Haptically Assisted Connection Procedure for the Reconstruction of Dendritic Spines

Loic Corenthy, Marcos Garcia, Sofia Bayona, Andrea Santuy, Jose San Martin, Ruth Benavides-Piccione, Javier DeFelipe, and Luis Pastor

Abstract—Dendritic spines are thin protrusions that cover the dendritic surface of numerous neurons in the brain and whose function seems to play a key role in neural circuits. The correct segmentation of those structures is difficult due to their small size and the resulting spines can appear incomplete. This paper presents a four-step procedure for the complete reconstruction of dendritic spines. The haptically driven procedure is intended to work as an image processing stage before the automatic segmentation step giving the final representation of the dendritic spines. The procedure is designed to allow both the navigation and the volume image editing to be carried out using a haptic device. A use case employing our procedure together with a commercial software package for the segmentation stage is illustrated. Finally, the haptic editing is evaluated in two experiments; the first experiment concerns the benefits of the force feedback and the second checks the suitability of the use of a haptic device as input. In both cases, the results shows that the procedure improves the editing accuracy.

Index Terms—Haptically assisted segmentation, Volume haptics, Incomplete dendritic spines

1 Introduction

As stated in the "Grand Challenges" of the 21*st* century of the National Academy of Engineering [1], understanding the functioning of the human brain is one of the great challenges of contemporary science. One of these challenges is to find out how the neurons are structured and mutually connected [2]. In the cerebral cortex, the most common type of neuron is the pyramidal cell. The dendritic surface of these cells is covered by thin protrusions named dendritic spines (for simplicity, spines) which represent the targets of most excitatory synapses in the cerebral cortex [3]. Furthermore, spines are considered critical in learning, memory and cognition [4]. Thus, there is considerable interest in obtaining accurate measures of the spine structure.

Technically, spines can be accurately reconstructed using electron microscopy, but it is time consuming and difficult, which makes it challenging to obtain large numbers of measurements [5]. Instead, light microscopic techniques, although limited by the lower level of resolution, remain the method of choice to obtain large-scale spatial information, see Benavides-Piccione et al. [6]. However, it is relatively common for the necks of the dendritic spines of pyramidal cells (Figs. 1a and 1b) to be so thin that they are close to the optical resolution limit of the imaging techniques.

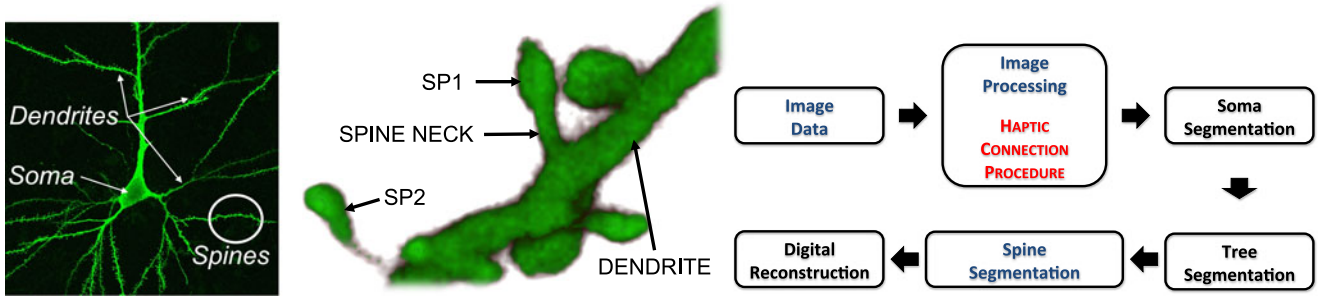
The volumetric datasets obtained with this methodology usually need to undergo a segmentation process in order to retrieve useful semantic information. Segmentation can be defined as the process of establishing relationships between regions of the 3D image and their meaning [7]. In general, segmentation processes can be classified as anything ranging from fully automatic to completely manual processes according to the degree of intervention by the user. Fully manual techniques rely on user expertise for result correctness, but they are time consuming and thus are not suitable for processing very large amounts of data. By contrast, automatic methods are capable of segmenting huge datasets. However, they are usually not robust enough to deal with any arbitrary dataset. Semi-automatic segmentation methods fall between these two extremes, and aim to combine the desirable features of both options. Generally speaking, these methods make the most of the users knowledge to interactively control an automatic process [8].

In this paper, we present a semi-automated haptically assisted procedure to restore spines with low fluorescence intensity values in order to accomplish an accurate and complete reconstruction. This procedure takes place according to a general segmentation workflow such as the one presented in [9] and shown in Fig. 1c.

Using a Phantom Omni haptic device as input, our method provides intuitive 3D navigation and facilitates a dynamic interaction with the 3D image. This was first supported with evidence by Avila and Sobierajski [10] and more recently highlighted by Pannëls and Roberts [11] "*the integration of haptic interaction into a scientific visualization process can lead to a better understanding of complex, three-dimensional data, and can result in more natural, intuitive methods for modifying these data*". In different stages of our proposed method, the haptic feedback facilitates steering and targeting tasks. The

- *L. Corenthy and A. Santuy are with the Politécnica de Madrid University, Madrid, Spain. E-mail: loic.corenthy@gmail.com, andrea. santuy@ctb.upm.es.*
- *M. Garcia, S. Bayona, J.S. Martin, and L. Pastor are with the Rey Juan Carlos University, Madrid, Spain. E-mail: {marcos.garcia, sofia.bayona, jose.sanmartin, luis.pastor}@urjc.es.*
- *R. Benavides-Piccione and J. DeFelipe are with the Instituto Cajal (CSIC), Center for Biomedical Technology, and Politécnica de Madrid University. E-mail: {rbp, defelipe}@cajal.csic.es.*

Manuscript received 31 Dec. 2013; revised 20 Aug. 2014; accepted 20 Aug. 2014. Date of publication 3 Sept. 2014; date of current version 15 Dec. 2014. Recommended for acceptance by M. Flanders.
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.2014.2354041



(a) Illustration of a neuron (b) Close-up of a dendrite and its spines (c) General workflow diagram

Fig. 1. (a) Soma and dendritic tree from a pyramidal neuron. (b) High-magnification image to show two examples of dendritic spines, one is accurately reconstructed with available automatic methods (SP1), whereas the neck of the other dendritic spine (SP2) is hardly visible due to the low fluorescence intensity. (c) General workflow and key processing steps for neuronal segmentation. The different steps are divided in sub-tasks which are not represented here. The procedure introduced here can be seen as a complementary image processing step which will facilitate the spine segmentation stage.

benefits of haptic feedback while executing these types of tasks were also supported by Dennerlein et al. [12].

The main contributions in the present paper are:

- A procedure specially designed for full and accurate reconstruction of spines that present too low fluorescence intensity values to be automatically reconstructed by available 3D methods (see Section 3).
- An interactive haptic control of the procedure that facilitates the user experience by integrating natural 3D navigation and providing different types of haptic feedback in most of the reconstruction steps (see Section 4).
- The illustration of the integration of this procedure within the workflow of a commercial software package for the segmentation of neuronal data: Imaris [13] (Section 5).
- The evaluation of both the benefits and the suitability of using a haptic interface for the haptic editing step of our procedure (Section 6).

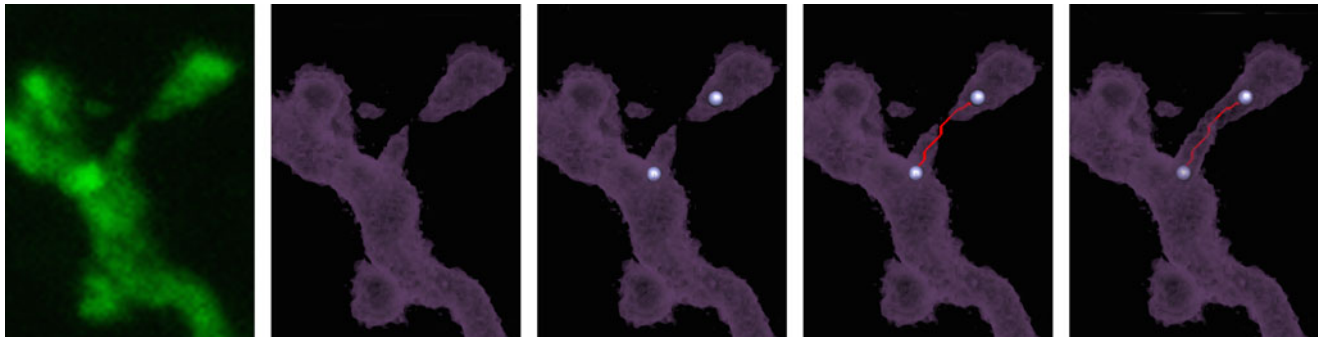
2 Related Work

2.1 Automatic Segmentation Methods

For more than 40 years, there have been numerous attempts to obtain digital reconstructions of neuronal morphology. Back in 1965, Glaser and Van Der Loos were able to obtain the length measurement of dendritic branches [14]. Since then, the increasing computational power has facilitated the development of a variety of automatic segmentation algorithms. These algorithms have become more and more efficient and are generally designed to require little human intervention. Taking advantage of data acquisition technique improvements, research concerning neuronal data segmentation progressively focused on smaller structures such as spines. For example, Imaris (Bitplane AG, Zurich, Switzerland) or NeuronStudio [13], are commonly used programs designed to allow reconstruction of neuronal structures from confocal images, providing tools for manual, semi-manual, and automatic tracing of the dendritic arbor as well as manual and automatic detection of dendritic spines. Both softwares deal with the detached spine problem. Imaris, provides a module (FilamentTracer) which performs a segmentation of volume data based on connectivity and thresholding. However, it does not accurately capture the morphology of the spine, although it can approximate its volume reasonably well. In [13] the workflow for spine processing is automatic and deals with detection and shape classification. Part of the process called Spine Stem Reattachment, focuses on disconnected spines. Their approach, involves linking detached spines to their corresponding spine stems, i.e. the end of the spine neck on the dendrite side, located close to them. The algorithm is based on a search phase in a bell-shaped region around a segment linking the detached spine to the dendrite. The radius of the search region is based on an empirical parameter. Similarly in [15], Janoos et al. presented a method to reconstruct dendrites using a surface representation of the neuron. This representation is based on a 2D projection and it facilitates skeleton extraction. Their image processing pipeline includes a specific step to connect spines to dendrites. The method is based on the observation that the spine heads tend to be oriented toward the dendrite, while the dendrite also has a protrusion pointing toward the spine. Moreover, spines and dendrites are in close proximity to each other. The algorithm used by Janoos et al. takes advantage of these factors for the spine reconstruction. Part of the algorithm depends on an empirical parameter that takes into account the distance between the spine head and the dendrite. Several other methods have been specifically designed to deal with the complex problem of spine detection (i.e. [16], [17], [18], [19]). However, very often, automatic segmentation methods require an empirical parameter in their implementations. This can lead to a wrong result in some of the more complicated cases. In such cases, users intervention using manual or semi-automatic methods is necessary.

2.2 Volume Haptics

Haptic interaction with volumetric datasets is a fairly active research topic. Throughout this paper, a volumetric dataset will correspond to a 3D grid. Each vertex of the grid, defined by a discrete position in space, is associated to a normalized scalar density value d (unitless values between 0 and 1). The word *voxel* refers to a vertex of the volumetric grid. Numerous algorithms are proxy-based implementations. The concept of proxy for haptic rendering was originally introduced by Zilles and Salisbury [20] and Ruspini et al. [21] for surface



(a) Confocal image (b) Isovalue selection (c) Targets positioning (d) Path creation (e) Spine reconstruction

Fig. 2. (a) Original confocal image. (b) Dendritic spine which fails to be completely reconstructed with standard available 3D tools. (c) Connection points are placed (blue spheres). (d) The red line represents the path linking the two targets. (e) The complete dendritic spine reconstruction is accomplished.

haptic rendering. This section briefly introduces a selection of volume haptic rendering algorithms.

Palmerius et al. also introduced proxy based algorithms for haptic rendering of volumetric datasets ([22] and [23]). The haptic feedback is calculated using a set of four haptic primitives (point, line, plane and directed force) that can be combined in order to represent different local properties of the volumetric data.

Ikits et al. [24] used directional constraints in their haptic rendering algorithm in order to provide intuitive exploration modes of volumetric datasets. The constraints were obtained by "*augmenting the proxy with a local reference frame and controlling its motion according to a set of rules and transfer functions*".

Kim et al. [25] presented a combined geometric and volumetric haptic rendering method. A proxy constrained on the surface of the mesh lets the user perceive the real geometry of the surface while the volumetric data allowed a smoother haptic feedback avoiding force discontinuities.

The procedure described in the present paper is based on the work of Corenthy et al. [26] in which the proxy is updated using a local isosurface dynamically extracted from the volumetric dataset.

2.3 Haptic Segmentation Methods

Several publications recognized the advantages of a haptic feedback in a semi-automatic segmentation process.

In their work about liver segmentation, Vidholm and Agmund [27] explored the connection between haptic sculpting and interactive volume image segmentation. They presented an editing tool based on morphological image processing operators. The tool is controlled by a Phantom Desktop haptic device. It ensures haptic feedback thanks to proxy-based haptic rendering. It includes two interaction modes for the volume data: erosion and dilation. As with our procedure, the volume interaction modes allow the volume data to be edited directly. Their editing algorithm is based on binary operators, therefore it cannot provide enough resolution to adequately solve the incomplete spine segmentation problem.

Sourin et al. [7] presented a pipeline for brain data segmentation with a haptic device. It consists of running an automatic segmentation process, correcting possible errors, and reiterating the automatic process to obtain the final result. The authors insist that guidance is a key factor for semi-automatic processes. In this case, the guiding procedure steers the user toward erroneous zones. Once the users acknowledge a zone with imperfections, they can interact with the generated mesh. The editing function is operated using a Phantom Omni and allows three types of operations on the mesh: push inside, pull outwards and rub. Finally, the edited mesh is used to guide the volume data correction. This last step requires the transformation of the mesh back into voxels. This technique cannot be applied to our target domain since it was designed to correct only small segmentation errors. In contrast to our technique, during the editing phase, the method relays only on the operator knowledge and does not take into account the original data to guide the surface correction.

Harders et al. [28] present a method for the segmentation of linear structures in medical datasets. The method is based on an interactive centreline extraction approach. A Phantom Premium guides the user along a path close to the centreline of the tubular structure. To calculate the force feedback and also the centerline, the volume data is first converted to a binary set taking into account a threshold to select the structures of interest. A distance map is then built from the binary structure and used to calculate a discrete gradient field. To obtain a continuous haptic feedback, trilinear interpolation is used on the gradient field. Users have to place seed points along the indicated path. Whenever they judge it necessary, they can leave the calculated path to place the seed points. The centreline is then used as the initialization point for a deformable surface. A final step based on six orthographic projections all around the tubular structure allows the 3D surface model to be turned back into voxels. In contrast to this approach, our technique also recovers data outside the given threshold.

3 Procedure

The method presented here intends to simplify and improve the quality of spine reconstruction by locally increasing the fluorescence intensity values in those dendritic spines that cannot be fully reconstructed with standard 3D methods. The procedure is divided in four main steps described in the following paragraphs and illustrated in Fig. 2. Once a spine that needs to be reconstructed is spotted in the dataset, the user starts by placing two *targets* that will serve as beginning and ending points of a polyline (or path). This polyline serves as the medial axis used to control the haptic editing process in the spine reconstruction.

3.1 Navigation and Isovalue Selection

In our system, the user can navigate using a Phantom Omni of the SensAble family of haptic devices, which offers six degrees-of-freedom (*DOF*) input (translation and rotation) and three *DOF* output (translation only). Concerning the input, the translation lets users navigate through the dataset using the stylus. In order to span the whole dataset, the navigation model is similar to [29]. Users can rotate the dataset around a vertical axis using the yaw rotation of the stylus. It is also possible to zoom in and out using the buttons on the stylus. The output renders the force feedback through point-based haptics. The haptic rendering is described in detail in Section 4.1.

Users need to select the density value which best charcterizes the spine (see Fig. 2b). Coherent structures in the image are typically characterized by clusters of points with similar density values. The selected value corresponds to the isovalue d_{iso} which will be used throughout the haptic connection procedure. The user can select the isovalue using the keyboard.[1]

3.2 Targets Positioning

During the second step of the procedure, the haptic device is used to place two connection points (referred to as *targets*) in the scene (see Fig. 2c). To do so, the user can switch between the two different modes: *navigation with touch* and *fully constrained navigation*. The first mode implements a volume haptic rendering algorithm that lets the user touch the different structures corresponding to d_{iso} in the scene. The second mode constrains the navigation to the surface of the dendrite or the spine. It provides the user with a sensation of being "attached" to a neuronal element of the scene, for instance a spine. The algorithms concerning these two modes are detailed in Section 4.1. The haptic cues enhance the perception of the dendritic shaft and spine shapes. Indeed, automatic algorithms often look for a small residue of the dendritic spine neck on the surface of the dendritic spine head or the dendrite to start their connection process. Here, users can also feel these small residues if they exist, and they can take advantage of this information to better place the targets. Moreover, the *fully constrained navigation* mode ensures users will indeed place the *targets* on the surface[2] of a neuronal element whereas if they do not have their movements constrained, they might unintentionally place the *target* off the surface of the neuronal element.

3.3 Path Creation

Once the target points are positioned in the scene, a path joining the two targets and equivalent to the medial axis of the future spine neck is computed. The user creates a path using a keyboard shortcut. The path consists of a set of nodes (discrete position corresponding to a voxel in the dataset) connected by linear segments, i.e. a 3D polyline. This polyline is continuous but not C^1. To ensure hard real-time haptic update rates, we favored a faster algorithm using linear segments instead of splines (or similar higher order interpolation method) to connect the nodes. The path is calculated using the pathfinding algorithm $A^\star$ presented by Hart et al. [30]. In Section 4.2, we detail how the $A^\star$ is adapted to our problem.

3.4 Complete Spine Reconstruction with Haptic Editing

The final step of our method involves editing the density values of all the voxels located around the path. The idea is to increase the density of the selected voxels in order to completely reconstruct the spine. The aim is not to replace the data with arbitrary density values but to truly recover the erroneously low density values. These values are multiplied by a *density correction function*. User movements in the editing area as well as user control over the editing process are both managed using the haptic device in a seamless manner: their movements are haptically constrained along the previously created path. Using keyboard shortcuts, users can activate two different modes to edit the dataset:

- The *global* mode thickens the entire path by increasing the density value of all the voxels around it at once.
- The *local* mode adjusts the thickness of the neck only around a specific position on the path (in this case, the proxy position). The *local* mode was designed to refine the first editing performed using the *global* mode if needed.

In both modes, users adjust the *density correction function* by simply pulling the haptic device in an orthogonal direction to the path. The force feedback attracts the haptic device toward the path during the editing phase. At the same time, when the user pulls the haptic device away from the path, the intensity of the force feedback indicates the editing intensity. The path haptic rendering, the *density correction function* and the editing modes are detailed in Sections 4.3, 4.4 and 4.5, respectively.

4 Algorithms

This section goes through the procedure in the following manner: paragraph A focuses on the second step, paragraph B on the third step and paragraphs C, D and E on the fourth step.

4.1 Volume Haptic Rendering

4.1.1 Navigation with Touch

This mode corresponds to the haptic rendering algorithm presented by Corenthy et al. [26]. Each isovalue is associated with an isosurface. Our algorithm constrains the proxy position to be outside the volume defined by the selected isosurface. As is usual in proxy-based algorithms, the force feedback $\mathbf{f}$ is proportional to the distance between a proxy $\mathbf{p_c}$ and a probe $\mathbf{p_d}$ (the unconstrained haptic position in the scene):

$$\mathbf{f} = k \cdot (\mathbf{p_c} - \mathbf{p_d}), \tag{1}$$

where k is a stiffness constant.

1. Any operation requiring the use of the keyboard can also be executed using the graphical user interface (**GUI**). However, it is impractical for the user to use both a haptic device and a mouse at the same time. For this reason, most of the interactions are available through keyboard shortcuts.

2. The targets are not actually placed on the surface because they have a discrete position in the 3D grid. The haptic navigation ensures that the distance between the target and the neuronal element is smaller than the side of a voxel.

Our algorithm iteratively searches the new proxy position starting from the current one, minimizing the distance between $\mathbf{p_c}$ and $\mathbf{p_d}$. In each step of the algorithm, the isosurface is locally computed and the distance between $\mathbf{p_c}$ and $\mathbf{p_d}$ is minimized inside the area containing the current proxy. To locally calculate the isosurface, a marching tetrahedra approach is used. The subdivision is based on an adapted version of the body-centered cubic lattice [26]. The isosurface is linearly constructed by interpolating the density values of the tetrahedron vertices. Inside each tetrahedron, the interpolated isosurfaces are planar and parallel to each other. This property facilitates the collision detection stage. Furthermore, a continuous modification of d_{iso} produces a continuous change on the isosurface. This is a key feature for the exploration of volumetric datasets because it allows the user to freely select the isovalue which best represents the neuronal elements in the volumetric scene.

To avoid fall-through problems and improve its robustness, this algorithm employs a continuous collision detection (*CCD*) with the tetrahedral mesh when updating the proxy position. Here, the notion of mesh is similar to the grid representing the dataset, with each voxel corresponding to a vertex of the mesh at the same position. The *CCD* iteratively updates the proxy tetrahedron per tetrahedron between its current position and the probe position. In each tetrahedron, the algorithm checks if the line defined by the proxy and the probe l_{pp} intersects a local portion of the isosurface. If not, the proxy is updated at the intersection point between the current tetrahedron and l_{pp}. When an intersection occurs, the proxy movements are restrained on the isosurface while minimizing its distance to the probe. Thus, the algorithm not only takes the isosurface into account at the proxy position, but at any position between the proxy and the probe.

4.1.2 Fully Constrained Navigation

With the above algorithm, it is possible to freely navigate in the volume data and to touch the current isosurface. However, when the probe is moved away from the isosurface, the proxy is updated at the same position as the probe, and the force feedback vanishes. A slight modification enables a new behavior—the *fully constrained navigation*. The idea is to constrain the proxy to the isosurface even when the user moves the probe outside the element being "touched". Thus, users cannot inadvertently lose track of the structure they are interacting with. This behavior is implemented by inverting the constraints defined by the isosurface each time the probe crosses the isosurface. Usually, density values in the volume data increase toward the center of the elements represented in the scene. This means that voxels inside an element have higher density values than the isovalue. The orientation of the constraints will then be updated according to the position of the probe, inside or outside. This relation is summarized in (2). d_{p_d} is the interpolated density value at the probe position.

$$\begin{cases} d_{p_d} > d_{iso} & \Leftrightarrow \quad \text{probe inside}, \\ d_{p_d} < d_{iso} & \Leftrightarrow \quad \text{probe outside}. \end{cases} \tag{2}$$

4.2 A* Algorithm

The $A^\star$ algorithm was originally presented by Hart et al. [30]. Its general purpose is to find a minimum cost path through a graph using information relative to the problem domain. The resulting path links a start node $\mathbf{n_s}$ to an end node $\mathbf{n_e}$. In this case, a node corresponds to a voxel. As defined [30], let $f(\mathbf{n_i})$ be the cost of an optimal path constrained to go through the current node $\mathbf{n_i}$, from $\mathbf{n_s}$ to $\mathbf{n_e}$. $f(\mathbf{n_i})$ is defined as the sum of two parts:

$$f(\mathbf{n_i}) = g(\mathbf{n_i}) + h(\mathbf{n_i}), \tag{3}$$

where $g(\mathbf{n_i})$ is the cost of an optimal path from $\mathbf{n_s}$ to $\mathbf{n_i}$ and $h(\mathbf{n_i})$ is the cost of an optimal path from $\mathbf{n_i}$ to $\mathbf{n_e}$. We have to provide estimate functions in order to calculate $g(\mathbf{n_i})$ and $h(\mathbf{n_i})$. In our particular case, those functions are defined as:

$$g(\mathbf{n_i}) = D(\mathbf{n_i}, \mathbf{n_{i-1}}) + g(\mathbf{n_{i-1}}) + \alpha \cdot |1 - d_{n_i}|, \tag{4}$$

$$h(\mathbf{n_i}) = D(\mathbf{n_i}, \mathbf{n_e}), \tag{5}$$

where $\mathbf{n_{i-1}}$ is the node preceding $\mathbf{n_i}$ in the path linking $\mathbf{n_s}$ to $\mathbf{n_i}$ and d_{n_i} is the density of the node $\mathbf{n_i}$. α is a weight coefficient whose value depends on the scale of the volume data (a value of 80 worked in the majority of the tested cases). $D(x, y)$ mesures the Euclidian distance between x and y.

We customized the original implementation of the cost function g adding the factor $\alpha \cdot |1 - d_{n_i}|$ that penalizes nodes with the lowest density values (see (4)). Indeed, the path must go through the nodes with the highest density values in order to be the centreline of the dendritic neck being reconstructed.

In our version of the $A^\star$ algorithm, all nodes in the computed path have to meet the following criteria:

$$criteria(n_{i-1}, n_i) : \begin{cases} d_{n_i} > T_{C1,} \\ |d_{n_{i-1}} - d_{n_i}| < T_{C2.} \end{cases} \tag{6}$$

The first criterion avoids nodes with very low density values which can be considered as noise in the data. All nodes in the path must have a density value higher than a predefined threshold value T_{C1}. The second criterion penalizes directions with high gradients, i.e., the density difference between two consecutive nodes must be lower than a threshold value T_{C2}. The latter criterion prevents the path from going toward the borders of the reconstructed structures, characterized by higher gradients. We used the default values $T_{C1} = 0.02$ and $T_{C2} = 0.5$ in the majority of the tested datasets. Users can refine the parameters α, T_{C1} and T_{C2} to adjust the path as needed.

In the implementation of the algorithm, Fig. 3, *closedset* is the set of nodes already evaluated, *openset* is the set of tentative nodes to be evaluated that initially contains $\mathbf{n_s}$. The functions used in the algorithm are the following:

- *reconstructPath* rebuilds the entire path. It adds the node passed as parameter to the path. Then it recursively adds the parent node of the last node added to the path until it reaches n_s.
- *neighborNodes* simply returns the neighbor nodes of the node passed as parameter. It is possible to select the connectivity type: six-connectivity or 26-connectivity.
- *criteria* returns true if the criteria in (6) are verified.

```
function A*(start,goal)

// Initialization
closedset = ∅, openset = {n_s},
g[n_s] = 0, h[n_s] = D(n_s,n_e), f[n_s] = g[n_s] + h[n_s]
// n_c is the current node
while openset ≠ ∅ do
   n_c = node in openset with the lowest f
   if n_c = n_e then
      return recontructPath(n_c)
   end if
   remove n_c from openset
   add n_c to closedset
   for all neighbor in neighborNodes(n_c) do
      if neighbor in closedset or not criteria(n_c, neighbor) then
         continue to next neighbor
      end if
      g̃ = g[n_c] + D(n_c, neighbor) + α·|1 − d_{n_c}|
      if neighbor not in openset or g̃ < g[neighbor] then
         parent[neighbor] = n_c
         g[neighbor] = g̃
         f[neighbor] = g[neighbor] + h[neighbor]
         if neighbor not in openset then
            add neighbor to openset
         end if
      end if
   end for
end while
```

Fig. 3. Pseudo code of the $A^\star$ algorithm.

4.3 Path Haptic Rendering

The haptic navigation along the path is implemented using an algorithm similar to the one introduced by Raya et al. [31], except that, in our case, the path does not have any branches. It is an iterative algorithm which constrains the movements of a proxy along a path while following the movements of the haptic device. The algorithm is initialized by placing the proxy on the path position that is closest to the probe. The first time, all the segments of the path are checked. Once the proxy is on the path, the iterative part of the algorithm updates the proxy using a gradient descent approach. First, the proxy is set at its previous position on the path $\mathbf{p_c^{prev}}$. While the distance between the proxy and the probe decreases, the proxy is updated in the direction of the curvilinear abscissa of the path toward the probe. The proxy reaches its final position when it reaches the first local minimum of the function $D(\mathbf{p_c}, \mathbf{p_d})$.

In practice, using linear segments greatly simplifies the computation. The algorithm is initialized defining the *active segment* as the one containing $\mathbf{p_c^{prev}}$. In a first step, the algorithm computes the point $\mathbf{P_A}$ on the *active segment* that is the closest to the probe. Two cases have to be taken into account:

i) If $\mathbf{P_A}$ does not fall on one of the extremities of the *active segment*, the new proxy position is set to $\mathbf{P_A}$ and the algorithm ends.

ii) If $\mathbf{P_A}$ is positioned at one of the two vertices defining the *active segment*, the latter is marked as visited and the segment adjacent to the current *active segment* becomes the new *active segment*. Two other subcases have to be taken into account:

a) If the new *active segment* has already been visited, the proxy is placed at $\mathbf{P_A}$ and the algorithm stops.

b) Otherwise, the algorithm continues to iterate from the first step, i.e. computing a new $\mathbf{P_A}$ as the point on the current active segment that is closest to the probe.

Concerning the border cases, when $\mathbf{P_A}$ falls on a vertex segment with no neighbor segment, the proxy is updated to $\mathbf{P_A}$ and the algorithm ends.

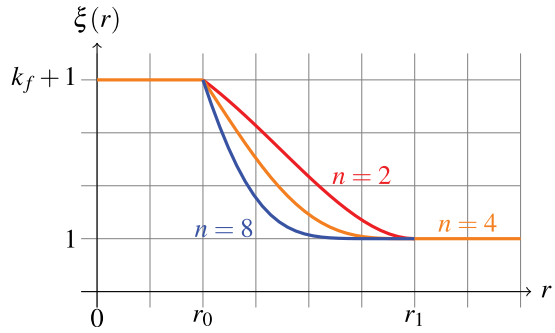


Fig. 4. Plot of the function $\xi(r) = 1 + k_f \cdot \phi(r)$ with $k_f = 3$, $r_0 = 2$, $r_1 = 6$ and $n \in \{2, 4, 8\}$.

4.4 Density Correction Function

In order to increase the density values of the selected voxels, they are multiplied by a *density correction function* $\xi(r)$. The variable r represents the distance between a processed voxel $\mathbf{v}$ and the path, $r = D(\mathbf{v}, path)$. ξ is designed in such a way that if the distance between a voxel and the path is smaller than a radius parameter r_0, the density is multiplied by a constant factor, preserving the local gradient direction. ξ decreases from $\xi(r_0)$ at r_0 to zero at r_1. This allows a smooth transition at the borders of the reconstructed area. The density values of the voxels situated at a distance higher than r_1 from the path are not modified.

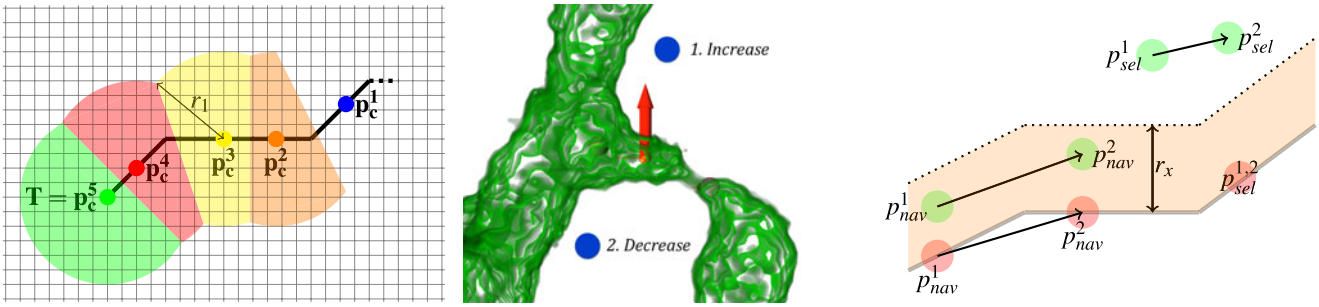
The function ξ is a radial piecewise continuous function defined in (7) and illustrated in Fig. 4.

$$\xi : \begin{vmatrix} \mathbb{R}^+ & \rightarrow & \mathbb{R}^+, \\ r & \mapsto & 1 + k_f \cdot \phi(r). \end{vmatrix} \tag{7}$$

During the editing, users haptically control the weight coefficient k_f to adjust the shape of the spine neck. The function ϕ is defined in (8). This function is the decreasing constituent of ξ and it depends on three discrete parameters: r_0 and r_1, as defined above, and n.

$$\phi : \begin{vmatrix} \mathbb{R}^+ & \rightarrow & \mathbb{R}^+ & \\ r & \mapsto & \begin{cases} 1 & \text{if } \mathrm{r} < \mathrm{r}_0, \\ \left(\frac{r_1^2 - r^2}{r_1^2 - r_0^2}\right)^n & \text{if } \mathrm{r}_0 \leq \mathrm{r} < \mathrm{r}_1, \\ 0 & \text{otherwise.} \end{cases} \end{vmatrix} \tag{8}$$

A small difference between r_0 and r_1 results in more abrupt transitions. The exponent n controls how fast the function ϕ decreases. In practice, r_1 and n are useful when editing the extremities of the path. Indeed, in these regions, the radial function ξ will tend to increase density values in voxels



(a) Local selection of the voxels (b) *Local* mode haptic control (c) Navigation on the path vs k_f selection.

Fig. 5. (a) 2D illustration of the selected voxels according to (9). The grid represents the voxels. In this sequence, the proxy is moved from $\mathbf{p_c^1}$ to $\mathbf{p_c^5}$. At each update of the proxy position, the region corresponding to the selected voxels around the proxy is colored. $\mathbf{T}$ represents a final node of the path. As the proxy cannot be updated any further after $\mathbf{T}$, the conditions in (9) correspond to a circular zone. (b) Illustration of the interaction with the *Local* editing mode. The arrow in red represents the vector $\mathbf{S_A^\perp}$ and the blue sphere represents the probe. In the first case, pulling the haptic device in the direction of the arrow increases the k_f value. In the second case, pulling in the opposite direction decreases the k_f value. (c) The plain black line is the path. The navigation region is the orange region between the path and the dotted line. When the probe (in green) moves from $\mathbf{p^1_{nav}}$ to $\mathbf{p^2_{nav}}$ within the navigation region, the proxy (in red) is updated from $\mathbf{p^1_{nav}}$ to $\mathbf{p^2_{nav}}$ on the polyline but k_f value is not updated. When the probe moves from $\mathbf{p^1_{sel}}$ to $\mathbf{p^2_{sel}}$ outside the navigation region, the proxy stays at $\mathbf{p^{1,2}_{sel}}$ and the k_f value is updated according to (10).

corresponding to the spine head or the dendrite. Compared to the voxels of the spine neck, these voxels typically have higher density values. This can result in an overly inflated part of the dendritic neck in those specific regions. Increasing the value of n avoids this effect if it is not desired. Default values $r_0 = 3$, $r_1 = 7$ and $n = 2$ were used in the majority of the tested datasets.

4.5 Haptic Editing Modes

The *global* mode can be activated whenever the proxy is constrained to move along the path. Users set the k_f value by radially pulling the probe away from the path. k_f is proportional to the distance between the constrained proxy and the probe, $k_f \propto D(\mathbf{p_c}, \mathbf{p_d})$. In this mode, the position of the proxy along the path does not influence the editing. The *global* editing mode overwrites the density values of all the voxels that are at a smaller distance than r_1 to the path. The force feedback indicates the intensity of the modification coefficient k_f to the user; the higher the intensity, the higher the feedback.

The *local* mode has been designed to enable users to refine their global editing. This mode only affects voxels that are close to the proxy. Let $D^{(t)}(\mathbf{v}, \mathbf{p_c})$ be the distance between a voxel $\mathbf{v}$ and the current proxy position $\mathbf{p_c}$ at an instance t. The modified voxels are the ones satisfying the following conditions:

$$\begin{aligned} D^{(t)}(\mathbf{v}, \mathbf{p_c}) &< r_1, \\ \beta \cdot D^{(t)}(\mathbf{v}, \mathbf{p_c}) &< D^{(i)}(\mathbf{v}, \mathbf{p_c}) \quad , \forall i \in [0, t-1], \end{aligned} \tag{9}$$

where β is a constant that is slightly lower than one and is used to avoid rounding errors. In this mode, the radius used in ξ is no longer the distance between a voxel $\mathbf{v}$ and the path, but rather the distance between $\mathbf{v}$ and the current proxy, $r = D^{(t)}(\mathbf{v}, \mathbf{p_c})$. These two conditions concerning $D^{(t)}(\mathbf{v}, \mathbf{p_c})$ define a selection region around the proxy, with the shape of this region varying between a sphere and a narrow band centered on the proxy and oriented in a radial direction to the path. These regions are illustrated in Fig. 5a.

In *local* mode, k_f does not simply correspond to the distance between the probe and the proxy. In order to refine the editing, the haptic device is used to increase or decrease the previously set k_f values, depending on the direction towards which the user is pulling. An arrow indicating the direction to increase the previously set k_f value is displayed on the screen. This arrow visually represents a vector $\mathbf{S_A^\perp}$ which is always orthogonal to the path and to the screen projection plane (see Fig. 5b). To decrease the k_f value, users have to pull in the opposite direction to the displayed arrow.

The haptic control of the editing intensity in the *local* mode should be as smooth as in the *global* mode. In the case of the *local* mode, a smooth interaction requires the user to manipulate the haptic device with enough dexterity to be able to dissociate the movements related to the selection of a local zone and the movements related to k_f refinement. Experience demonstrated that while novice users try to act upon k_f, they might inadvertently move the proxy on the path at the same time. The result is the user editing the wrong local region of the spine neck with a wrong k_f value. To avoid this undesired situation while trying to keep the interaction as smooth as possible, the proxy is updated on the path only when the probe is close to the path. When users move the probe out of a navigation region, the algorithm interprets their gesture as a coefficient selection and the proxy does not move (see Fig. 5c). The navigation region corresponds to a cylindrical region of radius r_x around the path. When the probe moves within this navigation region, the value of k_f is not updated. Let $D_{out} = D(\mathbf{p_c}, \mathbf{p_d}) - r_x$ be the distance from the region limit to the probe and $\mathbf{pp} = \mathbf{p_c} - \mathbf{p_d}$ be the vector from the probe to the proxy. The *local* mode calculates the k_f value at time t as:

$$k_f^{(t)} = \begin{cases} k_f^{(t-1)} + \gamma \cdot D_{out} \text{ if } \mathbf{pp} \cdot \mathbf{S_A^\perp} \geq 0 \text{ and } ||\mathbf{pp}|| > \mathrm{r_x}, \\ k_f^{(t-1)} - \gamma \cdot D_{out} \text{ if } \mathbf{pp} \cdot \mathbf{S_A^\perp} < 0 \text{ and } ||\mathbf{pp}|| > \mathrm{r_x}, \\ k_f^{(t-1)} \quad \text{otherwise}, \end{cases} \tag{10}$$

where γ is a positive scalar parameter used to emphasize the increase (or decrease) of $k_f^{(t)}$. Its value is increased (or decreased) by adding (subtracting) a value proportional to the distance between the current position of the probe and the limit of the navigation zone. The more the user pulls on the haptic device, the faster the $k_f^{(t)}$ value increases (or decreases).

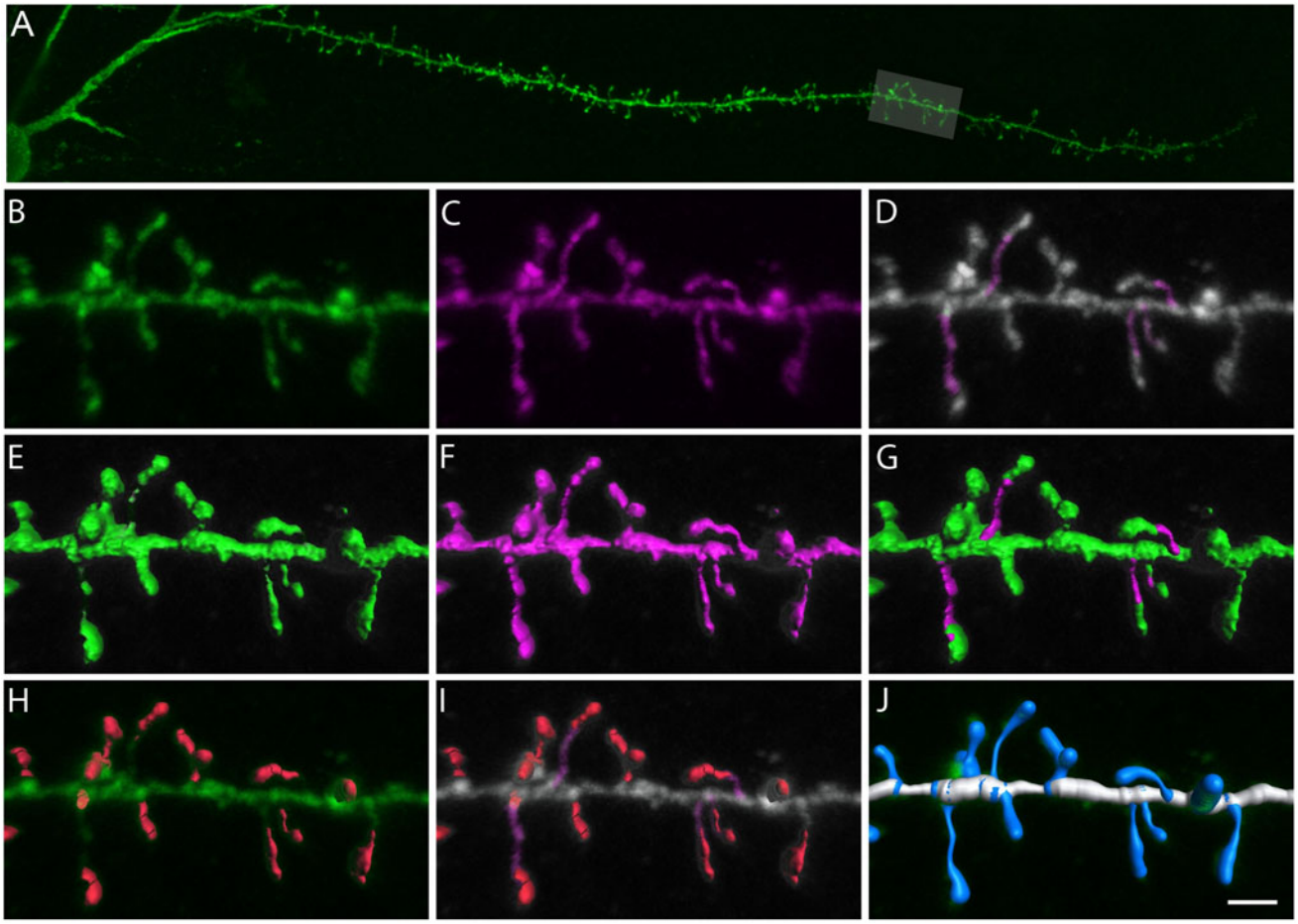


Fig. 6. (A) Confocal microscopy image of an intracellular injected layer III human pyramidal neuron dendrite, to illustrate the spines along the entire length of the dendrite from the soma to the distal tip. (B) High-magnification image of a dendritic segment highlighted in A. (C) Same dendritic segment as in A, after the haptic editing procedure was applied to restore spines with low fluorescence intensity values. (D) By combining B and C images, the relative location of the restored low fluorescence intensity values is shown. (E–G) Estimation of the spine volume values shown in (B-D). (H-I) Three-dimensional manual reconstructions of the morphology of each spine using Imaris software. (J) Estimation of spine volume values using filament tracer (Imaris). Scale bar (in J): 8.5 μm in A and 2 μm in (B–J).

5 Use of the Method in Spine Reconstruction

Intracellular injections in fixed brain tissue represent a powerful method to visualize the morphology of pyramidal neurons (Fig. 6). The labeled cells are reconstructed using high-resolution tile scan stacks of confocal microscopy images with currently available software tools (Fig. 6). Using this methodology we evaluated how practical was the four-step procedure for the complete reconstruction of dendritic spines. Thus, intracellularly injected pyramidal neuron dendritic spines were evaluated along the entire length of the dendrite from the soma to the distal tip.

We first reconstructed the spines from the original image (Figs. 6A and 6B) using a standard 3D reconstruction commercial tool (Imaris software, Bitplane). Using this tool, it was possible to fully reconstruct about 80 percent of the spines ($n = 210$), whereas the remaining spines ($n = 43$), were only partially reconstructed (Figs. 6E and 6H). Using the present procedure we were able to locally edit the low fluorescence intensity values in these 43 remaining spines (Figs. 6C and 6D). We used Imaris software again to evaluate these spines. As shown in Figs. 6F and 6G spines were then completely reconstructed. Fig. 6I illustrates how the current module Imaris software Filament tracer, which is intended to solve the problem of the detached spines, does not accurately capture the morphology of the spine, although it can approximate its volume quite well. Thus, we show the complementarity of this four-step procedure with other commercially available segmentation tools.

6 Haptic Editing Evaluation

To confirm the value added by the haptic editing in the complete spine reconstruction stage (Section 3.4), we designed two tests, T_A and T_B, comparing the use of the haptic device with that of another input device. The haptic device will be referred to as the modality M_{HD} and the other input device as modality M_A in T_A and modality M_B in T_B.

T_A is a pilot study that checks the benefits of the force feedback with M_A—a non-haptic 3D input device.[3]

T_B checks the suitability of the haptic device as 3D interface by comparing its use with a traditional input interface M_B which is a mouse and a slider on a GUI. Although the procedure was designed for a haptic device,

3. Implemented using a Phantom Omni from Sensable with the force feedback deactivated.

TABLE 1
Degree of Curvature and Node Count in Each Scene

Scene	A	B	C	D	E	F	G	H
Curvature	★	★	★★	★★	★★★	★	★★★	★★
Node count	9	12	14	16	14	26	20	24

all the movements involved in the complete spine reconstruction task are 1D. Such 1D movements are easy to perform using a slider. The aim here is to compare M_{HD} with one of the most common and natural types of interface (i.e. mice and GUIs) for users not familiar with the manipulation of a haptic device.

6.1 Common Aspects to Both Tests

6.1.1 Subjects

Fourteen participants, four females and 10 males, aged from 22 to 37, took part in each test. Participants were categorized based on their expertise concerning the use of haptic devices. There were three possible categories: novice, intermediate and expert. Novice participants had no knowledge whatsoever about haptic devices and had never manipulated one before the experiment; intermediate participants had manipulated a haptic device on scarce occasions (but at least once) before the experiment; and expert participants had used a haptic device extensively before the experiment.

6.1.2 Experimental Protocol

Each participant was first given a written explanation of the experiment and signed an informed consent form. The subjects' task was to recreate the connection between a spine and its corresponding dendrite using the *global* and *local* modes previously described. As the tests only concerned the haptic editing phase, the $targets$ and the paths were given to the participants.

Eight scenes (A to H) were available for the tests. The scenes are divided according to their level of difficulty. Scenes A to F are considered as "easy", and scenes G and H as "difficult". This classification is based on two distinguishing scene features: a categorical curvature parameter and the number of nodes per path in each scene. These values per scene are summed up in Table 1. The curvature categories are represented by a star (★), one star meaning a low level of curvature and three stars corresponding to a high level of curvature. Subjects had to execute the task twice for each scene, i.e. once per modality.

Since it is difficult to find enough participants with previous experience in segmenting neuroscience data, we designed our validation experiments for subjects with no previous experience in neuroscience. For each scene, participants were shown the result to be achieved, i.e. the complete spine already reconstructed. Neuroscientists have knowledge on how the shape of the connection should be. Giving the images of the reconstructed spines to the participants compensates their lack of knowledge. Consequently, a reference image, see Fig. 7, was shown to the subjects as a goal for them to achieve when performing the task. Before starting the real tests, subjects had a five-minute habituation session where they could manipulate the program on a training scene. The assignment order of the different input modalities and scenes was counterbalanced.

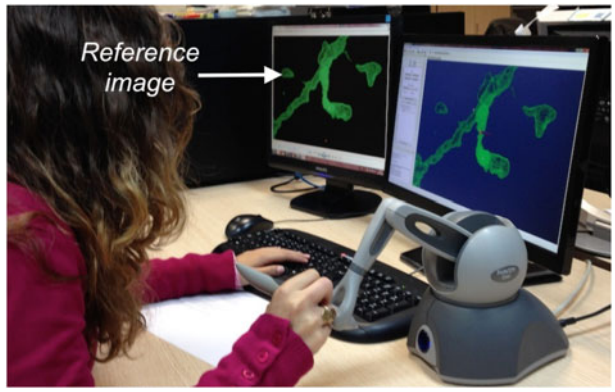


Fig. 7. Experimental setup for the two tests. The image on the screen on the left is the reference image. The spine being reconstructed by the user is displayed on the screen on the right.

6.1.3 Setup

The experimental setup is shown in Fig. 7. The procedure is executed on a PC with an i5 CPU running at 3.00 Ghz and with 8 GB of RAM. The graphic card is an NVIDIA GTX 670. The whole program is written in C++. The volume scene is rendered using the Coin3D and SimVoleon libraries. The GUI is created using Qt library.

6.1.4 Data Analysis

For each trial and each subject, completion time (in seconds) and density values of the edited scenes were recorded. For each test, the statistical analysis considers four dependent variables: the completion time and an $error$ value for each input modality. Those variables are characterized by their mean μ and their standard deviation σ. The $error$ value was calculated as shown in (11) and reflects the difference between the reference scene and the scene edited by the subjects. It takes into account the density values d in the set of voxels V defining the edited spine, as well as its cardinality $size(\mathrm{V})$.

$$error = \frac{\sqrt{\sum_{i \in \mathrm{V}} (d^i_{ref} - d^i_{subject})^2}}{size(\mathrm{V})}. \tag{11}$$

For each scene and for all variables (time and error), ND represents the normalized mean difference between M_{HD} and M_A or M_B as a percentage:

$$ND = 100 \frac{\mu^{M_{HD}} - \mu^{M_X}}{mean(\mu^{M_{HD}}, \mu^{M_X})}, \tag{12}$$

where X stands for A or B.

6.2 Specific Aspects and Results of T_A

6.2.1 Experimental Protocol

In this test, participants edited four scenes. The proxy was always updated on the path. There was no need to move the probe within the navigation zone. Indeed, it is

TABLE 2
Time and Error Measurements

	$\mu^{M_A}(\sigma)$	$\mu^{M_{HD}}(\sigma)$	$ND(\%)$	$sig.$
time	152.92 (105.46)	149.09 (92.010)	-2.54	0.974
error	0.1213 (0.4325)	0.1093 (0.0407)	-10.41	$^{\star}0.010$

Measures marked with a star reflect statistically significant results.

impossible for the subjects to maintain the probe in the navigation zone without haptic feedback (M_A).

6.2.2 Results

Since the dependent variables did not meet the assumption of normality, the mean difference between the modalities is assessed using the non-parametric Wilcoxon signed-rank test. The results are shown in Table 2.

Although the mean completion time is shorter when using the haptic device with the force feedback activated as input, the difference is not significant. However, there is significantly less *error* using the first modality ($W(56) = 28.35, Z = -2.59, p = 0.010, r = 0.24$). The results show greater accuracy when using the force feedback.

6.3 Specific Aspects and Results of T_B

6.3.1 Experimental Protocol

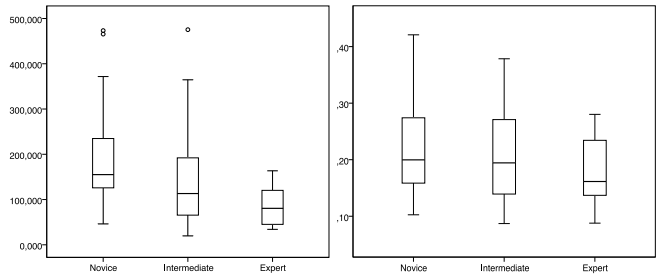
Participants had to execute the task on all the scenes.

In this test, the M_B GUI replaces the use of the haptic device in three respects: a first slider to move the proxy on the path, and second and third sliders to set the intensity in the *global* and the *local* mode, respectively. The intensity values selected with the sliders correspond to the distance between the probe and the proxy when using the haptic device. At the end of the experiment, subjects filled out a preference questionnaire.

6.3.2 Results

Since the data did not meet the assumption of normality, the non-parametric Wilcoxon signed-rank test was applied to assess if the mean ranks differed between the variables when using the haptic device and the mouse. The results of the performance evaluation for each scene are shown in Table 3.

For the scenes categorized as easy, the difference is significant in half of the cases concerning the error and in one case concerning the completion time. For scenes A, D and E, the Wilcoxon test showed subjects executed the task with significantly less error using the Phantom compared to the mouse: A ($W(14) = 14, Z = -2.42, p = 0.016, r = 0.46$), scene D ($W(14) = 9, Z = -2.73, p = 0.006, r = 0.52$) and scene E ($W(14) = 19, Z = -2.10, p = 0.035, r = 0.40$). For scene B, the same test indicates that using a Phantom to execute the task it is significantly faster than using a mouse ($W(14) = 19, Z = -2.10, p = 0.035, r = 0.40$). For the scenes categorized as difficult, there are no significant differences between the two input methods.



(a) Completion time (b) Error

Fig. 8. Completion time (in seconds) and normalized error results when performing the task using a haptic device. The participants are grouped according to their previous haptic expertise.

Figs. 8a and 8b show, for the haptic device, the time and error related to the haptic experience of the participants. The general tendency seems to indicate that more experienced users do better, both in terms of time and error. A Kruskal Wallis test revealed a significant effect of group on value ($\chi^2(2) = 10.53, p = 0.005$) only concerning the time. A post-hoc test using Mann-Whitney tests showed significant differences between the Novice and Intermediate categories ($p = 0.01, r = 0.25$) and between the Novice and Expert categories ($p = 0.003, r = 0.46$). There was no significant difference between Intermediate and Expert categories.

6.3.3 Discussion

From the scene classification, regarding the difficulty, we observed that significant results were obtained only for scenes classified as easy. A closer look at the mean differences for the four dependent variables between these two categories shows that, compared to easy scenes, the difficult scenes result in up to 1.45 times more error with the mouse, and take up to 1.96 times longer to be edited with the haptic

TABLE 3
Time and Error Measurements for the Two Input Methods

Scene	$\mu^{M_B}_{time}(\sigma)$	$\mu^{M_{HD}}_{time}(\sigma)$	$ND_{time}(\%)$	sig_{time}	$\mu^{M_B}_{error}(\sigma)$	$\mu^{M_{HD}}_{error}(\sigma)$	$ND_{error}(\%)$	sig_{error}
*A	93.33 (57.25)	73.69 (62.74)	-23.52	0.109	0.2762 (0.1193)	0.1948 (0.0703)	-34.59	$^{\star}0.016$
*B	115.60 (50.74)	78.89 (41.70)	-37.75	$^{\star}0.035$	0.2725 (0.0873)	0.2305 (0.0543)	-16.73	0.074
C	158.63 (99.61)	159.59 (95.40)	0.60	0.778	0.1753 (0.0438)	0.1562 (0.0190)	-11.57	0.064
*D	96.03 (56.35)	84.35 (57.58)	-12.96	0.433	0.1819 (0.0897)	0.1405 (0.0507)	-25.68	$^{\star}0.006$
*E	172.11 (79.60)	163.75 (101.32)	-4.98	0.778	0.2155 (0.0361)	0.1972 (0.0122)	-8.86	$^{\star}0.035$
F	129.69 (42.85)	158.09 (89.82)	19.73	0.300	0.1049 (0.0197)	0.1193 (0.0218)	12.78	0.300
G	190.06 (114.38)	225.33 (103.55)	16.98	0.158	0.2939 (0.0281)	0.2880 (0.0182)	-2.01	0.638
H	222.86 (63.31)	245.23 (87.53)	9.56	0.363	0.3013 (0.0307)	0.3027 (0.0373)	0.49	0.363

Scenes marked with a star (*) show significant differences between the two input methods. Measures marked with a star reflect statistically significant results.

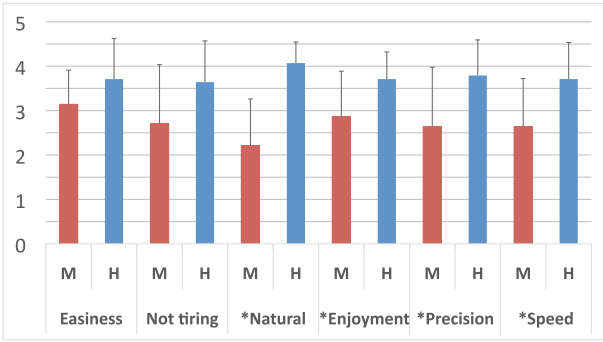


Fig. 9. Results of the questionnaire. M corresponds to the mouse and H to the haptic device. The vertical axis is the 5-level Likert-type scale, 1 corresponds to strongly disagree and 5 to strongly agree. For each subjective criterion, the star (★) indicates a significant difference between the two methods.

device than the easy scenes. The 5-minute habituation time may have been insufficient for this difficulty level which could explain why there is no significant difference between the two methods for these complex scenes. Hence, we will focus the rest of the discussion on the results obtained for the scenes classified as easy.

Among those scenes, half of them yielded significant differences between the two methods concerning the $error$. Although the results obtained for the scenes B and C are not significant at the 95 percent significance level, they are significant at the 90 percent significance level. However, results for scene F were not significant.

Spearman one-tailed correlation test between the error-related dependent variables and the curvature and node count gave the following results: the error associated with the mouse is significantly correlated to both the curvature ($r = 0.227, p = 0.016$) and the node count ($r = -0.203, p = 0.032$); the error associated with the haptic device is correlated only to the curvature ($r = 0.344, p = 0.0002$). For the mouse and haptic device cases, the error is correlated to the curvature. The particular features of scene F (a long straight dendritic spine, hence, the highest node count and one star of curvature) might have influenced the results, which did not show any significant differences..

In summary, using a haptic device seems to improve user performance for scenes with a small-to-medium polyline, which have a medium-to-high level of curvature. The results suggest that when scenes are too complex, the difference in error between the two methods fades. They also indicate that the experience of the user affects the time required to complete the task. Novice users will tend to take longer to do so with the haptic device than intermediate or expert users.

6.3.4 Subjective Evaluation

Participants filled out a preference questionnaire after the experiment. First, they had to select their favorite input device, either the mouse or the haptic device. 85.71 percent of the participants selected the haptic device as their favorite input device (compared to 14.28 percent for the mouse). The rest of the questionnaire evaluated to what extent the subjects agreed with a set of six statements using a five-level Likert-type scale. For each input device, participants were asked to rate the *ease of use (easiness)*, *lack of fatigue using the device (not tiring)*, *naturalness*, *enjoyment*, *precision* and *speed*. Each affirmation was proposed twice, once referring to the mouse and once to the Phantom. Fig. 9 shows the means and standard deviations of the two techniques for each subjective criterion. For all six subjective criteria, the Phantom obtained on average better results than the mouse. Wilcoxon signed rank tests showed significant differences for the naturalness ($W(14) = 2.5, Z = -3.18, p = 0.001, r = 0.60$), the enjoyment ($W(14) = 10.5, Z = -2.30, p = 0.022, r = 0.44$), the impression of precision ($W(14) = 7, Z = -2.11, p = 0.035, r = 0.40$) and the impression of speed ($W(14) = 7, z = -2.36, p = 0.018, r = 0.45$).

7 Conclusion and Future Work

This paper presents a haptically assisted procedure that facilitates the complete reconstruction of spines. In four steps, the method lets users locally edit the low fluorescence intensity values in those spines that cannot be fully reconstructed with standard 3D methods. The complementarity of this procedure with other segmentation tools was illustrated by showing its use with the commercial software Imaris. The haptic editing step of the procedure was evaluated with two tests. The first test focused on the benefits of the force feedback, which improved the quality of the editing. The second one compared the haptic procedure with a classic mouse and GUI interface as input. The results indicate that for the scenes categorized as easy, the haptic device also seems to improve the quality of the editing. Results did not show significant differences between the two methods for the difficult scenes. However, the habituation session lasted just 5 minutes which may have proved insufficient for these difficult scenes. A subjective analysis showed that users prefer and generally feel more confident using the haptic method than its mouse-based equivalent. Our procedure was shown to complement other segmentation tools well, as illustrated by its use with the commercial software Imaris.

The procedure can be improved in several respects. Instead of using a classic linear force feedback to control the editing intensity, the usability of the method would certainly benefit from having a custom force feedback function. Taking advantage of the haptics, additional cues could also be added to facilitate the detection of the spines that need to be edited. Future work will focus on implementing the above-mentioned features in order to keep improving the experience of the operator. We also plan to study the method learning curve and the influence of previous training on the performance for all types of scenes.

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] National Academy of Engineering, *Grand Challenges*. (2008) [Online]. Available: http://www.engineeringchallenges.org/cms/8996.aspx

[2] J. DeFelipe, "From the connectome to the synaptome: An epic love story," *Science*, vol. 330, no. 6008, pp. 1198–1201, 2010.

[3] J. DeFelipe and I. Fariñas, "The pyramidal neuron of the cerebral cortex: Morphological and chemical characteristics of the synaptic inputs," *Progress Neurobiol.*, vol. 39, no. 6, pp. 563–607, 1992.

[4] R. Yuste, *Dendritic Spines*. Cambridge, MA, USA: MIT Press, 2010.

[5] J. I. Arellano, R. Benavides-Piccione, J. DeFelipe, and R. Yuste, "Ultrastructure of dendritic spines: Correlation between synaptic and spine morphologies," *Frontiers Neurosci.*, vol. 1, no. 1, p. 131, 2007.

[6] M. M. de Lagran, R. Benavides-Piccione, I. Ballesteros-Yañez, M. Calvo, M. Morales, C. Fillat, J. DeFelipe, G. Ramakers, and M. Dierssen, "Dyrk1a influences neuronal morphogenesis through regulation of cytoskeletal dynamics in mammalian cortical neurons," *Cerebral Cortex*, vol. 22, no. 12, pp. 2867–2877, 2012.

[7] A. Sourin, S. Yasmin, and V. Zagorodnov, "Segmentation of MRI brain data using a haptic device," in *Proc. 10th IEEE Int. Conf. Inf. Technol. Appl. Biomed.*, Nov. 2010, pp. 1–4.

[8] M. Hardens and G. Szekely, "Enhancing human-computer interaction in medical segmentation," *Proc. IEEE*, vol. 91, no. 9, pp. 1430–1442, Sep. 2003.

[9] E. Meijering, "Neuron tracing in perspective." *Cytometry. Part A: J. Int. Soc. Anal. Cytol.*, vol. 77, no. 7, pp. 693–704, Jul. 2010.

[10] 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.

[11] S. Paneels and J. C. Roberts, "Review of designs for haptic data visualization," *IEEE Trans. Haptics*, vol. 3, no. 2, pp. 119–137, Apr. 2010.

[12] J. Dennerlein, D. Martin, and C. Hasser, "Force-feedback improves performance for steering and combined steering-targeting tasks," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, 2000, vol. 2, no. 1, pp. 423–429.

[13] Software imaris [Online]. Available: http://www.bitplane.com/imaris/imaris, 1993.

[14] E. M. Glaser and H. Van Der Loos, "A Semi-Automatic Computer-Microscope for the analysis of neuronal morphology," *IEEE Trans. Biomed. Eng.*, vol. BME-12, no. 1, pp. 22–31, Jan. 1965.

[15] F. Janoos, K. Mosaliganti, X. Xu, R. Machiraju, K. Huang, and S. T. C. Wong, "Robust 3D reconstruction and identification of dendritic spines from optical microscopy imaging," *Med. Image Anal.*, vol. 13, no. 1, pp. 167–179, Feb. 2009.

[16] W. Zhou, H. Li, and X. Zhou, "3d dendrite reconstruction and spine identification," in *Proc. 11th Int. Cof. Med. Image Comput. Comput.-Assist. Intervent.*, 2008, pp. 18–26.

[17] P. Shi, Y. Huang, and J. Hong, "Automated three-dimensional reconstruction and morphological analysis of dendritic spines based on semi-supervised learning," *Biomed. Opt. Exp.*, vol. 5, no. 5, pp. 1541–1553, 2014.

[18] D. Jungblut, A. Vlachos, G. Schuldt, N. Zahn, T. Deller, and G. Wittum, "Spinelab: Tool for three-dimensional reconstruction of neuronal cell morphology," *J. Biomed. Optics*, vol. 17, no. 7, pp. 0 760 071–0 760 077, 2012.

[19] H. Mukai, Y. Hatanaka, K. Mitsuhashi, Y. Hojo, Y. Komatsuzaki, R. Sato, G. Murakami, T. Kimoto, and S. Kawato, "Automated analysis of spines from confocal laser microscopy images: Application to the discrimination of androgen and estrogen effects on spinogenesis," *Cereb. Cortex*, vol. 21, no. 12, pp. 2704–2711, 2011.

[20] C. Zilles and K. Salisbury, "A constraint-based god-object method for haptic display," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. Human Robot Interaction Cooperative Robots*, 1995, pp. 146–151.

[21] D. C. Ruspini, K. Kolarov, and O. Khatib, "The haptic display of complex graphical environments," in *Proc. 24th Annu. Conf. Comput. Graph. Interactive Tech.*, 1997, pp. 345–352.

[22] K. Lundin, A. Ynnerman, and B. Gudmundsson, "Proxy-based haptic feedback from volumetric density data," in *Proc. Eurohaptic Conf.*, 2002, pp. 104–109.

[23] K. Lundin, B. Gudmundsson, and A. Ynnerman, "General proxy-based haptics for volume visualization," in *Proc. 1st Joint Eurohaptics Conf. Symp. Haptic Interfaces Virtual Environ. Teleoperator Syst.*, 2005, vol. Vi, no. 1, pp. 557–560.

[24] M. Ikits, J. Brederson, C. D. Hansen, and C. R. Johnson, "A constraint-based technique for haptic volume exploration," *IEEE Trans. Ultrason., Ferroelectr. Freq. Control*, pp. 263–269, 2003.

[25] L. Kim, G. S. Sukhatme, and M. Desbrun, "An implicit-based haptic rendering technique," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2002, vol. 3, pp. 2943–2948.

[26] L. Corenthy, J. S. Martin, M. A. Otaduy, and M. Garcia, "Volume haptic rendering with dynamically extracted isosurface," in *Proc. Haptics Symp.*, 2012, pp. 133–139.

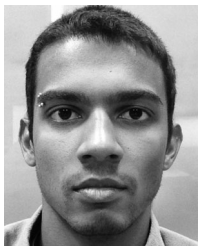
[27] E. Vidholm and J. Agmund, "Fast surface rendering for interactive medical image segmentation with haptic feedback," in *Proc. SIGRAD Environ.*, 2004, pp. 33–39.

[28] M. Harders, S. Wildermuth, and G. Székely, "New paradigms for interactive 3D volume segmentation," *J. Vis. Comput. Anim.*, vol. 13, no. 1, pp. 85–95, Feb. 2002.

[29] L. Dominjon, A. Lécuyer, G. Andrade-barroso, J.-m. Burkhardt, and S. Richir, "The 'bubble' technique: interacting with large virtual environments using haptic devices with limited workspace," in *Proc. 1st Joint Eurohaptics Conf., Symp. Haptic Interfaces Virtual Environ. Teleoperator Syst.*, 2005, vol. i, pp. 639–640.

[30] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.,*, vol. SSC-4, no. 2, pp. 100–107, Jul. 1968.

[31] L. Raya, M. A. Otaduy, and M. García, "Haptic navigation along filiform neural structures," in *Proc. World Haptics Conf.*, 2011, pp. 71–76.



Loïc Corenthy received the engineering degree from Telecom 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 Métier 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.

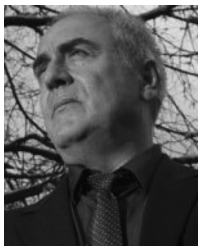
Marcos Garcia received the computer science and PhD degrees from Universidad Politécnica de Madrid (UPM), Madrid, Spain, in 2002 and 2007, respectively. In the PhD degree, he worked on simulation of elastic objects in interactive applications. From November 2007 to February 2008, he was a research fellow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

Sofia Bayona received the PhD degree in computer science from Universidad Rey Juan Carlos in Madrid in 2007. She was at INRIA Rhone-Alpes, France, in 2003 and at Imperial College London in 2009-2010 as a FP7 Marie Curie Intra European fellow (grant PIEF- GA-2009-236642). Currently, she is a researcher within the research group GMRV and she holds a position of a full-time assistant professor at the University Rey Juan Carlos, in Madrid. Her research interests include virtual reality, visualization, simulation, and education.

Andrea Santuy received the biology degree from the Universidad Complutense de Madrid, Spain, and the master's degree in neuroscience from the Universidad Autnoma de Madrid, Spain. She is currently working toward the PhD degree at the Cajal Laboratory of Cortical Circuits, CSIC-UPM, Spain. Her research interest is the morphological description of the microstructure of the brain cortex.

Jose San Martin received the Mechanical Engineer degree from UPCO-ICAI, Madrid, Spain, in 1997. He received the PhD degree from URJC, Madrid, in 2007. He was at ALSTOM and other firms until 2003 when he joined the Universidad Rey Juan Carlos-URJC. He has been an associate professor since 2007. He collaborated with Mechatronics Lab at Kyoto University in 2007-2008. His main research interests include haptics design and optimization, mechatronics, and virtual reality-based trainers.

Ruth Benavides-Piccione received the PhD degree in neuroanatomy from the Cajal Institute, Madrid, Spain, in 2004. She is a postdoctoral researcher leader of the microanatomical studies of labeled cells at the Laboratorio Cajal de Circuitos Corticales (Centro de Tecnologia Biomedica, Universidad Politecnica de Madrid, Spain). She is one of the few scientists to extensively use the powerful anatomical method of intracellular injections in fixed brain sections in order to unravel the structural dendritic design and to analyze variations in cell morphology. Using this technique, she has contributed to our understanding of pyramidal cells structure, the main cell type in the cerebral cortex, in various mammalian species, including both healthy and diseased humans (Down's syndrome, epilepsy, and Alzheimers disease).



Javier DeFelipe is a research professor of the CSIC at the UPM. He is an expert in the correlation of morphological electron microscopy data with the synaptic connectivity of nerve cells previously identified using functional markers, or by intracellular labeling. He has served as the associate editor of Brain Research (2006-2009) and is the chief editor at *Frontiers in Neuroanatomy* (2007-to date). He is a member of the editorial board of various journals (e.g., *Experimental Neurology and Cerebral Cortex*). Since 1991, the main focus of his team has been directed at understanding the neurochemical and microanatomical characteristics of the human neocortex and of the hippocampal formation. The information that has been obtained regarding the normal organization of these cortical regions has been used to investigate the possible alterations that might occur in these structures in epilepsy and Alzheimer disease.

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 Politecnica de Madrid in 1985. Currently, he is a full professor in the University Rey Juan Carlos, Madrid, Spain. His research interests include image processing and synthesis, virtual reality, 3D modeling, and parallel computing.