Volume Haptics with Topology-Consistent Isosurfaces

Loïc Corenthy, Miguel A. Otaduy, Luis Pastor, and Marcos Garcia

Abstract—Haptic interfaces offer an intuitive way to interact with and manipulate 3D datasets, and may simplify the interpretation of visual information. This work proposes an algorithm to provide haptic feedback directly from volumetric datasets, as an aid to regular visualization. The haptic rendering algorithm lets users perceive isosurfaces in volumetric datasets, and it relies on several design features that ensure a robust and efficient rendering. A marching tetrahedra approach enables the dynamic extraction of a piecewise linear continuous isosurface. Robustness is achieved using a continuous collision detection step coupled with state-of-the-art proxy-based rendering methods over the extracted isosurface. The introduced marching tetrahedra approach guarantees that the extracted isosurface will match the topology of an equivalent isosurface computed using trilinear interpolation. The proposed haptic rendering algorithm improves the consistency between haptic and visual cues computing a second proxy on the isosurface displayed on screen. Our experiments demonstrate the improvements on the isosurface extraction stage as well as the robustness and the efficiency of the complete algorithm.

Index Terms—Volume haptics, Marching tetrahedra, isosurface, proxy-based haptic rendering

1 Introduction

Volumetric data representations are very popular among the scientific community, notably in medicine, thanks to acquisition techniques such as Magnetic Resonance Imaging (MRI), Nuclear Magnetic Resonance Imaging (NMRI), Computed Tomography (CT) and Confocal Microscopy (CM) [1], [2].

The quality of the images obtained from those acquisition techniques greatly improved over the years. Alongside those technologies, the volume rendering techniques dedicated to visualization also enjoyed considerable enhancements. Nevertheless, the interpretation of dense volumetric datasets using only the visual channel remains a challenging task, especially when dealing with complex datasets.

Haptic interfaces offer a complementary sensory channel to interact with and manipulate 3D data. There are many successful examples combining visual and haptic interfaces for scientific visualization of multi-dimensional data and manipulation of large datasets, helping in research and comprehension tasks [3], [4], [5], [6]. In the case of volumetric data, *volume haptics* complement the visual cues of volume graphics, providing help in guidance and an improvement in information analysis [7]. The computation and display of forces that result from the interaction with digital objects or datasets is referred to as *haptic rendering* [8].

This work presents a robust volume haptic rendering algorithm enabling users to feel an isosurface directly from volumetric datasets. Robustness refers here to the algorithm capacity to avoid fall-through problems. The algorithm adapts proxy-based haptic rendering algorithms for navigation along isosurfaces. Coherent structures in the data are typically characterized by clusters of points with similar data values, therefore the navigation along isosurfaces may help users to perceive and understand the different structures in the dataset.

For visual rendering, we rely on state-of-the-art high-quality isosurface extraction methods based on trilinear interpolation [9], [10]. For haptic rendering, which must be executed at a much higher rate, we propose to extract a piece-wise linear isosurface instead, using a marching tetrahedra (MT) algorithm. Topological differences between the visual and haptic isosurfaces may produce inconsistent sensory stimuli. As a solution, we introduce an isosurface extraction method that retains the computational efficiency of piece-wise linear interpolation, but matches the topology of trilinear interpolation.

Our algorithm was designed according to the following desiderata:

- Isosurface navigation must be free of fall-through problems.
- It should allow the exploration of the full dataset, both in space and range of values, to enable the identification of potential structures at different isovalues.
- The haptic isosurface should be topologically equivalent to the visual isosurface.
- It must be computationally efficient, to ensure hard real-time haptic updates.
- It should support local modifications to the data, as well as time-varying data.
- It should enhance the consistency between the visual and haptic feedback.

• *L. Corenthy is with the Universidad Politécnica de Madrid, Madrid, Spain. E-mail: loic.corenthy@gmail.com.*
• *M.A. Otaduy, L. Pastor, and M. Garcia are with the Universidad Rey Juan Carlos, Madrid, Spain.
E-mail: {miguel.otaduy, luis.pastor, marcos.garcia}@urjc.es.*

*Manuscript received 10 Mar. 2015; revised 3 July 2015; accepted 31 July 2015. Date of publication 10 Aug. 2015; date of current version 11 Dec. 2015. Recommended for acceptance by M. Harders.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TOH.2015.2466239*

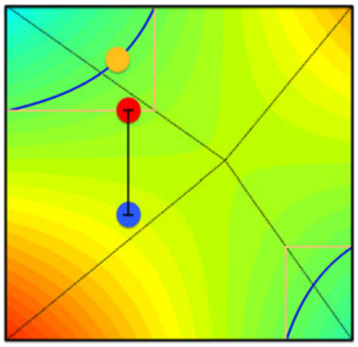


Fig. 1. The orange disk represents the *visual proxy*, the red disk represents the haptic proxy, and the blue disk represents the haptic device. The blue curves represent the visual isosurfaces obtained from the interpolated scalar field. The orange lines show the piecewise linear extracted isosurfaces, computed from the voxel decomposition in triangles (black dotted lines).

The algorithm fulfils all the previous features thanks to the following properties:

- Even though the algorithm relies only on local data, the underlying isosurface is continuous. This property, combined with our continuous collision detection algorithm, prevents fall-through problems.
- Isosurfaces are extracted dynamically.
- Each step of the rendering algorithm extracts only a local portion of isosurface, which ensures computational efficiency.
- Continuous collision detection relies on two extremely simple fundamental tests, tetrahedron-traversal tests and one plane-intersection test per tetrahedron, without the need for acceleration data structures.
- The isosurface changes smoothly in response to changes in the target isovalue, again helping the ability to explore the complete dataset.
- A new *visual proxy* is computed on the visual isosurface based on the position of the proxy on the haptic isosurface.

Fig. 1 resumes, in a 2D illustration, the different concepts above mentioned and detailed in this paper: trilinear interpolation, visual isosurface, haptic isosurface, tetrahedrization, haptic proxy, visual proxy and the haptic device position. This square is a 2D representation of a grid cell of a volumetric dataset. The values at each corner are known. The interior density values are calculated using trilinear interpolation. The resultant scalar field is drawn using a transfer function. The black segment shows the distance between the haptic feedback and the proxy used to compute the force feedback, see Eq. (11).

This paper is an extension of our previous work [11]. We inherit the constraint-based isosurface navigation algorithm, but we incorporate the novel isosurface extraction method. This paper is organized as follows: Section 2 provides a summary of related work. Section 3 details the isosurface extraction. Section 4 describes the haptic rendering algorithm per se and the computation of the visual proxy. Section 5 evaluates the benefits of the new Topology Preserving Body Cubic Centered (TPBCC) decomposition as well as its performance impact on the haptic rendering loop. Finally, Section 6 concludes and details future research directions.

2 Related Work

In general, volume haptic rendering methods can be grouped into two categories. A first category of methods renders forces that depend only on the value of the dataset at the exact location of the haptic interface point (HIP), and we refer to these methods as *local*. A second category of methods renders forces that depend on geometric structures obtained from a broader analysis of the dataset, such as isosurfaces, and we refer to these methods as *structural*.

Local methods process datasets at the current location of the haptic device in order to compute a force as a function of the data value [12], [13]. Early local methods used transfer functions to convey isosurface information [14]. The main advantage of these methods is their ability to render many different types of data, e.g., scalar, vector or tensor data. They are typically intended as a way to provide supplementary channels for high-dimensional data. Yet, their main disadvantages are the difficulty to convey global structure and the fact that the displayed force may disrupt the user's intent, because it is difficult to remain stationary at a point of interest while the user perceives a net force.

Structural methods, on the other hand, typically render isosurfaces on volumetric datasets and let the user change the desired isosurface value to explore the complete volume [15]. Users may navigate along isosurfaces and obtain global information about the structure of the data. Isosurfaces work best with scalar data; otherwise, some conversion from high-dimensional data to scalar values is necessary. A much more common approach consists in using the Marching Cubes (MC) algorithm [16] or some of its variants. Since its introduction, numerous attempts were made to improve the performance and the quality of resulting surfaces of MC, notably for large datasets. The visualization community, where MC originated, has widely discussed topological ambiguity problems [17], and several solutions have been proposed [18], [19]. Nielson and Hamann [20] focused on the ambiguities in the faces of a grid cell; Natarajan [21] and Chernyaev [22] showed the existence of additional ambiguities in the interior of a grid cell and how to deal with them. In both cases, the key to handle the ambiguities is to take into account the saddle points of the trilinear interpolant (bilinear interpolant in the case of the faces) representing the isosurface in a grid cell. Lopes and Brodlie [9] build upon the previously mentioned improvements and propose a global overview of an algorithm to extract an isosurface of good quality. Despite the possible solutions to the ambiguity problem, MC may produce local isosurfaces with multiple connected components on each cell, each composed of multiple planar segments. Our approach yields at most one planar segment per tetrahedron, which largely simplifies continuous collision detection.

It is also possible to use a MT algorithm for isosurface extraction from tetrahedral meshes [23], [24], [25]. The Topology Preserving Tetrahedra Decomposition (TPTD) algorithm presented by Sohn [26] is an equivalent approach to [9] for MT. The dynamic decomposition is divided into five cases. The resulting decomposition counts between

6 and 30 tetrahedra. In the simplest cases, the choice of six tetrahedra was made to avoid additional connection points. Both MC and MT have been employed in several haptics systems [27], [28], [29].

The work of Lundin et al. [7], [30], [31] is relevant in the area of point-based volume haptic rendering. Their work combines isosurface rendering with dynamic adaptation of the isosurface using the concept of penetrability, to naturally give the user global information. Their algorithm uses a Gradient-Based Isosurface Definition (GBID), i.e., the reconstructed isosurface depends only on the gradient at the current position. Others have introduced additional modifications [32]. Despite the benefits of their approach, it also suffers some limitations, because the isosurface is not guaranteed to be continuous at grid cell transitions. Our approach, based on isosurface extraction from a tetrahedral grid, ensures surface continuity.

In their work focused on virtual-reality-based surgical simulation, Chan et al. [28] presented a method for haptic rendering of deformable isosurfaces embedded within volumetric data. Their haptic rendering is a proxy-based method. Defining the isosurface through an implicit distance function and a gradient measurement, they were able to locate contact points on the isosurface by means of an interval bisection approach. Our Continous Collision Detection (CCD) step relieves us from having to use this type of optimization method while updating the proxy on an isosurface.

Proxy tools such as [28] are increasingly popular but single-point haptic rendering is still an active research area. Recently, Knott and Kuhlen [33] proposed a single-point haptic rendering approach for polygonal meshes using additional geometric information for a preventive collision response. The proposed contact model uses limited contact constraints reflecting the extent of the geometric features they correspond to, as opposed to the usually employed infinite half planes. Along with this new constraint formulation, they presented a solver for the dynamic complementarity problems created by the proposed time stepping implementation. The authors reported an experiment demonstrating that their technique alleviates the problem of over-constrained configuration spaces.

Vlasov et al. [34] presented a volumetric haptic rendering method similar to our approach. Same as in our algorithm, the proxy is progressively updated, one cell of the volumetric dataset at a time, toward the position of the haptic device. One notable difference between the two approaches is that we use a sub-cell resolution using the tetrahedra subdivision. Their method uses a ray-casting technique for collision detection, checking for each grid cell if the associated density value corresponds to what they refer to as "an obstacle". Collision responses was handled with a path finding approach.

Haptic rendering of implicit surfaces is also closely related to volume haptics. The works of Salisbury and Tarr [35] and Kim et al. [36] presented iterative methods to render nonlinear implicit functions, and as such they could be used to render the isosurface produced by trilinear interpolation. However, our method is considerably faster, as it avoids iterative searches altogether thanks to the extraction of a piecewise linear surface. In addition, those methods did not guarantee the detection of all possible surface interpenetrations, as they did not perform global consistency tests. Our constraint-based rendering algorithm analyzes all cells of the volume dataset along the path of the probe to ensure that the isosurface is never missed.

As previously mentioned, this work is based on the algorithm introduced in [11]. This new implementation improves the isosurface extraction step with a new method which guaranties that the topology of the extracted isosurface will match the topology of a trilinearly interpolated isosurface. As detailed in the following section, this improvement is achieved using our new TPBCC decomposition instead of our previous Body Cubic Centered (BCC) decomposition in our marching tetrahedra algorithm. This new approach also adds the notion of "visual proxy", a secondary proxy restrained on the isosurface displayed on screen that aims at improving the visuo-haptic consistency.

3 Isosurface Extraction

This section first defines the terminology that will be employed. Second, it presents the rules corresponding to the partition of the 3D dataset using the BCC decomposition. We will then explain the benefits and drawbacks of the TPTD approach compared to BCC before introducing our new TPBCC decomposition method which makes the most of both approaches. Finally, we will recall general properties of isosurface extraction inside one tetrahedron. Those properties are valid for any decomposition method.

3.1 Context and Definitions

Let $\mathcal{V}$ refer to a visual isosurface and $\mathcal{H}$ refer to a haptic isosurface. Our haptic rendering algorithm extracts haptic isosurfaces using a Marching Tetrahedra (MT) approach. MT applies a set of rules to create a set of tetrahedra partitioning each cell of the 3D grid. The goal of any MT algorithm is to choose the vertices of the tetrahedra for the partitioning to best approximate $\mathcal{V}$ while satisfying some predefined criteria, e.g., small number of tetrahedra, symmetry, topology correctness.

Once the cubic lattice is converted into a tetrahedral lattice, a local portion of $\mathcal{H}$ can be extracted from each tetrahedron in a consistent manner.

Let $\mathbf{D}$ refer to the regular 3D grid representing the volumetric dataset, i.e., a scalar field sampled regularly on a cubic lattice. Each vertex of $\mathbf{D}$, corresponding to a discrete position in space, is assigned a normalized scalar density value d (unitless values between 0 and 1).

Let $\mathbf{C}$ represent a unit cell of $\mathbf{D}$. Each vertex of $\mathbf{C}$ is assigned a density value d_i with $i \in \{0, 1, \ldots, 7\}$ as shown in Fig. 2a.

We compute $\mathcal{V}$ using trilinear interpolation in all the unit cells $\mathbf{C}$ in $\mathbf{D}$. Let F^c be a trilinear interpolant inside $\mathbf{C}$ and F^f be a bilinear interpolant inside a face of $\mathbf{C}$. These two polynomial functions are defined as follows:

$$\mathrm{F}^c : \left| \begin{array}{lcl} [0,1]^3 & \rightarrow & \mathbb{R}^+ \\ (x,y,z) & \mapsto & \alpha + \beta x + \gamma y + \delta z + \\ & & \epsilon xy + \eta xz + \nu yz + \rho xyz \end{array} \right. \tag{1}$$

with $\alpha = d_0$, $\beta = (d_3 - d_0)$, $\gamma = (d_4 - d_0)$, $\delta = (d_1 - d_0)$, $\epsilon = (d_7 + d_0 - d_3 - d_4), \eta = (d_2 + d_0 - d_4 - d_1) \nu = (d_5 + d_0 - d_4 - d_3)$, $\rho = (d_6 + d_3 + d_4 + d_1 - d_0 - d_5 - d_2 - d_7)$

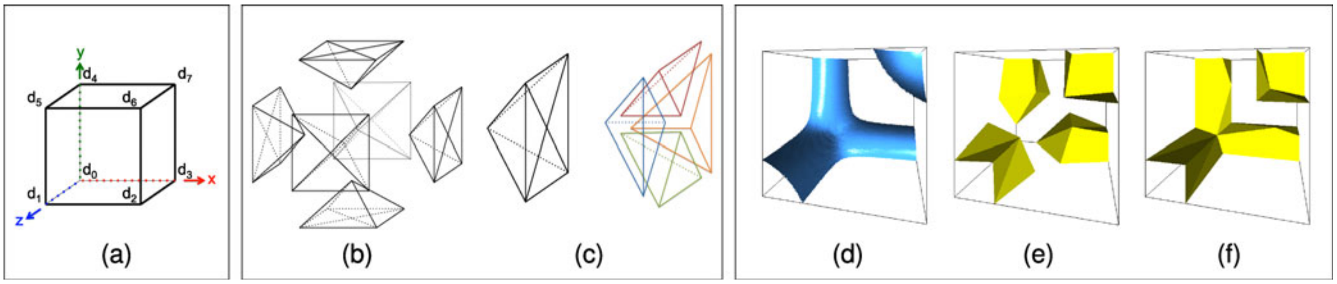


Fig. 2. (a) Density values and axis orientation in a unit cell. (b) Decomposition of a cube into $6 \times 4 = 24$ tetrahedra. Cube divided into six pyramids. (c) Each pyramid divided into four tetrahedra. (d) Visual isosurface defined through trilinear interpolation. (e) Haptic isosurface extracted using marching tetrahedra on a BCC lattice. (f) Our improved haptic isosurface, extracted using our novel topology-preserving decomposition TPBCC.

$$\mathrm{F}^f : \left| \begin{array}{lcl} [0,1]^2 & \rightarrow & \mathbb{R}^+ \\ (x,y) & \mapsto & \alpha' + \beta' x + \gamma' y + \delta' xy \end{array} \right. \tag{2}$$

with $\alpha' = d_0$, $\beta' = (d_3 - d_0)$, $\gamma' = (d_4 - d_0)$, $\delta' = (d_0 + d_7 - d_3 - d_4)$ for the face $z = 0$.

Given an isovalue τ, its corresponding isosurface $\mathcal{V}$ is defined as:

$$\mathcal{V}(\tau) = \{(x,y,z) | \mathrm{F}^c(x,y,z) = \tau\}. \tag{3}$$

$\mathcal{V}$ is a nonlinear continuous surface which is smooth inside each cell **C**. MT allows the extraction of $\mathcal{H}$, a continuous piecewise linear approximation of the isosurface. Both of these isosurfaces are continuous within the whole volume **D**. Contrary to $\mathcal{V}$, $\mathcal{H}$ maintains a continuous orientation only inside each tetrahedron. The linearity of $\mathcal{H}$ offers computational advantages. The counterpart is the loss of smoothness and visual appeal of the surface.

3.2 Static Decomposition: BCC

Chan and Purisima [37] originally introduced the BCC decomposition. The BCC lattice adds a central vertex $\mathbf{v}^c$ in the middle of **C**. The density value associated with $\mathbf{v}^c$ is computed using the trilinear interpolant Eq. (1). We build on BCC for its symmetry properties and its promising behavior when dealing with noisy data [38]. The original BCC needs to process two cubes of the lattice in order to recover an isosurface patch. We add a vertex $\mathbf{v}^f$ at the center of each face of **C**. Even if this modification results in a higher number of tetrahedra per cell, it allows the isosurface extraction algorithm to deal with a single cell at a time instead of two. Indeed, the tetrahedra of the regular BCC lattice span two grid cells. The density value associated with $\mathbf{v}^f$ is computed using the bilinear interpolant Eq. (2). Our version of BCC always produces 24 tetrahedra per grid cell. Figs. 2b and 2c illustrate the decomposition in two steps for an easier representation.

Besides its nice properties, the BCC decomposition remains a static decomposition. All the grid cells are consistently decomposed the same way, i.e., the vertices chosen to create the tetrahedra are always placed at the center of **C** and at the center of its faces, regardless of the density values d_0 to d_7. Unfortunately, as mentioned in the introduction, the BCC lattice does not guarantee that the topology of the resulting extracted haptic isosurface will always match the topology of the visual isosurface which is computed using trilinear interpolation. Fig. 2e illustrates the phenomenon (the parameters employed in this example are: $d_0 = 0.64$, $d_1 = 0.89, d_2 = 0.22, d_3 = 0.89, d_4 = 0.94, d_5 = 0.14, d_6 = 0.92$, $d_7 = 0.1, \tau = 0.69$). $\mathcal{H}$ is represented in yellow and $\mathcal{V}$ in blue. We can clearly see that for these particular density values in **C**, $\mathcal{H}$ computed using BCC gives four local disconnected components (Fig. 2e) instead of the two components corresponding to $\mathcal{V}$ (Fig. 2d). As comparison, $\mathcal{H}$ obtained using our TPBCC decomposition perfectly matches $\mathcal{V}$'s topology (Fig. 2f).

3.3 Dynamic Decomposition: TPTD

In order for $\mathcal{H}$ to have the same topology as $\mathcal{V}$, we consider a dynamic tetrahedral decomposition method which takes into account the density values of **C** to dynamically determine the shape and size of each tetrahedron. This method, called TPTD, was introduced by Sohn [26]. The main idea of Sohn's method is to insert what he refers to as *saddle points* and connect them to the corner vertices of **C** to generate tetrahedra. The saddle points correspond to the points where the gradient of the trilinear interpolant equals zero, i.e., stationary points of F^c and F^f, lying at the intersection of monotonic density regions in **C**.

By placing tetrahedral vertices at saddle points, the trilinear interpolation is guaranteed to be monotonic along the edges of the resulting tetrahedra. Then, an edge of the resulting tetrahedra intersects $\mathcal{V}$ if and only if it intersects $\mathcal{H}$, and it only intersects $\mathcal{H}$ once. As a result, the application of MT on the resulting tetrahedra yields an isosurface with the same topology as $\mathcal{V}$.

To compute the position of the saddle points, it is convenient to distinguish between **C** and its faces:

- A face saddle point $\mathbf{s}^f$ in a face of **C**, if it exists, is a point where:

$$\begin{cases} \frac{\partial \mathrm{F}^f}{\partial x}(x,y) = \beta' + \delta' y = 0 \\ \frac{\partial \mathrm{F}^f}{\partial y}(x,y) = \gamma' + \delta' x = 0. \end{cases} \tag{4}$$

  Solving the previous system gives:

$$\mathbf{s}^f = \left(-\frac{\gamma'}{\delta'}, -\frac{\beta'}{\delta'} \right). \tag{5}$$

  $\mathbf{s}^f$ is a valid point if it lies inside the corresponding face. $\mathbf{s}^f$ is actually a point with three coordinates. The missing one depends on the face considered to compute $\mathbf{s}^f$.

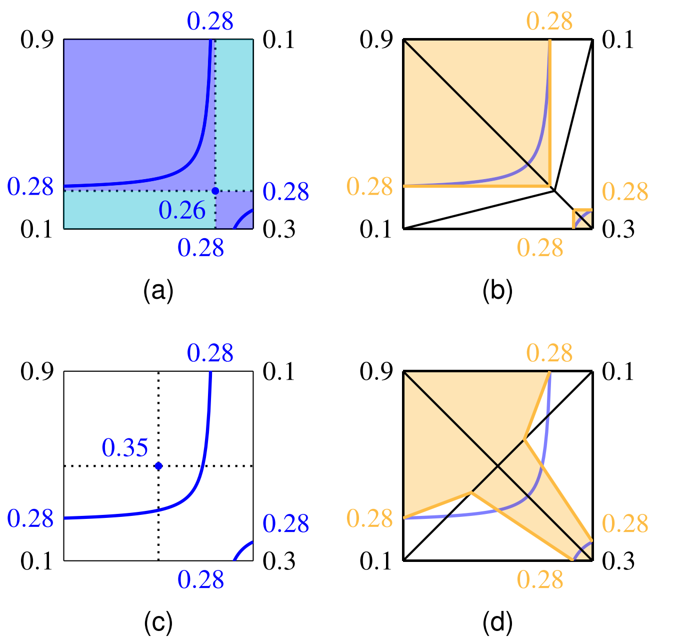


Fig. 3. 2D illustration of the properties of saddle points for isocurve extraction.

- A cell saddle point $\mathbf{s}^c$ inside $\mathbf{C}$, if it exists, is a point where:

$$\begin{cases} \frac{\partial F^c}{\partial x}(x,y,z) = \beta + \epsilon y + \eta z + \rho yz = 0 \\ \frac{\partial F^c}{\partial y}(x,y,z) = \gamma + \epsilon x + \nu z + \rho xz = 0 \\ \frac{\partial F^c}{\partial z}(x,y,z) = \delta + \eta x + \nu y + \rho xy = 0. \end{cases} \quad (6)$$

This system of quadratic equations can be solved by substitution. We obtain zero, one or two cell saddle points ($\mathbf{s}^c_1$ and $\mathbf{s}^c_2$):

$$\mathbf{s}^c_{1,2} = \begin{cases} x_{1,2} = -\frac{\gamma+\nu z_{1,2}}{\epsilon+\rho z_{1,2}} \\ y_{1,2} = \frac{\Omega_0+\Omega_1 z_{1,2}}{\Omega_2} \\ z_{1,2} = -\frac{\epsilon}{\rho} \pm \frac{\sqrt{\epsilon^2-(\rho/\Omega_1)(\epsilon\Omega_0+\beta\Omega_2)}}{\rho} \end{cases} \quad (7)$$

with $\Omega_0 = (\gamma\eta - \delta\epsilon), \Omega_1 = (\nu\eta - \delta\rho), \Omega_2 = (\nu\epsilon - \gamma\rho)$. $\mathbf{s}^c$ is a valid point if it actually lies inside $\mathbf{C}$.

For example, let us consider the 2D case in Fig. 3 which corresponds to a face case in $\mathbf{C}$. Fig. 3a shows in blue the isocurve corresponding to an isovalue of $\tau = 0.28$. The blue dot represents the position of the saddle point for this particular case. We can note that the saddle point lies at the intersection of two isolines (dotted lines), both with the same value, corresponding to the interpolated density value (0.26) at the saddle point position. For any value of $\tau > 0.26$, the corresponding isocurve will lie in the region marked in purple. Otherwise, the isocurve will lie in the region marked in turquoise. Fig. 3b illustrates the result of the 2D decomposition using the saddle point. The triangles representing the tetrahedra are shown in black and the extracted isocurve is shown in orange. The extracted isocurve's interior region, i.e., the areas where the density values are greater than τ, is shown in orange as well. There are two distinct regions corresponding to the two distinct parts of the isocurve in blue.

For comparison, Fig. 3c illustrates the same case using a BCC-like, 2D decomposition. The triangles are connected to

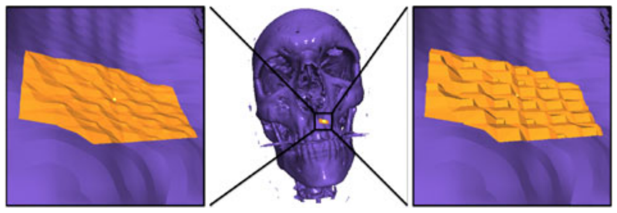


Fig. 4. Zoom on an extracted isosurface on the visible male model, see Fig. 10c. On the left, using BCC decomposition, and on the right, using TPTD.

the center of the face which has an interpolated density value of 0.35. The extracted isocurves corresponding to an isovalue of $\tau = 0.28$, shown in Fig. 3d, do not match the topology of the real isocurve. In this case, the interior region is incorrectly composed of a single component.

3.4 Combined Decomposition: TPBCC

TPTD guarantees to match the correct topology but yields worse results in the simplest decomposition cases than BCC. Indeed, TPTD introduces additional vertices only when there exist saddle points in $\mathbf{C}$. As illustrated in Fig. 4 and demonstrated in Section 5.1, this results in a smaller number of tetrahedra per grid cell but also a worse isosurface approximation.

We introduce here a new approach, the TPBCC decomposition, which combines the benefits of both BCC and TPTD. The following paragraphs detail the set of rules to partition $\mathbf{C}$ into tetrahedra. Each case is defined by the cardinality of $\mathbf{s}^f$ and $\mathbf{s}^c$. Let $\sigma^f = card(\mathbf{s}^f)$ and $\sigma^c = card(\mathbf{s}^c)$ be the number of face and cube saddle points respectively. Let Θ refer to a tetrahedron and $card(\Theta)$ refer to the maximum number of tetrahedra after the decomposition. The six different cases are the following:

1) $(\sigma^f = 0 \text{ and } \sigma^c = 0) \Rightarrow card(\Theta) = 24$
With no saddle points at all, TPBCC performs a BCC decomposition of $\mathbf{C}$.

2) $(\sigma^f = 1 \text{ and } \sigma^c = 0) \Rightarrow card(\Theta) = 20$
For the only face containing a saddle point, $\mathbf{v}^f$ is placed at the position of $\mathbf{s}^f$. In this case, there is no $\mathbf{v}^c$. $\mathbf{C}$ is divided into five pyramids, connecting $\mathbf{s}^f$ to the four corner vertices of each face except the one containing $\mathbf{s}^f$. Each pyramid is then divided into four tetrahedra.

3) $(2 \leq \sigma^f \leq 4 \text{ and } \sigma^c = 0) \Rightarrow card(\Theta) = 24$
In this case, TPBCC performs a decomposition similar to BCC. For each face with a saddle point, $\mathbf{v}^f$ is placed at the corresponding saddle point position. For the other faces, $\mathbf{v}^f$ stays at the center of the face. $\mathbf{v}^c$ is computed as the mean of the saddle points: $\mathbf{v}^c = 1/\sigma^f \cdot \sum_1^{\sigma^f} \mathbf{s}^f$. In this case, $\mathbf{C}$ is divided into six pyramids as in BCC, connecting $\mathbf{v}^c$ to the faces of $\mathbf{C}$. Again, each pyramid is then divided into four tetrahedra.

4) $(0 \leq \sigma^f \leq 4 \text{ and } \sigma^c = 1) \Rightarrow card(\Theta) = 24$
Here, $\mathbf{v}^c = \mathbf{s}^c$. Similarly to the previous case, for each face with a saddle point, $\mathbf{v}^f$ is placed at the corresponding saddle point position and for the other faces the vertices are placed at their centers. $\mathbf{C}$ is

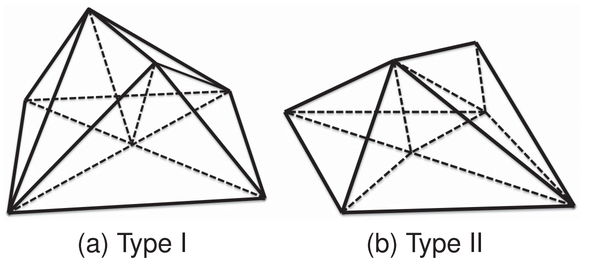


Fig. 5. Polyhedra in TPBCC.

divided into six pyramids, each one later subdivided into four tetrahedra.

5) $(\sigma^f = 6 \text{ and } \sigma^c = 0) \Rightarrow card(\Theta) = 28$

A diamond is created by connecting the six face saddle points. A vertex is placed inside the diamond, at the mean of the six face saddle points: $\mathbf{v}^c = 1/6 \cdot \sum_1^6 \mathbf{s}^f$. The diamond is decomposed into eight tetrahedraTwelve tetrahedra are created by connecting the two vertices of each of the 12 edges of **C** and the two face saddle points on the two faces which share that edge. Eight tetrahedra are created by connecting each of the eight faces in the diamond and a corresponding corner vertex of **C**.

6) $(\sigma^f = 6 \text{ and } 1 \leq \sigma^c \leq 2) \Rightarrow card(\Theta) = 30$

First, we consider the case of $\sigma^c = 2$. It generates two pyramids, two polyhedra of type I and two polyhedra of type II shown in Fig. 5. The pyramids and polyhedra are further decomposed into tetrahedra. The pyramids and polyhedra of each type always correspond to faces parallel to each other. The faces f of **C** are sorted according to the density value associated with the saddle point they contain: $d(\mathbf{s}_1^f) \leq d(\mathbf{s}_2^f) \leq \cdots \leq d(\mathbf{s}_6^f) \rightarrow f_1 \leq f_2 \leq \cdots \leq f_6$. Likewise, the two interior saddle points are also classified as $\mathbf{s}_{min}^c$ and $\mathbf{s}_{max}^c$ according to their associated density values. We first create the two pyramids connecting $\mathbf{s}_{min}^c$ to f_1 and $\mathbf{s}_{max}^c$ to f_6. Let $\mathbf{v}_{min}$ be the vertex of **C** adjacent to f_1, f_2, f_3 and $\mathbf{v}_{max}$ be the one for f_4, f_5, f_6. We first check which type of polyhedron should be used to connect f_2 and f_5 to $\mathbf{s}_{min}^c$ and $\mathbf{s}_{max}^c$. If the projection of $\mathbf{s}_{min}^c$ on f_2 is closer to $\mathbf{v}_{min}$ than the projection of $\mathbf{s}_{max}^c$ on f_2, then the polyhedra associated with f_2 and f_5 are of type I and the polyhedra associated with f_3 and f_4 are of type II. Otherwise, it is the contrary. The four polyhedra must be decomposed into tetrahedra in a way such that $\mathbf{v}_{min}$ should not be connected to the $\mathbf{s}_{max}^c$ and $\mathbf{v}_{max}$ should not be connected to $\mathbf{s}_{min}^c$. For the type I polyhedra, the decomposition is unequivocal. For the type II polyhedra, as illustrated in Fig. 5, it is necessary to choose between two possible opposite vertices to create the fifth tetrahedron. For f_1, f_2, f_3, the correct vertex is the one not included in f_1. Similarly, for f_4, f_5, f_6, the correct vertex is the one not included in f_6.

In the case $\sigma^c = 1$, we consider as if $\mathbf{s}_{min}^c$ or $\mathbf{s}_{max}^c$ moves to a face saddle point of a pyramid that is connected to the cell saddle point and hence the pyramid is collapsed. In this case, the pair of parallel faces to form the pyramids are chosen such that the face saddle point of the collapsed pyramid should not be f_1 or f_6.

Those six cases deal with all the possible combinations considering the following properties are always true: i) $0 \leq \sigma^f \leq 6$ and $0 \leq \sigma^c \leq 2$ ii) $\sigma^f \neq 5$ iii) $\sigma^c \neq 2$ unless $\sigma^f = 6$. All the possible isosurface topologies of the trilinear interpolant and their associated tetrahedral decomposition are illustrated in Fig. 11. This figure shows how the 30 cases that usually have to be distinguished in MC are reconstructed using TPBCC (we do not show the case where there is no isosurface in the cube). The haptic isosurface is shown in yellow and its equivalent visual isosurface in blue. We use the same notation as in [9] and [21]: the first label number denotes the original MC case number, the second indicates the number of faces with an ambiguity and the third the resolution of an interior ambiguity.

3.5 Isosurface Extraction in a Tetrahedron

Once **C** is partitioned into tetrahedra, barycentric interpolation allows us to compute the portion of $\mathcal{H}$ inside each tetrahedron Θ. The barycentric coordinates $(\theta_1, \theta_2, \theta_3, \theta_4)$ of any point $\mathbf{p} = (x, y, z) \in \Theta$ can be computed by the following equation:

$$(\theta_1, \theta_2, \theta_3, \theta_4)^T = \mathbf{A^{-1}} \cdot (x, y, z, 1)^T, \tag{8}$$

where the columns of the matrix **A** are the homogeneous coordinates of the four vertices of Θ. Thus, density values at point **p** can be interpolated as:

$$d(\mathbf{p}) = \sum_{i=1}^{4} \theta_i d_i, \tag{9}$$

where (d_1, d_2, d_3, d_4) are the densities associated with the vertices of Θ.

The gradient of an interpolated scalar field is constant when using barycentric coordinates to interpolate the scalar field inside a tetrahedron. The gradient at any point $\mathbf{p} = (x, y, z) \in \Theta$ is constant and orthogonal to all the isosurfaces. For $\mathbf{A}^{-1} = (a_{i,j})$, the local gradient can be computed by means of the following equation:

$$\mathbf{grad}(\mathbf{p}) = \left(\sum_{i=1}^{4} a_{i,1} \cdot d_i, \sum_{i=1}^{4} a_{i,2} \cdot d_i, \sum_{i=1}^{4} a_{i,3} \cdot d_i \right)^T. \tag{10}$$

In the static decomposition case (BCC), $\mathbf{A}^{-1}$ can be precomputed only once because the position of all the vertices of all the tetrahedra in **C** are known in advance. However, in the dynamic case (TPBCC), $\mathbf{A}^{-1}$ must be dynamically computed. Since the gradient is constant, all the isosurfaces inside a particular tetrahedron are parallel to each other. The scalar field being continuous, a continuous modification on τ produces a continuous change on $\mathcal{H}$ itself, a key feature for the exploration of volumetric datasets.

A MT approach extracts the isosurface locally on each tetrahedron, i.e., finding the points inside the tetrahedron where densities match a specified value of the isovalue τ as illustrated in Fig. 6. The algorithm computes the intersection between the interpolated isosurface and the edges of Θ. Those intersection points are arranged in triangles or quadrangles. The whole isosurface can be reconstructed simply

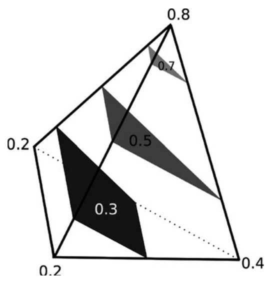


Fig. 6. Isosurfaces corresponding to $\tau = 0.3, \tau = 0.5,$ and $\tau = 0.7$ in a tetrahedron.

by connecting the triangles and the quadrangles of different tetrahedra by their common edges.

4 Constraint-Based Haptic Rendering

Proxy-based three-Degree-Of-Freedom (DoF) haptic rendering algorithms take as input the position of a haptic device, transform it into the corresponding position in the reference system of the 3D dataset, i.e., the **HIP** (also commonly referred to as probe), compute a force value based on this **HIP** position, and then display this force to the user. The following paragraphs first present a general description of our algorithm, which was already introduced in our earlier work [11]. Then the details about the CCD with the tetrahedral mesh, designed to ensure a robust constraint-based haptic rendering, are presented. Finally, we describe a novel contribution of this paper, the visual surface contact point, which ensures that visual and haptic feedback are consistent.

4.1 Description of the Algorithm

At every haptic update step, the haptic rendering algorithm must determine the next position of a Surface Contact Point ($\mathbf{SCP_H}$) (previously referred as proxy). The subscript **H** indicates that the surface corresponds to the extracted isosurface $\mathcal{H}$. Force feedback **f** is computed proportional to the separation between the **HIP** and the $\mathbf{SCP_H}$:

$$\mathbf{f} = k \cdot (\mathbf{SCP_H} - \mathbf{HIP}) \tag{11}$$

Our algorithm fits the family of constraint-based haptic rendering because the position of the $\mathbf{SCP_H}$ satisfies surface constraints at all times. These constraints maintain the $\mathbf{SCP_H}$ in contact with a target isosurface. In this implementation, even if it is visually represented as a small sphere, the $\mathbf{SCP_H}$ is effectively a point (with no radius).

When an isosurface corresponding to a specific isovalue exists in a tetrahedron containing the $\mathbf{SCP_H}$, the isosurface defines a constraint plane which restrains the movement of the $\mathbf{SCP_H}$. Thanks to the locality of volume haptic rendering, it is not necessary to extract the full isosurface. Only a local representation is dynamically extracted. The constraint plane's normal is given by the negative gradient Eq. (10) of the scalar field inside the tetrahedron. The next $\mathbf{SCP_H}$ position is computed using an iterative algorithm. At each iteration, the $\mathbf{SCP_H}$ moves from tetrahedron to tetrahedron until reaching its final position. This guarantees the $\mathbf{SCP_H}$ to never fall through the isosurface. This algorithm determines the $\mathbf{SCP_H}$ position in $O(n)$ where n is the number of tetrahedra crossed by the $\mathbf{SCP_H}$ while updated.

Each iteration combines three steps:

1) The first step looks for the target isosurface. It starts by creating a list referencing all the tetrahedra which contain the current $\mathbf{SCP_H}$. They constitute the active tetrahedra list. Then, the algorithm verifies, among all those tetrahedra, if there exists an isosurface corresponding to the currently specified isovalue. Each local piece of the isosurface defines a constraint.
2) The second step determines the **goal** position, i.e., the closest point to the Haptic Interface Point (**HIP**) which satisfies all the constraints. The algorithm takes into account two modes: the *Free Mode* and the *Constrained Mode*. The former is the mode set at the beginning of the simulation. It corresponds to the situation in which the $\mathbf{SCP_H}$ does not touch the isosurface. The latter is activated when the $\mathbf{SCP_H}$ is in contact with an isosurface and has therefore its motion constrained:
   - In *Free Mode*, the algorithm determines the intersection between the segment that joins the $\mathbf{SCP_H}$ and the **HIP** and the previously computed constraints (in this case, the constraints refer to the portion of the isosurface which is inside a tetrahedron). If the intersection point was outside of the current tetrahedron or if there was no isosurface in any of the active tetrahedra, the **goal** position is set to the **HIP** position. Otherwise, the $\mathbf{SCP_H}$ is updated to the intersection point and the algorithm switches to the Constrained Mode. The third step is skipped, no **goal** is computed.
   - In *Constrained Mode*, the algorithm sets the **goal** to a position as close as possible to the **HIP**, but respecting all the constraints previously determined. In this stage, the algorithm proposed by Ruspini et al. [39] can be used to solve the multiple constraint problem.
3) The third step first executes the CCD query to update the $\mathbf{SCP_H}$, see Section 4.2. Then, it detects if the $\mathbf{SCP_H}$ has been displaced into a zone with a lower density value. This would mean that the user is moving away from the selected isosurface. In this situation, the algorithm switches back to the Free Mode. Finally, the algorithm checks if the last updated **goal** corresponds to the last updated $\mathbf{SCP_H}$. In this case, the last $\mathbf{SCP_H}$ position is the output and the algorithm ends.

A pseudo-code of the algorithm is outlined in Fig. 8. Fig. 7 is a step by step 2D illustration of the algorithm, i.e., how the $\mathbf{SCP_H}$ is updated until it reaches its final position corresponding to the last **goal** position. Again, $\mathcal{V}$, shown for an isovalue $\tau = 0.55$, is represented by the blue curve. The linear piecewise orange curve represents $\mathcal{H}$. Each step corresponds to a $\mathbf{SCP_H}$ position. The algorithm starts in *Free Mode* and stays in the same mode in the first step ($\mathbf{SCP_H}$ 1). From step two to four ($\mathbf{SCP_H}$ 4) the algorithm is is in the *Constrained Mode*. The final position on $\mathcal{H}$ is "final $\mathbf{SCP_H}$".

4.2 CCD Query with Isosurfaces on Tetrahedral Meshes

The original constraint-based haptic rendering algorithm by Ruspini [39] performed CCD based on ray-intersections

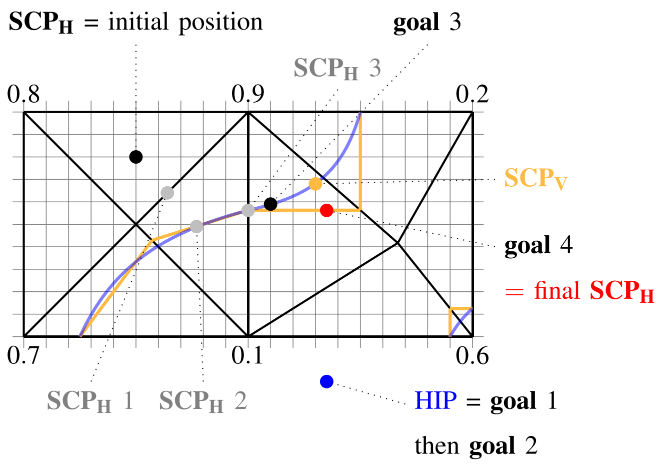


Fig. 7. Step by step illustration of the haptic rendering algorithm.

against an explicit triangle mesh. In our case, due to the use of a dynamically created isosurface, we have designed an algorithm that walks along the tetrahedral mesh. Tetrahedron boundaries are the only places where the isosurface might change its normal.

As summarized in Section 3, our isosurface definition based on a tetrahedral mesh allows us to compute the isosurface locally in an efficient manner, and we can guarantee that the isosurface is globally closed and continuous. The robustness of our haptic rendering algorithm is achieved by moving the $\mathbf{SCP_H}$ in an iterative manner, one cell and one tetrahedron at a time. In this way, the proxy never crosses the isosurface. CCD first determines if the segment defined by the $\mathbf{SCP_H}$ and the **goal** intersects the tetrahedral mesh. If it does, the $\mathbf{SCP_H}$ is placed at the intersection point. Otherwise, the $\mathbf{SCP_H}$ is placed at the **goal** position.

The CCD step ensures robustness compared to algorithms whose definition of the isosurface is based only on the local gradient. Indeed, those algorithms update the $\mathbf{SCP_H}$ position taking into account the constraints defined by the gradient at the $\mathbf{SCP_H}$ position, but do not ensure that the $\mathbf{SCP_H}$ does not cross the isosurface. These errors are avoided by our algorithm using CCD and the iterative update of the $\mathbf{SCP_H}$.

```
Input: {$\mathbf{SCP_H}$, HIP, $isovalue$, $freeMode$}
Output: {$\mathbf{SCP_H}$, $freeMode$}
// Initialization
$lookingForSCP_H$ = true
while $lookingForSCP_H$ is true do
    $executeStep3$ = true
    // Step 1
    $tetrahedraList$ = ComputeActiveTetrahedraList($\mathbf{SCP_H}$)
    $constraintsList$ = FindConstraints($tetrahedraList$, $isovalue$)
    // Step 2
    if $freeMode$ is true then
        $intersectionPoint$ = FindIntersection($\overline{\mathbf{SCP_H}\ \mathbf{HIP}}$, $constraintsList$)
        if $intersectionPoint$ is $\emptyset$ then
            goal = HIP
        else
            $\mathbf{SCP_H}$ = $intersectionPoint$
            $executeStep3$ = false
            $freeMode$ = false
        end if
    else {// Constained Mode}
        goal = SolveConstraints($constraintsList$, HIP)
    end if
    // Step 3
    if $executeStep3$ is true then
        $\mathbf{SCP_H}$ = CCD($\mathbf{SCP_H}$, goal, $tetrahedraList$)
        $currentDensity$ = GetDensityAt($\mathbf{SCP_H}$)
        if $currentDensity < isovalue$ then
            $freeMode$ = true
        end if
        if $\mathbf{SCP_H}$ = goal then
            $lookingForSCP_H$ = false
        end if
    end if
end while
```

Fig. 8. Pseudo code of the haptic rendering algorithm.

4.3 Visual Surface Contact Point on $\mathcal{V}$

$\mathbf{SCP_H}$ can be computed fast, avoid fall-through issues and follow the topology of $\mathcal{V}$. However, displaying $\mathbf{SCP_H}$ may lead to some visual artifacts specially due to the small gap between $\mathcal{H}$ and $\mathcal{V}$, and $\mathbf{SCP_H}$ may appear either floating or penetrating the visual isosurface $\mathcal{V}$ even if the user perceives a contact force. In order to increase visuohaptic consistency, we compute a Visual Surface Contact Point ($\mathbf{SCP_V}$) on $\mathcal{V}$. With $\mathbf{SCP_V}$, the visual probe is displayed exactly in contact with the visual isosurface $\mathcal{V}$ if and only if the $\mathbf{SCP_H}$ detects contact with the haptic isosurface $\mathcal{H}$. Differences in the actual location and orientation of the two isosurfaces cannot be perceived in practice.

Let $\mathbf{n_E}$ be the normal to $\mathcal{H}$ at the position of $\mathbf{SCP_H}$. $\mathbf{n_E}$ is interpolated using the normals at the vertices of the local portion of $\mathcal{H}$ surrounding $\mathbf{SCP_H}$. $\mathbf{SCP_V}$ is updated at the intersection between the line defined by the normal to $\mathcal{H}$ at the position of $\mathbf{SCP_H}$ and $\mathcal{V}$. $\mathbf{SCP_V}$ is computed only when the algorithm is in Constrained Mode. This intersection problem is equivalent to finding the roots of a third order polynomial. Fig. 1 illustrates the position of both $\mathbf{SCP_H}$ and $\mathbf{SCP_V}$ corresponding to the current **HIP** position. This figure shows a 2D representation equivalent to a zoom in the interior of one grid cell of the volume data. This feature is particularly useful when the point of view is quite close to the isosurface. $\mathbf{SCP_V}$ is also illustrated in orange in Fig. 7.

It appears tempting to use $\mathbf{SCP_V}$ for the computation of force feedback, as opposed to $\mathbf{SCP_H}$. However, this choice would produce wrong forces in cases where $\mathbf{SCP_V}$ and $\mathbf{SCP_H}$ lie on separate sides of the **HIP**.

5 Results

We evaluated three aspects of our haptic rendering algorithm. A first test focuses on the accuracy of TPBCC concerning the extraction of the haptic isosurface. A second test demonstrates the robustness of our algorithm compared to methods using the local gradient to constrain the Surface Contact Point. Finally, the last test shows that our algorithm meets the haptic framerate requirements to ensure a stable force feedback.

TABLE 1
Estimation of the Difference between $\mathcal{V}$ and $\mathcal{H}$ Using BCC, TPTD, and TPBCC

Method	BCC	TPTD	TPBCC
R_D^{max}	18.66%	15.18%	12.40%
R_D^{μ}	7.08%	7.95%	5.81%
R_D^{σ}	4.83%	3.26%	3.13%

5.1 Surface Reconstruction Accuracy

The notion of volumetric divergence is defined between two closed surfaces. It is a simple and natural metric to evaluate the closeness between two surfaces. As explained in [40], the volumetric divergence between $\mathcal{V}$ and $\mathcal{H}$ is the volume of region R_D defined as:

$$R_D = \left(\mathcal{V}^{in} \cap \mathcal{H}^{out}\right) \cup \left(\mathcal{V}^{out} \cap \mathcal{H}^{in}\right), \tag{12}$$

where the superscripts in and out refer to the inside and outside of the annotated isosurface respectively. We estimate R_D by voxelizing the input dataset at a resolution of $1000 \times 1000 \times 1000$ and interpolating the density field with the various reconstruction methods.

We evaluate R_D in each one of the thirty cases shown in Fig. 11 for the three extraction methods: BCC, TPTD and TPBCC. The parameters employed in each evaluation as well as the resulting value of R_D are given in Table 4. For each method, the maximum (max), mean (μ) and standard deviation (σ) of R_D over the thirty measurements are given in Table 1.

In each case, R_D is computed as a percentage value. We can see that our proposed TPBCC method gives the smallest R_D value on average: 5.81 percent. In other words, TPBCC is the method that globally best approximates $\mathcal{V}$. Since the data do not meet the assumption of normality, the non-parametric Wilcoxon signed-rank test is applied to assess if the mean ranks significantly differ. There is a significant difference between TPBCC and TPTD ($W(30) = 1$, $Z = -4.03$, $\rho < 0.01$, $r = 0.74$), demonstrating that our modification significantly improves TPTD. The difference between BCC and TPBCC is not significant. However, BCC does not guarantee to preserve the topology in all cases. Those results show that TPBCC is the method that both preserves the topology *and* significantly reduces the deviation w.r.t. the trilinear isosurface. A closer look at the values in Table 4 shows that the tendency is to have lower R_D values with BCC for the simpler cases but higher R_D values in all the cases 12 and 13. On the contrary, TPTD obtains relatively low R_D values for the cases 12 and 13 but much higher values in cases 1 and 2 for instance. TPBCC takes advantage of the strengh of the other two methods where they best operate. Usually, simple cases (1 and 2 for example) are a lot more frequent than complex cases (12 and 13 for example). This natural favorable distribution ensures a smooth haptic feedback during high point-of-view interaction schemes. Moreover, more complex cases usually require more attention from the user, i.e., a closer point-of-view and smaller displacements of Surface Contact Point. So once again, the interaction will remain fluid.

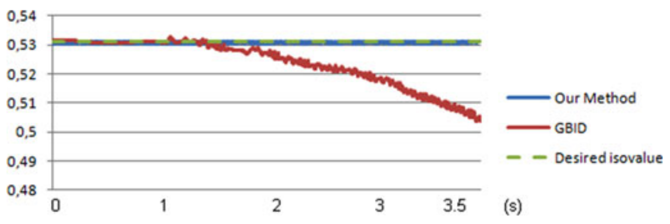


Fig. 9. Isovalue drift comparison between our method and GBID.

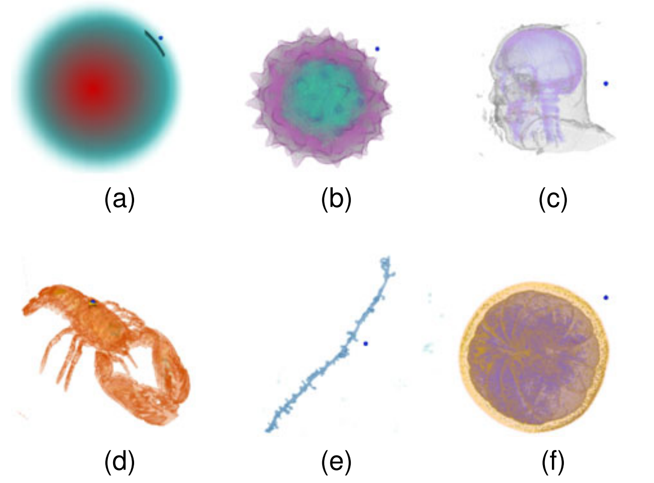


Fig. 10. Illustration of the datasets.

5.2 Robustness Test

This evaluation consists of initializing the target isovalue (0.53) as the one corresponding to the outer surface of the cylinder and positioning the $\mathbf{SCP_H}$ on the surface. The cylinder, Fig. 10a, has a radial distribution of its density, increasing toward its center. The $\mathbf{SCP_H}$ is repeatedly moved around the cylinder without leaving its surface. At each iteration, the density value at the position of the $\mathbf{SCP_H}$ is recorded. The results are compared with those obtained with the common algorithm using GBID. Fig. 9 shows how the original isovalue is robustly followed by our method, while with GBID it slowly decreases at each iteration. This is due to an accumulation of small unintentional fall-through into the dataset. An improved method to GBID is presented in [7], but it also defines the isosurface solely based on the local gradient, and therefore, as shown in their results, reduces drift but cannot prevent it.

5.3 Performance Evaluation

The evaluation of TPBCC is conducted on the datasets shown in Fig. 10. Their characteristics are given in Table 2. The "Cylinder" (Fig. 10a) is a synthetic dataset we generated, whereas the other ones are real datasets obtained using diverse technologies such as confocal microscopy, MRI or CT.

The evaluations are conducted on a PC with an i5 CPU running at 3.00 Ghz and with 8 GB of RAM. The graphics card is an NVIDIA GTX 670. The whole program is written in C++. $\mathcal{V}$ is displayed using ray casting and deferred shading techniques implemented on openGL 4.2. We use a Phantom Omni haptic device for force feedback.

The performance of the algorithm using TPBCC is evaluated by measuring the haptic loop running time. The results are summarized in Table 3. This table presents the mean

TABLE 2
Datasets Characteristics

Reference	Name and Source	Dimensions	Isovalue (τ)
(a)	*Cylinder*	$100 \times 100 \times 30$	0.12
(b)	*Daisy Particle**	$192 \times 180 \times 168$	0.02
(c)	*Visible Male**	$128 \times 256 \times 256$	0.12
(d)	*Lobster**	$301 \times 324 \times 56$	0.16
(e)	*Neuron***	$1024 \times 1024 \times 39$	0.12
(f)	*Orange**	$256 \times 256 \times 64$	0.22

**http://lgdv.cs.fau.de/External/vollib/*
***Instituto Cajal - Consejo Superior de Investigaciones Científicas (CSIC)*

TABLE 3
Mean Running Time of the Haptic Rendering Algorithm

Volume:	Cylinder	Daisy	Vis. male	Lobster	Neuron	Orange
LR - t^μ	0.27	0.35	0.31	0.21	0.32	0.30
LR - v^μ	19.29	7.87	11.62	8.29	8.14	8.73
LR - CM	63.11%	75.03%	85.46%	76.22%	66.74%	82.50%
HR - t^μ	0.44	0.33	0.36	0.33	0.33	0.41
HR - v^μ	13.92	5.03	7.54	5.59	5.79	5.40
HR - CM	84.41%	67.86%	79.92%	74.20%	69.72%	73.55%

time spent on the execution of the haptic loop in milliseconds (t^μ). The table also shows the mean velocity in centimeters per seconds of the **HIP** (v^μ) as well as the percentage of time spent in *Constrained Mode* in percent (CM) . These two last parameters demonstrate that the experiments were conducted within a "normal" scenario of exploration and interaction of the datasets. The user begins by translating and rotating the dataset in order to find a zone of interest. This exploration part is executed directly with the haptic device. The algorithm is mainly in Free Mode during this time. Once the zone of interest is located, the user adjusts τ value and starts to touch $\mathcal{H}$. During this exploration part, the algorithm is mainly in Constrained Mode.

The mean time depends on the grid resolution: the higher the resolution, the more tetrahedra the $\mathbf{SCP_H}$ has to go through before reaching its final position. The mean time

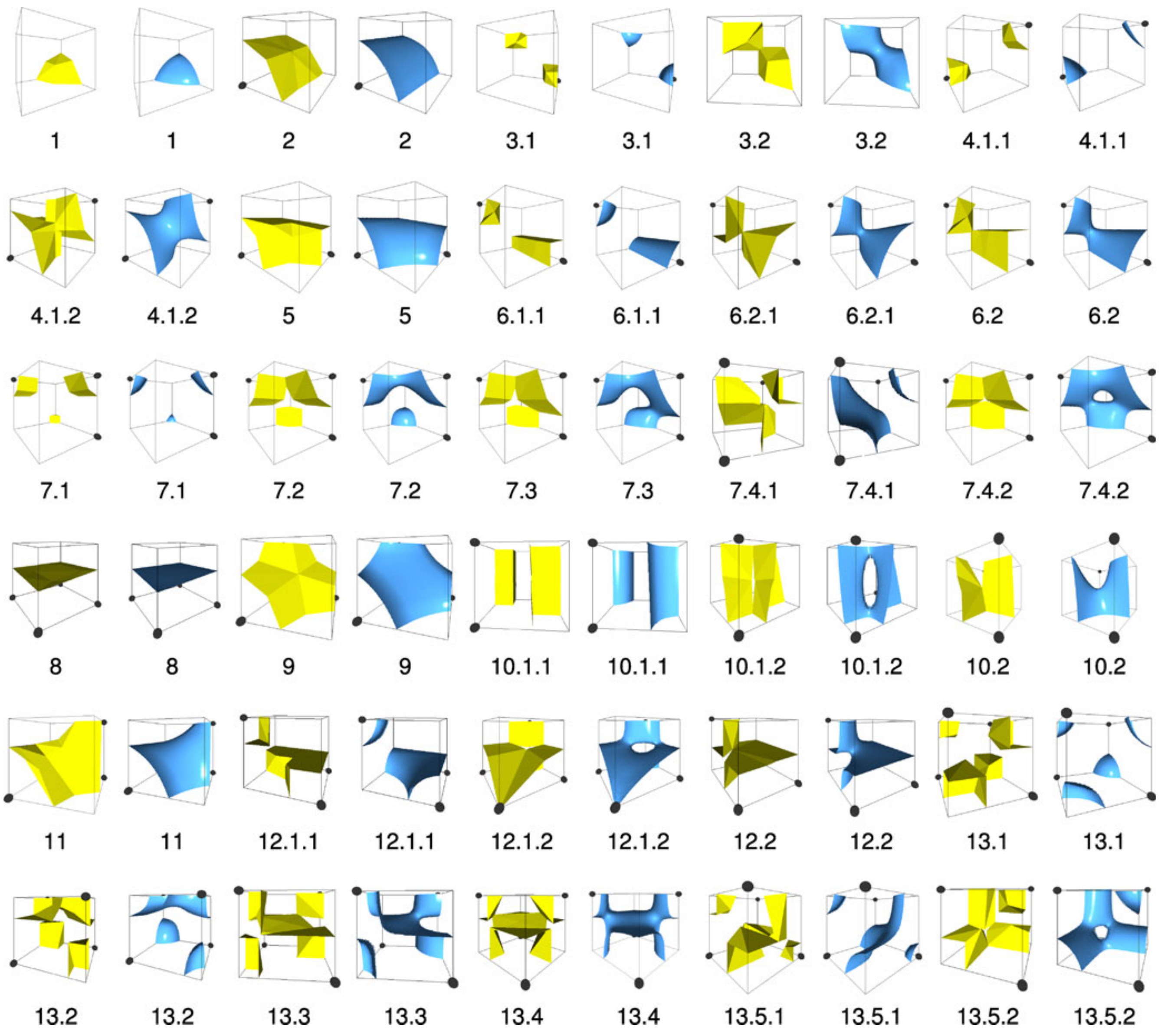


Fig. 11. All the decomposition cases. The visual isosurface $\mathcal{V}$ is shown in blue and the haptic isosurface $\mathcal{H}$ in yellow.

TABLE 4
Isovalue, Density Values, and R_D Results per Cases

Case	τ	d_0 d_1 ... d_7	R_D^{BCC}	R_D^{TPTD}	R_D^{TPBCC}	Case	τ	d_0 d_1 ... d_7	R_D^{BCC}	R_D^{TPTD}	R_D^{TPBCC}
1	0.41	0.87 0.14 0.12 0.24 0.15 0.10 0.08 0.18	1.45	15.18	1.45	8	0.44	0.81 0.85 0.79 0.88 0.04 0.13 0.13 0.07	0.20	1.14	0.64
2	0.37	0.87 0.91 0.12 0.24 0.15 0.10 0.08 0.18	1.58	11.93	1.58	9	0.51	0.95 0.06 0.91 0.97 0.17 0.04 0.08 0.97	2.18	4.47	2.18
3.1	0.63	0.87 0.04 0.12 0.24 0.15 0.90 0.08 0.18	2.07	2.89	2.46	10.1.1	0.40	0.95 0.16 0.94 0.08 0.92 0.17 0.85 0.09	7.44	8.11	7.50
3.2	0.46	0.87 0.04 0.12 0.24 0.15 0.90 0.08 0.18	4.50	9.14	6.22	10.1.2	0.40	0.52 0.15 1.00 0.06 0.81 0.00 0.74 0.00	6.28	4.84	2.05
4.1.1	0.57	0.87 0.04 0.12 0.24 0.15 0.10 0.88 0.18	2.87	5.83	3.24	10.2	0.46	0.90 0.04 0.62 0.11 0.72 0.05 0.86 0.36	6.24	8.33	5.86
4.1.2	0.29	0.87 0.04 0.12 0.24 0.15 0.10 0.88 0.18	10.69	12.73	10.60	11	0.40	0.90 0.04 0.82 0.91 0.12 0.05 0.61 0.06	4.10	7.08	5.90
5	0.45	0.04 0.78 0.89 0.91 0.15 0.10 0.13 0.18	2.14	8.36	9.56	12.1.1	0.56	0.09 0.82 0.82 0.83 0.84 0.11 0.07 0.11	6.24	7.93	7.34
6.1.1	0.59	0.82 0.78 0.12 0.11 0.15 0.10 0.93 0.18	4.39	4.53	4.49	12.1.2	0.46	0.10 0.84 0.67 0.83 0.81 0.11 0.07 0.11	7.20	4.94	3.05
6.1.2	0.41	0.90 0.58 0.12 0.24 0.15 0.12 0.88 0.18	4.44	10.24	3.29	12.2	0.48	0.09 0.72 0.82 0.83 0.97 0.11 0.07 0.11	7.08	7.13	4.07
6.2	0.46	0.82 0.78 0.12 0.11 0.15 0.10 0.93 0.18	5.07	7.80	6.91	13.1	0.34	0.95 0.05 0.81 0.02 0.11 0.91 0.06 0.88	10.20	10.31	10.31
7.1	0.66	0.08 0.78 0.12 0.11 0.95 0.10 0.93 0.18	3.40	3.57	3.36	13.2	0.38	0.95 0.24 0.81 0.11 0.11 0.56 0.06 0.67	12.62	8.44	8.44
7.2	0.51	0.08 0.78 0.12 0.11 0.95 0.10 0.93 0.18	8.17	8.38	7.02	13.3	0.44	0.95 0.34 0.81 0.11 0.11 0.56 0.06 0.67	17.24	6.54	6.54
7.3	0.51	0.29 0.78 0.12 0.11 0.95 0.10 0.93 0.18	8.34	8.12	6.63	13.4	0.43	0.95 0.24 0.81 0.11 0.11 0.56 0.06 0.67	18.66	6.80	6.80
7.4.1	0.38	0.06 0.85 0.03 0.06 0.94 0.03 0.93 0.07	9.39	14.19	10.05	13.5.1	0.51	0.05 0.81 0.42 0.82 0.71 0.02 0.83 0.04	16.24	12.40	12.40
7.4.2	0.44	0.06 0.85 0.03 0.06 0.94 0.03 0.93 0.07	8.57	7.69	4.89	13.5.2	0.58	0.05 0.81 0.42 0.82 0.71 0.02 0.83 0.04	13.50	9.54	9.54

also depends on the amplitude of the user motion. The measurements shown in Table 3 were taken at two different resolutions: a low resolution (LR) where a 1 mm displacement of the haptic device corresponds to half a grid cell, and a high resolution (HR) where a 1 mm displacement of the haptic device corresponds to three grid cells.

The results demonstrate that the haptic loop meets the 1 kHz requirements for stable feedback in all the tested cases.

6 Conclusions and Future Work

This paper presents a robust algorithm for haptic rendering of 3D datasets based on dynamic extraction of isosurfaces. The algorithm exploits tetrahedral mesh properties to define a closed surface and surface constraints that are constant within each tetrahedron. A new TPBCC isosurface extraction algorithm guarantees that the haptic isosurface matches the topology of the visual isosurface. Using a proxy combined with a constraint-based haptic rendering algorithm, users perceive a smooth and stable representation of the data. The algorithm goes one step further in improving the consistency between the visual and haptic cues using a secondary Surface Contact Point updated in real time on the visual isosurface according to the usual Surface Contact Point position used to compute the force feedback. Several tests showed that those improvements had a minor impact on the running time of the algorithm, ensuring the smoothness of the force feedback.

There are several directions for future work. We will update the penetrability features presented in [11] to work with our new implementation. With the improvements concerning the topology of the isosurfaces, this algorithm is now better suited for haptics tools targeting volumetric datasets such as [41]. Finally, this work can easily be extended into a 6-DoF haptic rendering algorithm.

Acknowledgments

This work was supported by grants from the following entities: the Spanish Ministry of Economy and Competitiveness (grant from Cajal Blue Brain Project, Spanish partner of the Blue Brain Project initiative from EPFL) and the European Commission (HBP: Human Brain Project—FET Flagship HBP604102 and RaSimAS: Regional Anaesthesia Simulator and Assistant—FP7-ICT 610425).

References

[1] A. Amruta, A. Gole, and Y. Karunakar, “A systematic algorithm for 3-D reconstruction of MRI based brain tumors using morphological operators and bicubic interpolation,” in *Proc. IEEE 2nd Int. Conf. Comput. Technol. Develop.*, 2010, pp. 305–309.

[2] D. Heeraman, J. Hopmans, and V. Clausnitzer, “Three dimensional imaging of plant roots in situ with X-ray computed tomography,” *Plant Soil*, vol. 189, no. 2, pp. 167–179, 1997.

[3] M. Ikits, J. Brederson, C. D. Hansen, and C. R. Johnson, “A constraint-based technique for haptic volume exploration,” in *Proc. IEEE Vis. Conf.*, 2003, pp. 263–269.

[4] E. Vidholm and I. Nystrom, “A haptic interaction technique for volume images based on gradient diffusion,” in *Proc. IEEE 1st Joint Eurohaptics Conf. Symp. Haptic Interfaces Virtual Environ. Teleoperator Syst.*, 2005, pp. 336–341.

[5] D. A. Lawrence, L. Pao, C. Lee, and R. Y. Novoselov, “Synergistic visual/haptic rendering modes for scientific visualization,” *IEEE Comput. Graph. Appl.*, vol. 24, no. 6, pp. 22–30, Nov. 2004.

[6] T. R. Coles, D. Meglan, and N. John, “The role of haptics in medical training simulators: A survey of the state of the art,” *IEEE Trans. Haptics*, vol. 4, no. 1, pp. 51–66, Jan./Feb. 2011.

[7] K. Lundin and G. Baravdish, “Higher precision in volume haptics through subdivision of proxy movements,” in *Proc. 6th Int. Conf. Haptics: Perception, Devices Scenarios*, 2008, vol. 5024, pp. 694–699.

[8] K. Salisbury, D. Brock, T. Massie, N. Swarup, and C. B. Zilles, “Haptic rendering: Programming touch interaction with virtual objects,” in *Proc. ACM Symp. Interactive 3D Graphics*, 1995, pp. 123–130.

[9] A. Lopes and K. Brodlie, “Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing,” *IEEE Trans. Vis. Comput. Graph.*, vol. 9, no. 1, pp. 16–29, Jan.–Mar. 2003.

[10] S. Stegmaier, M. Strengert, T. Klein, and T. Ertl, “A simple and flexible volume rendering framework for graphics-hardware-based raycasting,” in *Proc. IEEE 4th Int. Workshop Volume Graph.*, 2005, pp. 187–241.

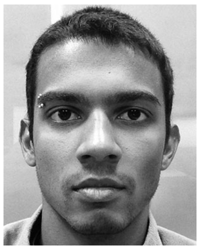
[11] L. Corenthy, J. San Martin, M. A. Otaduy, and M. Garcia, “Volume haptic rendering with dynamically extracted isosurface,” in *Proc. IEEE Haptics Symp.*, 2012, pp. 133–139.

[12] H. Iwata and H. Noma, “Volume haptization,” in *Proc. IEEE Symp. Res. Frontiers Virtual Reality*, 1993, pp. 16–23.

[13] D. A. Lawrence, C. D. Lee, R. Y. Novoselov, and L. Y. Pao, “Shock and vortex visualization using a combined visual/haptic interface,” in *Proc. IEEE Conf. Vis.*, 2000, pp. 131–137.

[14] R. S. Avila and L. M. Sobierajski, “A haptic interaction method for volume visualization,” in *Proc. 7th IEEE Vis. Conf.*, 1996, vol. VI, pp. 197–204.

[15] O. Körner, M. Schill, C. Wagner, H. Bender, and R. Männer, "Haptic volume rendering with an intermediate local representation," in *Proc. 1st Int. Workshop Haptic Devices Med. Appl.*, 1999, pp. 79–84.
[16] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM Siggraph Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.
[17] Y. Zhou, W. Chen, and Z. Tang, "An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes," *Elsevier Comput. Graph.*, vol. 19, no. 3, pp. 355–364, 1995.
[18] R. Strand and P. Stelldinger, "Topology preserving marching cubes-like algorithms on the face-centered cubic grid," in *Proc. IEEE 14th Int. Conf. Image Anal. Process.*, 2007, pp. 781–788.
[19] J. H. Ryu, H. S. Byun, and K. H. Lee, "High-quality isosurface generation using an oversampling method," *Springer. Int. J. Adv. Manufacturing Technol.*, vol. 28, no. 11–12, pp. 1161–1168, 2006.
[20] G. M. Nielson and B. Hamann, "The asymptotic decider: resolving the ambiguity in marching cubes," in *Proc. 2nd IEEE Conf. Vis.*, 1991, pp. 83–91.
[21] B. K. Natarajan, "On generating topologically consistent isosurfaces from uniform samples," *Vis. Comput.*, vol. 11, no. 1, pp. 52–62, 1994.
[22] E. V. Chernyaev, "Marching cubes 33: Construction of topologically correct isosurfaces," CERN, Meyrin, Switzerland, Tech. Rep. CN/95-17, 1995.
[23] A. Guéziec and R. Hummel, "Exploiting triangulated surface extraction using tetrahedral decomposition," *IEEE Trans. Vis. Comput. Graph.*, vol. 1, no. 2, pp. 328–342, Dec. 1995.
[24] G. M. Treece, R. W. Prager, and A. H. Gee, "Regularised marching tetrahedra: Improved iso-surface extraction," *Comput. Graph.*, vol. 23, pp. 583–598, 1998.
[25] J. Georgii and R. Westermann, "A generic and scalable pipeline for GPU tetrahedral grid rendering," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 1345–1352, Sep./Oct. 2006.
[26] B.-S. Sohn, "Topology preserving tetrahedral decomposition applied to trilinear interval volume tetrahedrization," *KSII Trans. Internet Inf. Syst.*, vol. 3, no. 6, pp. 350–357, 2009.
[27] M. Eriksson, M. Dixon, and J. Wikander, "A haptic VR milling surgery simulator-using high-resolution ct-data," *IOS Press Stud. Health Technol. Inf.*, vol. 119, p. 138, 2005.
[28] S. Chan, N. H. Blevins, and K. Salisbury, "Deformable haptic rendering for volumetric medical image data," in *Proc. IEEE World Haptics Conf.*, 2013, pp. 73–78.
[29] M. Audette, I. Hertel, O. Burgert, and G. Strauss, "A tissue relevance and meshing method for computing patient-specific anatomical models in endoscopic sinus surgery simulation," in *Advances in Computational Vision and Medical Image Processing*. New York, NY, USA: Springer, 2009, pp. 159–172.
[30] K. Lundin, A. Ynnerman, and B. Gudmundsson, "Proxy-based haptic feedback from volumetric density data," in *Proc. IEEE Eurohaptic Conf.*, 2002, pp. 104–109.
[31] K. Lundin, M. Cooper, and A. Ynnerman, "The orthogonal constraints problem with the constraint approach to proxy-based volume haptics and a solution," in *Proc. SIGRAD Conf.*, 2005, vol. 1, pp. 45–49.
[32] B. Ménélas, M. Ammi, and P. Bourdot, "A flexible method for haptic rendering of isosurface from volumetric data," in *Proc. 6th Int. Conf. Haptics: Perception, Devices Scenarios*, 2008, vol. 5024, pp. 687–693.
[33] T. Knott and T. Kuhlen, "Geometrically limited constraints for physics-based haptic rendering," in *Proc. 9th Int. Conf. Haptics: Neurosci., Devices, Model., Appl.*, 2014, vol. 8619, pp. 343–351.
[34] R. Vlasov, K.-I. Friese, and F.-E. Wolter, "Haptic rendering of volume data with collision detection guarantee using path finding," in *Transactions on Computational Science XVIII*, vol. 7848. New York, NY, USA: Springer, 2013, pp. 212–231.
[35] K. Salisbury and C. Tarr, "Haptic rendering of surfaces defined by implicit functions," *ASME Dynamic Syst. Control Division*, vol. 61, pp. 61–67, 1997.
[36] L. Kim, G. S. Sukhatme, and M. Desbrun, "An implicit-based haptic rendering technique," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2002, pp. 2943–2948.
[37] S. Chan and E. Purisima, "A new tetrahedral tesselation scheme for isosurface generation," *Elsevier Comput. Graph.*, vol. 22, no. 1, pp. 83–90, 1998.
[38] J. Patera and V. Skala, "Centered cubic lattice method comparison," in *Proc. Slovak Univ. Technol. Conf. Algoritmy*, 2005, pp. 309–318.
[39] D. C. Ruspini, K. Kolarov, and O. Khatib, "The haptic display of complex graphical environments," in *Proc. 24th ACM Annu. Conf. Comput. Graphics Interactive Techn.*, 1997, pp. 345–352.
[40] C. Wang and T. S. Newman, "New metric for evaluating the accuracy of marching isosurfacing algorithms," in *Proc. ACM Southeast Regional Conf.*, 2014, p. 22.
[41] L. Corenthy, M. Garcia, S. Bayona, A. Santuy, J. S. Martin, R. Benavides-piccione, J. Defelipe, and L. Pastor, "Haptically assisted connection procedure for the reconstruction of dendritic spines," *IEEE Trans. Haptics*, vol. 7, no. 4, pp. 486–498, Oct.–Dec. 2014.



Loïc Corenthy received the engineering degree from Telecom-Physique Strasbourg, France, and the master's degree in sciences and information technologies from the Université de Strasbourg, France. He also received the master's degree in simulation and virtual reality from the ENSAM Art et Metier PariTech, France. He is currently working toward the PhD degree at the Unversidad Politécnica de Madrid, Madrid, Spain. His research interests include haptic rendering, object oriented computer programming, and visualization.

Miguel A. Otaduy received the BS degree in electrical engineering from Mondragn University in 2000, and the MS and PhD degrees in computer science from the University of North Carolina at Chapel Hill, in 2003 and 2004, respectively. He is an associate professor in the Department of Computer Sciences Modeling and Virtual Reality Group, Universidad Rey Juan Carlos (URJC Madrid). From 2005 to 2008, he was a research associate at ETH Zurich. His main research interests include physics-based simulation and computational haptics. He has published more than 80 papers in these fields, and he has served on the editorial board for several journals and conferences, most notably *IEEE Transactions on Haptics*, *IEEE Transactions on Visualization and Computer Graphics*, 2013 and 2015 IEEE World Haptics Conference, 2013 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, and 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.

Luis Pastor received the BSEE degree from the Universidad Politecnica de Madrid in 1981, the MSEE degree from Drexel University in 1983, and the PhD degree from the Universidad Politcnica de Madrid in 1985. He is currently a full professor at the University Rey Juan Carlos, Madrid, Spain. His research interests include image processing and synthesis, virtual reality, 3D modelling, and parallel computing.

Marcos Garcia received the computer science degree at UPM, Madrid, Spain, in 2002 and the PhD degree from the same university in 2007. In his PhD, he worked on simulation of elastic objects in interactive applications. From November 2007 to February 2008, was a research fellow at the GV2 group in TCD (Dublin, Ireland) working with Prof. Carol O Sullivan. He has been an associate professor of computer science at URJC since 2011. His main research interests are in haptic rendering, scientific rendering, and physically based simulation.