Fast Generation of Virtual X-ray Images for Reconstruction of 3D Anatomy

Moritz Ehlke, Heiko Ramm, Hans Lamecker, Hans-Christian Hege, *Member, IEEE*, and Stefan Zachow

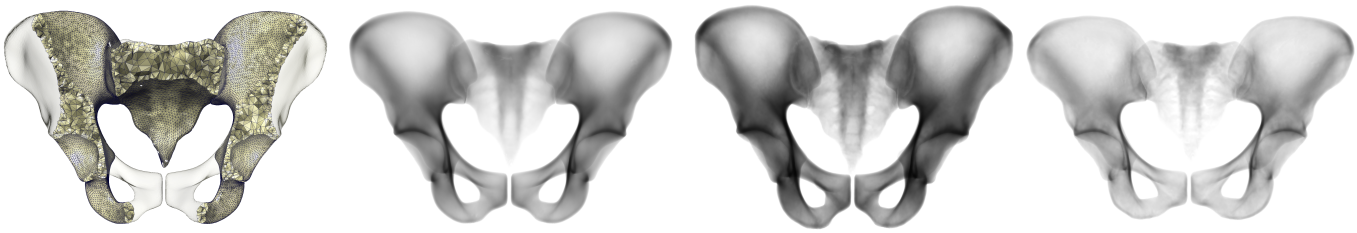

Fig. 1. From left to right: Deformable tetrahedral model of the pelvic bone enriched with density information, virtual X-ray projection of the mean model, two projections of the deformed model showing variation in both shape and density.

Abstract— We propose a novel GPU-based approach to render virtual X-ray projections of deformable tetrahedral meshes. These meshes represent the shape and the internal density distribution of a particular anatomical structure and are derived from statistical shape and intensity models (SSIMs). We apply our method to improve the geometric reconstruction of 3D anatomy (e.g. pelvic bone) from 2D X-ray images. For that purpose, shape and density of a tetrahedral mesh are varied and virtual X-ray projections are generated within an optimization process until the similarity between the computed virtual X-ray and the respective anatomy depicted in a given clinical X-ray is maximized. The OpenGL implementation presented in this work deforms and projects tetrahedral meshes of high resolution (200.000+ tetrahedra) at interactive rates. It generates virtual X-rays that accurately depict the density distribution of an anatomy of interest. Compared to existing methods that accumulate X-ray attenuation in deformable meshes, our novel approach significantly boosts the deformation/projection performance. The proposed *projection* algorithm scales better with respect to mesh resolution and complexity of the density distribution, and the *combined deformation and projection* on the GPU scales better with respect to the number of deformation parameters. The gain in performance allows for a larger number of cycles in the optimization process. Consequently, it reduces the risk of being stuck in a local optimum. We believe that our approach will improve treatments in orthopedics, where 3D anatomical information is essential.

Index Terms—Digitally reconstructed radiographs, volume rendering, mesh deformation, statistical shape and intensity models, image registration, GPU acceleration

1 Introduction

Despite the increasing availability of 3D image acquisition methods like computed tomography (CT) or magnetic resonance imaging (MRI), traditional 2D X-ray projections (in the following only called *X-ray images* or *X-rays*) are widely used for diagnosis or treatment planning in orthopedics. X-rays are especially suited to judge on the condition of bony structures, since bone is well distinguishable from surrounding soft tissue in the images. They are mandatory for imaging of weight-bearing situations and are applied for dynamic imaging of joint motion (fluoroscopy), neither of which can be performed in tomographic imaging. In addition, 2D X-ray imaging is widely available, rather inexpensive and exposes patients to less radiation compared to CT.

The reduction of dimensionality occurring while taking an X-ray, however, can lead to ambiguities in the image. In general, determining the 3D anatomical structure and pose from a single 2D X-ray image is an ill-posed problem. There are often uncertainties for example regarding the distance of an anatomy from the X-ray source and its scale (out-of-plane error). Structures project such that their order of appearance along the beam direction and their separation from each other remains unclear. Even for an experienced human observer, these ambiguities are hard to resolve [17].

The computer-aided reconstruction of a patient's 3D anatomy based on a single or few X-ray images has therefore received increasing interest in recent years [1, 21, 23, 24]. Zheng et al. [24] for example showed that by pose estimation of the pelvis from a single postoperative X-ray, its cup orientation with respect to the anterior pelvic plane can be predicted accurately. Their method aims at evaluating total hip replacement, where the assessment of the cup orientation plays an important role in the verification of the surgical outcome. Other applications include the diagnosis of osteoporosis from dual-energy X-ray absorptiometry (DXA) [21] or knee-joint motion tracking from a series of fluoroscopic images without a-priori knowledge about the patient-specific anatomy (e.g. CT).

A common concept of many 3D reconstruction approaches is to project a large amount of variations of a 3D shape onto the image plane, assuming an X-ray acquisition setup, i.e. a relative position of an X-ray source to a detector of particular size. The projections are then compared to a patient's clinical X-ray to find the projected shape that best represents the 2D information depicted in the reference image. This shape is assumed to be the best approximation of the true 3D anatomy.

We are developing an anatomy reconstruction framework that employs a shape and intensity based registration approach. The idea is to increase the robustness of the reconstruction process by considering as much information about the anatomy of interest in the reference X-ray as possible, including the interior density information of anatomical

- *The authors are with the Department Visualization and Data Analysis, Zuse Institut Berlin (ZIB), Berlin, Germany.*
- *E-mail: {ehlke, ramm, lamecker, hege, zachow}@zib.de.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 04 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

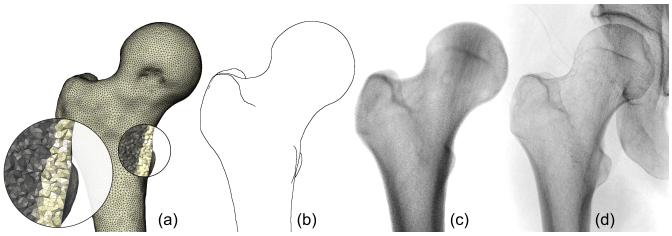

Fig. 2. A 3D model of the proximal femur (upper thigh bone) that incorporates volumetric density information (a). Compared to a projection of contour lines (b), a virtual X-ray (c) depicts the internal density structure of the bone and therefore better resembles the femur in the clinical X-ray (d).

structures. For this purpose, we project virtual X-ray images, often referred to as digitally reconstructed radiographs (DRRs), from many variations of a volumetric tetrahedral mesh with associated density information. The mesh is deformed prior to each projection such that it represents plausible candidates for a patient-specific shape and density distribution of the anatomy of interest. Our method searches for the best candidate fit by comparing the anatomy's X-ray attenuation in the clinical X-ray to the pixel intensities of the virtual X-ray.

Intensity-based registration methods are known to be prone to local optima due to the non-convex nature of the similarity measures (see for example [12]). Our framework evaluates many anatomical shape candidates in order to improve the reconstruction quality by identifying the best among several locally optimal solutions. It consequently has to produce large quantities of virtual X-rays (e.g. 10^4 or more images). An essential goal therefore is to quickly compute both deformed volumetric geometries and their corresponding projections. In addition, the virtual X-rays should depict the density information of the anatomy of interest as accurately as possible to allow for intensity-based comparison with the clinical X-rays.

This paper presents a GPU-only method that achieves these goals and thereby enables fast and robust intensity-based geometry reconstructions from X-rays. Its main contributions are:

- A GPU-algorithm for fast rendering of tetrahedral meshes with higher-order polynomial density functions. Our approach simplifies the tessellation of tetrahedra and can be implemented on the GPU without any explicit branching or looping (Section 4).
- A method that varies (deforms) both anatomical shape and density information on the GPU and concurrently generates virtual X-ray images from the deformed anatomy model (Section 5).
- A comparison to existing projected tetrahedra (PT) approaches. Our method achieves comparable quality to ground truth at higher frame rates (Section 7).

We present first promising results of our novel method within the developed 3D reconstruction framework in Section 8. A conclusion is given in Section 9.

2 Related Work

Existing approaches for the reconstruction of 3D anatomy from X-ray images often employ deformable surface models in combination with contour-based distance measures [1, 2, 23, 24]. Although contours or silhouettes of 3D anatomical models can be computed very efficiently, their correspondent contours/silhouettes are often difficult to extract from the patient's X-ray. Using contour information only neglects important information on the interior structures (i.e. bone density). For instance, cortical (compact) bone that typically forms the outer hull of a bony anatomy has a much higher X-ray absorption than spongy (cancellous) bone inside (cf. Figure 2).

Lamecker et al. [6] propose a method for 3D shape reconstruction that generates thickness projections of a deformable surface model, approximating bone by a homogeneous material. They conclude that the evaluation of intensity information improves the reconstruction but that thickness images do not sufficiently represent the heterogeneity of bony structures. One solution is to render high-resolution CT-like atlases that contain density information of the anatomy of interest. In [19] for instance, DRRs are generated from deformable hexahedral grids modeling the proximal femur. These projections are more suitable for intensity-based comparison to clinical X-rays, since they mimic the X-ray absorption in inhomogeneous anatomical structures. However, a dense sampling of intensity information in the atlas is required to accurately project bone mineral density, thus degrading the performance of the deformation and projection. Laney et al. [7] have shown how to accelerate the generation of radiographs from hexahedral grids using GPU-based volume rendering techniques.

With Fourier volume rendering [9], a method has been proposed to efficiently generate DRRs from datasets of high resolution. These techniques require a transformation of a deformed CT-like atlas into the frequency domain, e.g. through a fast Fourier transform (FFT). Since the model is deformed frequently but typically only few projections (often only one or two) are generated per deformed model, Fourier rendering becomes impractical for the reconstruction process.

An adaptive sampling of the interior densities of anatomical structures using unstructured meshes has been suggested. Yao [22] proposes higher-order Bernstein polynomials as an efficient way to encode density information on tetrahedral meshes. He represents anatomical structures as deformable point distribution models (PDM) that incorporate density values learned from CT data. In addition, Yao suggest a CPU-based projection algorithm that respects the non-linear density distribution in each tetrahedron and generates virtual X-rays for intensity-based reconstruction processes.

Sadowsky et al. [14, 15] build upon the idea of Yao and simulate X-ray images using a projected tetrahedra (PT) approach [18]. Their algorithm makes use of two observations: First, the tetrahedra can be processed independently from each other and do not have to be projected in visibility order. Second, barycentric coordinate systems can be applied to integrate the Bernstein polynomials in closed form and at the same time classify the projected outline of the tetrahedra. The outline classification is used to tessellate the projected tetrahedra and to determine ray parameters in barycentric and world coordinates on the front-facets of the individual tetrahedral cells. In the fragment shader stage of the graphics hardware, a perspective correction mechanism is applied to obtain the respective back-facet parameters. The polynomial density functions are then integrated in closed form and the overall X-ray attenuation is accumulated in the framebuffer according to the Beer-Lambert law of attenuation (see Equation 1).

The implementation presented by Sadowsky et al. performs both the deformation of the tetrahedral volume as well as the classification of its projected outline on the CPU. Consequently, the geometry and density information has to be copied to the GPU memory after the model is altered. Although GPU implementations for classifying projected tetrahedra exist [10, 11], they introduce branching and looping operations that have a negative effect on the overall rendering performance. The method proposed here instead adapts the idea of cell-based ray casting [4, 20] and relies on efficient ray-tetrahedron intersection tests in barycentric coordinates without prior classification. Moreover, we propose a method to combine both the deformation of tetrahedral meshes as well as their projection on the GPU.

3 Background

The overall attenuation encountered by a monochromatic X-ray beam $p(x) = w_{in} + (w_{out} - w_{in}) \cdot x$ passing through tissue is described by the Beer-Lambert law:

$$I_{out} = I_{in} \cdot e^{-\int_{p(x)} \alpha(w)\, dw} \tag{1}$$

where I_{in}/I_{out} are the input/output intensities of the beam, w_{in} and w_{out} are the entrance and exit points, and $\mu = \alpha(w)$ denotes the linear attenuation coefficient of some homogeneous tissue encountered by the ray at point w. The function α is referred to as the *density distribution* of an anatomical structure.

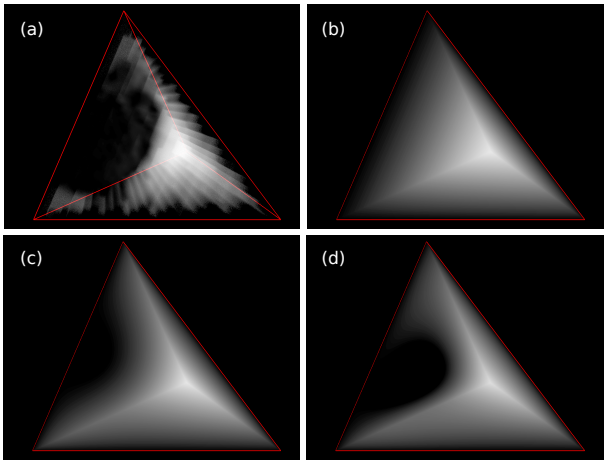

Fig. 3. Actual density distribution from CT data (a) and its approximations with Bernstein polynomial density functions of degree $d = 1$ (b), $d = 2$ (c) and $d = 3$ (d). The non-linear gray value distribution of the CT dataset is better approximated by polynomials of higher degree.

Following previous work [14, 15], we cast a monochromatic beam from the X-ray source to each pixel in the image plane. The spatial location of the source corresponds to the camera position in a perspective projection of the anatomy, where each pixel is assigned the intensity I_{out} of its associated beam after attenuation by the anatomical structure. Artifacts, however, that may occur due to scattering or beam hardening, are not considered in this attenuation model.

3.1 X-ray attenuation in tetrahedral meshes

We model anatomical structures using tetrahedral meshes according to the proposal of Yao [22]. Each cell in the mesh is associated with one Bernstein polynomial function that describes the density distribution at its spatial location in the anatomy.

A Bernstein polynomial of degree d is parametrized per tetrahedron t by Bernstein coefficients $c_t = \{c_{i,j,k,l}\}$ with $i+j+k+l = d$. Given a point $b = (b_x, b_y, b_z, b_w)^T$ in local barycentric coordinate space of t, the corresponding linear attenuation coefficient calculates as

$$\alpha_t(b) = \sum_{i+j+k+l=d} [c_{i,j,k,l} B^d_{i,j,k,l}(b)] \tag{2}$$

where

$$B^d_{i,j,k,l}(b) = \frac{d!}{i!j!k!l!}(b_x)^i (b_y)^j (b_z)^k (b_w)^l \tag{3}$$

is the Bernstein basis function of degree d.

The number of Bernstein coefficients assigned per tetrahedron depends on the degree of the polynomial that is used to describe the density distribution: $\binom{d+3}{3}$, with $n = 1, 4, 10, 20$ for degrees $d = 0, 1, 2, 3$ respectively. By applying higher polynomial degrees, non-homogeneous density distributions can be expressed in a single tetrahedron (cf. Figure 3).

To accumulate the attenuation encountered by a ray $p(x)$ passing through a tetrahedron (cf. Figure 4) the Bernstein polynomial density distribution is integrated along the ray:

$$\int_{p(x)} \alpha_t(b)\mathrm{d}b = ||w_{out} - w_{in}|| \cdot \sum_{i+j+k+l=d} [c_{i,j,k,l} \int_{b_{in}}^{b_{out}} B^d_{i,j,k,l}(b)\mathrm{d}b] \tag{4}$$

We refer you to Yao's work for a detailed closed form solution of Equation (4).

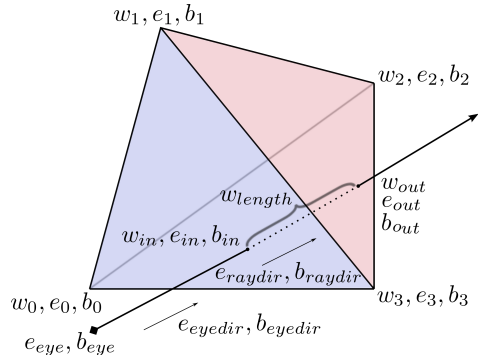

Fig. 4. Ray parameters on a single tetrahedron. The ray originates at the source e_{eye}, b_{eye}, intersects the tetrahedron at w_{in}, e_{in}, b_{in}, traverses the tetrahedron by length w_{length} and exits at the point $w_{out}, e_{out}, b_{out}$. The letter w stands for world coordinates, e for normalized eye coordinates, and b for local barycentric coordinates.

3.2 Statistical shape and intensity models (SSIMs)

An SSIM, as employed in this work, is a statistical model of shape and intensity (or density) that is generated from a set of training meshes by means of a principal component analysis (PCA). Each training mesh represents a patient-specific anatomy according to the tetrahedral model described in the previous subsection. The SSIM combines the mean shape and density of all training data and their specific variations, which are expressed in the PCA eigenvectors. We reuse the notation of Yao [22], who describes an SSIM as:

$$Y = \bar{Y} + Pr \tag{5}$$

Here, $Y = [Y_s, Y_\mu]$ denotes an SSIM instance (i.e. a representative of the training set) with tetrahedral vertex positions Y_s and coefficients Y_μ to Bernstein polynomial density distributions of a fixed degree. $\bar{Y}$ represents the mean model and P is the eigenvector matrix that holds information about the anatomical variation of the training data. The number of eigenvectors (and parameters) depends on the number of training data that are incorporated into the SSIM, and the contents of the eigenvectors are related to the deviation from the average shape and density distribution. The mean model is deformed by selecting values for the deformation parameters r and evaluating Equation (5). To reduce the model's parameter space, only a subset of eigenvectors can be regarded during deformation, e.g. those eigenvectors that describe the largest variability in the training data. Using the full set of deformation parameters leads to the maximum level of detail that instances of the SSIM may exhibit.

We apply this type of deformable volumetric model since SSIMs are capable of representing a set of anatomical shapes with associated densities in a compact form and allow for a plausible interpolation within this set. However, SSIMs do depend on the choice of training data that are integrated therein. An SSIM cannot represent any possible patient-specific anatomy in full detail but will rather be a approximation. For example, pathologies are not expressed by the model, unless they are contained in the training data. SSIMs are well suited for application scenarios such as pose estimation or for an extraction of anatomical landmarks or mechanical axes, where a perfect representation of the patient-specific anatomy is not necessary.

4 GPU-BASED ALGORITHM FOR PROJECTING MESHES

According to Equations (1) and (4), the total attenuation encountered by an X-ray depends on the ray traversal length $w_{length} = ||w_{out} - w_{in}||$ and its entrance and exit points in local barycentric coordinates b_{in} and b_{out} for each tetrahedron that is intersected by the ray (cf. Figure 4). In addition, the Beer-Lambert law states that the total attenuation equals the accumulated contributions of all tetrahedra along the ray, irrespective of the order they are traversed ("visibility order"). Our GPU-based

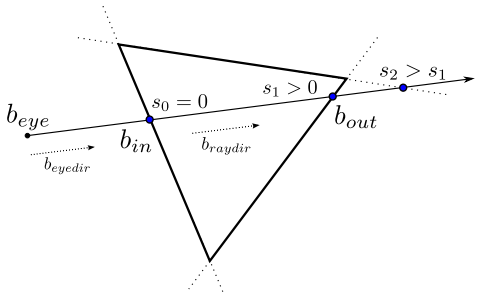

Fig. 5. Intersection of an X-ray with a cell (e.g. a triangle). The ray originates at the X-ray source b_{eye} and traverses the cell between b_{in} and b_{out} with direction $b_{raydir} = b_{eyedir} \cdot s$. Our intersection test determines three candidates for s that correspond to the intersection points with cell edges. Discarding the entrance point itself, the correct solution for $s = s_1$ is located at minimum distance to the X-ray source.

algorithm for simulating X-ray attenuation in tetrahedral meshes exploits these properties by independently processing each tetrahedron of the mesh in parallel.

We stream the tetrahedra and corresponding Bernstein polynomial coefficients through the per-vertex, per-geometry and per-fragment stages of the graphics pipeline. The entrance parameters of the rays on the front-facets of the tetrahedra, such as b_{in}, are interpolated linearly between tetrahedral vertices. The rays' exit parameters b_{out} and their traversal depths w_{length} are computed in the fragment stage using direct ray-facet intersection tests in barycentric coordinates. Both the entrance and exit parameters are then applied to integrate the Bernstein density functions, cf. Equation (4), in closed form using per-fragment operations, as proposed by Sadowsky et al. in [14]. The contributions of all individual tetrahedra are finally blended together to form the virtual X-ray image.

4.1 Computing the rays' entrance parameters

In a first processing step, *per-vertex* operations transform the tetrahedral vertices into normalized device coordinates and eye coordinates (e_i). The vertex positions are then handed to the per-geometry processing stage together with the Bernstein coefficients of the respective tetrahedron.

The *per-geometry* operations construct the matrix $M = [e_0, e_1, e_2, e_3]$. Its inverse M^{-1} is a linear transformation that projects eye coordinates into the local barycentric coordinate space of the respective tetrahedron. The method of Sadowsky et al. utilizes M^{-1} in order to classify the projected tetrahedral outline. We apply M^{-1} to compute four vectors b_{eyedir}, each pointing from the source (e.g. the eye position) to one of the four tetrahedral vertices: $b_{eyedir} = b_i - M^{-1} \cdot e_{eye}$. An additional processing step then determines their eye space correspondences $e_{eyedir} = e_{eye} - e_i = -e_i$. Note that $e_{eye} = (0,0,0,1)^T$.

Consecutive operations in the per-geometry stage triangulate the tetrahedron into its four facets. The values of b_i, b_{eyedir} and e_{eyedir} are assigned as parameters of the respective facet's vertices. They are linearly interpolated within the front-facets of the tetrahedron, thus making the perspective-correct entrance parameters of all rays intersecting the tetrahedron available in the fragment stage. We utilize the culling functionality of graphics hardware to discard the rasterization of tetrahedral back-facets. The Bernstein coefficients are pushed down the GPU pipeline as non-varying parameters.

In contrast to PT approaches, the triangulation of the tetrahedra is view-independent. Our method does not determine a thick point (see [15]) and avoids computing line intersections or sorting vertices into rendering order.

4.2 Computing the rays' exit parameters

Given the front-face parameters b_{in}, b_{eyedir} and e_{eyedir} of a ray, the goal is to find the exit parameter in barycentric coordinates b_{out} and the traversal length w_{length} (cf. Figure 4). In the following, $b_{i,j}$ denotes the j^{th} component of the vector b_i. Keeping in mind that local barycentric coordinates of a tetrahedron are non-negative and sum up to one, we observe the following:

1. At least one component of b_{in} and of b_{out} equals zero, since b_{in} and b_{out} are located on the facet of a tetrahedron: $\exists i.b_{in,i} = 0, \exists j.b_{out,j} = 0$.

2. In the "regular" case, a ray enters and exits a tetrahedron on two distinct facets, on opposite edges or on opposite vertices. This implies, that one zero component of b_{in} has a different index than the zero component of the corresponding b_{out}: $\exists i.(b_{out,i} = 0 \wedge b_{in,i} > 0)$.

3. As an exception from (2), the ray might enter and exit on the same vertex. In this case the traversal depth w_{length} is zero and the ray is not attenuated.

We model the ray between the barycentric entrance and exit points of a tetrahedron as $b_{raydir} = b_{out} - b_{in}$. Since the vectors b_{raydir} and b_{eyedir} lie on the same ray of sight, b_{raydir} can be calculated by $b_{raydir} = s\, b_{eyedir}$ with a positive factor s. The barycentric exit coordinates of a ray then compute as

$$b_{out} = b_{in} + s\, b_{eyedir} \tag{6}$$

According to (1), we know that at least one component of b_{out} is zero. Consequently, there exist four possible candidates for s, given that the parameters b_{in} and b_{eyedir} are known:

$$s_i = -\frac{b_{in,i}}{b_{eyedir,i}} \tag{7}$$

The s_i represent solutions for intersecting the ray with all four tetrahedral facets.

To find the correct intersection s_i, all results with $b_{eyedir,i} \geq 0$ or $b_{in,i} = 0$ are ignored. Among the remaining candidates, the smallest positive s_i is the correct solution for s (cf. Figure 5). If no candidate is found, then $s = 0$ and therefore $b_{in} = b_{out}$. In the following section we present a shader implementation that computes s^{-1} using only three vector operations.

Once the transformation factor s is determined, we use it to compute the actual exit point of the ray on the tetrahedron b_{out} according to Equation (6). Given b_{out}, it would now be possible to determine e_{out} by applying the transformation matrix M and determine $w_{length} = ||e_{out} - e_{in}||$. However, we propose a more efficient method based on the fact that s also scales the ray direction vector in normalized eye coordinates to the traversal length of the ray (s is positive):

$$\begin{aligned} w_{length} &= ||e_{out} - e_{in}|| \\ &= ||M \cdot (b_{out} - b_{in})|| \\ &= ||M \cdot (b_{raydir})|| \\ &= ||M \cdot s \cdot (b_{eyedir})|| \\ &= s \cdot ||e_{eyedir}|| \end{aligned} \tag{8}$$

Our algorithm computes w_{length} according to Equation (8), applying the interpolated e_{eyedir}. Given b_{out} and w_{length}, all parameters are available to solve the rendering integral, Equation (4), in the per-fragment stage. The attenuation encountered by the ray is then returned as the fragment color and can be summed up (e.g. blended) with the contributions of other tetrahedral cells in the mesh.

5 GPU IMPLEMENTATION

In this section, we first propose an efficient implementation of our algorithm to render static tetrahedral meshes with higher-order density functions that was introduced earlier. The rendering pipeline is then extended to deform the tetrahedral meshes in parallel on the GPU. We implemented the rendering pipeline based on OpenGL version 4.0, featuring the OpenGL Shading Language (GLSL) as of version 4.00.

5.1 Projecting tetrahedral meshes

In the following, we describe the rendering process for a single tetrahedral cell in the mesh. To generate the final image, i.e. accumulate the attenuation of all tetrahedra in a 2D texture, we set the OpenGL blending functionality to `glBlendFunc(GL_ONE, GL_ONE)`, `glBlendEquation(GL_FUNC_ADD)` and bind a framebuffer object (FBO). Taking the exponential of the summed-up contributions in the 2D textures is then performed in the post-processing step using an additional rendering pass.

We issue one `GL_POINT` primitive per tetrahedron. The respective tetrahedral vertex positions in world coordinates and the Bernstein coefficients describing the density function are assigned as point vertex attributes. A vertex shader is executed once for every tetrahedron (`GL_POINT` primitive), and therefore has access to the four vertex coordinates. It performs the transformation into normalized eye and device coordinate space. The transformed vertices and the Bernstein coefficients are then streamed further down the rendering pipeline.

In the geometry shader stage, M^{-1} is computed using the GLSL `inverse()` call on $M = [e_0, e_1, e_1, e_3]$. We experienced that under certain views the matrix M becomes nearly singular, for example when the rendered mesh is located far from the source and covers few pixels in the image plane. This is not a problem in the reconstruction framework, as such situations won't occur in a clinical X-ray setup. One might alternatively use the 64Bit precision inverse operations on the OpenGL `dmat` matrix types which eliminate the problem.

We compute $-b_{eyedir}$ in one vector operation, exploiting that the b_i are equal to zero in all but one component. A small offset is added to the zero components of the b_i afterwards in order to circumvent division by zero in the fragment shader stage. The tetrahedron is decomposed into its four triangle facets and the e_i, b_i and $-b_{eyedir}$ are linearly interpolated in between the geometry and fragment shader stages.

The fragment program performs the ray-tetrahedron intersection tests by solving for ${s_i}^{-1} = \frac{-b_{eyedir,i}}{b_{in,i}}$ rather than s_i. The correct intersection at s^{-1} is the maximum of all four possible candidates and is extracted using two `max()` calls. To obtain b_{out} and w_{length}, we divide by ${s_i}^{-1}$ instead of multiplying with s_i, cf. Equation (6) and Equation (8). Note that the intersection candidates for $b_{in,i} = 0$ and $b_{eyedir,i} \geq 0$ do not contribute to the overall attenuation since w_{length} becomes (practically) zero or the corresponding ${s_i}^{-1}$ becomes negative and is therefore discarded. Finally, we utilize hard-coded multinomial factors as proposed in [14] to solve the volume rendering integral, Equation (4), in the fragment shader stage.

Our method requires less than four vector or matrix operations in the geometry shader stage to compute all variables for tessellation. Compared to the PT method proposed by Sadowsky et al. [14], this significantly reduces the number of per-geometry operations while introducing at most three additional fragment operations.

5.2 Combining deformation and projection

To deform and project a tetrahedral mesh entirely on the GPU, a statistical model is varied in the vertex shader stage and the resulting model instance projected in the consecutive stages. We store the SSIM in the graphics hardware memory. Our implementation utilizes OpenGL texture buffer objects (TBOs) to hold the mean vertex coordinates $\bar{Y}_s$ and the mean Bernstein coefficients $\bar{Y}_\mu$. Their components can be accessed using unique vertex identifiers and unique tetrahedron identifiers respectively.

The eigenvector components of the vertex positions and of the Bernstein coefficients are each split in one texture array (`GL_TEXTURE_2D_ARRAY`). Every eigenvector maps to exactly one 2D texture where the respective eigenvector components are stored consecutively. Additionally, two vertex arrays hold the static tetrahedral indices and the corresponding vertex indices. The width and height of the 2D eigenvector textures and the number of eigenvectors are handed to the shader stages as uniform parameters.

In the vertex shader stage, the mean values are extracted from the `samplerBuffer` (TBO) using the current tetrahedral and vertex ids. The normalized texture coordinates for accessing the eigenvector textures are then precomputed, applying the uniform texture width and height. Afterwards, the vertex shader varies both vertex positions and Bernstein coefficients according to Equation (5). The "deformed" tetrahedron is then projected in subsequent shader stages by applying the projection method introduced in the preceding section.

To avoid multiple deformation of vertices that are shared between tetrahedra, a preliminary rendering pass is issued that only performs the geometric deformation. Here, the OpenGL transform feedback buffer is utilized to avoid data exchange between CPU and GPU. One primitive is rendered per vertex, and the deformation of the vertex positions is computed in the vertex shader stage. The implementation discards the fragment rasterization using the OpenGL `glEnable(GL_RASTERIZER_DISCARD_NV)` functionality. Rather than putting the result on the screen, the deformed vertices are stored directly on the GPU. They are then bound to the shader programs as vertex buffer objects in a second rendering pass. A different vertex shader program varies the Bernstein coefficients and proceeds with the projection accordingly.

6 RECONSTRUCTION FRAMEWORK

Within the 3D reconstruction process, the transformation and deformation parameters of the SSIM are optimized with respect to the similarity measure. Our framework utilizes a normalized mutual information metric (NMI) [13] between the virtual X-ray projection of the deformed/transformed SSIM and the clinical X-ray. We chose NMI instead of mean squared differences or other intensity-based measures since it is robust against overlapping structures and artifacts in the clinical X-ray images that are not present in the virtual X-rays.

To further increase the robustness of the similarity evaluation, we currently segment the anatomy of interest from surrounding soft tissue in the reference image and additionally mask out pathologies and implants covering the anatomy. The tissue (that does not overlap with the anatomy in the image) is replaced by a black background such that the reference X-ray more resembles a virtual X-ray projection. The masked out regions in turn are ignored for the evaluation of the similarity measure.

The mean model of the SSIM is positioned roughly in the virtual X-ray setup to initialize the optimization process. A gradient descend method then searches for the best fit between virtual and clinical X-ray. Before each optimization cycle, the gradient of the similarity measure in terms of transformation and deformation parameters is approximated based on finite-differences. We perform a line search along the direction of the gradient to find an improved model fit, that again acts as an initialization for the next cycle. The method is described in more detail in our previous work [2].

The gradient descend method is prone to local minima and there is no guarantee that a global maximum of similarity is reached. To avoid local optima in early stages of the optimization, we sub-sample the clinical X-ray images and perform the similarity evaluation on a reduced image resolution while fitting the model with only a subset of the deformation parameters. This prevents the optimization from getting stuck in local minima due to subtle image details. The resolution as well as the number of deformation parameters are increased in later optimization cycles, where it is important to consider finer image details to enhance the model fit. Once a (local) optimum is found, a new set of cycles can be triggered to again widen the search window and increase the chance of reaching a better solution.

6.1 Integration of the GPU solution

We deform and project the SSIM on the GPU after each parameter change and read the virtual X-ray image back to main memory for sim-

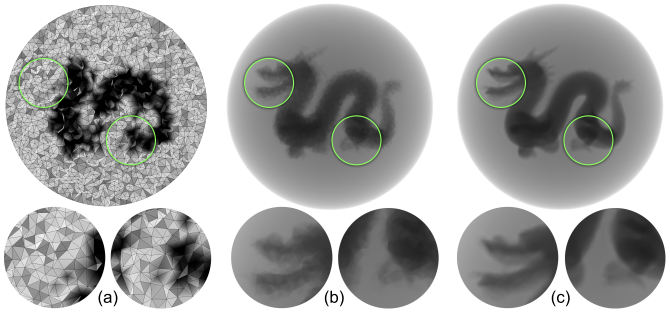

Fig. 6. A mesh of approximately 174k tetrahedra generated from the dragon dataset for performance evaluation (a). Whereas individual tetrahedra can be identified in virtual X-ray images generated with density functions of degree $d = 0$ (b), they are not distinguishable in $d = 2$ projections (c) due to a better approximation of the dragon's shape by higher-degree density polynomials.

ilarity evaluation. The deformation parameters are transferred to the GPU prior to a deformation/projection of the model using a TBO and by setting the modelview matrix accordingly. Three rendering passes are then issued, implementing the GPU-processing steps described in Section 5: First the vertex positions are deformed and written into the transform feedback buffer, then the density functions are varied and the deformed mesh is projected, and finally the exponential is taken of the result according to the Beer-Lambert law.

Note that the SSIM is deformed and projected entirely on the GPU. For every transformation/deformation cycle, only the deformation parameters, the model view matrix and the rendered image are transferred between main memory and GPU memory.

7 Experiments and Results

We evaluated the projection algorithm and the combined deformation and projection method in terms of their rendering speed. Additionally, we compared the rendering quality of our projection approach to ground truth DRRs that were generated via ray-casting of regular scalar fields, given by CT datasets of high resolution. The experiments were performed on two computer systems: *System-1* was equipped with an NVIDIA GTX680 graphics card and a 2.67 GHz Intel Xeon CPU with four cores (eight threads). *System-2* was featured with an NVIDIA GeForce GTX 570 graphics hardware and a six-core (twelve threads) Intel Xeon CPU running at 3.47 GHz clock speed.

7.1 Rendering performance

Our goal in the rendering performance evaluation was to compare the method of Sadowsky et al. [14] to our approach in terms of projection speed on the GPU. In the original implementation proposed by Sadowsky, the classification of the projected tetrahedral outline is performed on the CPU, making a direct comparison of the two approaches difficult. For that reason, we reimplemented the classification as GPU geometry shader programs, following the hardware-assisted PT approach presented in [11]. This way, Sadowsky's original algorithm, including the outline classification based on barycentric coordinates, was not altered and ported directly to the GPU. We employed GPU optimizations such as hardware-accelerated vector and copy operations wherever applicable.

We tested the rendering performance on tetrahedral meshes with varying number of cells. The Stanford *dragon*[1] surface was first converted into an artificial high-resolution voxel dataset. Bernstein polynomials of degree $d = 0$ to $d = 3$ were sampled from the voxel data onto spherical volume meshes with equal diameter but varying mesh resolution between 17k tetrahedra and 4M tetrahedra (see Figure 6 for an example). The spherical shape guarantees a constant pixel coverage on the image plane when changing the viewing direction.

We recorded the rendering speed while rotating the camera around the data in 12 degree increments. Therefore, in total 30 projections were generated per tetrahedral mesh and polynomial degree for a full 360° trajectory. The viewport of size 1000^2 showed a constant pixel coverage of 75%.

The results are depicted in Figure 7 for *System-1* and *System-2* in terms of rendering time in milliseconds. Both implementations reach projection rates above 10 frames per second (fps) when rendering meshes of up to 670k tetrahedra. However, our rendering method scales better with respect to the number of tetrahedra. On *System-1* for example, the algorithm proposed in this work is 0.5-2.7 times faster than the PT method when rendering meshes of 468k to 2.8M tetrahedra and density functions of degrees $d = 2$ and $d = 3$. Our method projects meshes of higher resolution, e.g. 1.2M tetrahedra and degree $d = 2$, at frame rates above 40 fps.

7.2 Deformation and projection

We evaluated the rendering speed of our GPU-based deformation and projection approach in order to compare its performance to previous methods that deform the tetrahedral volume on the CPU. For this purpose, we created an SSIM from 47 CT datasets with an approximate resolution of $0.9 \times 0.9 \times 1\,mm^3$ that were available from a previous study [16]. For each dataset, a polyhedral model of the pelvis was generated with vertex correspondences over all training datasets. We then transferred a reference volumetric mesh consisting of 195k tetrahedra to each individual anatomy. After sampling the CT data with density distributions of all four polynomial degrees, we applied the PCA on the resulting 47 tetrahedral meshes, incorporating the density information as described in Section 3.2.

The combined deformation and projection performance of our implementation was recorded on *System-1* while rendering the SSIM onto a 1000^2 viewport in anterior (frontal) view. A number of 0 to 46 eigenvectors were considered and 100 combined deformations and projections issued for every set of deformation parameters. We randomly chose the parameter values equally distributed within the range of the minimum and maximum values of the training data. To provide a comparison to CPU approaches, we measured a multi-threaded (eight threads) deformation on the CPU as well as a hybrid CPU/GPU implementation of Sadowsky's approach, in which the model is first deformed on the CPU, and then projected on the GPU as described previously. We applied the OpenMP framework to parallelize the deformation such that each thread independently varies one model component, e.g. a vertex position or Bernstein coefficient, at a time.

The results of the evaluation are given in Figure 8 in terms of duration in milliseconds per deformation/projection cycle (referred to as the rendering time). Our GPU-based approach deforms and projects the tetrahedral meshes more than 45 times per second in all experiments. The CPU-based deformation time increases significantly faster with respect to the number of deformation parameters applied to the model. Note that the number of deformation parameters corresponds to the number of linear combinations performed with the eigenvectors in Equation (5). When regarding more than 40 deformation parameters and higher polynomial degrees of the density function ($d = 2$ and $d = 3$), our GPU approach is 6 to 7 times faster than a multi-threaded deformation on the CPU that does not account for the projection time. A comparison of the GPU approach to the hybrid implementation reveals a performance gain of even 8 to 9 times in the same scenario.

7.3 Comparison to DRR from CT

To assess the rendering quality, we compared our projection approach to ground truth images that were generated from clinical CT data of a pelvis (resolution $512 \times 512 \times 531$). We first segmented the pelvis and extracted tetrahedral meshes of four resolutions (20k, 58k, 252k and 731k tetrahedra) with density functions sampled from the original CT data. The meshes served as input data to our projection method. Ground truth images were then projected from the CT data by means of a ray-casting approach, masking out the surrounding tissue of the

[1] Source: Stanford University Computer Graphics Laboratory, The Stanford 3D Scanning Repository

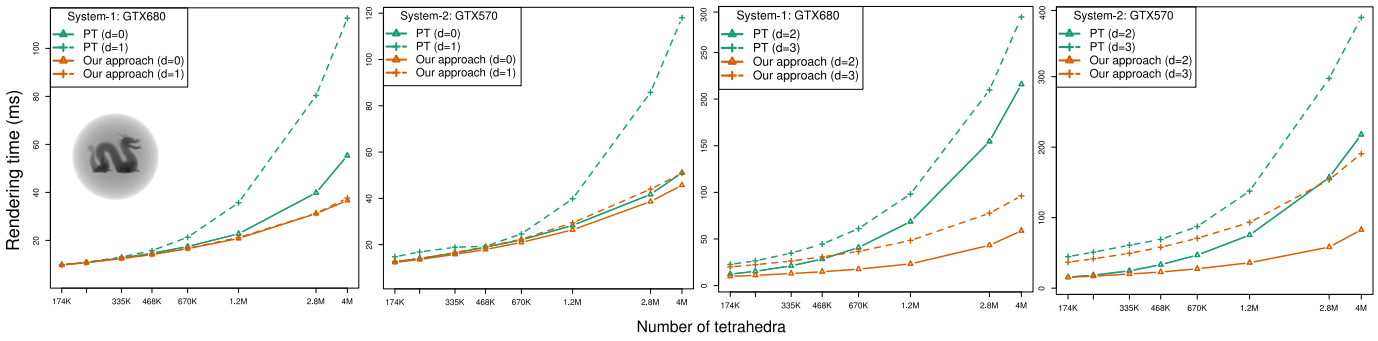

Fig. 7. Rendering performance (in milliseconds, averaged over 30 measurements) when generating virtual X-rays from the dragon datasets. Depicted here are the rendering times of both Sadowsky's projected tetrahedra (PT) method as well as our approach, recorded for tetrahedral meshes of various resolutions and density functions (degrees $d = 0$ to $d = 3$) on *System-1* and *System-2*.

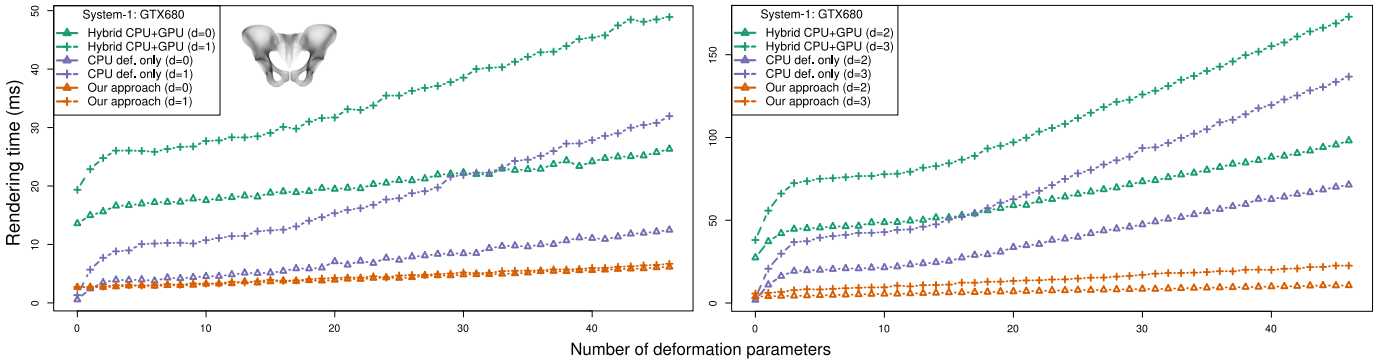

Fig. 8. Rendering time (in milliseconds, averaged over 100 experiments) of our GPU-based deformation and projection method. The experiments were performed on *System-1* using a pelvis SSIM that incorporates density functions of polynomial degrees $d = 0$ to $d = 3$. The CPU measurements (blue) account for multi-threaded deformations on the CPU only and do not include the time required to project the model. Also shown is the performance of a hybrid CPU (deformation) and GPU (projection) method that was implemented based on previous work.

	d=0	d=1	d=2	d=3
20k rms	0.085	0.072	0.070	0.068
abs	0.058	0.048	0.047	0.044
58k rms	0.058	0.046	0.042	0.038
abs	0.042	0.034	0.030	0.026
252k rms	0.043	0.035	0.030	0.027
abs	0.031	0.025	0.020	0.017
731k rms	0.035	0.029	0.025	0.024
abs	0.025	0.020	0.015	0.014

Table 1. Root mean square and mean absolute error between X-rays from model instances and ground truth projections from the segmented pelvis CT (depicted in Figure 9). The maximum pixel intensity in the ground truth image is 0.993.

pelvis. The ray-caster is a GPU implementation of the algorithm described in [5] that accurately solves the Beer-Lambert law for every pixel in the image plane.

During experiments, the viewport was set to a resolution of 1000^2 pixels with the projected pelvis covering 38% of the viewport pixels in anterior view. Similar to [14], we measured the quality of the projections generated with our method in terms of RMS distance and mean absolute error to the ground truth. Only those pixels were regarded in the distance measures that were either covered by the projected pelvis in the ground truth or in the virtual X-ray image compared to it.

Table 1 summarizes the results and Figure 9 provides example projections and difference images. Note how the image quality increases with higher degrees of the polynomials on a fixed mesh resolution.

We repeated the experiments and evaluated the projections generated with the GPU implementation of Sadowsky's method. The results are almost identical to those of our approach, differing by at most 7×10^{-5} in terms of RMS distance.

8 Discussion

The performance of our approach (in terms of rendering speed) exceeds those of the PT method when increasing the resolution of the rendered tetrahedral mesh to more than 335k cells. We attribute this to the reduced number of per-geometry operations and the lack of branching and looping in our approach. The per-geometry operations are executed for every tetrahedral cell and consequently more operations are performed the more tetrahedral cells are projected. When large numbers of tetrahedra are rendered, our GPU-approach benefits from executing fewer operations compared to PT. This is what we were aiming for in order to apply anatomical models of high resolution in the optimization process. From Table 1, we conclude that increasing the tetrahedra count, e.g. to 700k tetrahedra for a pelvic model, does have a positive influence on the projection quality and is thus desirable for future SSIMs.

On smaller mesh resolutions, per-fragment operations dominate the computation of virtual X-rays and the two methods perform equally well. Note that only three additional per-fragment operations are executed by our approach compared to Sadowsky's method. These additional operations did not impose a drawback in terms of performance compared to PT.

With density distributions of higher degree ($d = 2$ or $d = 3$), the

(a) (b) (c) (d) (e) (f)

Fig. 9. Comparison of virtual X-ray images to ground truth data. The projections (a) and (b) show close-ups of the right ilium (pelvis) and were generated by applying our method to a tetrahedral mesh of 252k tetrahedra and density function degrees $d = 0$ and $d = 3$ respectively. A ground truth projection from CT (c) and a clinical X-ray image (d) of the same pelvis are given for reference. Images (e) and (f) depict the differences of the virtual X-rays (a) and (b) to the ground truth (c) with red indicating positive error, blue negative error.

performance of both rendering methods is decreasing. This matches well with the investigation of Sadowsky et al. in [14]. They argue that the decrease is caused by the Bernstein density function terms integrated in the fragment shader, which show a "trend to exponential growth". However, our method still reaches interactive rates even for $d = 3$ and more than 1 million tetrahedra.

There are only minor differences in speed when projecting tetrahedral meshes with up to 750k tetrahedra and density functions $d = 0$ and $d = 1$. We believe this is due to hardware-accelerated vector operations, which allow to process one Bernstein coefficient ($d = 0$) just as efficiently as four coefficients ($d = 1$) stored in a vector.

The quality evaluation indicates that our method generates virtual X-rays similar to ground truth images from CT. We confirm the observation of previous work [14, 15, 22] that projections from higher-order density functions increase the similarity of virtual X-rays to ground truth data compared to those from lower degrees ($d = 0$ and $d = 1$). Our implementation renders identical results as Sadowsky's approach since it solves the same volume rendering integral and adopts the closed-form integration of higher-order density distributions.

The difference images to ground truth data show discrepancies at the boundaries of the projected pelvic bone (cf. Figures 9e and f). We attribute this to the geometric simplification that occurs when approximating the curvature of the pelvis with tetrahedral meshes. The tetrahedra fail to represent the thin, high-density cortical shell of the bone. This has to be considered for model generation in future studies.

Compared to a CPU-based deformation of SSIMs, our GPU-based deformation and projection approach benefits from highly parallel execution of the linear combination, fast texture access and hardware-accelerated vector operations. It is important to note that the CPU measurements do only account for a multi-threaded deformation of the model and not for projecting a virtual X-ray image. We consider them to reflect the upper bound of possible performance of previous hybrid CPU/GPU methods, since the deformation step remains a bottleneck when rendering SSIMs with higher-polynomial density functions and when regarding larger parameter spaces (e.g. more than 10 parameters). The rendering time of the CPU implementation grows non-linearly with approximately the first 5 parameters due to thread management overhead that outweighs the actual computation.

If a CPU-based deformation is applied, additional resources are required to push the deformed model to the GPU and to project it accordingly. The measurements of the hybrid solution give an example how these operations further decrease the overall rendering performance. It would be possible to deform the model in parallel on the CPU and concurrently project the deformed elements on the GPU. This would, however, also introduce additional overhead through thread synchronization. Our GPU-only approach in contrast does not copy the deformed mesh and density information between CPU and GPU but deforms and projects the SSIM in parallel on the GPU.

8.1 Reconstruction performance

To show the advantages of our GPU-only approach, we reconstructed five pelvic bones from standard X-ray images (see Figure 10). On average, approximately 6000 projections (and deformations) were executed for each experiment while fitting the pelvis SSIM (46 deformation parameters, density function degree $d = 2$) to the X-ray images.

In spite of this large number of virtual X-ray images that had to be generated, we were able to reconstruct a pelvic bone in 1:41 minutes average time using our GPU method. In contrast, it takes 9 to 10 minutes to perform the same reconstructions with the hybrid deformation/projection approach. Our GPU-based method generates 4 to 6 times more virtual X-rays in the same time frame. These additional samples can be utilized to avoid local optima during optimization, e.g. by executing the optimization process several times using different initialization parameters of the anatomical model.

From the performance evaluation in Section 7 it becomes obvious that by applying our GPU-only approach in theory a speed-up in reconstruction performance of a factor 9 and more is possible. The current implementation of the reconstruction framework reads back the virtual X-ray image from the GPU and evaluates the distance measure on the CPU, which leaves the GPU idle and thus slows down the overall process. This issue will be addressed in future work.

We are in the process of evaluating the reconstruction framework with regard to measuring pelvic tilt and orientation of the acetabulum (cotyloid cavity) from single X-ray images. Results will be published in a separate study.

9 Conclusion

We presented a fast and efficient method to generate virtual X-ray images from deformable anatomical models. The approach deforms and projects volumetric meshes in parallel on the graphics hardware and reaches interactive frame-rates while shifting computational burden from the CPU to the GPU.

In the context of reconstruction processes, we see several advantages of our GPU-only approach compared to previous hybrid CPU/GPU methods: First, while deforming and projecting the anatomical model on the GPU, other computations related to the reconstruction can be performed concurrently on the CPU, e.g. to control the optimization process. Second, the combined deformation and projection on the GPU shows a performance increase over CPU-based methods and accelerates the reconstruction process. Third, our method scales better with respect to the number of deformation parameters and the complexity of the underlying model. The GPU-based approach

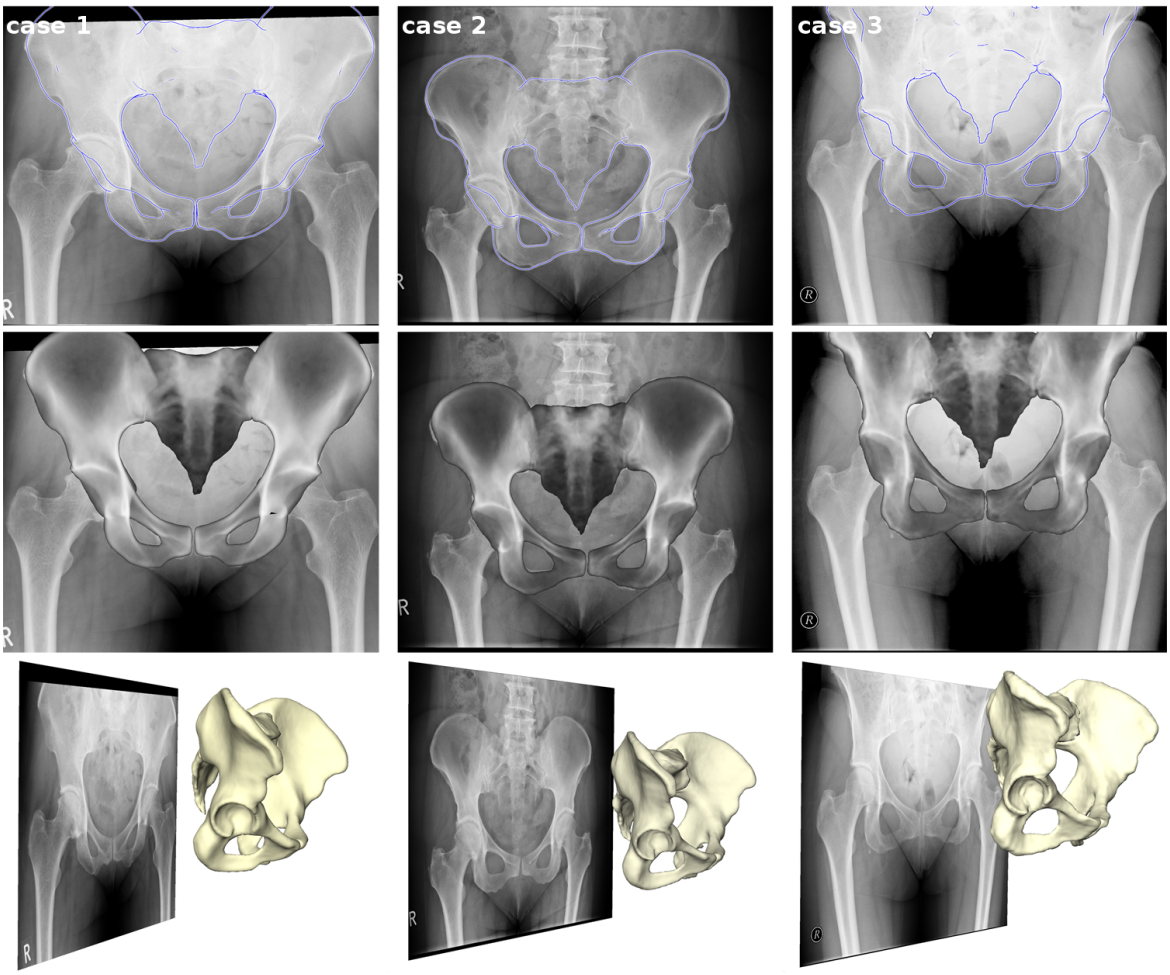

Fig. 10. Reconstruction of three pelvic bones from standard 2D X-ray images using our combined deformation and projection approach on the GPU. Every column depicts a different reconstruction case. From top to bottom: Silhouettes of reconstruction results projected into clinical reference images, reconstructed shape and density distributions (result), reconstructed pose of the pelvis in front of the image plane. In average, 54 deformations and projections were performed per second, with an average reconstruction time of 1:41 minutes.

allows for processing anatomical models with larger degrees of freedom and higher accuracy, both in terms of mesh resolution and density function. Future work will take advantage of those models to increase reconstruction accuracy while keeping reconstruction time as low as possible. This will include an in-depth evaluation of the application of our projection method to the problem of reconstructing a 3D anatomical model from clinical X-ray data.

In principle, the method supports different polynomial degrees on different tetrahedra in the same mesh, by issuing a separate rendering pass for each polynomial degree. This allows tetrahedral meshes to be tailored for certain anatomical structures and applications and will be investigated in the future.

Due to the feed-forward nature of the proposed pipeline, our method might be coupled with other deformation techniques on the GPU. It is possible that after a match is established between anatomy model and X-ray, slight deformations of the mesh, for example by applying green coordinates [8], might further enhance the reconstruction result. We will investigate whether such techniques can help to approximate patient-specific anatomies that are not expressed in the statistical model applied for reconstruction.

Our reconstruction method utilizes similarity measures such as mutual information to compare virtual X-ray images and clinical data. Currently, the virtual X-rays are read back from GPU framebuffer memory to evaluate the similarity measure on the CPU. The similarity measure could instead be evaluated on the GPU as well.

Additionally, we expect that a combination of the proposed projection method with GPU-based image-registration methods [3] will further increase both speed and accuracy of the 3D reconstruction process.

Acknowledgements

This work was in part supported by the EU-FP7 Project MXL (ICT-2009.5.2), the Berlin-Brandenburg School for Regenerative Therapies and the DFG Research Center Matheon. We would like to thank Markus Heller (University of Southampton) and the Charité Berlin for providing clinical CT data and X-ray images. Sincere thanks also go to the reviewers for their valuable comments.

References

[1] N. Baka, B. L. Kaptein, M. de Bruijne, T. van Walsum, J. E. Giphart, W. J. Niessen, and B. P. F. Lelieveldt. 2D-3D shape reconstruction of the distal femur from stereo X-ray imaging using statistical shape models. *Medical Image Analysis*, 15(6):840–850, Dec. 2011.

[2] J. Dworzak, H. Lamecker, J. von Berg, T. Klinder, C. Lorenz, D. Kainmüller, H. Seim, H.-C. Hege, and S. Zachow. 3D reconstruction of the human rib cage from 2D projection images using a statistical shape model. *International Journal of Computer Assisted Radiology and Surgery*, 5(2):111–124, Mar. 2010.

[3] O. Fluck, C. Vetter, W. Wein, A. Kamen, B. Preim, and R. Westermann. A survey of medical image registration on graphics hardware. *Computer Methods and Programs in Biomedicine*, 104(3):e45–e57, Dec. 2011.

[4] J. Georgii and R. Westermann. A Generic and Scalable Pipeline for GPU Tetrahedral Grid Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1345–1352, Sept./Oct. 2006.

[5] J. Krüger and R. Westermann. Acceleration Techniques for GPU-based Volume Rendering. In *IEEE Visualization 2003*, pages 287–292. IEEE, 2003.

[6] H. Lamecker, T. H. Wenckebach, and H.-C. Hege. Atlas-based 3D-Shape Reconstruction from X-ray Images. In *International Conference on Pattern Recognition 2006*, volume 1, pages 371–374. IEEE, 2006.

[7] D. Laney, S. P. Callahan, N. Max, C. T. Silva, S. Langer, and R. Frank. Hardware-Accelerated Simulated Radiography. In *IEEE Visualization 2005*, pages 343–350. IEEE, 2005.

[8] Y. Lipman, D. Levin, and D. Cohen-Or. Green Coordinates. *ACM Transactions on Graphics*, 27(3):78:1–78:10, Aug. 2008.

[9] T. Malzbender. Fourier Volume Rendering. *ACM Transactions on Graphics*, 12(3):233–250, July 1993.

[10] R. Marroquim, A. Maximo, R. Farias, and C. Esperanca. GPU-Based Cell Projection for Interactive Volume Rendering. In *Brazilian Symposium on Computer Graphics and Image Processing 2006*, pages 147–154. IEEE, 2006.

[11] A. Maximo, R. Marroquim, and R. Farias. Hardware-Assisted Projected Tetrahedra. *Computer Graphics Forum*, 29(3):903–912, Aug. 2010.

[12] S. Ourselin, A. Roche, S. Prima, and N. Ayache. Block Matching: A General Framework to Improve Robustness of Rigid Registration of Medical Images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention 2000*, pages 557–566. Springer, 2000.

[13] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever. Mutual-Information-Based Registration of Medical Images: A Survey. *IEEE Transactions on Medical Imaging*, 22(8):986–1004, Aug. 2003.

[14] O. Sadowsky and J. D. Cohen. Projected Tetrahedra Revisited: A Barycentric Formulation Applied to Digital Radiograph Reconstruction Using Higher-Order Attenuation Functions. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):461–473, July/Aug. 2006.

[15] O. Sadowsky, J. D. Cohen, and R. H. Taylor. Rendering Tetrahedral Meshes with Higher-Order Attenuation Functions for Digital Radiograph Reconstruction. In *IEEE Visualization 2005*, pages 303–310. IEEE, 2005.

[16] H. Seim, D. Kainmueller, M. Heller, S. Zachow, and H.-C. Hege. Automatic extraction of anatomical landmarks from medical image data: An evaluation of different methods. In *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pages 538–541. IEEE, 2009.

[17] A. Serrurier, S. Quijano, R. Nizard, and W. Skalli. Robust femur condyle disambiguation on biplanar X-rays. *Medical Engineering & Physics*, 34(10):1433 – 1440, Dec. 2012.

[18] P. Shirley and A. Tuchman. A Polygonal Approximation to Direct Scalar Volume Rendering. *ACM SIGGRAPH Computer Graphics*, 24(5):63–70, Nov. 1990.

[19] P. Steininger, K. Fritscher, and G. Kofler. Comparison of Different Metrics for Appearance-model-based 2D/3D-registration with X-ray Images. In *Bildverarbeitung für die Medizin 2008*, pages 122–126. Springer, 2008.

[20] M. Weiler, M. Kraus, M. Merz, and T. Ertl. Hardware-Based View-Independent Cell Projection. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):163–175, Apr./June 2003.

[21] T. Whitmarsh, L. Humbert, M. De Craene, L. M. Del Rio Barquero, and A. F. Frangi. Reconstructing the 3D Shape and Bone Mineral Density Distribution of the Proximal Femur From Dual-Energy X-ray Absorptiometry. *IEEE Transactions on Medical Imaging*, 30(12):2101–2114, Dec. 2011.

[22] J. Yao. *A statistical bone density atlas and deformable medical image registration*. PhD thesis, The Johns Hopkins University, 2002.

[23] G. Zheng. Statistical shape model-based reconstruction of a scaled, patient-specific surface model of the pelvis from a single standard AP x-ray radiograph. *Medical Physics*, 37(4):1424–1439, Mar. 2010.

[24] G. Zheng, J. Von Recum, L.-P. Nolte, P. Grützner, S. Steppacher, and J. Franke. Validation of a statistical shape model-based 2D/3D reconstruction method for determination of cup orientation after THA. *International Journal of Computer Assisted Radiology and Surgery*, 7(2):225–231, Mar. 2012.