Eurographics Workshop on Visual Computing for Biomedicine (2008)
Charl Botha and Gordon Kindlmann and Wiro Niessen and Bernhard Preim (Editors)

Computational Steering for Patient-Specific Implant Planning in Orthopedics

Christian Dick[1], Joachim Georgii[1], Rainer Burgkart[2], and Rüdiger Westermann[1]

[1]Computer Graphics and Visualization Group, Technische Universität München, Germany
[2] Klinik u. Poliklinik für Orthopädie u. Sportorthopädie am Klinikum Rechts der Isar, Technische Universität München, Germany

Abstract
Fast and reliable methods for predicting and monitoring in-vivo bone strength are of great importance for hip joint replacement. To avoid adaptive remodeling with cortical thinning and increased porosity of the bone due to stress shielding, in a preoperative planning process the optimal implant design, size, and position has to be determined. This process involves interactive implant positioning within the bone as well as simulation and visualization of the stress within bone and implant due to exerting forces. In this paper, we present a prototype of such a visual analysis tool, which, to our best knowledge, provides the first computational steering environment for optimal implant selection and positioning. This prototype considers patient-specific biomechanical properties of the bone to select the optimal implant design, size, and position according to the prediction of individual load transfer from the implant to the bone. We have developed a fast and stable multigrid finite-element solver for hexahedral elements, which enables interactive simulation of the stress distribution within the bone and the implant. By utilizing a real-time GPU-method to detect elements that are covered by the moving implant, we can automatically generate computational models from patient-specific CT scans in real-time, and we can instantly feed these models into the simulation process. Hardware-accelerated volume ray-casting, which is extended by a new method to accurately visualize sub-hexahedron implant boundaries, provides a new quality of orthopedic surgery planning.

Categories and Subject Descriptors (according to ACM CCS): G.1.8 [Numerical Analysis]: Partial Differential Equations - Finite Element Methods, Multigrid and Multilevel Methods I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling - Physically Based Modeling I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Raytracing

1. Introduction and Related Work

Methods for predicting and monitoring in-vivo bone strength are of great importance in clinical applications such as fracture fixation or endoprostheses for hip joint replacement. Such procedures call for highly efficient and reliable analysis tools, which allow during a preoperative design loop to find the optimal design, size, and position of an implant by matching its mechanical properties with those of the individual bone. The clinical relevance of such a planning approach is due to the well known fact that an essential determinant factor for the long term stability of an implant is a physiological load transmission to the adjacent bone stock. For instance, an optimized femoral stem should provide bone stress patterns that closely replicate the preoperative physiological stress state in order to prevent stress shielding with the consecutive effects of osteopenia, fracture and aseptic loosening [OH78, SV02]. The main objective is thus to simulate the mechanical response of patient-specific bone to a load that is applied to the implant, and to find of all possible implant shapes and positions the one that results in the most natural stress distribution. The challenge in developing such an analysis tool results from the complexity of simulating the stress due to exerting forces in a physically correct way, and from the difficulty of considering a moving implant with specific mechanical properties in such a simulation. Furthermore, such a tool is highly demanding on advanced visualization technology because it requires simultaneous visualization of surface and dynamic volume structures at rates that allow for interactive monitoring and analysis.

Due to the aforementioned challenges, so far no surgical planning system recruits—to our knowledge—patient specific 3D data according to the individual mechanical properties of the involved bone and implant. In the majority of clinical centers worldwide the preoperative planning for the selection of an implant, e.g., an endoprosthesis for total hip joint replacement, is performed on an X-ray of the patient's hip joint. Therefore, in preparation for the surgical intervention, the surgeon is only able to select on a 2D X-ray the approximately best fitting size of an endoprosthesis using simple, transparent template sheets with the outlines of the implants. The drawbacks and limitations of such a two-dimensional approach are obvious, especially because rotational misalignment is not controlled and the position of the endoprosthesis is only revisable in the coronar plane.

To overcome these problems new approaches were pursued in the last years to use 3D information from patient specific CT data [HEPP01, BBL97, VCT*04]. With these virtual 3D planning systems the surgeon can visualize the position of the implant components three-dimensionally in the bone or can plan a custom-made implant for a specific bony contour. But these systems provide only geometrical data to the surgeon and he has to decide about the best implant design, size, and position according to his subjective medical experience. However, most important would be the additional knowledge of the patient-specific biomechanical properties of the affected bone to select the optimal implant according to the prediction of individual load transfer from the implant to the bone. This information is still missing in current 3D planning systems.

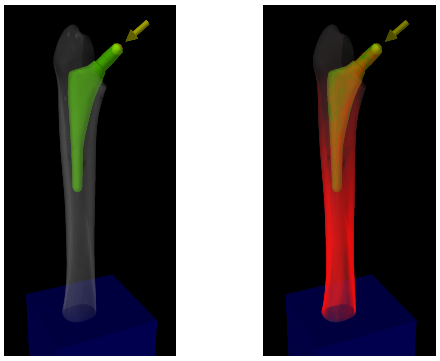


Figure 1: *Screenshots of the simulation environment: Left: Semi-transparent rendering of the femur and the implant (green color) as well as the volume rendering of the femur's interior structures support the positioning of the implant. Right: The load on the femur is indicated by the yellow arrow and sphere, and the resulting stresses in the volume are visualized with red color.*

In this paper, we present a first prototypical 3D planning system for hip joint replacement that addresses the aforementioned requirements (see Figure 1). Following previous work in orthopedics [BOM*07, TSH*07], we use 3D finite element (FE) analysis to predict loads in the proximal femur consisting of cortical stiff tissue and trabecular cellular spongy tissue. The physical model underlying our approach is based on linear elasticity and thus mimics the behavior of the bone at the macro-level during normal movements [KGW*94]. Mechanical properties of all femur regions are derived directly from the Hounsfield units of the cells in a measured CT scan as proposed in [KLS94, KF03].

In particular, we have developed a h-version FE method (h-FEM), which assigns material properties on a per-element basis in an orthogonal hexahedral 3D grid. At first glance such an approach seems to have substantial accuracy limitations compared to high-order p-FEM methods, which have recently shown good correlation between simulation results and real-world experiments [YTM07]. Such methods can model the variations of material properties within each element using high-order polynomials, thus reducing the number of elements to be used and allowing for the accurate representation of curved structure boundaries. However, we will show in this paper that accuracy limitations of our method can be avoided by using a highly optimized h-FEM method including a multigrid solver for improved convergence. Not only does such a method allow for interactive load simulations using reasonably sized finite element models, but we will also demonstrate that it can be used to simulate bone and implant loads on a desktop PC system at the resolution of the CT scan, i.e., one hexahedral finite element per CT voxel, and at simulation rates of less than one minute.

To further improve the proposed system towards a computational steering environment, we have integrated 3D visualization and interaction mechanisms. Specifically, we enable the user to interactively place the implant in the bone and we provide immediate visual feedback of simulated stresses via texture-based volume ray-casting. To the best of our knowledge, this is the first time that changes in bone stress due to variations in implant position and exerting external forces can be monitored and analyzed in quasi-real-time. This has been made possible by a fully automatic approach for the generation of a computational model based on the initial CT scan. A fast voxelization technique is employed to determine all CT voxels covered by the implant, which then get assigned the material properties of this implant. The FE analysis considers these voxels and simulates their interaction with the surrounding bone voxels under load. As the voxelization process is performed entirely on the GPU, it does not impose any performance constraints, and it allows integrating the determined sub-volume into the rendering process.

The remainder of this paper is organized as follows: In the following section, we give an overview of our prototypical 3D planning system, including a description of the different components used as well as their interplay in the current application. Next, we briefly outline the FE method we use, and we describe the novel extensions we have developed to make this method applicable for 3D planning of hip joint replacements. The following section addresses visualization issues, and it demonstrates the different functionalities we have integrated. We conclude the paper with a detailed analysis of all components of the 3D planning system, and we outline a number of issues that will be addressed in the near future.

2. System Overview

Figure 2 illustrates the different components of the 3D planning system for hip joint replacement. Components colored white indicate preprocesses, which are performed once before interactive implant placement and load monitoring starts. Components colored black indicate processes that are performed on the CPU and the GPU during runtime. In particular, while the simulation engine is entirely implemented on the CPU and sends a simulated scalar stress field to the GPU for rendering, the voxelization of the implant is performed on the GPU, which sends a binary solid voxel model of the implant in the current position to the CPU.

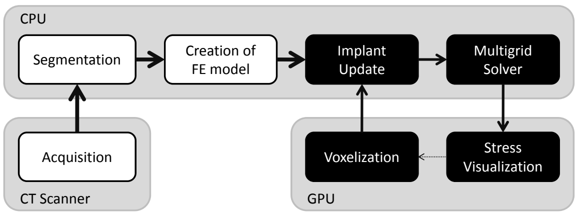


Figure 2: *Overview of the proposed surgery support system.*

Our 3D planning system operates on a voxel model of the bone, which is obtained from a high-resolution CT scan. The axial scan was performed on a Siemens Sensation Cardiac 64 with 1.0 mm slice thickness and 0.74 mm pixel size. Samples are given on a $256 \times 256 \times 512$ lattice. In a preprocess, the bone is segmented from the CT data, and dependent on the Hounsfield unit h, for each segmented voxel the Young's modulus E (i.e., the elastic material parameter) is assigned according to [KLS94, KF03] as:

$$E(h) = \begin{cases} 33900 \left(8.2106 \cdot 10^{-4} h + 0.057663\right)^{2.20} & h \leq 320 \\ 10200 \left(8.2106 \cdot 10^{-4} h + 0.057663\right)^{2.01} & h \geq 660 \\ 5307 \left(8.2106 \cdot 10^{-4} h + 0.057663\right) + 469 & 320 < h < 660 \end{cases} \quad [\text{MPa}] \; .$$

For the Poisson's ratio ν a value of 0.3 is used. The removal of the trabecular head region is simulated by masking all voxels belonging to this region.

Starting with the segmented voxel model at the initial CT resolution, a hierarchical octree representation is constructed in bottom-up order. Properties of voxels at coarser resolution levels are averages of the properties of all voxels at the next finer level contained in this voxel. Once the octree hierarchy is built, at each level a FE model is constructed, by creating a hexahedral finite element for each voxel containing at least one segmented CT voxel.

Once the hierarchy has been constructed, the user first selects the resolution level at which to perform the simulation. Then, the user interactively navigates the implant to the desired position in the bone. Whenever the implant is moved, GPU voxelization of the implant geometry is performed to determine those CT voxels covered by this geometry. The indices of these voxels are transferred to the simulation engine, which assigns the material properties of the implant to these voxels, recomputes the material properties of the respective finite elements, and updates the simulation matrices accordingly. The simulation engine then computes per-element stresses in the bone, updates a 3D texture containing the entire bone with these values, and transfers this texture to the GPU. On the GPU, an integrated view of the 3D stress distribution, the 3D CT scan of the bone, as well as the implant geometry is generated. Specifically, to allow for a precise analysis of the stress distribution, we perform volume ray-casting at sub-element accuracy if the FE model does not represent the bone at the finest resolution.

3. Simulation

In this section, we describe the physical model and the numerical machinery used to simulate the stress distribution in the interior of the proximal femur. According to the widely accepted assumption of linear elastic response of the bone under normal load, our model is based on linear elasticity using heterogeneous material properties described by the Young's modulus E. We also assume the material to be isotropic, which is a common abstraction from the real bone behavior due to the difficulty in identifying the anisotropic material parameters.

One central aspect of our computational steering approach is the possibility to obtain immediate visual feedback of induced loads while the surgeon positions the implant in the simulation environment. Therefore, we provide a novel approach to handle a moving implant in the numerical simulation. This approach is described in Section 3.3.

3.1. Linear Elasticity Theory

In static elasticity theory, the deformation of a volumetric object is described by a displacement field $u(x)$, $u : \mathbb{R}^3 \rightarrow \mathbb{R}^3$, which maps the reference configuration Ω to the deformed configuration $\{x + u(x) \mid x \in \Omega\}$. Driven by external forces f, a deformed solid is governed by the well-known equation from static elasticity theory

$$Ku = f, \tag{1}$$

where K is known as the stiffness matrix, u consists of the displacement vectors of all vertices, and f consists of the force vectors applied to these vertices. The stiffness matrix K is assembled from the so-called element stiffness matrices K^e, which are obtained by applying the principle of virtual work [Bat02] to one specific element. These matrices can be computed as $K^e = \int_{\Omega^e} B^{\mathrm{T}} D B \,\mathrm{d}x$, where B is the strain matrix and D is the material law [Bat02]. Typically, every element in a finite element discretization has only a very small number of neighbors, and thus the resulting stiffness matrix is very sparse.

3.2. Hexahedral Elements

To calculate the element matrices, the displacements are expressed in a basis of shape functions Φ as

$$u(x) = \Phi(x)\, u^e,$$

where $u^e = (\underline{u}_0^T, \ldots, \underline{u}_7^T)^T$ contains the single node displacements of each element. To account for the grid structure given by the initial CT scan, in the current work we are using hexahedral elements with tri-linear nodal basis functions:

$$\begin{aligned} N_i(x_1,x_2,x_3) &= c_0^i + c_1^i x_1 + c_2^i x_2 + c_3^i x_3 + \\ &\quad + c_4^i x_1 x_2 + c_5^i x_1 x_3 + c_6^i x_2 x_3 + c_7^i x_1 x_2 x_3 . \end{aligned}$$

To determine the coefficients $c_j^i, 0 \leq i,j < 8$ of the shape functions, a system of linear equations is built from the interpolation conditions of the hexahedron's vertices v_j:

$$N_i(v_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}.$$

3.3. Implant Simulation

One possible approach for simulating the interaction between an implant and the surrounding bone is to a) remove from the bone voxel model those voxels representing the material that would have been drilled away by the surgeon, b) build a new FE model from this voxel model, and c) simulate the interaction of the FE model with a separate model of the implant. However, this approach requires rebuilding the FE model whenever the implant is moved, including the recomputation of the structure of the system matrices. As these are operations that cannot be performed at interactive rates for reasonably sized data sets, we have developed a different approach which is based on a voxelization of the implant with respect to the initial CT voxel grid (see Section 4 for a detailed description of the voxelization algorithm).

By means of this voxelization, we obtain a classification of the voxels in the CT scan into bone voxels and implant voxels, i.e., voxels that are covered by the implant. After assigning the implant material properties to the implant voxels, the FE model hierarchy is rebuilt as described in Section 2. It is worth noting that in our approach the structures (i.e., the positions of the potential non-zero entries) of the simulation system matrices remain unchanged, because neither are hexahedral cells removed nor added to the simulation.

Furthermore, instead of modeling the contact between the bone and the implant explicitly, as a reasonable approximation for the contact of an object fixed in the bone, our method simulates a non-slip boundary between both objects. At the downside, the resolution of this boundary is determined by the resolution of the regular hexahedral grid. Even though this drawback is reduced because we assign an average stiffness value to finite elements that contain both implant and bone voxels, it is clear that at coarser resolution levels the simulation cannot resolve this boundary adequately.

To achieve a fast update of stiffness values in the FE model, we store a copy of each hexahedron's element matrix. Fortunately, since element matrices are translational invariant and all hexahedral elements have exactly the same size, we only have to store one element matrix K^e. This matrix is precomputed with a fixed elastic modulus E_0. Due to the linearity of the material law, the element matrix of a particular hexahedron can then be obtained by scaling K^e by the stiffness value of this element relative to E_0.

At runtime, we reassemble the stiffness matrix K once the implant is moved. Reassembling is greatly accelerated by storing for each hexahedral element a list of pointers to those entries in the matrix data structure that represent this element, allowing to quickly access the required entries within the assembling process. As the structure of the stiffness matrix is not changing during simulation, these lists can be precomputed. Without these references, the implementation would have to determine the respective positions by using row/column indices. Since we use a row-compressed sparse matrix format, such an index lookup requires a binary search on the entries of the respective row, which has a logarithmic runtime complexity in the number of non-zero entries in this row. This complexity can be reduced to $\mathcal{O}(1)$, if the respective entries are determined once in a preprocess and then stored as references within each hexahedral element. To keep the number of references as small as possible, we employ a block-row-compressed format in which each non-zero entry is a dense 3×3 matrix. Since the element matrix K^e consists of 3×3 blocks (according to the element vertices), this format accounts for the fact that units that have to be written into the global matrix are 3×3 blocks.

To reassemble the global stiffness matrix K, for every hexahedral element that is affected by the implant movement, all blocks of the reference element matrix K^e are first scaled according to the current element's elastic modulus and they are then written into the stiffness matrix.

3.4. Solution Method

Iterative methods are the most popular approaches for solving systems of linear equations as they arise in the current application, because they effectively allow for the exploitation of the systems' sparsity. Numerical multigrid solvers [Hac85, BHM00], in particular, are known to be among the fastest methods for solving elliptic partial differential equations of the form described above, as they scale linearly in the number of supporting vertices. In geometric multigrid schemes, relaxation methods like Gauss-Seidel are typically used to effectively damp high-frequency errors on a fine resolution grid, with the remaining low-frequency errors being solved for on a coarser grid. Recursive application of this basic idea to each consecutive system in the hierarchy of grid levels leads to a multigrid V-cycle.

Building upon previous work in the field [GW06], we have implemented an implicit multigrid solver which is well suited for the simulation of heterogeneous linear materials exhibiting a wide range of stiffness values. The multigrid solver internally requires to construct matrices on the coarser grids, which are used in the V-cycles. These matrices are built by means of sparse matrix products involving the respective interpolation and restriction operators derived from the geometric grid hierarchy. We use a data structure that is especially designed for the efficient computation of sparse matrix products [Geo08], as they occur when the implant is moved and the matrices on the coarser grids have to be recomputed, too. The stream-like layout of this data structure allows one to perform these operations in a very cache-efficient way. By reformulating the problem into the simultaneous processing of a sequential data and control stream, cache miss penalties are significantly reduced.

3.5. Stress Calculation

Once the displacement field u is computed according to Equation 1, we can determine the internal stress of the bone. Here we should note that computed displacements are typically so small, that these displacements do not result in any perceivable change of the bone's shape, and they are thus not considered in the visualization process. To compute the internal stress, we utilize the element stress matrices

$$S = \frac{1}{V}\int_{\Omega^e} DB\,\mathrm{d}x.$$

Here, $V = \int_\Omega 1\,\mathrm{d}x$ is the volume of the element, D is the material law, and B is the element strain matrix. By applying the element displacement field u^e to this matrix, an averaged stress tensor σ is derived for each element (note that σ stores the six entries of the symmetric tensor in a linearized form):

$$\sigma = S u^e.$$

To visualize the internal stress, we calculate for each element and corresponding tensor the so-called von Mises stress norm [Bat02]:

$$\sigma_{Mises} = \sqrt{3\sum_{k=4}^{6}\sigma_k^2 + \frac{3}{2}\sum_{k=1}^{3}(\sigma_k - \bar{\sigma})^2} \quad \text{with} \quad \bar{\sigma} = \frac{1}{3}\sum_{k=1}^{3}\sigma_k.$$

This norm gives us a scalar value for each hexahedral element. These values are stored in a 3D texture that is transferred to the GPU, where a volume ray-caster is used to visualize the 3D stress distribution.

4. GPU-Based Voxelization

In our computational steering approach, the implant is modeled by adapting the stiffness values of those voxels that are covered by the implant. By using a GPU-based voxelization method, and thus by exploiting the rasterization and parallel processing capabilities available on recent GPUs, we can accurately determine these voxels on a per-frame basis. Furthermore, as the resulting implant voxel model is stored in a 3D texture in GPU memory, it can directly be employed in the rendering process to provide immediate visual feedback while the surgeon is moving the implant inside the bone.

For GPU-based voxelization of solid objects several algorithms have been presented in the past [KPT99, FL00]. The algorithm showing best performance has recently been proposed by Eisemann and Décoret [ED08]. It performs the voxelization of a closed polygonal surface in a single rendering pass, by encoding the voxelization along the columns of the 3D voxel grid in large bit vectors. These vectors can be built directly in the framebuffer hardware using bitwise XOR blending. Because bitwise blending operations are only available in OpenGL, we propose an only slightly slower approach to GPU-based voxelization, which is suited for Direct3D 10 and, at the same time, allows us to efficiently pack the resulting voxelization into a compressed 3D texture representation.

Our voxelization method is specifically designed for the current application, in that it performs the implant voxelization in only two rendering passes. Multipass voxelization approaches, in contrast, require the implant model to be rendered several times, and would therefore result in a significant loss in performance due to the geometric complexity of this model. Specifically, our method employs the stencil routed k-buffer proposed by Myers and Bavoil [MB07], which allows capturing of multiple fragments per texel in a single rendering pass. When writing to a multisampled texture while multisample antialiasing is disabled, an incoming fragment is spread to all sub-samples of the respective texel, but the stencil is tested individually for each sub-sample. At the beginning, the stencil buffer is initialized with the values $2,3,4,\dots,9$ for the eight sub-samples of each texel. The stencil test is set to "passing if equal to 2", and the stencil fail and pass operation is set to "decrementing". Depth testing is disabled. With respect to a specific texel, for the first incoming fragment the stencil test passes exactly for the first sub-sample, and thus the fragment is written into this sub-sample. After execution of the stencil operation, the resulting stencil values are $1,2,3,\dots,8$. For the second incoming fragment the stencil test thus passes exactly for the second sub-sample. In this way, the incoming fragments for a texel are successively routed to different sub-samples, i.e., the i^{th} fragment is stored in the i^{th} sub-sample. This functionality allows us to capture multiple fragments per texel in a single rendering pass.

Our voxelization method uses two rendering passes. In the first pass, the implant mesh is rendered into the k-buffer to obtain its depth layers. In the second pass, the depth information is used to build a 3D binary volume representing the voxelization of the implant.

In the first rendering pass, we render the implant's surface mesh into the k-buffer to capture the depth layers of the implant. The only attribute being associated with the fragments is the world space depth, thus the k-buffer consists of a single component floating point texture. Current graphics hardware supports up to eight sub-samples per texel, and thus allows capturing up to eight depth layers in a single rendering pass—to voxelize objects with higher depth complexity further rendering passes would be required [MB07]. We use an orthographic projection with a view frustum matching the previously determined bounding box, and we choose a view port that aligns the texels in the k-buffer with the bone voxel grid. Front/back face culling and depth testing is disabled. By rendering the implant's mesh, each texel captures the depth values of the entry and exit points of an imaginary ray through the implant.

In the second pass, we build the 3D binary volume representing the implant. This volume is stored in a four component unsigned integer 3D texture (R32G32B32A32), with each voxel being encoded into one bit. Thus, 128 voxel slices are stored in one 3D texture slice. By adding the SV_RenderTargetArrayIndex semantic to the geometry shader output declaration, which enables the geometry shader to specify the respective target slice within the 3D texture, the entire volume can be created in a single rendering pass.

The bit patterns representing the voxelization are created in the fragment shader. By using up to 8 render targets and thus accessing 8 texture slices, the fragment shader can output a vector of up to 1024 voxels at once. First, the k-buffer entry corresponding to the respective ray is read and the depth values are sorted in ascending order. Each consecutive pair of depth values then represents an entry and an exit point into and from the implant. If $z_{\text{entry}}, z_{\text{exit}} \in [0,1]$ denote the depth values, the entry and exit voxel indices $k_{\text{entry}}, k_{\text{exit}}$ are determined by $k_{\text{entry}} = \left\lceil z_{\text{entry}} \cdot d - \frac{1}{2} \right\rceil$ and $k_{\text{exit}} = \left\lfloor z_{\text{exit}} \cdot d - \frac{1}{2} \right\rfloor$, with d denoting the depth (number of slices) of the voxel volume. The corresponding voxel block between the entry and the exit voxel is created by adding $\sum_{i=k_{\text{entry}}}^{k_{\text{exit}}} 2^i = 2^{k_{\text{exit}}+1} - 2^{k_{\text{entry}}}$ to the bit pattern. Note that due to the triangle rasterization rules and due to rounding in the computation of k_{entry} and k_{exit}, a voxel is created iff the voxel center is covered by the implant.

Finally, the 3D binary volume can directly be used on the GPU for visualization purposes. Furthermore, it is downloaded from GPU memory into main memory, where it is used to update material properties in the FE model as described in Section 3.3.

5. Visualization

For the rendering of the virtual simulation environment and visualization of the simulation results we use GPU-based semi-transparent rendering of surface meshes and volume ray-casting. These techniques enable us to simultaneously render the simulation objects as well as the simulation results and thus to show the results in their respective context, without limiting perception due to occlusions. The bone and implant meshes are rendered as semi-transparent surfaces. To render opaque and semi-transparent geometry as well as the volumetric von Mises stress scalar field in the correct visibility order, we use a multi-pass approach, which again utilizes the stencil-routed k-buffer already in use in the voxelization process.

5.1. Rendering

First, we render all opaque geometry into the frame buffer with enabled depth testing. The content of the depth buffer is later used during ray-casting to correctly handle occlusions of semi-transparent geometry and the ray-cast volume by fully opaque objects. Then, we render all semi-transparent geometry into an off-screen k-buffer, with depth testing as well as front/back face culling being disabled. By means of the k-buffer we can capture for each pixel up to eight incoming fragments, independently of the rendering order of the geometry. In our current implementation, we store with each fragment its depth value and its surface normal—needed for diffuse lighting—in camera space as well as an object id, which is later used to access a small GPU lookup table storing the material colors of the respective object. Since these values are encoded into 2×32 bits, we choose a two component unsigned integer texture format (R32G32) for the k-buffer.

We then use a full-screen rendering pass to ray-cast the volume as well as to simultaneously render the semi-transparent geometry. Our ray-casting approach is based on the technique proposed by Krüger and Westermann [KW03]. For each pixel, we first determine the corresponding ray's entry and exit point into and from the volume by two texture lookups. These two textures have been computed in two previous rendering passes. Furthermore, we fetch the pixel's k-buffer entry, i.e., the fragments of the semi-transparent geometry for that pixel, and sort them with respect to ascending camera depth values. We also fetch the pixel's depth from the depth buffer and back-project the depth value into camera space. The depth is used to handle occlusions by fully opaque objects.

If a ray does not intersect the volume, which is indicated by a special value in the entry point texture, we blend the k-buffer fragments lying in front of the opaque geometry front-to-back, and we write the resulting color into the frame buffer using alpha blending. Otherwise, we sample the volume equidistantly along the ray and simultaneously accumulate the respective color contributions by using front-to-back blending. The semi-transparent geometry is incorporated into the ray-casting process by blending a fragment as soon as the depth of the next sampling position is larger than the depth of the fragment. If the ray hits opaque geometry, it is terminated. The accumulated color is finally written into the frame buffer using alpha blending.

Due to the FE element discretization of the bone, the von Mises stress field slightly sticks out from the bone surface. To achieve a more appealing visualization of the von Mises stress, we therefore clip its rendering at the bone surface, i.e., the von Mises stress volume is only sampled when the sampling position is lying in the interior of the bone mesh. This is determined by maintaining a flag during casting, which is toggled whenever a bone surface fragment is blended.

5.2. Sub-Element Stress Visualization

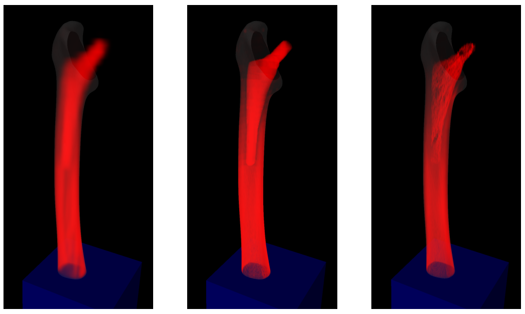


Figure 3: *Stress visualization using per-element stress (left) and sub-element stress (middle) for a FE model resolution of 8^3 CT voxels/hexahedron. For comparison, we also show the stress computed on the initial CT resolution (right).*

In this subsection, we describe a novel method to visualize the 3D stress distribution at sub-element accuracy with respect to the FE simulation grid. In Section 3.5, we have described how to compute an average stress tensor for a hexahedral element. If we look closer at the underlying theory, we see that, because we have a linear material law, the stress scales lineary with the elastic modulus E. Consequently, we can also compute the stress for an arbitrary elastic modulus first, and then scale the result by the specific elastic modulus of the element.

From this observation, we can immediately derive a method to visualize the stress tensor field on a per-voxel basis rather than on a per-hexahedron basis. All we have to do at a particular sample point along a ray is to scale the element's stress according to the elastic modulus of the current CT voxel. As a result, the stress field corresponds much better to the given implant position in the visualization. It is clear that at the highest resolution simulation level, which matches the resolution of the CT scan, this improvement does not have to be used.

Figure 3 demonstrates the effectiveness of the approach (see Figure 5 for a close-up view). First, we show the stress distribution as it was simulated on a low resolution grid with constant stress tensor per hexahedron. Next, we show a visualization of the same distribution at sub-element accuracy. Finally, we show the result obtained by using the FE model at CT resolution. As can be seen, the visualization at sub-element accuracy already resembles the high-resolution stress distribution very closely, but it takes considerably less time for the simulation. In addition, the performance of volume ray-casting only slows down marginally. Though we now have to adapt the sampling distance along the rays to the CT resolution, at every sample it only requires one additional texture look-up into the CT voxel model to obtain the respective stiffness value as well as one additional scalar multiplication.

6. Results and Discussion

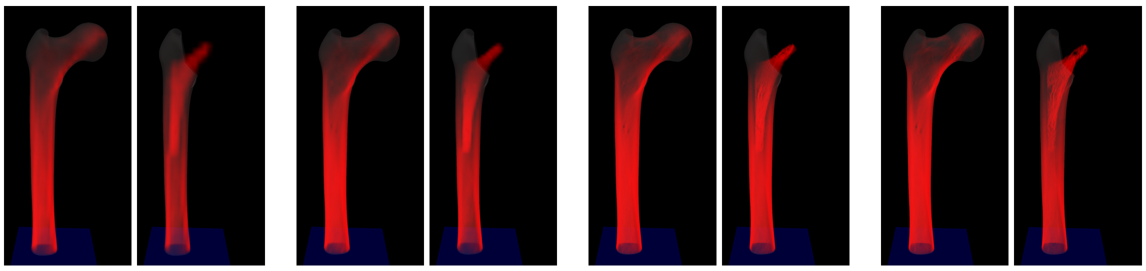
In this section, we give a detailed analysis of the performance and accuracy of our simulation support system. All benchmarks were run on a standard desktop PC with an Intel Core2 Quad Q6600 2.4 GHz processor, 8 GB of RAM, and an NVIDIA GeForce 8800GTX graphics card with 768 MB of video memory. The view port resolution was set to 1280×1024.

Figure 4 shows the simulation of a real world scenario in our virtual 3D planning system (see Figure 6 for a close-up view). The segmented femur is derived from a clinical CT scan of the patient. In the figure, we demonstrate both the natural stress distribution in the femur as well as the impact of a specific implant once it is inserted into the femur. From image to image, the resolution of the FE model is doubled. Even for the coarsest resolution, the stress distribution without the implant is already close to what is known from experiments on real bone structures. However, the resolution is not fine enough to properly represent the implant. The right image shows the simulation performed on the resolution of the initial CT scan, with the stress in the implant exhibiting very fine details due to the trabecular structures within the femur. Since these trabeculae are tangential to the implant, the forces of the implant are directly fed into the trabeculae, resulting in high stress in these fibers. Note that these accurate results were obtained within less than one minute.

CT Voxels/Hexahedron	8^3	4^3	2^3	1^3
# Hexahedra	2,257	14,277	88,783	520,416
# Vertices	3,280	18,020	115,076	782,420
Voxelization+Download	49 ms	49 ms	49 ms	49 ms
Update	95 ms	335 ms	2,038 ms	13,168 ms
Solving (# V-cycles)	18 ms (2)	113 ms (2)	3.63 s (10)	30.7 s (10)
Simulation Total	162 ms (6.17 rps)	497 ms (2.01 rps)	5,720 ms	43,977 ms
Rendering Frame Rate	73 fps	68 fps	65 fps	63 fps

Table 1: *Timing statistics of the surgery support system.*

Table 1 shows the performance of our simulation support system at different FE model resolutions. The first two rows



8^3 CT voxels/hexahedron 2,257 hexahedra — 4^3 CT voxels/hexahedron 14,277 hexahedra — 2^3 CT voxels/hexahedron 88,783 hexahedra — 1^3 CT voxels/hexahedron 520,416 hexahedra

Figure 4: *Visualization of the (per-element) stress distribution inside the bone at different FE model resolutions, without and with the implant. Note that the principal stress distribution can already be deduced from the coarsest resolution, while at the highest resolution, the trabecular structures of the femur (right) are visible.*

list the number of hexahedral elements as well as the number of vertices in the respective FE model. The next three rows give the time for a single run of the simulation, split into the time for the voxelization of the implant (including the download of the constructed voxel model), the time for the update of the stiffness matrix and the multigrid matrix hierarchy, as well as the time for the solution process to obtain a displacement field u and to calculate the per-element stresses. The following row contains the total time of a simulation run, as well as for the interactive FE model resolutions (8^3 and 4^3 CT voxels/hexahedron) the number of simulation runs per second (rps). For the finer FE model resolutions (2^3 and 1^3 CT voxels/hexahedron), the simulation is not updated on-the-fly, but only on demand of the user. The last row shows the number of frames per second for the rendering. Note that the frame rate is decoupled from the simulation update rate, since we use two separate threads for the rendering and the simulation, thus enabling a smooth interaction with the system.

The time for the implant voxelization and the download of the voxel model is constant for all FE model resolutions, since the voxelization is always performed at the resolution of the initial CT voxel model. Note that the voxelization time in Table 1 is the CPU time, i.e., the time from issuing the respective draw call to the availability of the voxel model in main memory. The CPU time is much larger than the needed GPU time, since CPU and GPU have to synchronize, i.e., the CPU is stalled until the GPU has executed all pending commands in the command buffer. Using the voxelization algorithm proposed in Section 4, the GPU time is less than 1 ms for an implant consisting of 30,000 triangles and a resolution of 128^3 voxels, thus the GPU-based voxelization does not conflict with maintaining interactive frame rates for the visualization.

The time for updating the stiffness matrix and the multigrid matrix hierarchy as well as the time for solving the linear system scale roughly linear with the number of vertices. The aprupt rise of the time needed for solving between the FE model resolutions 4^3 and 2^3 CT voxels/hexahedron results from using as many V-cycles as are required for convergence in case of on-demand simulation, but only using 2 V-cycles in case of on-the-fly simulation, where convergence is achieved over multiple simulation runs.

With simulation update rates of 2-6 rps for the 8^3 and 4^3 CT voxels/hexahedron resolutions, our system enables a visual steering approach. Even for the finest resolution with more than half a million finite elements, the simulation runs in less that one minute on a standard desktop PC. With respect to the visualization, the frame rate slightly decreases from coarser to finer resolutions, since a higher resolution of the von Mises stress scalar volume requires a smaller step size during ray-casting, but even for the finest resolution interactive frame rates of more than 60 fps can be maintained.

7. Conclusion and Future Work

In this paper, we have presented a prototype of a computational steering environment for optimal implant selection and positioning. As our results show, by using advanced numerical schemes for FE-based model analysis, interactive yet highly accurate simulations are possible today on desktop PC systems. Combined with efficient visualization schemes including surface and volume rendering, a powerful visual computing tool for orthopedics has been developed. In the future, we will validate the results of our simulation with respect to real-world experiments. Furthermore, we will develop more sophisticated methods for the visualization of the directional stress tensor field and we will integrate haptic feedback devices into our system. Further directions of research are the integration of an anisotropic material law into the simulation as well as the explicit modeling of the contact zone between implant and bone.

Acknowledgments

The first author is funded by the International Graduate School of Science and Engineering (IGSSE) of Technische Universität München. We thank our cooperation partners at the Chair for Computation in Engineering of Technische Universität München for fruitful discussions.

References

[Bat02] BATHE K.-J.: *Finite Element Procedures*. Prentice Hall, 2002.

[BBL97] BÖRNER M., BAUER A., LAHMER A.: Computer-guided robot-assisted surgery in hip endoprostheses. *Orthopäde 26*, 3 (1997), 251–257.

[BHM00] BRIGGS W. L., HENSON V. E., MCCORMICK S. F.: *A Multigrid Tutorial*, 2 ed. SIAM, 2000.

[BOM*07] BESSHO M., OHNISHI I., MATSUYAMA J., MATSUMOTO T., IMAI K., NAKAMURA K.: Prediction of strength and strain of the proximal femur by a CT-based finite element method. *Journal of Biomechanics 40*, 8 (2007), 1745–1753.

[ED08] EISEMANN E., DÉCORET X.: Single-pass GPU solid voxelization for real-time applications. In *Proc. Graphics Interface* (2008), pp. 73–80.

[FL00] FANG S., LIAO D.: Fast CSG voxelization by frame buffer pixel mapping. In *Proc. IEEE Symposium on Volume Visualization* (2000), pp. 43–48.

[Geo08] GEORGII J.: *Real-Time Simulation and Visualization of Deformable Objects*. PhD thesis, Technische Universität München, 2008. http://mediatum2.ub.tum.de/node?id=627732.

[GW06] GEORGII J., WESTERMANN R.: A multigrid framework for real-time simulation of deformable bodies. *Computer & Graphics 30*, 3 (2006), 408–415.

[Hac85] HACKBUSCH W.: *Multi-Grid Methods and Applications*. Springer Series in Computational Mathematics. Springer, 1985.

[HEPP01] HANDELS H., EHRHARDT J., PLÖTZ W., PÖPPL S. J.: Simulation of hip operations and design of custom-made endoprostheses using virtual reality techniques. *Methods of Information in Medicine 40*, 2 (2001), 74–77.

[KF03] KEYAK J. H., FALKINSTEIN Y.: Comparison of in situ and in vitro CT scan-based finite element model predictions of proximal femoral fracture load. *Medical Engineering and Physics 25*, 9 (2003), 781–787.

[KGW*94] KEAVENY T. M., GUO E., WACHTEL E. F., MCMAHON T. A., HAYES W. C.: Trabecular bone exhibits fully linear elastic behavior and yields at low strains. *Journal of Biomechanics 27*, 9 (1994), 1127–1136.

[KLS94] KEYAK J. H., LEE I. Y., SKINNER H. B.: Correlations between orthogonal mechanical properties and density of trabecular bone: Use of different densitometric measures. *Journal of Biomedical Materials Research 28*, 11 (1994), 1329–1336.

[KPT99] KARABASSI E.-A., PAPAIOANNOU G., THEOHARIS T.: A fast depth-buffer-based voxelization algorithm. *Journal of Graphics Tools 4*, 4 (1999), 5–10.

[KW03] KRÜGER J., WESTERMANN R.: Acceleration techniques for GPU-based volume rendering. In *Proc. IEEE Visualization* (2003), pp. 287–292.

[MB07] MYERS K., BAVOIL L.: Stencil routed A-buffer. In *Proc. ACM SIGGRAPH Technical Sketch Program* (2007), p. 21.

[OH78] OH I., HARRIS W. H.: Proximal strain distribution in the loaded femur. *Journal of Bone and Joint Surgery, American Volume 60*, 1 (1978), 75–85.

[SV02] SIMÕES J. A., VAZ M. A.: The influence on strain shielding of material stiffness of press-fit femoral components. *Journal of Engineering in Medicine 216*, 5 (2002), 341–346.

[TSH*07] TADDEI F., SCHILEO E., HELGASON B., CRISTOFOLINI L., VICECONTI M.: The material mapping strategy influences the accuracy of CT-based finite element models of bones: An evaluation against experimental measurements. *Medical Engineering and Physics 29*, 9 (2007), 973–979.

[VCT*04] VICECONTI M., CHIARINI A., TESTI D., TADDEI F., BORDINI B., TRAINA F., TONI A.: New aspects and approaches in pre-operative planning of hip reconstruction: a computer simulation. *Langenbeck's Archives of Surgery 389*, 5 (2004), 400–404.

[YTM07] YOSIBASH Z., TRABELSI N., MILGROM C.: Reliable simulations of the human proximal femur by high-order finite element analysis validated by experimental observations. *Journal of Biomechanics 40*, 16 (2007), 3688–3699.

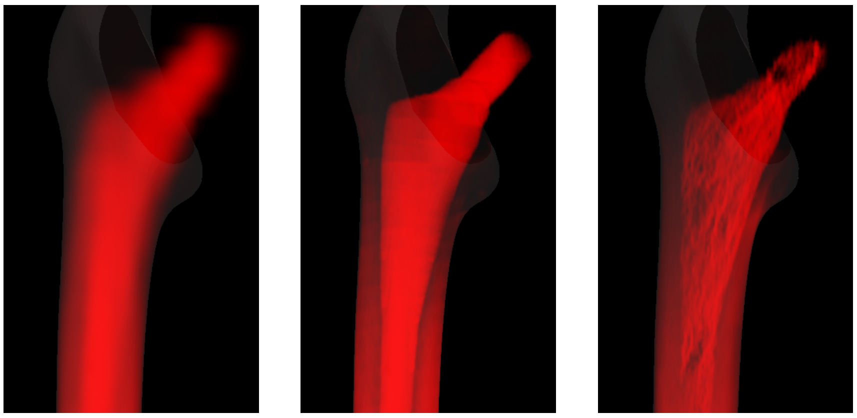


Figure 5: *Stress visualization using per-element stress (left) and sub-element stress (middle) for a FE model resolution of* 8^3 *CT voxels/hexahedron. For comparison, we also show the stress computed on the initial CT resolution (right).*

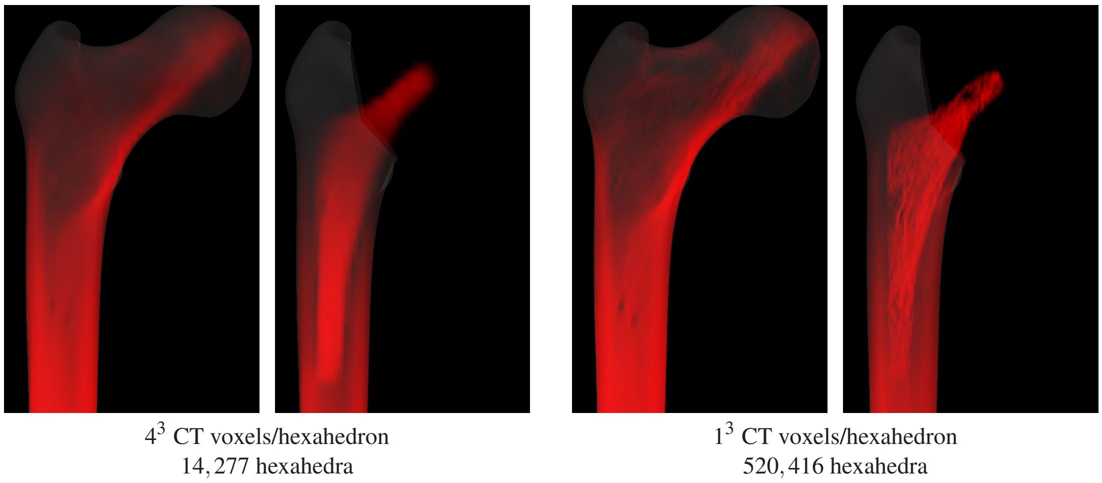


Figure 6: *Visualization of the (per-element) stress distribution at FE model resolutions of* 4^3 *and* 1^3 *CT voxels/hexahedron, without and with the implant.*

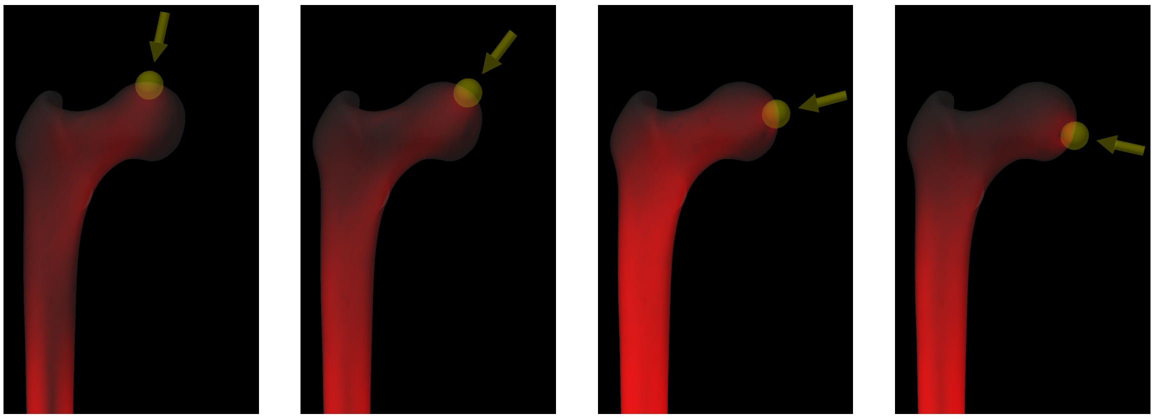


Figure 7: *Visualization of the (per-element) stress for various load directions using a FE model resolution of* 4^3 *CT voxels/hexahedron.*