillars we show the diffusion of heat. From left to right the complexity increases (4×4 up to 32×32 pillars, resolved on a 257^3 grid). In the top row results are depicted at time $t = 5$ and in the bottom row at time $t = 30$. A color ramp from blue to green to red renders the temperature u on the boundary of the domain Ω_-. The number of virtual nodes is from left to right: 1 192 986, 1 875 326, 3 188 938 and 5 521 294, where the number of DOF is 4 350 226, 4 545 512, 4 990 894 and 6 010 360, respectively.

sity structure of prolongation and restriction, the matrix product (10) can be computed efficiently. For each entry of the coarse grid operator A_{l-1}, at most 15 entries of $R_{l\to l-1}$ and $P_{l-1\to l} = R^T_{l\to l-1}$ need to be considered (for two multiplications and one addition). Thus the total workload for generating A_{l-1} from A_l is bounded by $3 \cdot 15^2 \cdot n_l = O(n_l)$ flops. Finally let us mention that for the block Gauss-Seidel iteration in the multigrid method for vector valued problems, we can improve the computational efficiency by caching the inverses of 3×3 matrices arising. This, however, is at the cost of a higher memory requirement.

4.5 Computational and storage costs

The computation and storage of lookup table adds a fixed amount to the computational and storage costs. In fact, it is possible to compute the lookup table once for all times and then reload the corresponding data from a file. In our implementation the lookup table is of size

2.261 KB and it takes less than 0.05 seconds to build it on a standard PC.

For storing additional information for virtual nodes as needed by our CFE algorithm presented in Section 2.3, the implementation uses approximately 220 Bytes per virtual node. Thus, it is possible to run simulations on standard PCs, even for large data-sets and geometries inducing many virtual nodes.

5 Applications

5.1 Heat diffusion

As a first model problem we consider the heat diffusion in a homogeneous material. Thus, we consider the heat equation on $D \subset \Omega$ and search for $u : [0,T] \times D \to \mathbb{R}$ such that

$$\begin{aligned} \partial_t u - \lambda \Delta u &= 0 && \text{in } [0,T] \times D, \\ \partial_\nu u &= 0 && \text{on } [0,T] \times \partial D, \\ u(0,\cdot) &= g && \text{in } D, \end{aligned}$$

for a diffusion constant $\lambda > 0$ and initial data $g \in L^2(D)$. Let us suppose we have determined a level-set function ϕ describing D such that $D = \Omega_-$ (cf. Section 2.2). Let us furthermore discretize the domain Ω and its subdomains $\Omega_\pm$ by the algorithm described in Section 2.2. We discretize the problem in time by a backward Euler approach with time step size $\tau > 0$. For the spatial discretization we employ the standard Galerkin procedure as described in Section 2.3. This leads to the bilinear forms

$$\begin{aligned} a(u,v) &:= \int_{\Omega_-^{\boxtimes}} \left(uv + \tau \lambda \nabla u \cdot \nabla v \right) dx, \\ f(u,v) &:= \int_{\Omega_-^{\boxtimes}} uv \, dx. \end{aligned}$$

Taking the ansatz space $\mathcal{V}^h_{\mathrm{CFE}}$ into account as introduced in Section 2.2, and starting with $u^0 = g$ we have to solve

$$a(u^n, v) = f(u^{n-1}, v)$$

in each timestep n and for all test-functions $v \in V^h_{\mathrm{CFE}}$.

In Figure 7 we show the result of the computation of the heat equation on a sample geometry consisting of an array of pillars. Note that—due to the one-dimensional pillar-structure—the heat isosurfaces do not approximate spheres but diamond-shapes (i. e. spheres in the l_1-norm). In the finest computations the number of regular DOF and virtual nodes is approximately equal. Thus, in this case the CFE-approach needs only half the number of DOF compared to a standard FE approach on the virtual grid.

Moreover, in Figure 8 the computational domain is based on segmented CT-data of the human liver. We compute heat diffusion on the segmented liver from which the segmented vascular tree has been removed. This application scenario is of interest in the so called ablation therapy of liver cancer, where energy is applied inside the organ to cause a local heating (cf. Section 5.3). Transport of heat on the vascular system is not considered here.

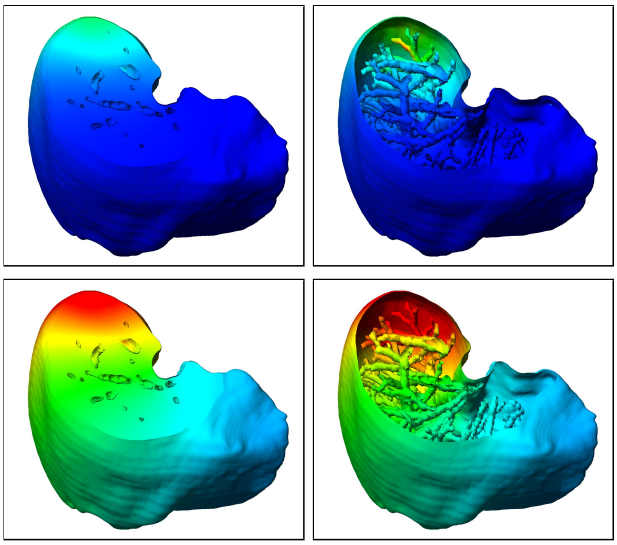

Fig. 8. The diffusion of heat is depicted on a domain which has been segmented from CT-data of a human liver. The upper row shows the result at time $t = 5$ whereas the bottom row shows the result at time $t = 30$. Again, color indicates the temperature (cf. Figure 7). To better visualize the internal structures of the domain we depict the temperature on a cut through the domain in the left column. In the right column the internal vascular structures are shown, which are not part of the computational domain.

5.2 Elasticity

As a second model problem, we consider linear elasticity in the case of an isotropic material. Thus we are dealing with the vector valued problem on $D \subset \Omega$: Find $u : D \to \mathbb{R}^3$ such that

$$\begin{aligned} -\mathrm{div}\, \mathcal{C}\mathcal{E}(u) &= f && \text{in } D \\ \nu \cdot \mathcal{C}\mathcal{E}(u) &= 0 && \text{on } \partial D \setminus \Gamma_0 \\ u(x) &= 0 && \text{on } \Gamma_0 \end{aligned} \tag{11}$$

for $f \in \mathbb{R}^3$ where $\mathcal{C}$ is the elasticity tensor and $\mathcal{E}(u) := \frac{1}{2}(\nabla u^T + \nabla u)$ the linearized strain tensor and $\Gamma_0 \subset \overline{\Omega}$ has positive surface measure. In the case of an isotropic material, we have $\mathcal{C}_{ijmn} = \lambda \delta_{ij}\delta_{mn} + \mu(\delta_{im}\delta_{jn} + \delta_{in}\delta_{jm})$ with the so called Lamé constants λ and μ. In this case, the PDE (11) becomes

$$-\mathrm{div}\Big(\lambda (\mathrm{tr}\, \mathcal{E}(u)) \mathrm{Id} + 2\mu \mathcal{E}(u) \Big) = f.$$

Again let us suppose we have a level-set function ϕ describing D such that $D = \Omega_-$ (cf. Section 2.2). We follow the general discretization approach presented in Section 2.2. In this case, the resulting bilinear and linear

forms are given by

$$a(u,v) := \lambda \int_{\Omega_-^{\boxtimes}} \operatorname{div} u \operatorname{div} v \, dx + 2\mu \int_{\Omega_-^{\boxtimes}} \mathcal{E}(u) : \mathcal{E}(v) \, dx$$

$$l(v) := \int_{\Omega_-^{\boxtimes}} f \cdot v \, dx.$$

Since we have a vector valued problem here, we take the ansatz space $(\mathcal{V}_0^h)^3$ into account. Thus, the basis functions for the vector valued problem are $\psi_i^{\mathrm{CFE}} e_j \in (\mathcal{V}_0^h)^3$ with $j = 0, 1, 2$, where $\{e_0, e_1, e_2\}$ is the standard basis of $\mathbb{R}^3$. Note that this leads to three times as many DOF compared to the scalar problem.

In Figure 9 we show the elastic deformations of the pillar-structures which we have already used for the scalar test problem in Figure 7. We apply a force that shears the upper boundary of the volume to the right, whereas the lower boundary plate is kept fixed.

In Figures 10 and 11 we show the elastic deformation of the different samples of the internal structure (spongiosa) of a porcine vertebral bone. The corresponding level-set function ϕ has been obtained by a segmentation of micro-CT image data. Again we apply a force f that shears the upper boundary of the bone volume to the right. The lower boundary plate is kept fixed. The red boxes in the lower right image of Figure 10 correspond to zoomed version in Figure 11. For further details on elasticity computation for human vertebra we refer to [43].

5.3 Ablation in liver cancer therapy

Let us finally consider a scalar model problem with non-continuous coefficients to demonstrate the use of the composite finite element method in this case. The explanation is rather brief here and we refer the reader to [37] for details.

We consider the electrostatic equation in a domain containing two different materials. Consequently, the electric conductivity σ is discontinuous across the material interface γ. It is well known that the solution is continuous across the interface ($[u]_\gamma = 0$), its gradient is continuous in tangential to the interface ($[\partial_\tau u]_\gamma = 0$) but discontinuous in normal direction, satisfying the jump condition

$$[\sigma \partial_\nu u]_\gamma = 0 \qquad \text{a. e. on } \Gamma. \tag{12}$$

For this problem we use the virtual grid $\mathcal{G}^{\triangle}$ to resolve $\gamma^{\triangle}$ and to define a CFE space whose DOF lie on the vertices of the tetrahedral grid $\mathcal{G}^{\boxtimes}$ and which fulfil the jump condition (12) in an approximate sense.

So we assume $\overline{\Omega} = \overline{\Omega}_- \cup \overline{\Omega}_+$, $\Gamma_0 \subset \partial\Omega$ to be of positive surface measure and we consider the following scalar valued problem: Find $u : \Omega \to \mathbb{R}$ such that

$$\begin{aligned} -\operatorname{div}(\sigma(x)\nabla u) &= 0 && \text{in } \Omega \\ \nu \cdot \nabla u &= 0 && \text{on } \partial\Omega \setminus \Gamma_0 \\ u(x) &= g(x) && \text{on } \Gamma_0 \end{aligned} \tag{13}$$

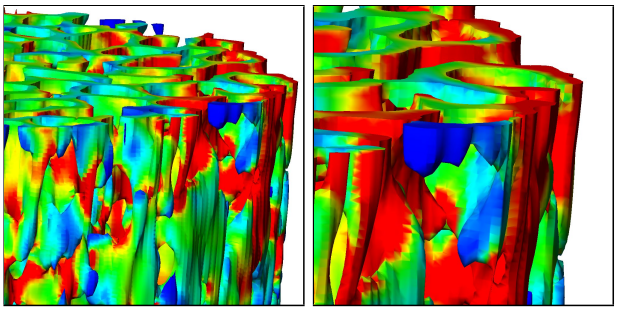

Fig. 11. Detailed view of the upper right corner of the elastic stress plot in Figure 10 in the lower right sample.

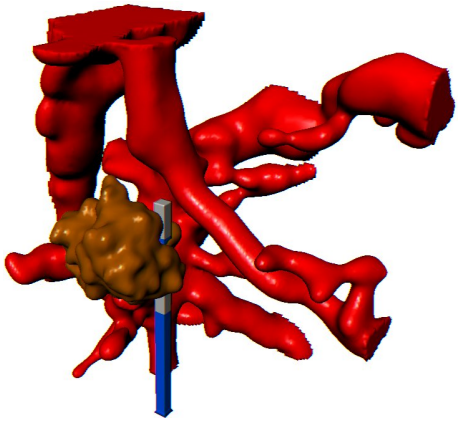

Fig. 12. The geometrical configuration for our computations is shown. The domain contains a vascular structure (red) and a tumor (brown) which have complicated shape. These structures form the set Ω_-. An RF-probe (blue) with two electrodes (gray) is placed into the vicinity of the domain. The electrodes form the Dirichlet boundary $\Gamma = \Gamma_- \cup \Gamma_+$. The vascular structure as well as the tumor have been segmented from image data acquired with CT.

for boundary data $g \in L^2(\Gamma_0)$. Here, the electric conductivity σ is uniformly positive and piecewise smooth. In our model problem, we consider a conductivity piecewise constant in the material regions $\Omega_\pm$, i. e.

$$\sigma(x) = \begin{cases} \sigma_+ & \text{if } x \in \Omega_+, \\ \sigma_- & \text{else.} \end{cases}$$

As usual we transform (13) to homogeneous boundary conditions and consider the weak form

$$\int_\Omega \sigma \, \nabla u \cdot \nabla v \, dx = \int_\Omega \sigma \, \nabla \hat{g} \cdot \nabla v \, dx,$$

for a test function $v \in H_0^1(\Omega)$ with appropriate boundary values on Γ_0 where $\hat{g} \in H^1(\Omega)$ is an extension of g onto whole Ω.

In analogy to the procedure in Section 2, we discretize the domain $\Omega = \Omega_+ \cup \Omega_-$ by the virtual grid $\mathcal{G}^{\triangle}$ such that the approximated interface $\gamma^{\triangle}$ is resolved. We build a set of composite basis functions on the original tetra-

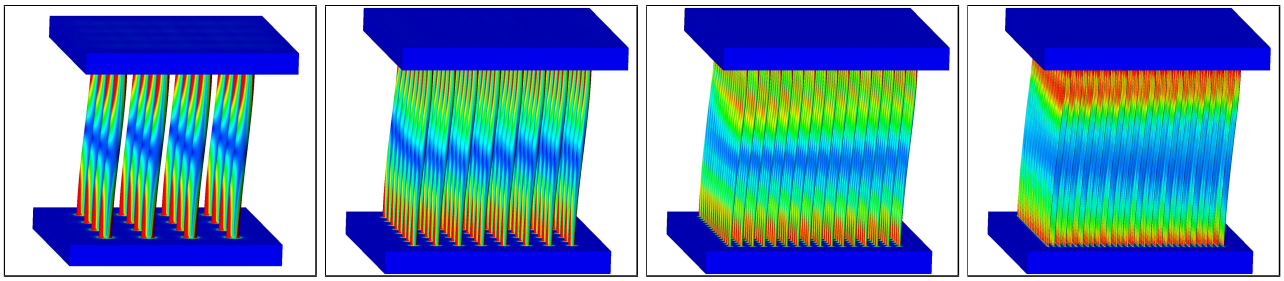

Fig. 9. The elastic deformation of a sample structure is depicted. From left to right the number of pillars increases (4×4 up to 32×32 resolved on a regular grid with 257^3 grid points). A color ramp from blue to green to red indicates the von Mises surface stress.

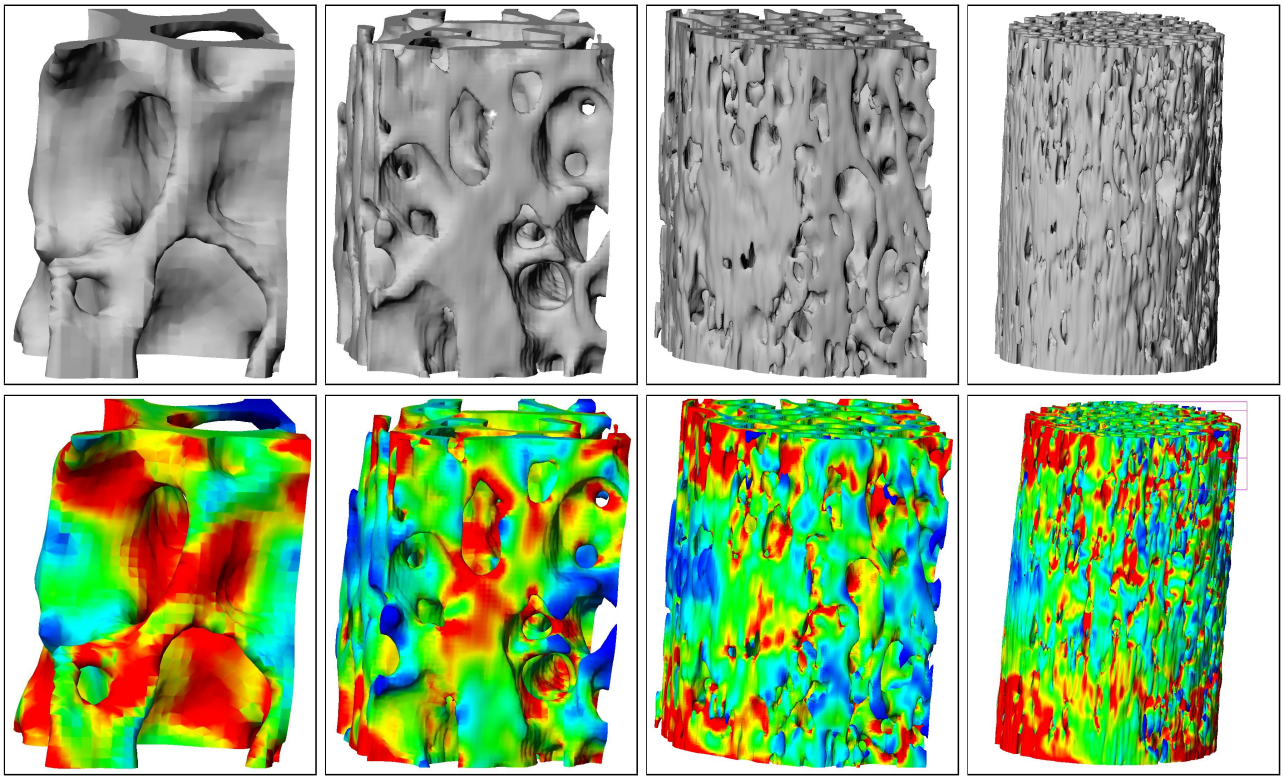

Fig. 10. The elastic deformation of the internal structure of a porcine vertebral bone is depicted. From left to right the level of detail (and thus the grid-depth) increases ($h = 2^{-j}, j = 5, \ldots, 8$). The top row shows the original non-deformed structure, whereas the bottom row shows the deformed structure. A color ramp from blue to green to red indicates the von Mises surface stress. The number of virtual nodes was from left to right: 16 887, 142 234, 1 004 417, and 5 606 274. The corresponding number of DOF was 27 921, 243 477, 1 847 286 and 10 124 160, respectively.

hedral grid $\mathcal{G}^{\boxtimes}$ in the spirit of (5), i. e. such that

$$\psi_i^{\mathrm{CFE}} := \sum_{l=1}^{c_l} \mu_{i,l} \psi_{i_l}^{\triangle}$$

The set of weights $\{\mu_{i,l}\}$ is chosen such that the linear combination of fine grid basis functions in the vicinity of $\gamma^{\triangle}$ respects the appropriate jump condition (12). For details we refer the reader to [37,41].

We proceed with the space spanned by the CFE functions

$$\mathcal{V}^h := \mathrm{span}\{\psi_i^{\mathrm{CFE}}\}$$

which has fewer DOF than ${\mathcal{V}^{\triangle}}^h$ but which still resolves the interface $\gamma^{\triangle}$ in the sense that its functions satisfy the jump condition (12). The standard Galerkin approach with the ansatz $u \in \mathcal{V}^h$ delivers a linear system of equations which can be solved numerically.

Obviously both CFE approaches—the one for domains with complicated boundary and the one for discontinuous coefficients—can be combined. The algorithm for matrix assembly differs from Algorithm 2 only in the weights which are used.

In Figures 14 and 15 we show results where structures Ω_- of the computational domain have been segmented from CT-data of the human liver. The geometrical configuration is depicted in Figure 12: Ω_- has two components and consists of a segmented vascular system and a tumor. The boundary Γ_0 consists of two separate

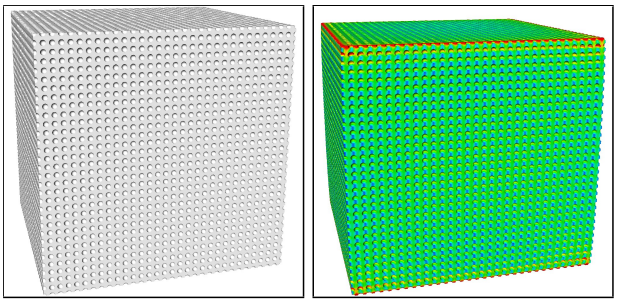

Fig. 13. A "swiss cheese" type domain with 32^3 holes in undeformed and deformed configuration, using 49 448 025 DOF.

small cylinders $\Gamma_0 = \Gamma_+ \cup \Gamma_-$ such that $g(\Gamma_+) = +1\,\mathrm{V}$ and $g(\Gamma_-) = -1\,\mathrm{V}$. Such a configuration arises from the simulation of the electrostatic field of a radio-frequency probe in RF-ablation [25]. RF-ablation is a therapy for the thermal destruction of lesions/tumors in e. g. the human liver. Thereby, one considers a second parabolic PDE for the energy-density in the biological tissue

$$\partial_t(c\rho T) - \mathrm{div}(\lambda \nabla T) = \rho |\nabla u|^2. \qquad (14)$$

The density ρ, the heat capacity c and the heat conductivity λ are piecewise constant in the regions of tumor, blood-vessel and native tissue.

In Figure 14 we depict isosurfaces of the potential u as the solution of (13). In this computation we set $\sigma_+ = 1\,\mathrm{S/m}$ and $\sigma_- = 0.1\,\mathrm{S/m}$. It is clearly visible from the pictures how the different conductivities of the tissue leads to iso-surfaces with kinks at the interfaces. Finally, in Figure 15 we depict isosurfaces of the solution temperature T. Here we choose $c_\pm = 1\,\mathrm{J/kg\,K}$, $\rho_\pm = 1\,\mathrm{kg/m^3}$ and $\lambda_+ = 1\,\mathrm{W/Km}$, $\lambda_- = 0.1\,\mathrm{W/Km}$.

5.4 Multigrid Performance

As a stopping criterion for the multigrid solver we considered the reduction of the norm of the residual by 10^{-8} relative to the norm of the initial residual. In the liver example in Figure 8 (resolved on a 129^3 grid), the first diffusion time step takes 12 V cycles, the convergence rate in the last cycle is 0.219 and one cycle requires 4.3 seconds of cputime. The corresponding numbers for the 4×4 and 32×32 pillar example (leftmost and rightmost picture) in Figure 7 (resolved on a 257^3 grid) are 12 and 142 V cycles, convergence rates 0.251 and 0.944, and 37 and 40 seconds, respectively.

For the elasticity example in Figure 13 (resolved on a 257^3 grid), we need 19 V cycles (taking 579 seconds each) and obtain a final onvergence rate of 0.580. The corresponding numbers for the bone examples on 33^3 and 129^3 grids (left and second-to-right example in Figure 10) are 71 and 384 V cycles (0.5 and 56 seconds each) with final convergence rates 0.860 and 0.978.

The poor performance of the multigrid solver in some of the examples is due to the shape of the geometry. If they are shaped such that supports of coarse grid basis functions have at least two connected components

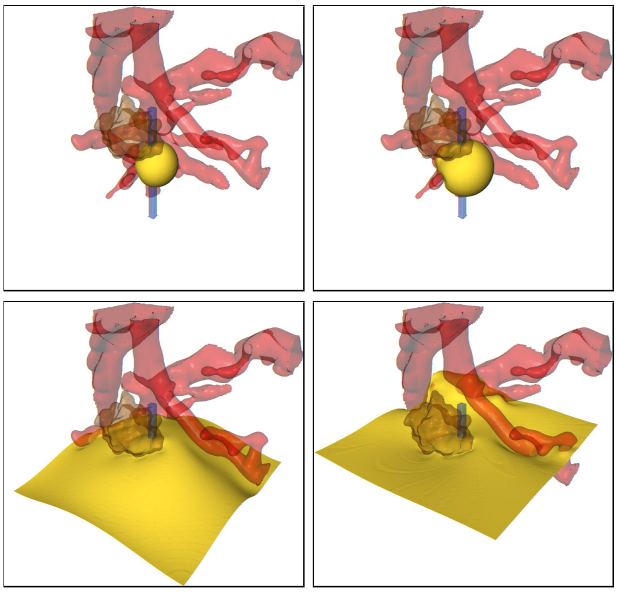

Fig. 14. Numerical results for the electrostatic potential are shown. From top left to bottom right isosurfaces for the values 0, 0.05, 0.14, and 0.16 (Volt) of the potential are shown. To underline how the non-continuous conductivity leads to kinks in the isosurfaces, the images are supplemented with a transparent rendering of the geometrical configuration of the domain (cf. Figure 12.)

of large geodesic distance in the object, our coarsening scheme produces poor coarse grid corrections. An improved coarsening scheme splitting and recombining coarse grid DOF is currently being investigated.

6 Conclusions

We have presented an approach for the solution of partial differential equations on complicated computational domains where the geometry is defined by previously segmented image-data. Starting from a hexahedral discretization induced by the underlying image data, the method generates a virtual tetrahedral grid that resolves the domain boundaries or approximated interior interfaces. The virtual grid is used to define a composite finite element space, whose functions respect the complicated domain boundaries or jump sets of coefficients.

The resulting composite finite element approach retains the efficiency of a finite element methods on structured hexahedral grids far from interfaces or domain boundaries. In the vicinity of such structures the efficiency is increased by hashing the local intersection topology as well as additional information associated with virtual nodes. The resulting linear systems of equations can be solved by standard numerical solvers. A suitable multigrid method has been presented which yield convergence rates independent of the grid size and hierarchical grid level. A further analysis and improvement of the multigrid method presented here as well as a multigrid method for the CFE in the case of discontinuous coefficients will be discussed in a forthcoming paper.

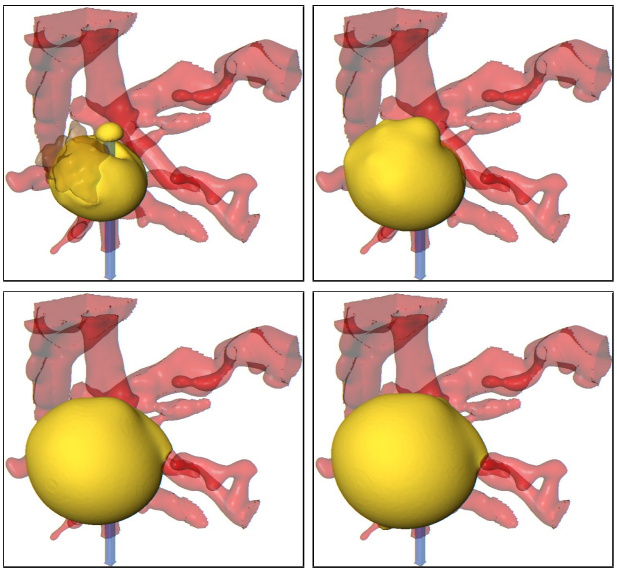

Fig. 15. Temperature isosurfaces are rendered at different times of a heat diffusion simulation. From top left to bottom right the isosurface $T = 333\,K$ ($\approx 60°C$) is rendered at times 0.01, 0.03, 0.06, and 0.1 (seconds) respectively. Again the images are supplemented with a transparent rendering of the geometrical configuration of the domain.

Acknowledgements

Lars Ole Schwen was supported by the DFG project "Multiscale Simulation and Validation of the Elastic Microstructures of Vertebral Bodies".

The authors would like to thank Dipl.-Ing. Uwe Wolfram from the Institute of Orthopaedic Research and Biomechanics, University of Ulm, for providing the data sets shown in Figure 10. Furthermore we acknowledge Ulrich Weikard for fruitful discussions on the CFE methods.

References

1. L. Adams and Z. Li. The immersed interface / multigrid methods for interface problems. *SIAM Journal on Scientific Computing*, 24(2):463–479, 2002.
2. L. Ambrosio and V. M. Tortorelli. On the approximation of free discontinuity problems. *Bollettino dell'Unione Matematica Italiana, Sezione B*, 6(7):105–123, 1992.
3. I. Babuška and J. Melenk. The partition of unity method. *International Journal for Numerical Methods in Engineering*, 40:727–758, 1997.
4. T. Belytschko, N. Moës, S. Usui, and C. Parimi. Arbitrary discontinuities in finite elements. *International Journal for Numerical Methods in Engineering*, 50(4):993–1013, 2001.
5. R. P. Beyer and R. J. LeVeque. Analysis of a one-dimensional model for the immersed boundary method. *SIAM Journal on Numerical Analysis*, 29(2):332–364, 1992.
6. F. Bornemann and C. Rasch. Finite-element discretization of static hamilton-jacobi equations based on a local variation principle. *Computing and Visualization in Science*, 9(2):57–69, 2004. arXiv:math.NA/0403517.
7. D. Calhoun. A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions. *Journal of Computational Physics*, 176:231–275, 2002.
8. D. Calhoun and R. J. LeVeque. A Cartesian grid finite-volume method for the advection-diffusion equation in irregular geometries. *Journal of Computational Physics*, 157:143–180, 2000.
9. V. Caselles, F. Catté, T. Coll, and F. Dibos. A geometric model for active contours in image processing. *Numerische Mathematik*, 66:1–31, 1993.
10. V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *International Journal of Computer Vision*, 22(1):61–79, 1997.
11. T. Chan and L. Vese. A level set algorithm for minimizing the Mumford-Shah functional in image processing. In *Proceedings of the 1st IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 161–168, 2001.
12. S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng. Sliver exudation. *Journal of the ACM*, 47(5):883–904, September 2000.
13. P. G. Ciarlet. *The finite element method for elliptic problems*. Number 40 in Classics in applied mathematics. SIAM, 2002.
14. D. Cremers, C. Schnörr, and J. Weickert. Diffusion-snakes: Combining statistical shape knowledge and image information in a variational framework. *IEEE Workshop on Variational and Levelset Methods*, pages 137–144, 2001.
15. P. Deuflhard, M. Weiser, and M. Seebaß. A new nonlinear elliptic multilevel FEM applied to regional hyperthermia. *Computing and Visualization in Science*, 3(3):115–120, 2000.
16. M. Droske, T. Preusser, and M. Rumpf. A multilevel segmentation method. In *Proceedings of Vision, Modeling and Visualization (VMV)*, pages 327–336, Saarbrücken, Germany, 2000.
17. N. Frauböse and S. Sauter. Composite finite elements and multi-grid. part I: Convergence theory in 1-d. In *Proceedings of the 17th GAMM-Seminar Leipzig on Construction of Grid Generation Algorithms*, pages 69–86, 2001.
18. M. Goméz-Benito, J. García-Aznar, and M. Doblaré. Finite element prediction of proximal femoral fracture patterns under different loads. *Journal of Biomechanical Engineering*, 127(1):9–14, 2005.
19. W. Hackbusch. *Multi-Grid Methods and Applications*, volume 4 of *Springer Series in Computational Mathematics*. Springer, 1985.
20. W. Hackbusch and S. Sauter. Composite finite elements for the approximation of PDEs on domains with complicated micro-structures. *Numerische Mathematik*, 75:447–472, 1997.
21. H. K. Hahn, M. G. Lentschig, M. Deimling, B. Terwey, and H.-O. Peitgen. MRI-based volumetry of intra- and extracerebral liquor spaces. In *CARS*, pages 401–407, 2001.
22. K. Höllig, U. Reif, and J. Wipper. Weighted extended B-spline approximation of Dirichlet problems. *SIAM Journal on Numerical Analysis*, 39(2):442–462, 2001.
23. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1988.

24. R. Kornhuber, R. Krause, O. Sander, P. Deuflhard, and S. Ertel. A monotone multigrid solver for two body contact problems in biomechanics. *Computing and Visualization in Science*, 2006. accepted for publication.
25. T. Kröger, I. Altrogge, T. Preusser, et al. Numerical simulation of radio frequency ablation with state dependent material parameters in three space dimensions. In *MICCAI 2006*, volume 4191 of *LNCS*, pages 380–388. Springer, 2006.
26. J. M. Kuhnigk, V. Dicken, L. Bornemann, A. Bakai, D. Wormanns, S. Krass, and H. O. Peitgen. Morphological segmentation and partial volume analysis for volumetry of solid pulmonary lesions in thoracic CT scans. *IEEE Transactions on Medical Imaging*, 25(4):417–434, 2006.
27. R. J. LeVeque and Z. L. Li. The immersed interface method for elliptic equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, 31(4):1019–1044, 1994.
28. Z. Li. A fast iterative algorithm for elliptic interface problems. *SIAM Journal on Numerical Analysis*, 35(1):230–254, 1998.
29. Z. Li. The immersed interface method using a finite element formulation. *Applied Numerical Mathematics*, 27:253–267, 1998.
30. Z. Li, T. Lin, and X. Wu. New Cartesian grid methods for interface problems using the finite element formulation. *Numerische Mathematik*, 1996(1):61–98, November 2003.
31. W. E. Lorensen and H. E. Cline. Marching cube: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4), 1987.
32. J. M. Melenk and I. Babuška. The partition of unity finite element method. Research Report 96-01, Eidgenössische Technische Hochschule Zürich, Seminar für angewandte Mathematik, January 1996.
33. H. Müller and M. Wehle. Visualization of implicit surfaces using adaptive tetrahedrizations. In *Proceedings of Scientific Visualization Conference (Dagstuhl '97)*, 1997.
34. S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.
35. J. Parvizian, A. Düster, and E. Rank. Finite cell method. *Computational Mechanics*, Online First, 2007.
36. T. Preusser and M. Rumpf. An adaptive finite element method for large scale image processing. *Journal of Visual Communication and Image Representation*, 11:183–195, 2000.
37. T. Preusser, M. Rumpf, S. Sauter, L. O. Schwen, et al. Three-dimensional composite finite elements for scalar problems with jumping coefficients. 2007. in preparation.
38. T. Preusser, M. Rumpf, and L. O. Schwen. Finite element simulation of bone microstructures. In *Proceedings of the 14th Finite Element Workshop*. University of Ulm, July 2007. To appear.
39. M. Rech, S. Sauter, and A. Smolianski. Two-scale Composite Finite Element method for the Dirichlet problem on complicated domains. *Numerische Mathematik*, 102, 2006.
40. G. Russo and P. Smereka. A remark on computing distance functions. *Journal of Computational Physics*, 163(1):51 – 67, 2000.
41. S. A. Sauter and R. Warnke. Composite finite elements for elliptic boundary value problems with discontinuous coefficients. *Computing*, 77:29–55, 2006.
42. A. Schenk, G. Prause, and H.-O. Peitgen. Efficient semi-automatic segmentation of 3D objects in medical images. In *Proceedings of MICCAI*, volume 1935 of *LNCS*, pages 186–195. Springer, 2000.
43. L. O. Schwen et al. Computing elastic deformation of vertebrae using composite finite elements. in preparation, 2007.
44. F. Segonne, A. M. Dale, E. Busa, M. Glessner, D. Salat, H. K. Hahn, and B. Fischl. A hybrid approach to the skull stripping problem in MRI. *Neuroimage*, 22(3):1060–1075, 2004.
45. J. A. Sethian and A. Wiegmann. Structural boundary design via level set and immersed interface methods. *Journal of Computational Physics*, 163(2):489 – 528, 2000.
46. F. L. Stazi, E. Budyn, J. Chessa, and T. Belytschko. An extended finite element method with higher-order elements for curved cracks. *Computational Mechanics*, 31:38–48, 2003.
47. G. Strang and G. J. Fix. *An Analysis of the Finite Element Method*. Wellesey-Cambridge Press, 1973.
48. T. Strouboulis, K. Copps, and I. Babuška. The generalized finite element method. *Computer Methods in Applied Mechanics and Engineering*, 190:4081–4193, 2001.
49. S.-H. Teng and C. W. Wong. Unstructured mesh generation: Theory, practice and applications. *International Journal of Computational Geometry & Applications*, 10(3):227–266, 2000.
50. A. Wiegmann and K. P. Bube. The immersed interface method for nonlinear differential equations with discontinuous coefficients and singular sources. *SIAM Journal on Numerical Analysis*, 35(1):177–200, Feb. 1998.
51. A. Wittek, R. Kikinis, S. K. Warfield, and M. K. Brain shift computation using a fully nonlinear biomechanical model. In *Proceedings of MICCAI*, LNCS, pages 583–590. Springer, 2005.
52. C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, 1998.

Bestellungen nimmt entgegen:

Institut für Angewandte Mathematik
der Universität Bonn
Sonderforschungsbereich 611
Wegelerstr. 6
D - 53115 Bonn

Telefon: 0228/73 4882
Telefax: 0228/73 7864
E-mail: link@wiener.iam.uni-bonn.de http://www.sfb611.iam.uni-bonn.de/

Verzeichnis der erschienenen Preprints ab No. 355

355. Löbach, Dominique: On Regularity for Plasticity with Hardening

356. Burstedde, Carsten; Kunoth, Angela: A Wavelet-Based Nested Iteration – Inexact Conjugate Gradient Algorithm for Adaptively Solving Elliptic PDEs

357. Alt, Hans-Wilhelm; Alt, Wolfgang: Phase Boundary Dynamics: Transitions between Ordered and Disordered Lipid Monolayers

358. Müller, Werner: Weyl's Law in the Theory of Automorphic Forms

359. Frehse, Jens; Löbach, Dominique: Hölder Continuity for the Displacements in Isotropic and Kinematic Hardening with von Mises Yield Criterion

360. Kassmann, Moritz: The Classical Harnack Inequality Fails for Non-Local Operators

361. Albeverio, Sergio; Ayupov, Shavkat A.; Kudaybergenov, Karim K.: Description of Derivations on Measurable Operator Algebras of Type I

362. Albeverio, Sergio; Ayupov, Shavkat A.; Zaitov, Adilbek A.; Ruziev, Jalol E.: Algebras of Unbounded Operators over the Ring of Measurable Functions and their Derivations and Automorphisms

363. Albeverio, Sergio; Ayupov, Shavkat A.; Zaitov, Adilbek A.: On Metrizability of the Space of Order-Preserving Functionals

364. Alberti, Giovanni; Choksi, Rustum; Otto, Felix: Uniform Energy Distribution for Minimizers of an Isoperimetric Problem Containing Long-Range Interactions

365. Schweitzer, Marc Alexander: An Adaptive hp-Version of the Multilevel Particle-Partition of Unity Method

366. Frehse, Jens; Meinel, Joanna: An Irregular Complex Valued Solution to a Scalar Linear Parabolic Equation

367. Bonaccorsi, Stefano; Marinelli, Carlo; Ziglio, Giacomo: Stochastic FitzHugh-Nagumo Equations on Networks with Impulsive Noise

368. Griebel, Michael; Metsch, Bram; Schweitzer, Marc Alexander: Coarse Grid Classification: AMG on Parallel Computers

369. Bar, Leah; Berkels, Benjamin; Rumpf, Martin; Sapiro, Guillermo: A Variational Framework for Simultaneous Motion Estimation and Restoration of Motion-Blurred Video; erscheint in: International Conference on Computer Vision 2007

370. Han, Jingfeng; Berkels, Benjamin; Droske, Marc; Hornegger, Joachim; Rumpf, Martin; Schaller, Carlo; Scorzin, Jasmin; Urbach Horst: Mumford–Shah Model for One-to-one Edge Matching; erscheint in: IEEE Transactions on Image Processing

371. Conti, Sergio; Held, Harald; Pach, Martin; Rumpf, Martin; Schultz, Rüdiger: Shape Optimization under Uncertainty – a Stochastic Programming Perspective

372. Liehr, Florian; Preusser, Tobias; Rumpf, Martin; Sauter, Stefan; Schwen, Lars Ole: Composite Finite Elements for 3D Image Based Computing