

# Haptic Navigation along Filiform Neural Structures

Laura Raya

Miguel A. Otaduy

Marcos García

Rey Juan Carlos University (URJC Madrid)  
Grupo de Modelado y Realidad Virtual (GMRV)

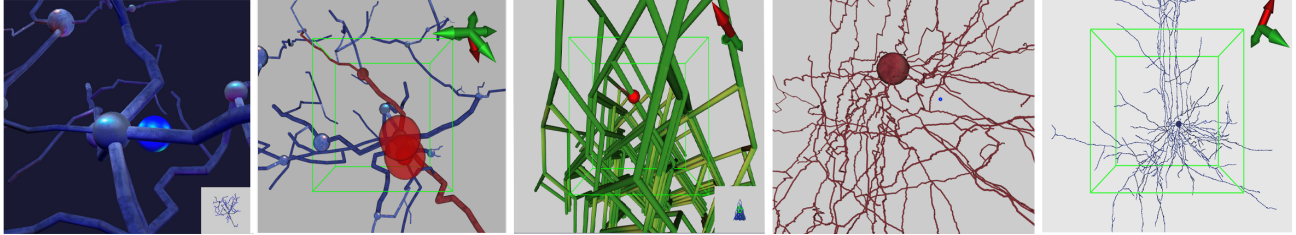


Figure 1: Examples of haptic navigation along filiform structures. The three left-most scenes were created synthetically, while the two right-most scenes were obtained from real neuron data.

## ABSTRACT

Neural connections in the brain are arranged in dense, complex networks of filiform (i.e. thread-like) structures. Navigation along complex filiform networks entails problems including path following, path selection at complex branching nodes, and the combination of large-scale with fine-scale exploration. This work proposes haptic interaction techniques to aid in the navigation along such complex filiform structures. It introduces a constraint-based path following approach with smooth node transition and intuitive branch selection during fine-scale exploration, as well as user-friendly camera control and visual aids for large-scale exploration. We demonstrate the successful application of these techniques to the analysis and understanding of dense neural networks.

**Keywords:** Haptic, filiform structures, navigation, path selection.

**Index Terms:** H.5.2 [INFORMATION INTERFACES AND PRESENTATION]: User Interfaces—Haptic I/O; H.5.1 [INFORMATION INTERFACES AND PRESENTATION]: Multimedia Information Systems—Artificial, augmented, and virtual realities

## 1 INTRODUCTION

Brain neurons exhibit a tremendous complexity that severely hinders the ability of scientists to understand their connections and mechanisms. Each human brain is estimated to contain roughly  $10^{11}$  neurons, and this complexity grows by orders of magnitude for neural dendrites (Figures 1). Therefore, visualization and interpretation of even a small portion of the brain becomes a daunting task. Existing visualization methods for neural data exploit only the visual modality. However, the interpretation of large and complex datasets, such as neural data, can be simplified by communicating information via multiple sensory channels with the aid of multi-modal interfaces [16]. Incorporating the sense of touch to information visualization has been shown to successfully increase user’s performance and situation awareness in other settings [2].

Neural structures consist of multiple connected components of neurons. Each connected component can be modeled using piecewise-linear segments meeting at nodes, and often multiple

segments meet at one node along arbitrary 3D directions. One of the most complex tasks during neural exploration is to intuitively select and explore a specific path at complex branching nodes. Motivated by this challenge, one of our main objectives was to develop path exploration techniques that would require little training and could be used intuitively by neurologists.

In this work, we present interaction techniques that exploit the haptic modality to help neurologists navigate and understand neural structures. The advantages offered by the use of haptic devices are two-fold. First, a stylus-type haptic device combined with our navigation techniques serves as an intuitive interface to navigate along neural structures naturally. And second, tactile information helps during fine-scale exploration for intuitive path following and path selection at complex branching nodes. The algorithms remain transparent to users, and haptic information is presented in a natural user-friendly manner, simply helping or preventing the user’s desired motion.

Our novel navigation techniques are classified into two groups. Section 3 introduces a proxy-based technique for navigating along filiform structures. This technique employs a multi-path optimization approach, and it incorporates a novel simple method to escape from local minimum by inferring the user’s intention based on the feedback force. Our technique smooths node transitions and allows intuitive branch selection at complex branching nodes. Section 4 presents camera control methods to combine large-scale and fine-scale exploration under the workspace limitations of haptic interfaces. Also, this section describes several visual aid to make the navigation easier.

We demonstrate the application of our haptic navigation techniques to complex neural filiform structures, but these techniques are general and could be extended to other types of filiform data.

## 2 RELATED WORK

Several researchers have proposed the use of haptic device to aid in path following tasks. Proxy-based techniques have been used to help users follow geometric primitives such as lines, segments or complex surfaces [15]. On the other hand, techniques such as those presented in [7, 8, 9] provide navigation tools for path-planning applications, where the goal is to find optimal collision-free paths to carry an object to a goal. These techniques build on approaches such as force fields, configuration-space navigation, preliminary workspace discretizations, or scene graphs. Others have designed

haptically aided path planning algorithms for filamentary structures such as nanotubes [6]. The user navigates through space moving the nanotubes along obstacle-free optimal paths, but unlike neural filiform structures, the paths present no branching.

Neural filiform structures can be regarded as 3D graphs, and haptic visualization techniques are often used to display 2D graphs to visually impaired subjects [17]. However, haptic graph visualization techniques typically represent the graph data as a 2D texture, and are not concerned with path guidance.

Haptic display of filiform structures also resembles in a way medical interventions such as needle insertion [4, 5] and catheter navigation [10], because in all cases an object navigates along a filiform structure. However, in needle insertion and catheter navigation the configuration space of the object is of much higher dimension, and the filiform structure is deformable. We exploit the low dimensionality of the haptic display problem to design more efficient algorithms, and we also pay special attention to the problem of path selection at complex branching nodes.

Our path following algorithm falls in the category of proxy-based haptic rendering methods. Given a position of the haptic interface point (in configuration space), these methods constrain the position of a proxy point to a valid manifold. The first proxy-based methods were designed for points or spheres constrained to lie on the exterior of 3D objects [18, 14]. In our case, the proxy is constrained to lie on a filiform network. In proxy-based methods, haptic feedback is computed as a function of the separation between the proxy and the haptic interface point.

Early proxy-based methods suffered discontinuities at convex regions of the constraint surface, because the proxy would suddenly jump to distant locations. Morgenbesser et al. [11] introduced force shading techniques to smooth forces on piecewise linear constraint surfaces. Later, Ruspini et al. [14] improved force shading to deal with cases where the proxy is in contact with multiple intersecting shaded planes at the same time. Force shading techniques are based on a well-defined surface normal, making them applicable only to 2-manifold surfaces. They can be extended to 1-manifold curves such as a filiform path without branches, but they do not apply to branching filiform structures, as they do not constitute valid manifolds. In our experiments, we have verified that force shading does not perform well at complex branching nodes and some branches remain inaccessible. In contrast, we propose a path-search approach that infers user’s intention through the feedback force magnitude, and allows smooth branch selection and access to all possible branches in a natural manner.

### 3 HAPTIC NAVIGATION

In this section, we describe a novel proxy-based navigation algorithm for filiform structures. We first describe the basic multi-path optimization algorithm, and we discuss its limitations. Next, we describe a force-based method to escape from local minima, which smooths discontinuities and increases accessibility to all paths in the filiform structure.

#### 3.1 Basic Constraint-Based Navigation

We represent a filiform structure as a set of nodes  $\{n\}$  connected by linear segments  $\{s\}$ . Two segments meet at nodes forming elbows, and several segments may meet at branching nodes. We denote as  $\mathbf{h}$  the (unconstrained) position of the haptic probe in the virtual workspace, and as  $\mathbf{p}$  the position of a proxy point that (locally) minimizes the distance to  $\mathbf{h}$  and is constrained to lie on the filiform structure. For visual feedback, we display the proxy as a small sphere, and the haptic probe is not displayed. On each step of the haptic update loop, we read the new position of the haptic probe, compute the new position of the proxy, and compute a feedback force

$$\mathbf{f} = K(\mathbf{p} - \mathbf{h}), \quad (1)$$

where  $K$  is a stiffness constant that can be defined by the user. The force is transformed from the virtual workspace to the device workspace to be consistent with camera motion, and then it is displayed to the user with a haptic device. Damping can be easily added to the previous equation, but in practice we found the physical damping of our system to be sufficient.

Given a proxy position  $\mathbf{p}_0$  in the previous update step, and the new haptic probe position  $\mathbf{h}$ , we seek a new proxy position  $\mathbf{p}$  constrained to the filiform structure and which (locally) minimizes the distance to  $\mathbf{h}$ . To capture the user’s intended motion, we propose a search algorithm that travels along the filiform structure by reducing the distance to the haptic probe, i.e. following a path of negative distance gradient. The navigation approach can thus be regarded as a gradient-descent optimization approach. To account for the branching topology of the filiform structure, we propose a multi-path optimization approach. Then, given the proxy from the previous step, we follow all the paths with negative gradient until we reach a local minimum along each path, and we set the new proxy at the lowest local minimum, i.e. the one that is closest to the haptic probe.

The multi-path optimization can be implemented as a recursive depth-first search on the graph defined by the filiform structure. For each branch, we define a sub-goal proxy  $\tilde{\mathbf{p}}$ , and when the algorithm terminates the proxy is set as the sub-goal that is closest to the haptic probe. We initialize the first sub-goal as the old proxy, i.e.,  $\tilde{\mathbf{p}} = \mathbf{p}_0$ . We set as active element the node or segment where it lies, and we start the recursion by calling successively two possible procedures, depending on the active element.

- If the active element is a segment  $s$ , we mark it as visited, and search for the point  $\mathbf{q} \in s$  that is closest to the haptic probe  $\mathbf{h}$ .
  - If  $\mathbf{q}$  is one of the end nodes of  $s$  and it was already visited, then we end this branch of the recursion.
  - If  $\mathbf{q}$  is one of the end nodes of  $s$  and it was not visited before, then it becomes the new sub-goal for this branch, i.e.  $\tilde{\mathbf{p}} = \mathbf{q}$ , we set it as active element and continue the recursion. We also check if it is closer to  $\mathbf{h}$  than the current proxy, and in that case we update the proxy,  $\mathbf{p} = \mathbf{q}$ .
  - If  $\tilde{\mathbf{p}}$  is an interior point of  $s$ , we check if it is closer to  $\mathbf{h}$  than the current proxy, and in that case we update the proxy, i.e.,  $\mathbf{p} = \mathbf{q}$ . Either way, we end this branch of the recursion, as we have reached a local minimum.
- If the active element is a node  $n$ , we mark it as visited. Then we loop over all unvisited segments incident on  $n$ , set them as active elements and start branches of the recursion. For each branch we initialize a new sub-goal proxy at the node’s position.

In our implementation, we let the user select a constraint-based navigation mode or a free navigation mode. In free navigation, the user moves freely through the workspace, and when the constraint-based mode is set, we initialize the proxy at the node or segment closest to the haptic probe. The free navigation mode allows users to move to distant parts of the dataset without following the filiform structure.

#### 3.2 Discontinuities and Complex Branching

In the basic algorithm described above, the proxy may suddenly undergo a large translation in one update step, producing a noticeable force discontinuity. The reason is that a small displacement of the haptic probe may clear a path along the filiform structure that allows the proxy to move to a distant location. Figure 2 shows a 2D example, where the transition across a convex elbow is smooth, but the transition across a concave elbow produces a discontinuity. Besides the disturbing force discontinuity, sudden jumps of the proxy also limit navigability, as the user may miss the intended path along the data.

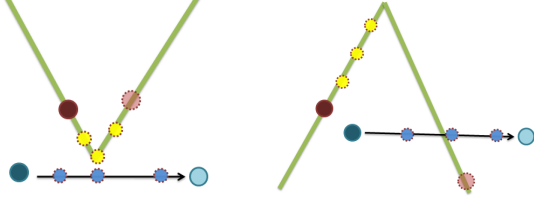


Figure 2: The basic navigation algorithm may suffer discontinuities. Left: 2D convex elbow where a smooth path of the haptic probe (in blue) induces a smooth path of the proxy (in red). Right: 2D concave elbow, where the proxy suffers a discontinuity when it reaches the node.

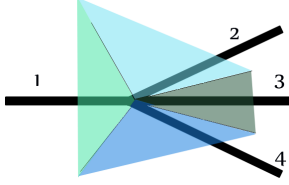


Figure 3: At complex branching nodes, segments with narrow Voronoi regions (e.g. segment 3) are very hard to access with the basic navigation algorithm.

In complex branching nodes, in particular when three or more segments stem along similar directions, some paths are very hard to access. The reason is that the proxy update based on gradient-descent is very unlikely to select segments whose Voronoi regions are very narrow, which is often the case for some segments at complex branching nodes. In the 2D example shown in Figure 3, segment 3 is hardly accessible.

### 3.3 Force-Based Path Optimization

Given that feedback force is computed as a function of the separation between proxy and haptic probe, and the proxy is constrained to the filiform structure at all times, we postulate that a small feedback force is a good indication of natural and insightful navigation. A small feedback force implies that the proxy is successfully following the user’s desired path, it is not trapped in local minima, force discontinuities are smaller, and the user is subtly perceiving all details in the structure. Therefore, if the feedback force grows (i.e. if the distance between probe and proxy grows), it means that the user is not able to reach the desired location and/or a discontinuity is about to happen.

Based on these observations, we have extended the basic navigation algorithm with a force-based path optimization to escape from local minima. In the recursive multi-path optimization algorithm described above, we incorporate the possibility to take small steps in the direction of positive distance gradient, i.e. we let the distance to the haptic probe grow slightly. To account for the user’s intent, we grow the size of the positive-gradient steps as a function of the feedback force, thereby increasing the chances to escape from a local minimum.

Specifically, we modify the recursive procedure for segments. After computing the closest point  $\mathbf{q} \in s$  to the haptic probe  $\mathbf{h}$ , we handle three possible cases:

- If  $\mathbf{q}$  is one of the end nodes of  $s$  and it was already visited, then we end this branch of the recursion.
- If  $\mathbf{q}$  is one of the end nodes of  $s$  and it was not visited before, we check whether it is closer to  $\mathbf{h}$  than the current sub-goal. If it is, then it becomes the new sub-goal for this branch, i.e.  $\tilde{\mathbf{p}} = \mathbf{q}$ , we set it as active element and continue the recursion. We also

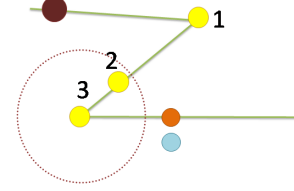


Figure 4: The blue dot represents the haptic probe, and the proxy moves from the brown dot to the red dot. The yellow dots indicate steps of the optimization algorithm. Dots 1 and 2 constitute proxy sub-goals, while 3 is a local maximum that is skipped over thanks to our force-based optimization.

check if it is closer to  $\mathbf{h}$  than the current proxy, and in that case we update the proxy,  $\mathbf{p} = \mathbf{q}$ . If the node is not closer to  $\mathbf{h}$  than the current sub-goal for this branch, then we stop the recursion for this branch.

- If  $\mathbf{q}$  is an interior point of  $s$ , we check if it is closer to  $\mathbf{h}$  than the current sub-goal proxy, and in that case we update the sub-goal, i.e.,  $\tilde{\mathbf{p}} = \mathbf{q}$ . We also check whether we should update the proxy. If the sub-goal is not updated, we end the recursion, otherwise we consider the possibility of continuing the recursion on the end nodes of  $s$  to escape from a local minimum. For each unvisited end node  $n$  of  $s$ , we set  $n$  as active element and continue the recursion if the distance between  $n$  and  $\mathbf{q}$  is smaller than a threshold  $r$ .

Effectively, the radius threshold allows the algorithm to escape from a local minimum on a segment. Note that the escape procedure will not continue beyond the first node if the next active element is not closer to the haptic probe than the current sub-goal, thereby limiting the escape procedure to one element. We do this to avoid stepping over far local maxima. Figure 4 shows a complete iteration of the solver.

The search radius  $r$  is set as a function of the distance between the sub-goal proxy  $\tilde{\mathbf{p}}$  and the haptic probe  $\mathbf{h}$ .

$$r = \min(r_{max}, G \cdot \|\tilde{\mathbf{p}} - \mathbf{h}\| + r_{min}), \quad (2)$$

with user-defined parameters for minimum radius  $r_{min}$ , maximum radius  $r_{max}$ , and growth factor  $G$ . The parameter values must be defined in terms of the neural segment length. In test,  $r_{min}$  is the length of the shortest neural segment and  $r_{max}$  is the length of the longest neural segment. For the purpose of computing the radius, we measure the distance between  $\tilde{\mathbf{p}}$  and  $\mathbf{h}$  in world coordinates, not in the virtual workspace. In this way, we adapt the radius to the scale of the workspace, and the user will perform finer motions when the virtual workspace is small (zoom in), and larger motions will be facilitated when the workspace is large (zoom out).

To sum up, we include pseudocode for the complete multi-path optimization. Algorithms 1, 2 and 3 list, respectively, the pseudocode for that overall optimization procedure (*solverInit*), and the recursive procedures for active node (*getNodeClosestProxy*) and active segment (*getSegmentClosestProxy*). *solverInit* receives as inputs the initial active element  $e$  (node or segment), the probe position  $\mathbf{h}$ , and the old proxy position  $\mathbf{p}_0$ , and it outputs the new proxy position  $\mathbf{p}$  and the new active element. *getNodeClosestProxy* and *getSegmentClosestProxy* both receive as inputs the current active element  $e$  (node or segment), the probe position  $\mathbf{h}$  and the distance between the current proxy and the probe,  $dist$ . They output the (temporary) proxy position ( $\mathbf{p}$ ), the new active element and the distance between the new proxy and the probe. In the pseudocode, the symbol  $_$  represents an unused argument.

---

**Algorithm 1**  $[p, e] = \text{solverInit}(e, \mathbf{h}, \mathbf{p}_0)$ : this function starts the recursive optimization.

---

```

//Initialize proxy-to-probe distance
dist = distance( $\mathbf{p}_0, \mathbf{h}$ )

if  $e$  is a node then
   $[p, e, \_]$  = getNodeClosestProxy( $e, \mathbf{h}, dist$ )
else
   $[p, e, \_]$  = getSegmentClosestProxy( $e, \mathbf{h}, dist$ )
end if

return  $[p, e]$ 

```

---

**Algorithm 2**  $[p, e, dist] = \text{getNodeClosestProxy}(e, \mathbf{h}, dist)$ : this function computes sub-goal proxies for all branches starting at this node.

---

```

//Prune if the node is already visited
if Already visited then
  return  $[-, -, \text{maximum float}]$ 
end if

//Start recursive searches on all segments incident on the node
for all  $segment_i$  in  $e$  do
   $[p', e', dist'] =$ 
  getSegmentClosestProxy( $segment_i, \mathbf{h}, dist$ )
  if  $dist' < dist$  then
     $[p, e, dist] = [p', e', dist']$ 
  end if
end for

return  $[p, e, dist]$ 

```

---

#### 4 CAMERA CONTROL AND VISUAL AIDS

The physical workspace of the haptic device maps to a virtual workspace in the reference system of the dataset. In large complex scenarios this virtual workspace should be intuitively adapted to provide full visual and haptic accessibility to the complete dataset. Stylus-type haptic devices provide a natural interface to navigate along 3D data, and we follow the approach by [1, 13], which centralizes all navigation and transformation aspects in one hand.

During navigation, a user can perform either proxy navigation actions, as described in the previous section, or three types of camera and workspace transformation actions: (i) virtual workspace translations, (ii) virtual workspace rotations, and (iii) virtual workspace area zooming. We center rotations and area zooming at the proxy's location, therefore, they are haptically invariant, i.e. they do not modify the feedback force unless the proxy itself moves. To rotate the workspace, we transform gimbal rotations of the stylus into X-axis and Y-axis rotations, as proposed by [1]. Rotations in the Z-axis (the viewing direction) were omitted, as users found them unintuitive. To zoom the workspace, two additional commands let users zoom in and out the scene.

Translations are not haptically invariant because they entail a translation of the proxy. When the virtual workspace is transformed, the camera must be transformed as well, to keep their relative transformation invariant. To separate proxy navigation from virtual workspace translation, we separate the workspace into two regions: a *static* region in the center of the virtual workspace, and a *dynamic* region near the boundary. If the proxy is inside the static region, we apply our proxy navigation algorithm. If the proxy moves into the dynamic region, the workspace is translated to keep the proxy inside the static region. In our experiments, the static region occupied 75% of the whole virtual workspace in each di-

---

**Algorithm 3**  $[p, e, dist] = \text{getSegmentClosestProxy}(e, \mathbf{h}, dist)$ : this function computes the the sub-goal proxy for the branch along this segment.

---

```

//Prune if the segment is already visited.
if Already visited then
  return  $[-, -, \text{maximum float}]$ 
end if

//Get the closest point for the edge,
//and prune if it is further than the sub-goal proxy.
 $\mathbf{q} = \text{getSegmentNearestPoint}(e, \mathbf{h})$ 
 $dist' = \text{distance}(\mathbf{q}, \mathbf{h})$ 
if  $dist' > dist$  then
  return  $[-, -, \text{maximum float}]$ 
end if
 $dist' = dist$ 
 $\mathbf{p} = \tilde{\mathbf{p}} = \mathbf{q}$ 

//Check if the new subgoal proxy satisfies the force-based radius.
 $r = \text{setForceBasedRadius}(dist')$ 
 $[n_1, n_2] = \text{getSegmentNodes}(e)$ 
if  $\tilde{\mathbf{p}}$  is inside sphere( $n_1, r$ ) then
   $[p', e', dist'] = \text{getNodeClosestProxy}(n_1, \mathbf{h}, dist)$ 
  if  $dist' < dist$  then
     $[p, e, dist] = [p', e', dist']$ 
  end if
end if
if  $\tilde{\mathbf{p}}$  is inside sphere( $n_2, r$ ) then
   $[p', e', dist'] = \text{getNodeClosestProxy}(n_2, \mathbf{h}, dist)$ 
  if  $dist' < dist$  then
     $[p, e, dist] = [p', e', dist']$ 
  end if
end if

return  $[p, e, dist]$ 

```

---

mension

All other transformations are controlled through user's commands, either via keyboard or using device buttons (if available). One command lets users center the virtual workspace on the proxy. The translation of the workspace is smoothed over time to prevent sharp force changes.

To better assist the user during navigation, we have incorporated additional visual aids, such as glyphs and stereo vision. (i) The virtual workspace, including static and dynamic regions, is displayed in wireframe to give users a better sense of the (de)activation of the different navigation and workspace transformation modes. (ii) When the user approaches a node, we display a glyph on the top right corner to help understand the local topology at the node, because the actual node may be occluded by the overall filiform structure. The glyph consists of arrows in the directions of the branches incident in the node. We show in red the branch where the proxy lies, and we show the other branches in green.

With simple perspective projection, the depth of filiform structures is hard to perceive, in particular in dense datasets. Different perceptual cues can provide basic depth cues with the objective of increasing depth perception. However, in the presence of complex structures, such simple depth cues are not sufficient for discriminating all possible 3D paths. We have implemented a stereo rendering module to improve depth perception. In our tests, we found that stressing the perspective by increasing the camera projection angle allows the use of shorter interocular distances. This simplifies the fusion of the two images, thereby improving the user's sensation. The stereo rendering module adapts the interocular distance in

a dynamic manner, depending on the position of the screen’s center. This dynamic adaptation avoids large positive parallax values when the user zooms in, which can cause visual merging errors and eyestrain.

## 5 RESULTS

We have tested our algorithm both on synthetic and real datasets of neural structures (see Figure 1). The real datasets were extracted from segmented information, obtained from the largest collection of publicly accessible 3D neuronal reconstructions [12]. All experiments were performed using a Phantom OMNI haptic device, interfaced through Chai3D [3], and running on an Intel Core i5 3.2 GHz machine with 2 GB of RAM and a Nvidia GForce 210 graphics card. We run the haptic update loop on a separate core at the highest update rate possible.

We have compared our navigation algorithm described in Section 3 to the force shading technique by Ruspini et al. [14], extended to deal with multiple branching. Note that standard force shading cannot deal with non-manifold structures such as branching filiform paths. Our extension selects dynamically three branch segments to find an interpolation direction, with the segment where the proxy currently lies as one of these segments. The other two are chosen based on proximity. We have evaluated both our basic multi-path optimization approach from Section 3.1 and the complete algorithm with the addition of the force-based optimization technique from Section 3.3. We have used the three test scenes shown in Figure 5. In all of them, the users had to follow the path marked in red. They can use the camera control, the visual aids and stereo vision to facilitate navigation. The first one is a path with no branches, while the second and third tests use the same data but different paths (shown in red). The path in the third test presents more complex branching, and users failed to traverse it with the basic algorithm.

In relation to the **quantitative analysis** table 1 shows the following performance results for the three scenes and for all three methods: the mean time spent by each solver in one iteration, and the mean number of segments visited in one iteration. One obvious conclusion is that performance is not an issue for any method. Performance does not depend on scene complexity but on the number of segments visited, because all three algorithms are based on local searches. Due to the high update frequency, most iterations visit only one segment. We can also observe that, as expected, the force-based optimization may some times visit several segments, as it skips over local maxima.

The navigability and usability of our algorithms require further analysis, but we have designed metrics to evaluate the effort carried out by a user while traversing a neural path. We consider the following two metrics: the integral of the haptic force over time,  $\Sigma = \int \|f\| dt$ , and the average haptic force, both measured for a complete path traversal. As we discuss in Section 3.3, the force exerted by the user is a good indication of his/her satisfaction with the current proxy position. Table 2 shows the results for a representative user with all three algorithms and on the three test scenes. Our novel navigation algorithm exhibits best results in all cases. In the test scene with no complex branching it is comparable to the other algorithms, but it clearly excels in the presence of complex branching. Finally, Figures 6 and 7 show the magnitude of the haptic force for each iteration on the first and second scene for the same representative user. The basic algorithm renders larger haptic forces, suffers higher discontinuities, and requires more time to finish the path traversal task. With complex branching (i.e. scene 3) our algorithm clearly outperforms force shading, requiring smaller forces and a shorter time to complete the task.

Regarding **qualitative analysis**, we did not perform exhaustive perceptual experiments, but we ran some preliminary tests with a group of volunteers. Some were computer scientist, used to work with haptics, and others were neurobiologists, with no previous ex-

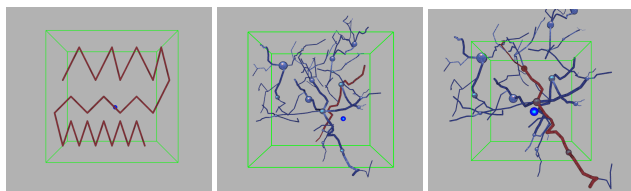


Figure 5: Scenes used for performance and navigability evaluation. Left: Scene1 (no branching); Middle: Scene2; Right: Scene3 (with complex branching).

Table 1: Average time per iteration and average number of segments visited per iteration for the different algorithms.

Scene	Algorithm	Average Time[ $\mu$ s]	Average Segments
Scene1	Basic Algorithm	0.89	1.43
	Force Shading	1.31	1.47
	Force-based Opt.	1.01	1.91
Scene2	Basic Algorithm	0.91	1.15
	Force Shading	1.21	1.11
	Force-based Opt.	1.03	3.56
Scene3	Force Shading	1.42	1.17
	Force-based Opt.	1.17	4.34

Table 2: Time-integral of the haptic force magnitude for a complete path traversal,  $\Sigma$ , and average haptic force magnitude.

Scene	Algorithm	$\Sigma$ [N*s]	Average Force[N]
Scene1	Basic Algorithm	11.86	1.81
	Force Shading	4.46	0.96
	Force-based Opt.	4.27	0.97
Scene2	Basic Algorithm	4.32	1.03
	Force Shading	2.01	0.76
	Force-based Opt.	1.40	0.53
Scene3	Force Shading	7.24	0.61
	Force-based Opt.	2.86	0.46

perience with haptics. The results indicate that, with force shading or the basic optimization algorithm, in many of the scenes, users cannot access some paths. However, to get an understanding of performance and navigability, we have compared the techniques on paths that are navigable by at least two methods. The neurobiologists stated that our haptic navigation framework is very useful for analyzing and exploring 3D models of large neuronal columns. According to their comments, haptic guidance along the segments helped them to arrive faster to points of interest. Most of the subjects said that force-based optimization makes navigation easier and found the whole system (both the constrained navigation and the camera control) user-friendly and intuitive.

## 6 CONCLUSIONS AND FUTURE WORK

The force-based optimization improves navigation performance at complex branching situations and smooths discontinuities. We have demonstrated that it outperforms basic gradient-descent algorithms, which may get stuck in local minimum, increasing the discontinuities and making some paths inaccessible. Our algorithm also outperforms existing force shading techniques, which are not well-defined at non-manifold branching locations. In all the experiments performed, the force-based optimization combined with the constrained navigation avoids inaccessible paths.

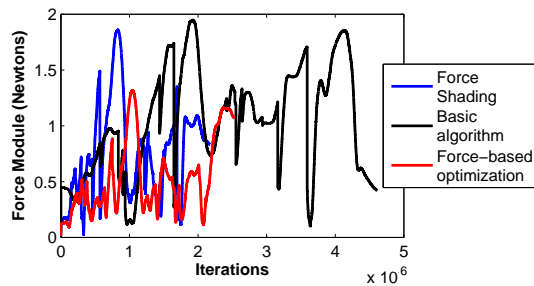


Figure 6: Force magnitude during path navigation for Scene2.

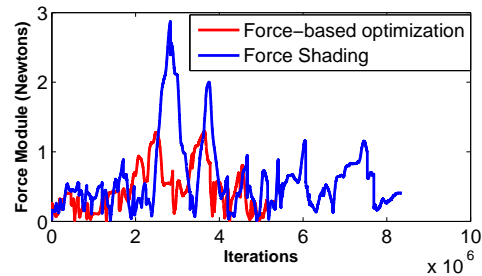


Figure 7: Force magnitude during path navigation for Scene3.

One interesting aspect of our technique is that it exploits haptic feedback in two ways. First, it constrains the user to the filiform path, allowing the perception of structural details. Second, we infer the user's intention from the feedback force, and use this information in the optimization process. It would be interesting to explore such approach in other haptic visualization settings.

The performance of the algorithm does not depend significantly on the value of  $(r_{min}, r_{max}, G)$ . Our method is limited to escape a single local minimum. Although this property have not caused any issue in the test performed (some of them with real data), we are working to eliminate this limitation.

Regarding future research directions, we plan to perform exhaustive evaluations of the navigability of our algorithm. Although our first subjects have expressed satisfaction with the algorithm, larger quantitative and qualitative evaluations and tests must be conducted. These tests should address, separately, both the optimization-based navigation algorithm and the camera control. In addition to user-based evaluation, we also plan to investigate formal conditions for stable performance of the algorithm.

Several aspects in our framework rely on user-dependent parameters (e.g. anatomical characteristics for the stereo configuration, velocity parameters for camera control, parameters of the radius function for force-based optimization, etc.). We plan to perform user studies to design automatic configuration procedures for such parameters.

Finally, although our navigation approach has been demonstrated on neural filiform structures, it is general and applicable to arbitrary 3D graphs. We expect that the method will find applicability in other fields.

## ACKNOWLEDGEMENTS

This work has been partially funded by the Spanish Ministry of Science and Innovation (grants TIN2007-67188, TIN2009-07942, TIN2010-21289 and the Cajal Blue Brain Project). The authors also wish to thank NeuroMorpho [12] database where the neural information was extracted.

## REFERENCES

- [1] J. D. Boeck and K. Coninx. Haptic camera manipulation: Extending the camera in hand metaphor. In *Proceedings of Eurohaptics 2002*, pages 36–40, 2002.
- [2] F. J. Brooks, M. Ouh-Young, J. Batter, and P. J. Kilpatrick. Project grope: Haptic displays for scientific visualization. In *Proceedings of SIGGRAPH 90. Computer Graphics*, volume 24, pages 177–185, 1990.
- [3] Chai3D.org <http://www.chai3d.org/>. Last visit: January 15th, 2011.
- [4] N. Chentanez, R. Alterovitz, D. Ritchie, L. Cho, K. K. Hauser, K. Goldberg, J. R. Shewchuk, and J. F. O'Brien. Interactive simulation of surgical needle insertion and steering. *ACM Transactions on Graphics*, 28(3):88:1–88:10, July 2009.

- [5] C. Duriez, C. Guebert, M. Marchal, S. Cotin, and L. Grisoni. Interactive simulation of flexible needle insertions based on constraint models. *Proc. of MICCAI*, 2009.
- [6] Z. Gao and A. Lecuyer. Path-planning and manipulation of nanotubes using visual and haptic guidance. In *Proceedings of IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems IEEE VECIMS*, volume 1, pages 1 – 5, 2009.
- [7] A. M. Howard and C. H. Park. Haptically guided teleoperation for learning manipulation tasks. In *Proceedings of Robotics: Science and Systems: Workshop on Robot Manipulation*, 2007.
- [8] C. V. Hurtado and J. R. Gratacos. Haptic guidance based on harmonic functions for the execution of teleoperated assembly tasks. In *Proceedings of 2007 IFAC Workshop on Intelligent Assembly and Disassembly*, pages 88–93, 2007.
- [9] N. Ladeveze, J.-Y. Fourquet, and B. Puel. Interactive path planning for haptic assistance in assembly tasks. *Computers and Graphics. Special section: IEEE Virtual Reality (VR) 2009*, 34(1):17–25, 2009.
- [10] J. Lenoir, S. Cotin, C. Duriez, and P. Neumann. Interactive physically-based simulation of catheter and guidewire. *Computers and Graphics*, 30(3):416–422, 2006.
- [11] H. B. Morgenbesser and M. A. Srinivasan. Force shading for haptic shape perception. In *Proceedings of the ASME Dynamic Systems and Control Division*, pages 407–412, 1996.
- [12] NeuroMorpho.org <http://neuromorpho.org/neuroMorpho/>. Last visit: January 13th, 2011.
- [13] M. A. Otaduy and M. C. Lin. User-centric viewpoint computation for haptic exploration and manipulation. In *Proceedings of IEEE Visualization Conference*, pages 311–318, 2001.
- [14] D. C. Ruspin, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. In *Proceedings of ACM SIGGRAPH*, pages 36–40, 1997.
- [15] N. Turro, O. Khatib, and E. Coste-Maniere. Haptically augmented teleoperation. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, volume 1, pages 386–392, 2001.
- [16] T. van Reimersdahl, F. Bley, T. Kuhlen, and C. H. Bischof. Haptic rendering techniques for the interactive exploration of cfd datasets in virtual environments. In *Proceeding EGVE '03 Proceedings of the workshop on Virtual environments*, 2003.
- [17] W. Yu, R. Ramloll, and S. Brewster. Haptic graphs for blind computer users. *Lecture Notes in Computer Science*, 2058, pages 41–51, 2000. ISSN: 0302-9743.
- [18] C. B. Zilles and J. K. Salisbury. A constraint-based god-object method for haptic display. In *Proceedings of 2007 IFAC Workshop on Intelligent Assembly and Disassembly*, pages 146–151, 1995.