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

Short Papers

Feature-Aware Reconstruction of Volume Data via Trivariate Splines

Bo Li[1] and Hong Qin[1]

[1] Stony Brook University, USA

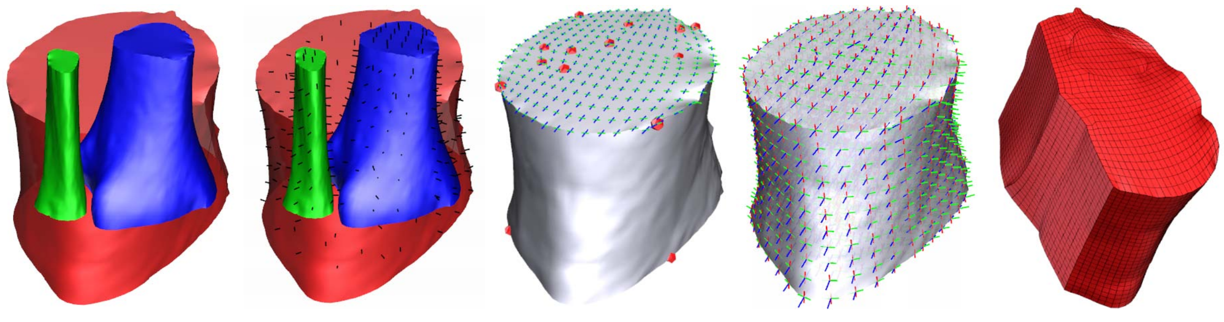

Figure 1: *Main steps of the reconstruction. (a) Input model with boundary surfaces. (b) A set of direction vectors are predetermined as the constraints. (c) Corner points are manually selected to determine the domain structure. (d) In a frame field optimization procedure, an as-smooth-as-possible frame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

Abstract

In this paper, we propose a novel approach that transforms discrete volumetric data directly acquired from scanning devices into continuous spline representations 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 parametrization is computed, with iso-parameter curves following the frame field directions. We utilize the parametrization to remesh the data and construct a set of regular-structured volumetric patch layouts, consisting of a small number of 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, GPC computing, data reduction, scientific visualization and finite element 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 a set of discrete samples, the reconstruction of the data is a fundamental problem with very significant applications. For instance in visualization, the size of volume data we have been dealing with increases dramatically to 1024^3 voxels commonly or even larger. This trend of ever-increasing data size poses a great challenge in terms of both storage and rendering costs thus model reconstruction is required.

An appropriate reconstruction should follow several quality requirements: **Accuracy** The reconstructed model should faithfully preserve the density function. **Feature alignment** In regions with well-pronounced feature directions, parametric lines should guide and follow the shape feature. **Compactness** The number of patch layout as well as the degree of freedom (e.g., control points/coefficents) for each patch 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 narrowed in favor of approximation accuracy. **Continuity** A continuous representation supports high-order derivatives for high quality visualization and physical analysis [HCB05].

An ideal reconstruction framework should optimize the output simultaneously with respect to all above criteria. However, existing techniques typically prefer offering a tradeoff between above conflicting requirements. For example, [RZNS03] has developed super splines to reconstruct discrete samples but the parametric domain is tetrahedral mesh; Other regular hierarchical structure methods like [BNS01], [LHJ99] have only produced axis-aligned cube block (i.e.,“flat block”) without feature alignment; Hexahedral mesh [She07] is another widely used reconstruction technique but it always leads to complicated domain with a lot of singularities.

Contributions and overview We provide a novel framework to reconstruct a discrete volume data into regular patches and spline representations. Our representation has significant advantages: Each patch has regular structure while maintaining the shape features. The whole data is compactly represented by a very small number of patches. The density in each patch is as-homogenous-as-possible thus both the shape and density function can be accurately approximated by a high-order spline representation.

In order to achieve these advantages, 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 parametrization 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). (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 operate 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 directions (i.e., along gradient $\nabla u, \nabla v$ or ∇w direction). Both tasks are discussed as follows:

Boundary surfaces and constraints It is natural to take features on the boundaries of all segmentations as constraints, because the final parametrization result must respect the shape of boundaries. Moreover, each sub-space in 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 at effectively classifying 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, for the sake of fast computation and easy programming, we use k-mean clustering in this texture-based high-dimension parameter space to automatically detect different volumetric components as segments. Consequently, we choose the normal directions one 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 choose one parametric direction from $\nabla u, \nabla v$ or ∇w for each direction constraint. This preprocessing 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 preprocessing includes the following interactive operation. (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 every input. Consequently, the shortest paths between corners partition

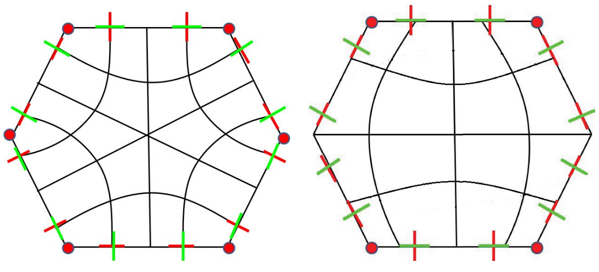

Figure 2: *Two frame fields: The boundary direction constraints are not aligned (a) / aligned (b).*

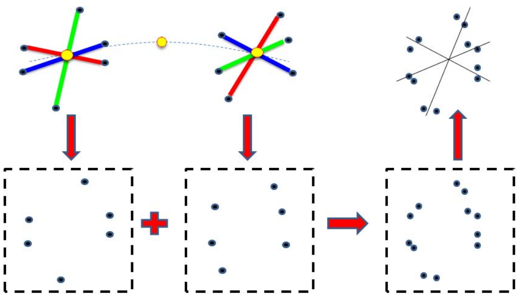

Figure 4: *Major steps of optimization: (1) Union of ending points. (2) ICP-registration. (3) Compute rotation to get updated frame.*

the boundary surfaces into patches, and each patch corresponds to an iso-parametric cube face. (3) For each direction constraint on one patch, we decide that 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 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 get 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 method fails.

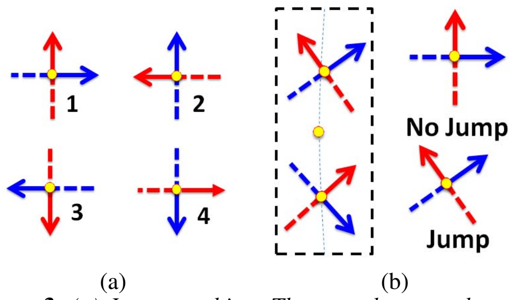

Figure 3: *(a) Jump matching: The smooth energy between 4 cases should be zero ideally. (b): The smoothing results with/without considering period jump.*

To overcome these problems, our key idea is to compute the registration energy [BM92] between center frame and its neighbor frames. Each frame gives 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 $\{S_1\} = \{\mathcal{P}(\mathbf{v}_i)\}$ is 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 computed on each local frame iteratively until we get a promisingly 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 Parametrization

In order to follow the generated frame field, the parametrization is computed as a solution to the following energy minimization problem:

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

where u_i, v_i, w_i are the unknown parameters and $\vec{u}_i, \vec{v}_i$ and $\vec{w}_i$ are 3 frame field directions. In practice, our parametrization algorithm has following steps:

1. Parameter constraints: Our previous preprocessing (Section 2.1) partition boundary surfaces to iso-parametric patches mapping 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 parametrization 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 Equation 1, we have to design a linear formulation of the gradient operator ∇. In order to get a local polynomial function $I^H(u,v,w)$ around center voxel $\mathbf{v}_i$, we 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 fitting cubic polynomial can be formulated as:

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

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

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

where $\vec{P}_u$ 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 describe other derivatives I_v^H and I_w^H.

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

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

where $\mathbf{Q}$ is the fitting matrix. Each row $\mathbf{Q}_{j:}$ in the matrix depends on a voxel $\mathbf{Q}_{j:} = \vec{P}(u_j,v_j,w_j), j \in i \bigcup \mathcal{N}(i)$. $\vec{I}^D$ is the vector of discrete value I_j^D 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:

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

We notice 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.

Equation 3 and 5 together describe the gradient operator. For instance, we represent ∇u_i as :

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

where $\vec{U}^D$ 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:

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

Equation 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 least square method.

Modified norm It is obvious that feature orientation is more important than exact edge length. The orientation can be 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:

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

with $\varepsilon \leq 1$.

4. Spline Approximation and Experimental Results

The previous steps generate a set of regular structured parametric patches thus it is very straight forward 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] for final approximation. 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. 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 $B_i(u,v,w) = N_{i0}^3(u)N_{i1}^3(v)N_{i2}^3(w)$, where $N_{i0}^3(u)$, $N_{i1}^3(v)$ and $N_{i2}^3(w)$ are cubic B-spline basis functions along u,v,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 processing on four models. It showcases that our system effectively reconstructs the model 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 object. Figure 6 shows more details about our parametrization: the corner points, parameter domain, surface parametrization and volumetric parametrization respectively.

Limitation Our framework does not perform well for highly textured scenes, or over-partitioned objects. It fails to handle highly branched models and fluid-like simulation. "Cracks" [PB06] may also occur when adjacent object

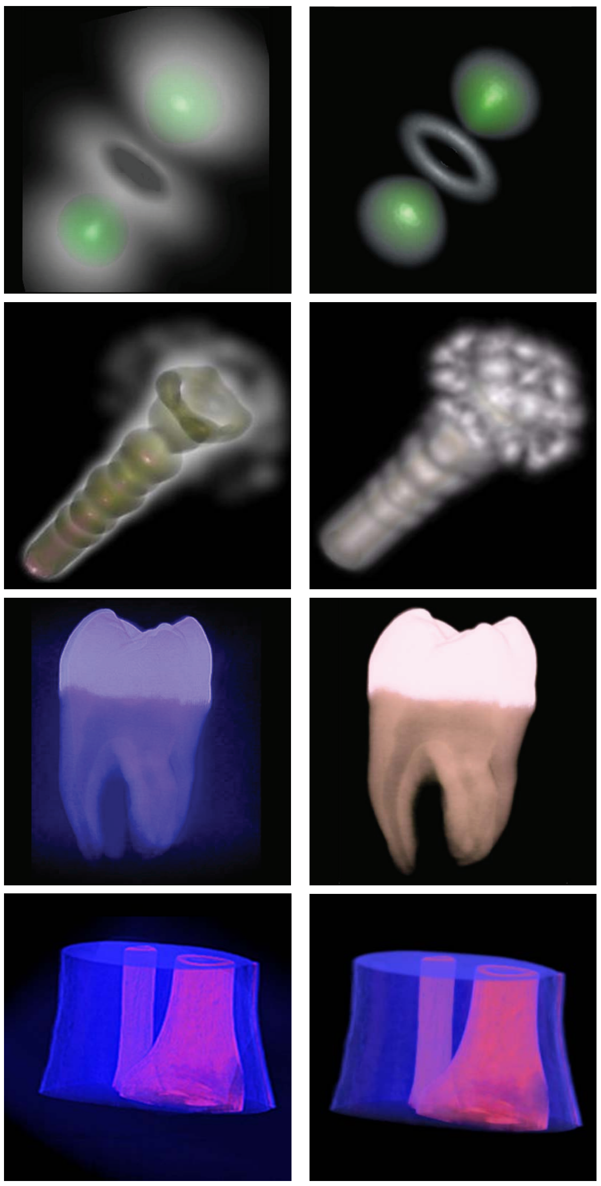

Figure 5: *Left column: Volume visualization using input discrete models; Right column: Reconstructed models.*

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	$5.1*10^4$

boundaries do not coincide precisely. Meanwhile, for some complicated input, interactive corner selection and domain construction is very time consuming.

5. Conclusion and Future Work

We have proposed a method that reconstructs the discrete volumetric data into the regular continuous representation. Our conversion promises a lot of 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 uncharted potential in the near future. We anticipate further novel GPU-accelerated visualization techniques based on our high-order regular representations. Meanwhile, the conjunctions between material-based physical analysis/simulation and our continuous hyper-volume shape functions are of great interest for potential physics-based applications.

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

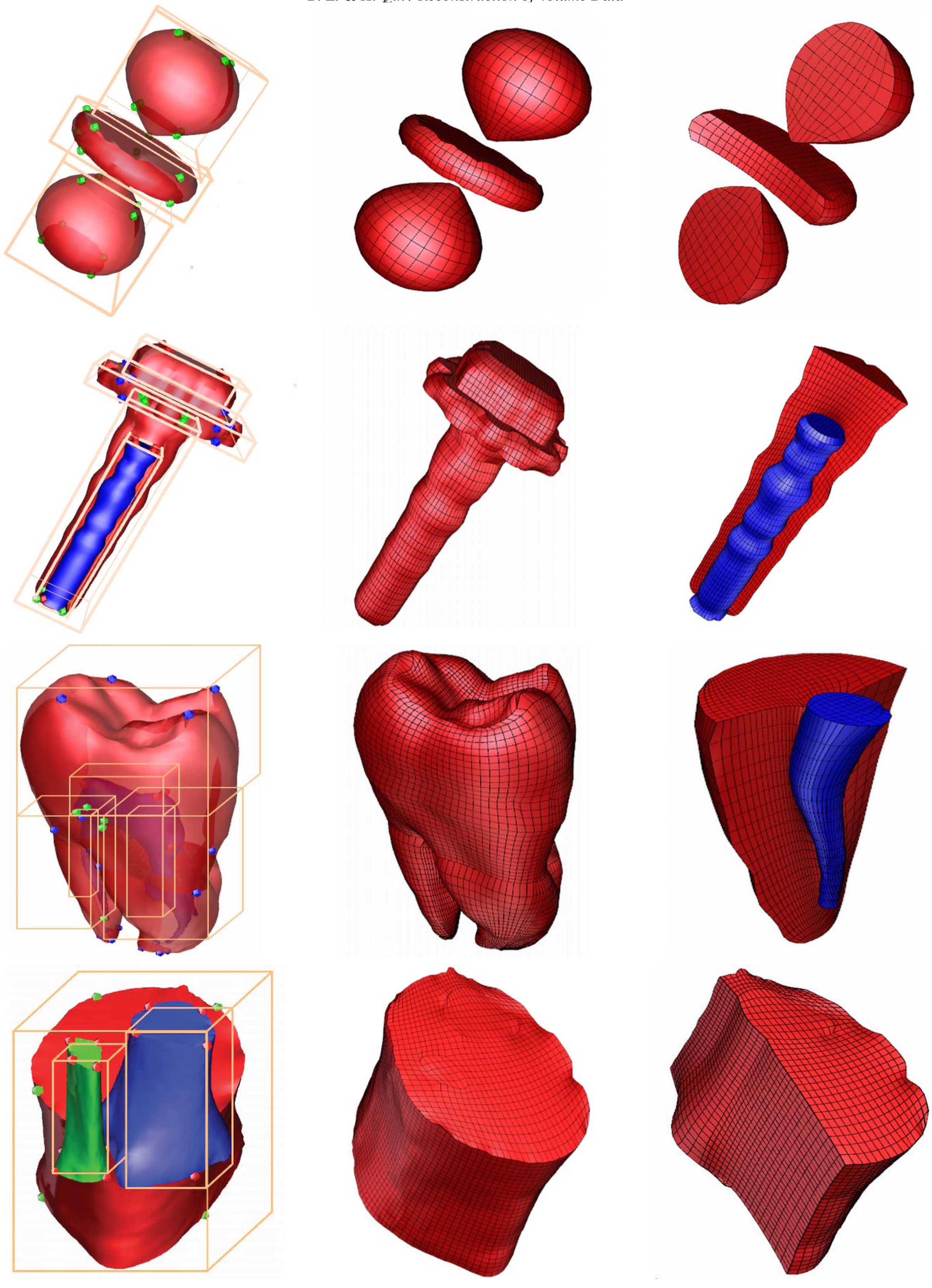

Figure 6: *Left: Corners and cube domain. Middle: Surface parametrization. Right: Interior parametrization.*