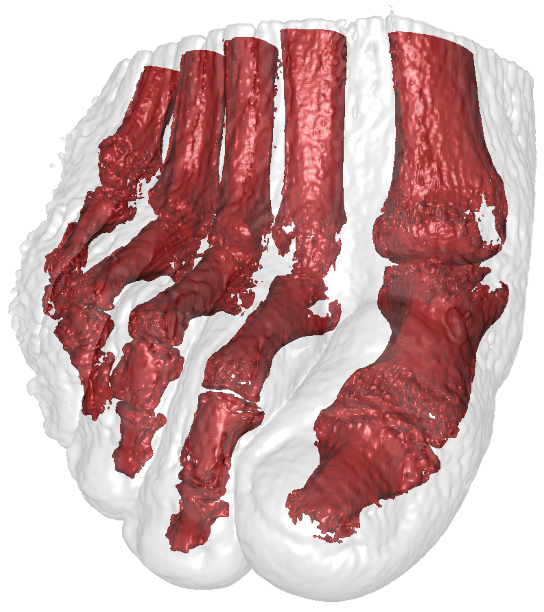
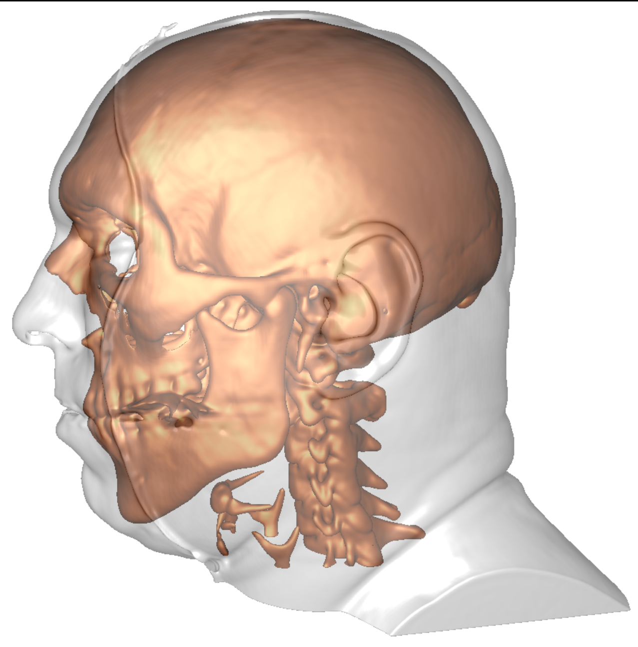
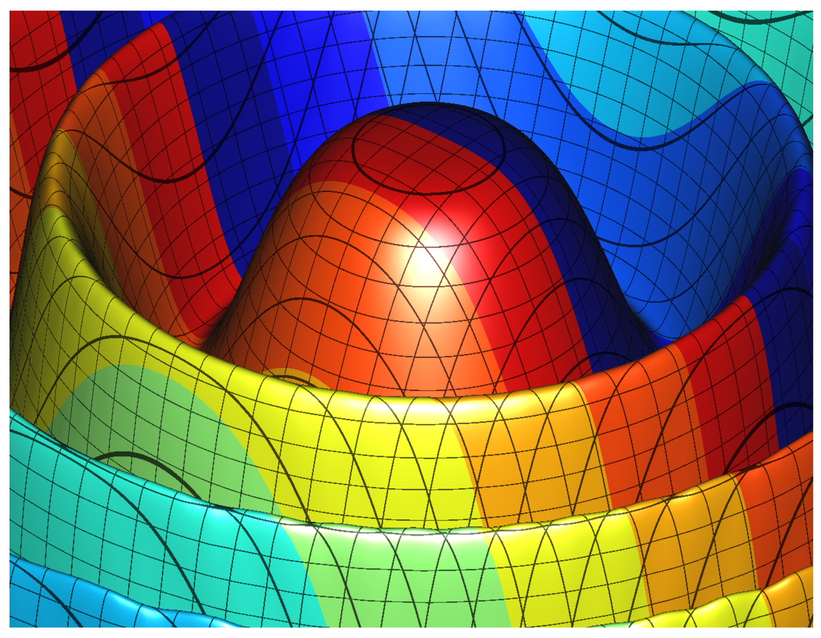
High-Quality Rendering of Interactively Varying Isosurfaces with Cubic Trivariate C^1-Splines

Thomas Kalbe Thomas Koch Michael Goesele

TECHNISCHE UNIVERSITÄT DARMSTADT







Introduction

We propose a novel rendering approach based on smooth trivariate splines defined w.r.t. uniform tetrahedral partitions. The splines are given in piecewise Bernstein-Bézier-form (BB-form) and are well suited for high-quality visualizations of isosurfaces from scalar volumetric data. Compared to tri-linear or higher-order tensor product splines, trivariate splines in BB-form have several advantages:

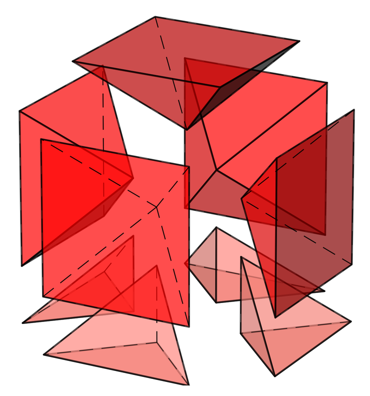
- well-known algorithms (de Casteljau, blossoming) exist
- artifacts resulting from tri-linear interpolation (e.g., staircasing) are significantly reduced, while the total degree of the splines does not exceed three
- the low polynomial degree makes stable evaluation and intersection of spline patches very efficient
- smooth gradients are directly available as a by-product
- the convex hull property of the BB-form allows for quick tests if a spline patch contributes to the surface
- data stencils are small (the direct 27-neighborhood is used)

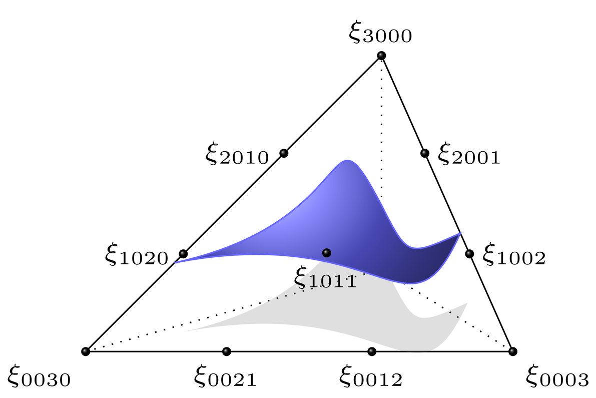
While our rendering algorithm is based on previous work (e.g., [1, 2, 3]), it is non-trivial to handle millions of smoothly connected spline patches simultaneously. We thus have significantly improved the usability of trivariate splines in real-world applications:

- interactive reconstruction and visualization of isosurfaces using a combined CUDA and graphics pipeline
- shader complexity and overall memory usage are significantly reduced, e.g., from using instancing
- spline coefficients are computed on-the-fly on the GPU

Mathematical Background

We use smooth quasi-interpolating splines [4] defined w.r.t. uniform *type-6 tetrahedral partitions*, where each data cube is split into 24 congruent tetrahedra.





On each tetrahedron T, the spline pieces are given in their piecewise BB-form, i.e.,

$$s|_T = \sum_{i+j+k+\ell=3} B_{ijk\ell} b_{ijk\ell},$$

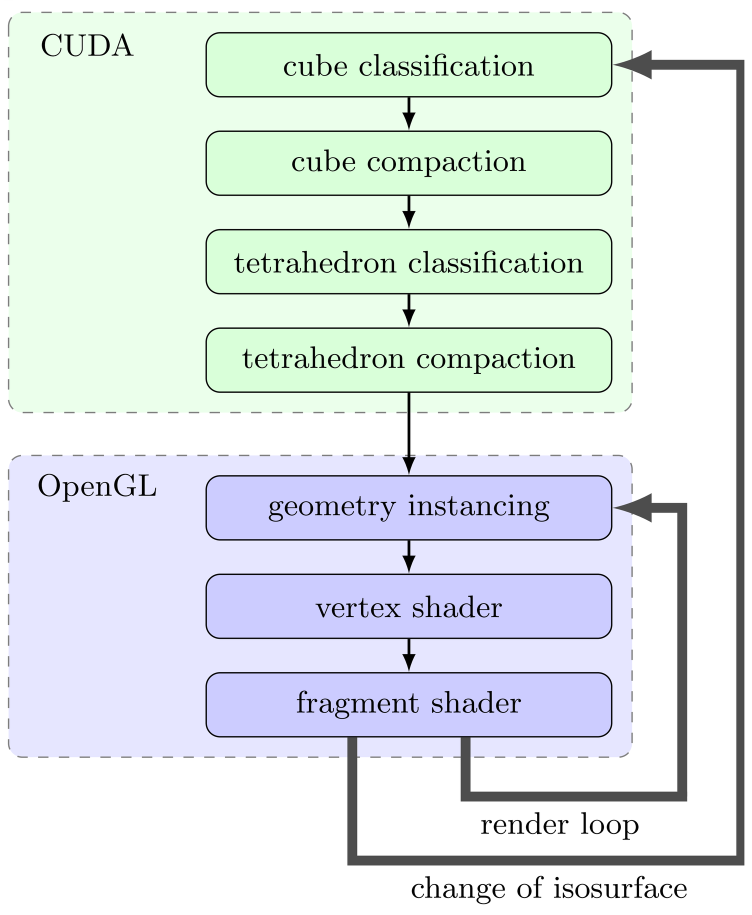
with the cubic *Bernstein polynomials* $B_{ijk\ell}$ and the scalar *BB-coefficients* $b_{ijk\ell}$. On each T, the $b_{ijk\ell}$ are associated with the 20 domain points $\xi_{ijk\ell}$ and are directly available from appropriate weightings of the data values in a symmetric 27-neighborhood of the centering value $f(\mathbf{v}_{ijk})$,

$$b_\xi = \sum_{i_0,j_0,k_0} \omega_{i_0 j_0 k_0} f(\mathbf{v}_{i+i_0,j+j_0,k+k_0}), i_0, j_0, k_0 \in \{-1, 0, 1\},$$

with $\omega_{i_0,j_0,k_0} \in \mathscr{R}^+$. The weights are chosen such that smoothness conditions as well as reconstruction guarantees for the splines and its derivatives are fulfilled.

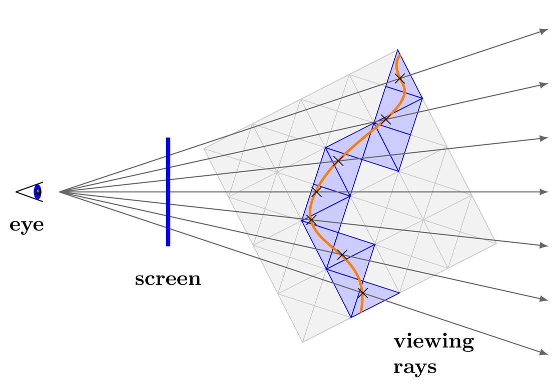
Details of the GPU Algorithm

Our GPU algorithm is organized as a combined CUDA and OpenGL pipeline:



The CUDA kernels are invoked for each change of isosurface:

- determine all the tetrahedra contributing to the surface
- use BB-hull property for quick tests if cubes or tetrahedra can be discarded
- the world coordinate translation of each contributing tetrahedron is packed into an array $\mathbf{T}_{\text{active},i}$, where $i = 0, \ldots, 23$, i.e., we have an array for each of the different tetrahedra in the type-6 partition
- use parallel prefix scans [5] from the *CUDA data parallel primitives* library for array compaction
- the arrays $\mathbf{T}_{\text{active},i}$ describe the bounding geometry of our spline surface



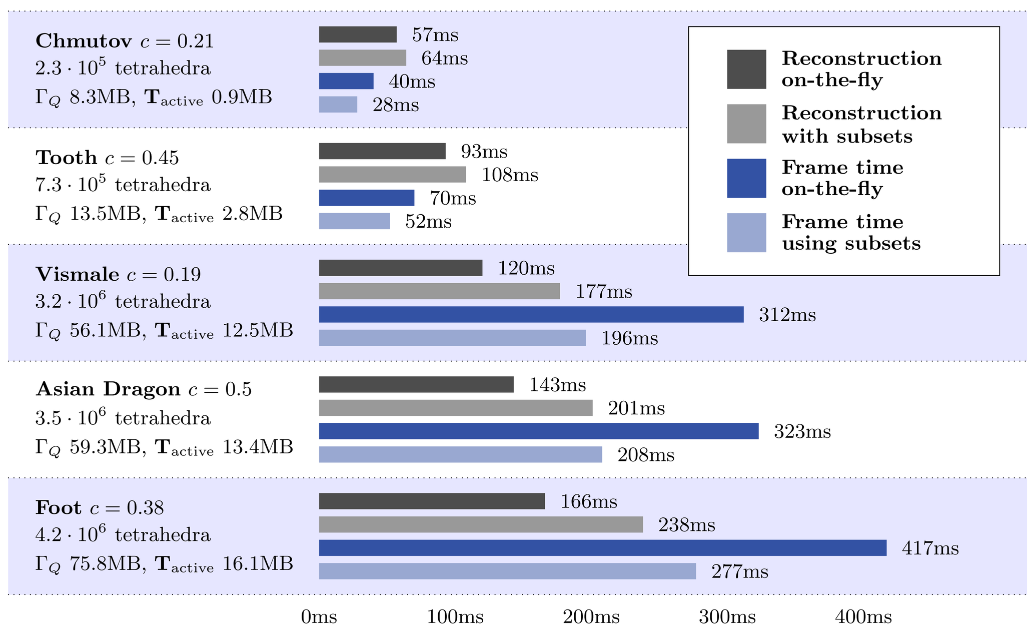
Visualization is done in a hybrid rasterization / ray casting approach where we use *instancing* to draw the bounding geometry.

- associated vertex and fragment programs for each type of tetrahedron to avoid conditional branches
- vertex programs set up various parameters for later ray-patch intersection tests
- spline coefficients are calculated on-the-fly from the volume data
- alternatively, a *pre-calculated subset* Γ_Q is used, from which the remaining coefficients are obtained by simple averaging

The fragment programs perform the ray-patch intersection tests:

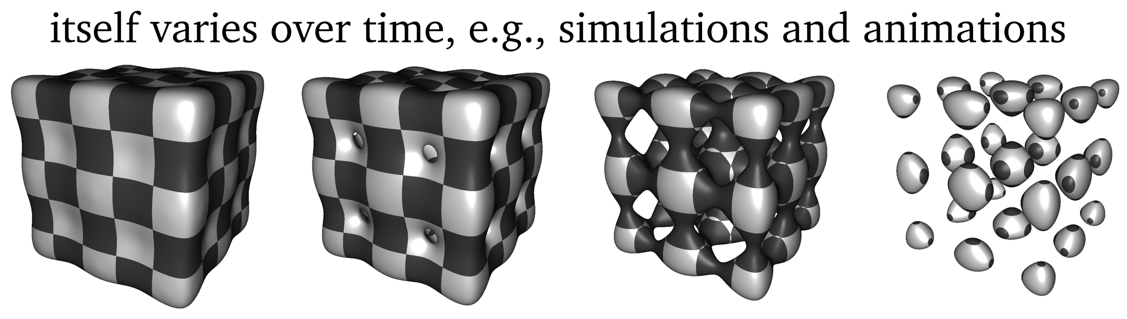
- we obtain a univariate cubic BB-curve restricted along the ray from *trivariate blossoming*
- solve monomial form $\sum_{i=0}^{3} x_i \cdot t^i$ for the ray parameter t with a simple and efficient Newton root finding algorithm: five iterations with $t_{j+1} = \frac{t_j^2 \cdot (x_2 + 2 \cdot t_j \cdot x_3) - x_0}{t_j \cdot (2 \cdot x_2 + 3 \cdot t_j \cdot x_3) + x_1}$ are sufficient to find precise intersections without notable artifacts
- take the first valid zero where all components of the associated barycentric coordinates (obtained from a linear interpolation) are non-negative
- no trigonometrics or recursion needed
- no ray exit points w.r.t. T needed in order to clip the intersections to the geometry
- gradients for later illumination are almost directly available from further linear interpolations and a few dot products

Results



We have shown that interactive and high-quality visualization of volume data with varying isosurfaces can be efficiently performed on modern GPUs. We conclude with a summary of the main features:

- our approach benefits strongly from the mathematical properties of the splines
- surface reconstruction and rendering are performed interactively using a combined CUDA and graphics pipeline
- surface reconstruction in less than one frame
- the obtained ray-patch intersections are precise, which is necessary for, e.g., texturing and volume clipping
- we can compute the spline coefficients on-the-fly or, alternatively, we use pre-calculated subsets of coefficients (higher memory consumption with slightly better frame times)
- the method scales well with the fast developing performance of modern GPUs and will directly benefit from increased numbers of multiprocessors and texture units
- the proposed algorithm can be used for an interactive variation of isolevels, as well as for applications where the data itself varies over time, e.g., simulations and animations



References

[1] C. Loop, J. Blinn. Real-time GPU rendering of piecewise algebraic surfaces. In *ACM Trans. on Graphics*, 25(3):664–670, 2006.

[2] J. Seland, T. Dokken. Real-Time Algebraic Surface Visualization. In *Geometric Modelling, Numerical Simulation, and Optimization*, Springer, 163–183, 2007.

[3] T. Kalbe, F. Zeilfelder. Hardware-Accelerated, High-Quality Rendering Based on Trivariate Splines Approximating Volume Data. *Eurographics*, 331–340, 2008.

[4] T. Sorokina, F. Zeilfelder. Local Quasi-Interpolation by cubic C^1 splines on type-6 tetrahedral partitions. In *IMJ Numerical Analysis*, 27:74–101, 2007.

[5] M. Harris, S. Sengupta, J. D. Owens. Parallel Prefix Sum (Scan) with CUDA. In *GPU Gems 3*, Addison-Wesley, 677–696, 2007.

HIGH-PERFORMANCE GRAPHICS hpg

TUD Graphisch-Interaktive Systeme