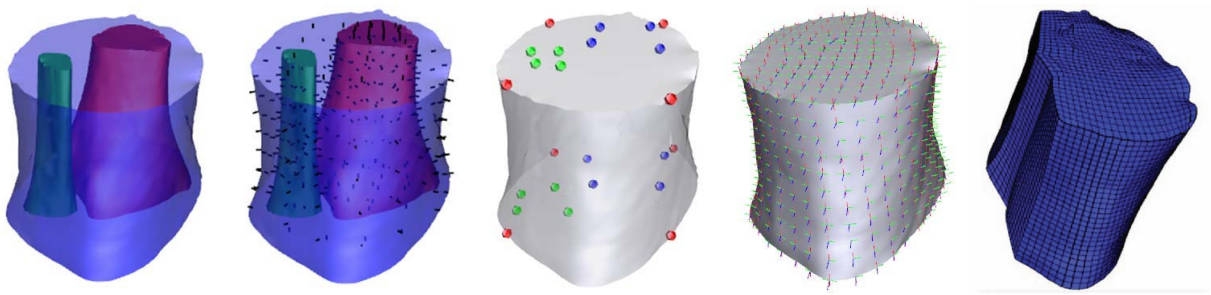
Pacific Graphics (2011)
Bing-Yu Chen, Jan Kautz, Tong-Yee Lee, and Ming C. Lin (Editors)

Feature-Aware Reconstruction of Volume Data via Trivariate Splines

Bo Li and Hong Qin

Stony Brook University (SUNY at Stony Brook), USA



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

Figure 1: *Main steps of the reconstruction. (a) Input model with boundary surfaces. (b) A set of direction vectors are predetermined as boundary constraints. (c) Corner points are manually selected to determine the domain structure. (d) In a frame field optimization procedure, an as-smooth-as-possible frame field is generated while enforcing the given constraints. (e) A volumetric parameterization.*

Abstract

In this paper, we propose a novel approach that transforms discrete volumetric data directly acquired from scanning devices into continuous spline representation with tensor-product regular structure. Our method is achieved through three major steps as follows. First, in order to capture fine features, we construct an as-smooth-as-possible frame field, satisfying a sparse set of directional constraints. Next, a globally smooth parameterization is computed, with iso-parameter curves following the frame field directions. We utilize the parameterization to remesh the data and construct a set of regular-structured volumetric patch layouts, consisting of a small number of volumetric patches while enforcing good feature alignment. Finally, we construct trivariate T-splines on all patches to model geometry and density functions simultaneously. Compared with conventional discrete data, our data-spline-conversion results are more efficient and compact, serving as a powerful toolkit with broader application appeal in shape modeling, GPU computing, data reduction, scientific visualization, and physical analysis.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling —Curve, surface, solid, and object representations

1. Introduction

For volumetric scalar fields defined over discrete samples, the reconstruction of the data is a fundamental problem with very significant applications: For instance, the trend of ever-increasing data size poses a great challenge in terms of both storage and rendering costs; Meanwhile, a reconstructed continuous representation is more desired in many graphics applications such as physical analysis and simulation.

An appropriate reconstruction should satisfy several quality requirements: **Accuracy.** The reconstructed model should faithfully preserve the density function; **Feature**

DOI: 10.2312/PE/PG/PG2011short/049-054

alignment. In regions with well-identified feature directions, parametric lines should guide and follow the shape feature; **Compactness.** The number of patch layout and the degrees of freedom (e.g., control points/coefficents) should be as few as possible; **Structured regularity.** Locally, each 3D patch is a subdivided cube-structured domain. Globally, the gluing between patches should avoid singularity; **As-homogenous-as-possible.** The density distribution in one single patch should be smoothly varying in favor of approximation accuracy (material-aware); and **Continuity.** A continuous representation supports high-order derivatives for high quality visualization and physical analysis [HCB05].

Unfortunately, existing reconstruction techniques rarely respect to all above criteria. For example, [RZNS03] has developed super splines but only on irregular elements (tetrahedral mesh); The hierarchical bounding box techniques [BNS01], [LHJ99] are not able to represent boundaries and features accurately; The conventional hexahedral mesh [She07] reconstruction includes a huge number of small volumes without global structure at all, which sabotages applications like subdivision, GPU computing, and tensor-product NURBS approximation. All of these serious limitations inspire us an ambitious goal: to create an integrated structured hexahedral volume that combines all aforementioned requirements together.

Contributions and overview. We provide a novel framework to reconstruct a discrete volume data into only a small number of regular patches and spline representations. The significant advantages include: Unlike bounding box, each patch maps to a regular tensor-product (cube) domain while maintaining the shape features. Thus applications like multiresolution editing, subdivision and spline approximation can be applied efficiently and GPU friendly. Each patch is as-homogenous-as-possible (material-aware) to simplify applications such as texturing and physical attributes modeling. Meanwhile, our framework guarantees the geometric continuity between cube patches in favor of a global continuous represention.

Our approach consists of the following major steps: (1) Starting with local direction vectors as constraints, we generate an optimized frame field to respect the shape feature (Section 2); (2) A regular structured parameterization of (u,v,w) is generated, whose gradients align with the generated frame field everywhere. Then we produce a set of volumetric patches through remeshing (Section 3); and (3) We construct on each patch a trivariate T-spline to approximate the shape and density function using as-few-as-possible control points (Section 4).

2. Frame Field

In order to generate the frame field, we start from selecting the most important features as constraints, which our frame field must respect. We introduce all the user interaction in this step (Section 2.1). In the second step we compute the optimization of a 3-direction frame field (Section 2.2).

2.1. Feature Constraints

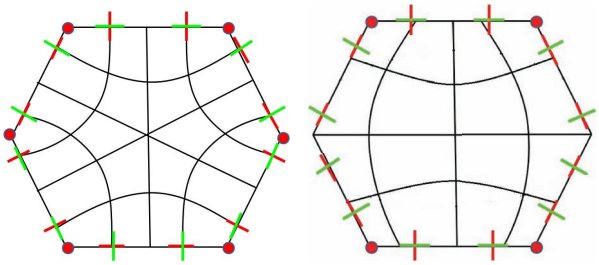
In order to generate a feature aligned frame field, we must pre-compute the most important features as the direction constraints. Furthermore, in order to get a simple and regular parametric domain structure, we also need to determine the domain shapes as well as alignment of each constraint direction (i.e., along gradient ∇u, ∇v or ∇w direction, respectively). Both tasks are detailed as follows:

Boundary surfaces and constraints. It is natural to consider features on the boundaries of all segmentations as constraints, because the final parameterization result must respect the shape of boundaries. Moreover, each sub-region within a boundary always tends to be as-homogenous-as-possible, which is an ideal property for final shape and density approximation. Therefore, we extract the boundary of segmentation and take the normal directions as our direction constraints.

Frequently, input datasets contain multiple structures and segments that need to be differentiated. However, if those features have the same density and gradient values, existing clustering methods are limited when trying to effectively classify those similar features accurately. Thus, we apply the texture-based classification method for the boundary surface extraction. First, statistical attributes can be extracted following the metrics defined in [HSD73], and each attribute is normalized into the range $[0,1]$. Then, in the interest of of fast computation and easy implementation, we use k-mean clustering in this texture-based high-dimensional parameter space to automatically detect different volumetric components as segments. Consequently, we choose the normal directions on all boundary surfaces as the direction constraints.

Domain and direction alignment. After determining direction constraints, we need to decide the alignment of each direction. In particular, it means that we must choose one parametric direction from ∇u, ∇v, or ∇w for each direction constraint. This necessary pre-processing step has a huge advantage in favor of generating parametric cube domain, as demonstrated in Figure 2. We construct a 2D frame field which respects the direction constraints on boundary edges. In Figure 2(a), we do not have any alignment requirement and the resulted frame field represents a complicated domain with central singularity. In Figure 2(b), we use 4 corners to divide the boundary into 4 segments and each segment corresponds to one boundary edge on the rectangular domain. Naturally, we align the direction constraint orthogonal to the iso-parameter on the boundary edge. Consequently, the resulted frame field represents a rectangular domain.

Motivated by the above 2D illustration, our pre-processing step includes the following interactive operations. (1) We predetermine the shape of the cube domain by manually constructing a group of cube domain to approximate the boundary shape; (2) On the boundary surfaces, we choose 8 corners for each cube domain. Figure 6 (Column 1) shows our cube domain construction and corner selection for



(a) (b)

Figure 2: *Two frame fields: The boundary direction constraints are not aligned in (a), but are aligned (b).*

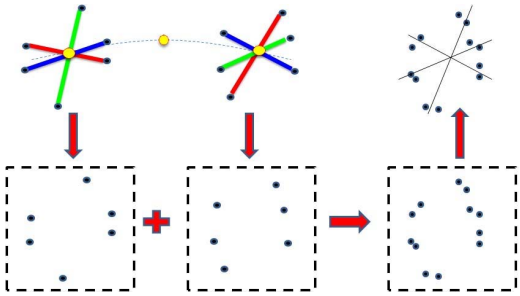
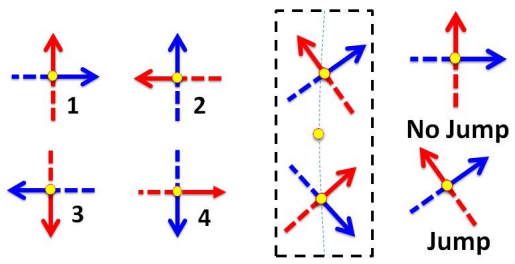


Figure 4: *Major steps of optimization: (1) Union of ending points. (2) ICP-registration. (3) Compute rotation to get a new frame.*

every input. As a result, the shortest paths between corners partition the boundary surfaces into patches, and each patch corresponds to an iso-parametric cube face; (3) For each direction constraint on one patch, we make a decision how it is aligned with the parametric coordinate gradient, which is orthogonal to the iso-parametric cube face.

2.2. Field Smoothing

[RVLL08] has studied the energy of a 2D cross field and simplified it to a linear representation. In our 3D volume, the challenge lies at smoothing 3 vectors in separate directions while maintaining their orthogonality. Another huge challenge for smoothing is "jump matching". It means that all permutation cases of direction alignment. Figure 3(a) shows all 4 "jump matching" cases for a 2-direction field. Similarly, we can have 24 "jump matching" cases for a 3-direction field. A "smart" optimization algorithm should dynamically change direction alignment to arrive at the best result. Figure 3(b) shows a simple frame smoothing with two adjacent neighbors. It demonstrates that using jump matching we are able to get a better smoothing result between neighbors, while traditional methods would fail.



(a) (b)

Figure 3: *(a) Jump matching: The smoothing energy between* 4 *cases should be zero ideally. (b) The smoothing results with/without considering period jump.*

To overcome these problems, our key idea is to compute the registration energy [BM92] between the central frame and its neighboring frames (Figure 4). Each frame would introduce 6 end positions $\{\mathcal{P}(\mathbf{v}_i)\} = \{\mathbf{p}_0, \ldots, \mathbf{p}_5\}$ at the end of 3 frame lines.

1. Get the union of all frame end positions on neighboring voxels: $\{S_2\} = \bigcup_{\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)} \mathcal{P}(\mathbf{v}_j)$.
2. The original point set $\{S_1\} = \{\mathcal{P}(\mathbf{v}_i)\}$ includes the frame ending positions of $\mathbf{v}_i$. Using the ICP-based registration [BM92], we compute a matrix T that approximately transforms voxels of $\{S_1\}$ to those of the approximated set $\{S_2\}$.
3. Decompose the transformation matrix T into a rotation matrix R and a shear matrix S using polar decomposition. Add the rotation R to the frame of $\mathbf{v}_i$.

The above algorithm is applied to each local frame iteratively until we obtain a satisfactorily smooth field. For any frame with a predetermined direction constraint, we first apply the above algorithm without considering constraints. Then we search for the closest direction $\vec{d}$ in the updated frame and rotate the frame to project $\vec{d}$ onto the direction constraint.

3. Volumetric Parameterization

In order to follow the generated frame field, the parameterization is computed as a solution to the following energy minimization problem on each node:

$$\mathbf{E} = \sum_{\mathbf{v}_i \in \mathbf{V}} ||\nabla u_i - \mathbf{u}_i^f||^2 + ||\nabla v - \mathbf{v}_i^f||^2 + ||\nabla w_i - \mathbf{w}_i^f||^2, \quad (1)$$

where u_i, v_i, w_i are the unknown parameters and $\mathbf{u}_i^f, \mathbf{v}_i^f$ and $\mathbf{w}_i^f$ are 3 frame field directions on each node. In practice, our parameterization algorithm has following steps:

1. Parameter constraints: Our previous pre-processing step (Section 2.1) partitions boundary surfaces to iso-parametric patches being mapped to cube faces. Now we set parameter constraints to guarantee that the nodes on each patch have the same parameter on u, v or w.
2. Energy minimization: Add these parameter constraints into the energy minimization equation. Compute the minimization to get the final parameterization result.
3. Remeshing: Guided by the generated parameter, we trace the iso-parametric lines and generate a small set of volumetric patches.

3.1. Energy Minimization

In order to minimize Eq.(1), we have to design a linear formulation of the gradient operator ∇. Our strategy is to approximate this gradient operator using a local polynomial function $I^H(u,v,w)$ around center voxel $\mathbf{v}_i$, we first assign a local parameter value (u_0,v_0,w_0) to $\mathbf{v}_i$. For each of its adjacent k-ring neighbor voxels $\mathbf{v}_j \in \mathcal{N}(\mathbf{v}_i)$, the local parameter is $(u_j,v_j,w_j) = (u_0+x_j-x_i, v_0+y_j-y_i, w_0+z_j-z_i)$. Then our cubic polynomial can be fitted as:

$$I^H(u,v,w) = \sum_{i,j,k\geq 0}^{i+j+k\leq 3} c_m u^i v^j w^k = \mathbf{P}(u,v,w)\mathbf{C}^T, \quad (2)$$

where $\mathbf{C}$ denotes the vector of unknown coefficients c_m, and $\mathbf{P}$ is the vector of $u^i v^j w^k$. Similarly, we can also formulate derivatives of u,v,w. For instance,

$$I^H_u(u,v,w) = \sum_{i,j,k\geq 0}^{i+j+k\leq 3} c_m i u^{i-1} v^j w^k = \mathbf{P}_u(u,v,w)\mathbf{C}^T, \quad (3)$$

where $\mathbf{P}_u$ is the vector of $iu^{i-1}v^j w^k$ (we set $u^m = 0$ if $m<0$). In the same way, we can also formulate other derivatives I^H_v and I^H_w.

In order to compute the currently unknown coefficients $\mathbf{C}$, we construct a fitting equation:

$$\mathbf{Q}\mathbf{C}^T = \mathbf{I}^D, \quad (4)$$

where $\mathbf{Q}$ is the fitting matrix. Each row $\mathbf{Q}_j$ in the matrix depends on a voxel $\mathbf{Q}_j = \mathbf{P}(u_j,v_j,w_j), j \in i \bigcup \mathcal{N}(i)$. And, $\mathbf{I}^D$ is the vector of discrete value I^D_j on each voxel. Because the size of unknown variables is very small, we can solve this linear least-square problem through multiplying the matrix $\mathbf{Q}$ by its transpose:

$$\mathbf{C}^T = (\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\mathbf{I}^D. \quad (5)$$

We observe that $(\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T$ is constant for every local function if we choose the same k for k-ring neighbors of each voxel.

Eq.(3) and Eq.(5) together compute the gradient operator. For instance, we represent ∇u_i as:

$$\nabla u_i = (\mathbf{P}_u\mathbf{C}^T, \mathbf{P}_v\mathbf{C}^T, \mathbf{P}_w\mathbf{C}^T) = (\mathbf{P}_u,\mathbf{P}_v,\mathbf{P}_w)(\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\mathbf{U}^D, \quad (6)$$

where $\mathbf{U}^D$ represents the vector of unknown scalar value u on $\mathbf{v}_i$ and its neighboring voxels. Then, we substitute them into the energy equation. for example, we can get from $\sum_{\mathbf{v}_i\in\mathbf{V}} ||\nabla u_i - \mathbf{u}_i^f||^2$ the following equation:

$$\sum_{\mathbf{v}_i\in\mathbf{V}} ||(\mathbf{P}_u,\mathbf{P}_v,\mathbf{P}_w)(\mathbf{Q}^T\mathbf{Q})^{-1}\mathbf{Q}^T\mathbf{U}^D - \mathbf{U}^f||^2, \quad (7)$$

where $\mathbf{U}^f$ is the vector of all $\mathbf{u}_i^f$. Eq.(7) is a typical fitting problem, which can be converted into a linear system $AU^T = B$ through computing $\frac{\partial \mathbf{E}}{\partial u} = 0$, where U^T is the vector of unknown value u on all voxels. We can simply solve it by the least square method.

Modified norm. It is obvious that feature orientation is more important than exact edge length. The orientation can be further improved by less penalizing stretch which is in the direction of the desired iso-lines. In order to achieve this, [BZK09] has introduced an anisotropic norm and we extend it to 3D vector computing:

$$||(u,v,w)||_{(\alpha,\beta,\gamma)} = \alpha u^2 + \beta v^2 + \gamma w^2.$$

This norm penalizes the deviation along the major directions with different weights. Then we modify the energy equation to the new form:

$$\sum_{\mathbf{v}_i\in\mathbf{V}} ||\nabla u_i - \mathbf{u}_i^f||_{(\varepsilon,1,1)} + ||\nabla v_i - \mathbf{v}_i^f||_{(1,\varepsilon,1)} + ||\nabla w_i - \mathbf{w}_i^f||_{(1,1,\varepsilon)}, \quad (8)$$

with $\varepsilon \leq 1$.

4. Spline Approximation and Experimental Results

The previous steps generate a set of regular structured parametric patches thus it is straightforward to define a regular high-order representation to approximate the shape and the density function of each patch. In our framework, we utilize T-splines [SCF*04] as a powerful approximation scheme. A trivariate T-spline [WLL*11] can be formulated as:

$$\mathbf{F}(u,v,w) = \frac{\sum w_i \mathbf{p}_i B_i(u,v,w)}{\sum w_i B_i(u,v,w)}, \quad (9)$$

where (u,v,w) denotes parameter coordinates, $\mathbf{p}_i = (X_i,Y_i,Z_i,I_i)$ denotes each control point, w_i and B_i are the weight and blending function sets, respectively. Each pair of $< w_i, B_i >$ is associated with a control point $\mathbf{p}_i$. Each $B_i(u,v,w)$ is a blending function given by $B_i(u,v,w) = N^3_{i0}(u)N^3_{i1}(v)N^3_{i2}(w)$, where $N^3_{i0}(u)$, $N^3_{i1}(v)$, and $N^3_{i2}(w)$ are cubic B-spline basis functions along u, v, and w, respectively. The detailed approximation techniques are discussed in [WLL*11].

4.1. Experimental Results

Table 1 summarizes the statistics of the performance of our reconstruction method on four models. It shows that our system effectively reconstructs the models with lower number of control points without sacrificing visual quality. Figure 5 visualizes the continuous representation results. It shows that our reconstructed models are able to preserve the shape and density information of the initial objects. Figure 6 illustrates more details about our parameterization by showing: the corner points, parameter domain, surface parameterization, and volumetric parametrization, respectively.

Limitation. It may be noted that, our framework may not perform well for highly textured scenes, or over-partitioned objects. Meanwhile, it may also encounter some difficulties when being used to handle highly branched models and fluid-like simulation. "Cracks" [PB06] may also occur when adjacent object boundaries do not coincide precisely. We have observed that, for rather complicated input, interactive corner selection and domain construction tends to be very time consuming.

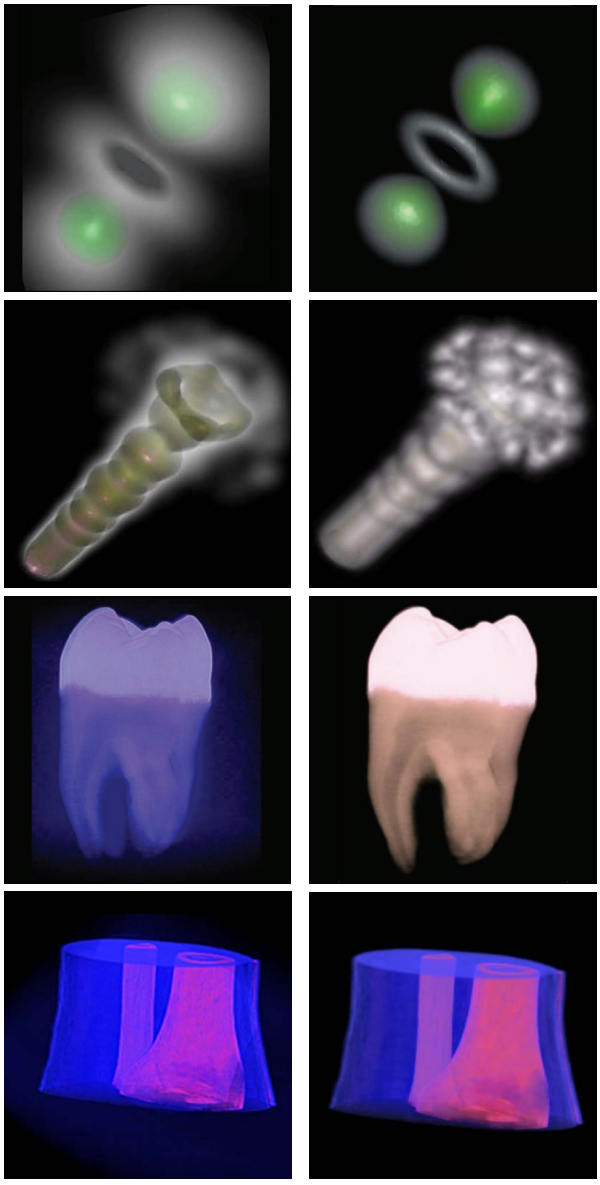


Figure 5: *Left column: Volume visualization using input discrete models. Right column: Reconstructed models.*

5. Conclusion and Future Work

We have proposed a method that reconstructs the discrete volumetric data into the regular continuous representation. Our conversion promises many good properties such as feature-alignment, compactness, regular structure, high-order representation, and as-homogenous-as-possible. These modeling advantages naturally prompt us to explore its unchartered potential in the near future. We anticipate further novel GPU-accelerated visualization techniques based on our high-order regular representations. Meanwhile, the connections between material-sensitive physical analysis/simulation and our continuous hyper-volume shape functions are of great interest for potential physics-based applications.

Table 1: *Statistics of various test examples: N_d, # of voxels; RMS, root-mean-square fitting error (density only, 10^{-2}); N_c, # of corners; N_c', # of control points.*

Model	N_d	RMS	N_c	N_c'
Atom	256^3	0.122	12	$1.5 * 10^4$
Fuel	64^3	0.877	16	$7.2 * 10^4$
Ankle	128^3	0.422	12	$1.6 * 10^4$
Tooth	$256^2 \times 161$	0.393	24	$5.1 * 10^4$

6. Acknowledgement

This research is supported in part by US NSF grants IIS-0710819, IIS-0949467, IIS-1047715, and IIS-1049448.

References

[BM92] BESL P. J., MCKAY N. D.: A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligences 14*, 2 (1992), 239–256. 3

[BNS01] BOADA I., NAVAZO I., SCOPIGNO R.: Multiresolution volume visualization with a texture-based octree. *The visual computer 17*, 3 (2001), 185–197. 2

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *Transactions of Graphics 28*, 3 (2009), 77:1–77:10. 4

[HCB05] HUGHES T., COTTRELL J., BAZILEVES Y.: Isogeometric analysis: Cad, finite elements, nurbs, exact geometry, and mesh refinement. *Computer methods in applied mechanics and engineering 194* (2005), 4135– 4195. 2

[HSD73] HARALICK R., SHANMUGAM K., DINSTEIN I.: Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics 3* (1973), 610–621. 2

[LHJ99] LAMAR E., HAMMANN B., JOY K.: Multiresolution techniques for interactive texture-based volume visualization. In *Visualization '99* (1999), pp. 355–362. 2

[PB06] PRICE B., BARRETT W.: Object-based vectorization for interative image editing. *The Visual Computer 22* (2006), 661–670. 4

[RVLL08] RAY N., VALLET B., LI W., LÉVY B.: N-symmetry direction field design. *ACM Transactions on Graphics 27*, 2 (2008), 1–13. 3

[RZNS03] RÖSSL C., ZEILFELDER F., NÜRNBERGER G., SEIDEL H.-P.: Visualization of volume data with quadratic super splines. In *Visualization '03* (2003), pp. 393–400. 2

[SCF*04] SEDERBERG T., CARDON D., FINNIGAN G., NORTH N., ZHENG J., LYCHE T.: T-spline simplification and local refinement. *ACM Transactions on Graphics 23*, 3 (2004), 276–283. 4

[She07] SHEPHERD J. F.: *Topologic and geometric constraint-based hexahedral mesh generation*. PhD thesis, Salt Lake City, UT, USA, 2007. 2

[WLL*11] WANG K., LI X., LI B., XU H., QIN H.: Restricted trivariate polycube splines for volumetric data modeling. *Technical report* (2011). 4

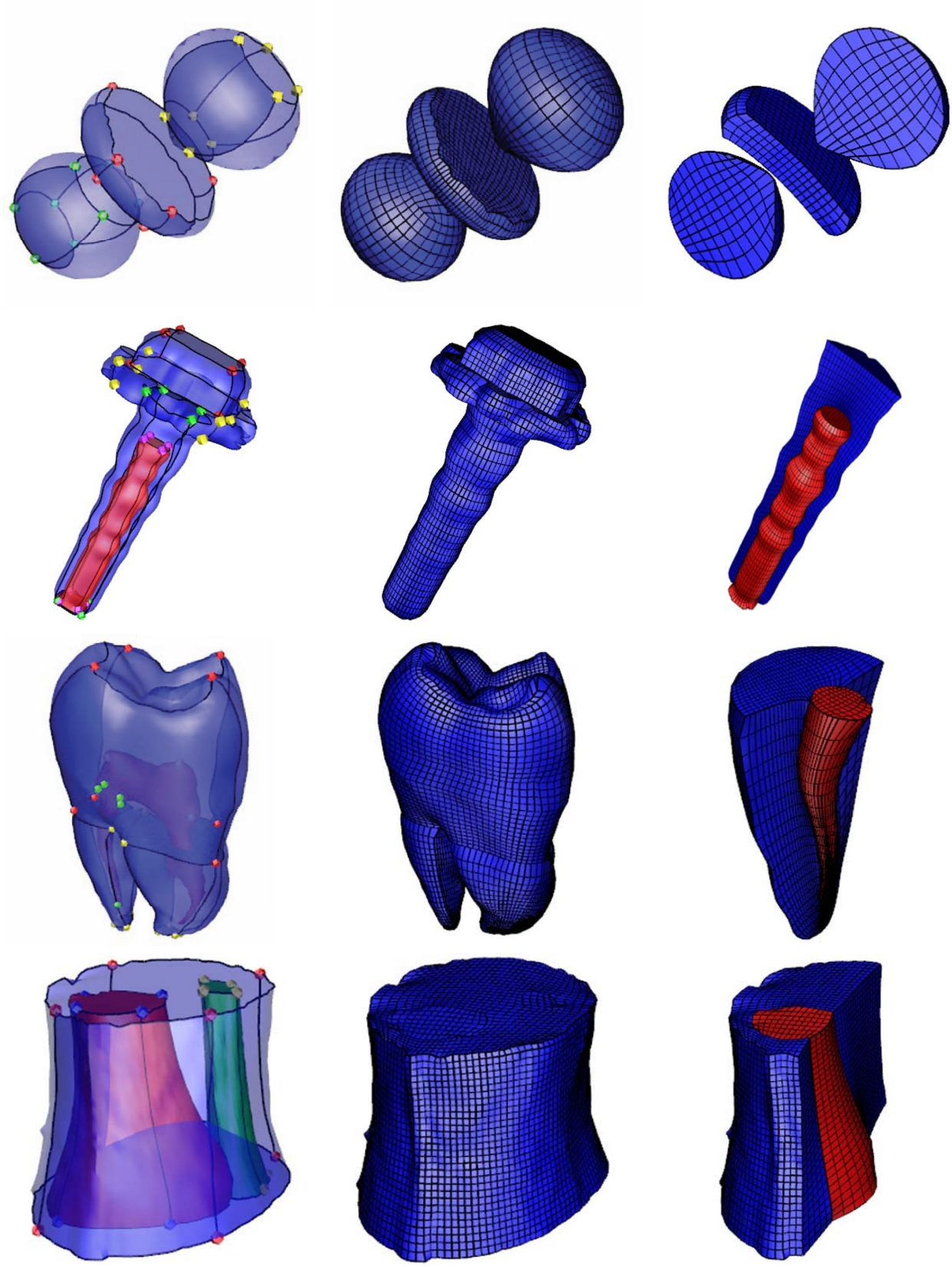


Figure 6: *Left: Corner points and their cube structures are highlighted by curves projected onto the inputs. Middle: Surface parameterization. Right: Interior volumetric parameterization.*