

University of Groningen

Interactive Segmentation and Visualization of DTI Data Using a Hierarchical Watershed Representation

Jalba, Andrei C.; Westenberg, Michel A.; Roerdink, Jos B. T. M.

Published in:
IEEE transactions on image processing

DOI:
[10.1109/TIP.2015.2390139](https://doi.org/10.1109/TIP.2015.2390139)

IMPORTANT NOTE: You are advised to consult the publisher's version (publisher's PDF) if you wish to cite from it. Please check the document version below.

Document Version
Publisher's PDF, also known as Version of record

Publication date:
2015

[Link to publication in University of Groningen/UMCG research database](#)

Citation for published version (APA):

Jalba, A. C., Westenberg, M. A., & Roerdink, J. B. T. M. (2015). Interactive Segmentation and Visualization of DTI Data Using a Hierarchical Watershed Representation. *IEEE transactions on image processing*, 24(3), 1025-1035. <https://doi.org/10.1109/TIP.2015.2390139>

Copyright

Other than for strictly personal use, it is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license (like Creative Commons).

The publication may also be distributed here under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license. More information can be found on the University of Groningen website: <https://www.rug.nl/library/open-access/self-archiving-pure/taverne-amendment>.

Take-down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from the University of Groningen/UMCG research database (Pure): <http://www.rug.nl/research/portal>. For technical reasons the number of authors shown on this cover page is limited to 10 maximum.

Interactive Segmentation and Visualization of DTI Data Using a Hierarchical Watershed Representation

Andrei C. Jalba, Michel A. Westenberg, and Jos B. T. M. Roerdink, *Senior Member, IEEE*

Abstract—Magnetic resonance diffusion tensor imaging (DTI) measures diffusion of water molecules and is used to characterize orientation of white matter fibers and connectivity of neurological structures. Segmentation and visualization of DT images is challenging, because of low data quality and complexity of anatomical structures. In this paper, we propose an interactive segmentation approach, based on a hierarchical representation of the input DT image through a tree structure. The tree is obtained by successively merging watershed regions, based on the morphological waterfall approach, hence the name watershed tree. Region merging is done according to a combined similarity and homogeneity criterion. We introduce filters that work on the proposed tree representation, and that enable region-based attribute filtering of DTI data. Linked views between the visualizations of the simplified DT image and the tree enable a user to visually explore both data and tree at interactive rates. The coupling of filtering, semiautomatic segmentation by labeling nodes in the tree, and various interaction mechanisms support the segmentation task. Our method is robust against noise, which we demonstrate on synthetic and real DTI data.

Index Terms—DTI, segmentation, watershed, waterfall algorithm, multiple views.

I. INTRODUCTION

MAGNETIC resonance Diffusion Tensor Imaging (DTI) [1] measures diffusion of water molecules, and it is mainly used to characterize orientation of white matter fibers and connectivity of neurological structures. The spatial resolution is usually quite low, with a voxel size of about $2 \times 2 \times 2$ mm, and it is hard to obtain good signal-to-noise ratios. There are also a number of other factors that can compromise data quality, such as movements, eddy currents, and discontinuities in magnetic properties at tissue interfaces. Not only these issues, but also the complexity of anatomical structures in the brain make analysis and visualization of 3D DT images challenging.

Manuscript received March 10, 2014; revised September 5, 2014; accepted December 21, 2014. Date of publication January 8, 2015; date of current version January 30, 2015.

A. C. Jalba and M. A. Westenberg are with the Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven 5612 AZ, The Netherlands (e-mail: a.c.jalba@tue.nl; m.a.westenberg@tue.nl).

J. B. T. M. Roerdink is with the Johann Bernoulli Institute for Mathematics and Computing Science, University of Groningen, Groningen 9712 CP, The Netherlands (e-mail: j.b.t.m.roerdink@rug.nl).

This paper has supplementary downloadable material available at <http://ieeexplore.ieee.org>, provided by the author. The material includes a video that demonstrates the interactive aspects of the segmentation approach. The total size of the video is 65.3 MB. Contact m.a.westenberg@tue.nl for further questions about this work.

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TIP.2015.2390139

Numerous methods have been proposed for visualizing DTI data, see [2] for an overview. A popular technique is fiber tracking (or tractography), which computes streamlines in the vector field defined by the principal directions of the DTI tensors obtained by eigen analysis. A limitation of this approach is that only part of the tensor information is used (the first eigenvector direction), which is acceptable for (clinical) applications in which structural connectivity between regions is studied. An important problem with fiber tracking is choosing seed points; manual approaches are involved and automatic methods generate many seed points, resulting in cluttered visualizations. Clustering of fiber tracts has also been proposed for producing visualizations of fiber bundles rather than of individual fibers [3]–[5]. However, the main problem is that these methods rely on the results of the fiber tracking algorithm, where part of the tensor information is discarded.

Segmentation of DT images is the process of partitioning a DT image into multiple, non-overlapping segments, *i.e.*, sets of DT-valued voxels. Unlike fiber clustering, where one first extracts fibers and then clusters them, most segmentation methods (including ours) work with the full diffusion tensor information. Segmentation offers a viable alternative to fiber tracking for extracting useful information from the data, and several methods have been proposed [6]–[12]. However, since none of these methods achieve a full segmentation completely automatically, there is a need for interactive approaches that allow the user to steer the segmentation process.

In this paper, we propose such a method based on a hierarchical representation of the input DT image. This representation, which we call a *watershed tree*, is obtained by merging watershed regions according to a combined similarity and homogeneity criterion. In the scalar case, it was found that hierarchical, region-based image representations offer many advantages over traditional, pixel-based representations [13]–[16]. For example, hierarchical, region-based representations are much more efficient, as the number of regions is typically much smaller than the number of original pixels [15]. The same property enables our system to respond in real-time to user actions, and moreover, as the representation scale can be made arbitrarily close to the interpretation (or decision) scale through simplified views of data aggregated in regions, the user can focus on the important aspects of the (segmentation) task. Specifically, the main contributions of our paper are:

- An *interactive* method for segmentation of DT images, based on a novel and efficient hierarchical, region-based representation – the watershed tree.

- *Filters* that work on this representation, which enable region-based attribute filtering of DTI data.
- Linked views of both the simplified DT image and the very representation itself allow a user to *visually explore* the data.
- A collection of tools, coupling filtering, semi-automatic segmentation and other interaction mechanisms, to support the segmentation task.

II. RELATED WORK

In recent years, several methods have been developed for segmenting DT images. Here we focus our overview only on *interactive* and closely-related DTI segmentation techniques.

As in the scalar case, level-set methods constitute the most popular approach for segmenting tensor-valued images, see [6]–[10], [17]. Although the level-set formalism allows elegant formulations of segmentation problems, the resulting methods have usually high computational demands: Level-set methods rely on iterative solvers to seek a local solution of the variational minimization describing the segmentation problem. Therefore, especially in the presence of noise, the solution can be far from the desired one. Furthermore, most methods are sensitive to initialization and parameter settings. Moreover, as shown in [10], only a certain formulation based on a specific tensor-dissimilarity measure is able to correctly segment a simple yet noisy synthetic DT image, see also section IV-A3.

With respect to interactivity, methods based on graph cuts constitute promising alternatives to level sets. They have been popularized in image segmentation by Boykov and collaborators, see [18]. Graph cuts are faster than level-set approaches, and, for scalar data, interactive methods have been introduced [19]. However, there is relatively little research in developing graph-cuts segmentation methods for DTI: Malcolm *et al.* [20] and Weldeselassie *et al.* [11] extend the scalar method of Boykov *et al.* [18] to tensor-valued data. They focus on methodology only and not on interactivity.

Despite being very efficient, and thus potentially interactive, there are only a handful of watershed-based methods for segmenting DT images [21]–[24]. The over-segmentation problem, inherent to the (standard) watershed-segmentation paradigm [25], and the non-optimality of the solution compared to globally-optimal graph cuts, most likely hampered the development of novel watershed-based DTI segmentation methods. In Rodrigues *et al.* [23], a Gaussian scale-space in the Log-Euclidean framework is used to overcome over-segmentation. Waterfall methods [25] provide another alternative to deal with over-segmentation.

Hierarchical, region-based representations of scalar images have been thoroughly studied, see [14]–[16], [19] and references therein. Most methods rely on a tree representation, whose nodes represent regions of aggregated pixels at different representation scales. In the scalar case, the initial partition (of regions) from which the tree is built, is given by the set of flat zones [15] or regional maxima/minima [13], [14] of the input image, or by an initial over-segmentation [19], through the morphological watershed transform [25], [26]. Note that,

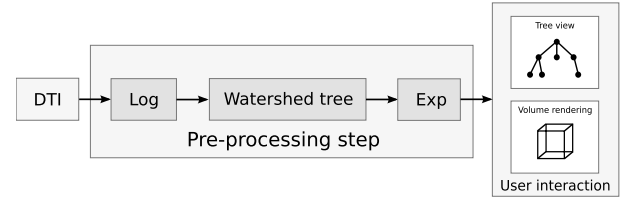


Fig. 1. Computational flow diagram of our method.

apart from Salembier *et al.* [15] and Armstrong *et al.* [19] who use their representations for segmentation, the other methods above have image simplification as their foremost target.

Non-hierarchical, region-based segmentation approaches, relying on non-parametric clustering in the Log-Euclidean space were also proposed, see [12]. This iterative segmentation scheme requires an initial segmentation as an initialization, and then it essentially performs kernel density estimation to obtain a fuzzy segmentation.

Simplified representations of scalar images based on tree data structures have been used for interactive image simplification and visualization. The Max-Tree [14], a data structure in which the nodes represent connected components of all level sets in a data set, has been extended for the purpose of interactive volumetric simplification (also called filtering) and visualization [27]. While the Max-tree offers a more compact, hierarchical representation of the data set, the tree itself is still too large to be used as a visual aid to support the (interactive) segmentation task. In addition, extending the Max-Tree data structures to tensor-valued data is difficult, since a total ordering of the level sets of the input tensor image has to be defined.

The watershed tree, which we introduce in Section III-B, is much smaller and simpler than the Max-Tree and the Binary-Partition Trees as used by Salembier and Wilkinson [16]. To achieve this, we extend the waterfall method to accommodate tensor-valued data, and build a multiscale representation. We then demonstrate that the resulting hierarchical representation constitutes a very useful visual aid to support the (interactive) segmentation task in Section III-D, which also performs better than the segmentation approach based on graph cuts.

III. PROPOSED METHOD

The computational flow diagram of our method is shown in Fig. 1. In the first pre-processing step (the Log operation), the input DT image is mapped to a Euclidean tangent space (see section III-A). In the next step, a hierarchical representation of the (regularized) image is constructed, based on a watershed tree (section III-B). This representation is central to our approach, as it enables interactive exploration and segmentation of the input DT image (section III-D). Through this representation, the user interacts with simplified views of the data aggregated in regions, which can be performed at interactive frame rates. Finally, in the last pre-processing step (the Exp operation), the data is mapped back to tensors for, *e.g.*, visualization purposes.

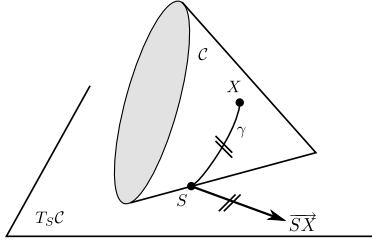


Fig. 2. Log_S maps $X \in \mathcal{C}$ to tangent vectors $\vec{S_X}$ at S . Here \mathcal{C} may represent the 3D cone of 2×2 tensors.

A. Tensor Computing

The space of symmetric nonnegative-definite matrices is not a vector space, but instead it has the structure of a Riemannian manifold that forms a convex cone \mathcal{C} [28]. Accordingly, for example, distance computations may not be simply Euclidean, but geodesically on \mathcal{C} . This can be accomplished using the so-called *log map*, $\text{Log}_S : \mathcal{C} \rightarrow T_S \mathcal{C}$, which maps points $X \in \mathcal{C}$ to the tangent plane $T_S \mathcal{C}$ at S . Specifically, Log_S maps points $X \in \mathcal{C}$ to the tangent vector $\vec{S_X}$ at S , given by the initial velocity of the geodesic curve γ with $\gamma(0) = S$ to $\gamma(1) = X$, see Fig. 2. Therefore, given points $X, Y \in \mathcal{C}$, the log map at X is given by [8], [29]

$$\text{Log}_X(Y) = X^{\frac{1}{2}} \log(\Sigma) X^{\frac{1}{2}}, \quad \text{with } \Sigma = X^{-\frac{1}{2}} Y X^{-\frac{1}{2}}, \quad (1)$$

with $\log(\cdot)$ the matrix logarithm, so that the geodesic distance D_R between the two points can be evaluated, *e.g.*, by

$$D_R(X, Y) = \|\log(\Sigma)\|_F, \quad (2)$$

where $\|\cdot\|_F$ is the Frobenius norm on symmetric matrices. Unfortunately, such accurate estimates have a high computational overhead, due to the use of matrix inverse, root, and logarithm operations. However, mapping to the tangent space at the identity tensor (a symmetric nonnegative definite matrix on \mathcal{C}), becomes equivalent to the Log-Euclidean (LE) metric [30]. It has been shown [30] that the LE framework delivers results similar to the full Riemannian framework [29], while greatly simplifying the computations involved. In fact, processing tensors in the LE framework, corresponds to performing Euclidean computations on vectors. Tensor logarithms are turned into vectors by a mapping ϕ , by keeping only the independent coefficients and multiplying off-diagonal elements by $\sqrt{2}$, so that the Frobenius norm on tensors is comparable to the Euclidean norm on vectors. Thus, given a tensor X , its vector mapping is $\text{Log}(X) = \phi(\log(X))$; compare to $\text{Log}_X(Y)$ from Eq. (1). Finally, the inverse mapping from vectors to tensors is $\text{Exp}(X) = \exp(\phi^{-1}(X))$, where $\exp(\cdot)$ denotes matrix exponentiation, see [30] and references therein for details.

Let $F : \Omega \in \mathbb{R}^3 \rightarrow S^+(3)$ denote the input DT image, so that for all $x \in \Omega$, $F(x)$ is a diffusion tensor in $S^+(3)$, the space of 3×3 real, symmetric, nonnegative-definite matrices. Note that, in this case, $\mathcal{C} \equiv S^+(3)$, and through the LE framework and ϕ , diffusion tensors are mapped to 6D vectors. Thus, given two diffusion tensors $X, Y \in S^+(3)$, the LE

distance D_{LE} between them becomes [30]

$$D_{LE}(X, Y) = \|\text{Log}(X) - \text{Log}(Y)\|_2, \quad (3)$$

where $\|\cdot\|_2$ is the Euclidean norm on vectors. The squared norm of the gradient at location x can be computed similar to [10], but in the LE framework, by

$$|\nabla F(x)|^2 = \frac{1}{2} \sum_{k=1}^3 \sum_{s=\pm 1} D_{LE}^2(F(x), F(x + s\mathbf{e}_k)), \quad (4)$$

where \mathbf{e}_k , $k = 1, 2, 3$ denote the canonical basis of \mathbb{R}^3 , and together with s are used to access neighbors of x in Ω .

The LE mean \bar{X} of a set $\{X_i\}$ of N diffusion tensors is given by [30]

$$\bar{X} = \text{Exp}\left(\frac{1}{N} \sum_{i=1}^N \text{Log}(X_i)\right). \quad (5)$$

Furthermore, given two sample sets $\{X_i\}$, $\{Y_i\}$ of sizes N_x and N_y , drawn from two independent multivariate normal distributions with sample means \bar{X} and \bar{Y} , the unbiased, pooled covariance matrix estimate is

$$W = \frac{\sum_{i=1}^{N_x} (\text{Log}(X_i) - \text{Log}(\bar{X}))(\text{Log}(X_i) - \text{Log}(\bar{X}))^T}{N_x + N_y - 2} + \frac{\sum_{i=1}^{N_y} (\text{Log}(Y_i) - \text{Log}(\bar{Y}))(\text{Log}(Y_i) - \text{Log}(\bar{Y}))^T}{N_x + N_y - 2}, \quad (6)$$

so that *Hotelling's* two-sample, *T-square statistic* [31] is

$$t^2 = \frac{N_x N_y}{N_x + N_y} (\text{Log}(\bar{X}) - \text{Log}(\bar{Y}))^T W^{-1} (\text{Log}(\bar{X}) - \text{Log}(\bar{Y})). \quad (7)$$

The quantities from Eqs. (4), (5) and (7) are required by our method. Although similar expressions can be established within the full Riemannian framework, the computations involved are much more expensive, while the results are comparable to those of the LE framework. Moreover, after mapping tensors to vectors through a global $\text{Log}(F)$ operation, all required quantities can be computed once, and only at the end of the pre-processing step (for rendering purposes), the resulting 6D vectors are mapped back to tensors, by $\text{Exp}(F)$. For our application, we found that the results delivered by the full Riemannian framework were very similar to those of the LE framework, and due to the substantial difference in computational requirements between the two, we rely on the LE framework for all our tensor computations.

B. Hierarchical Representation: Watershed Tree

In our tree-based, multiscale representation, the *leaves* of the tree represent the initial partition of the image domain. *Internal nodes* represent regions obtained by merging the regions corresponding to their children. The *root node* represents the entire image support. Thus, the tree represents a set of regions at different scales, and it can be regarded as a hierarchical, region-based representation of the input image. Clearly, the tree does not encode all possibilities for merging regions

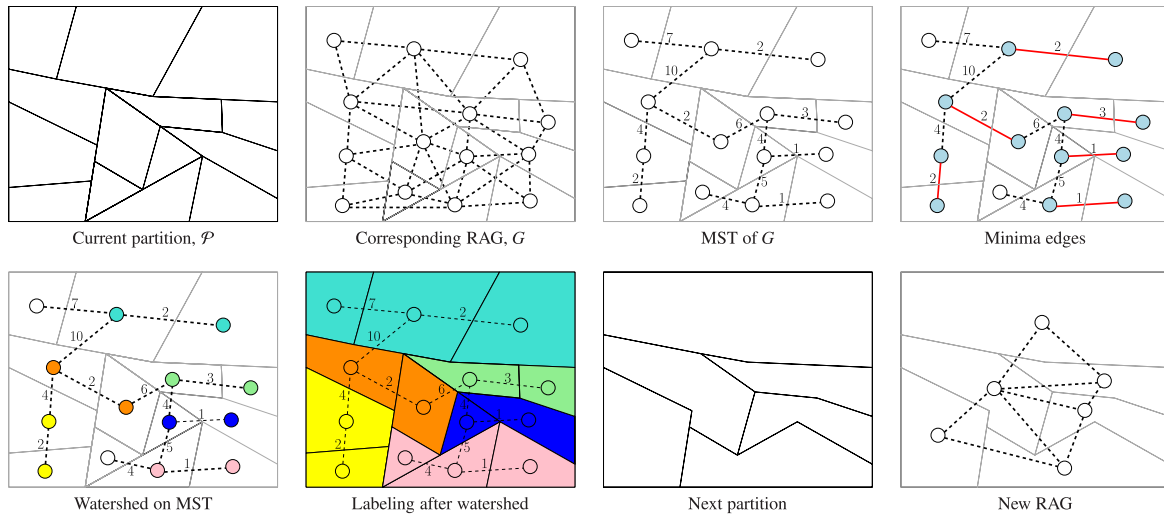


Fig. 3. Synthetic example illustrating the steps (corresponding to the while loop in Algorithm 1) of the waterfall algorithm by [32]; see also text.

Algorithm 1 Construction of the Watershed Tree T , See Fig. 1 and Text

- 1: Compute gradient-magnitude image E of F ;
 - 2: Apply the watershed transform on E to get initial partition \mathcal{P} ;
 - 3: Compute region model of all regions of \mathcal{P} ;
 - 4: Assign regions of \mathcal{P} to leaves of T ;
 - 5: Compute Region Adjacency Graph (RAG) G of \mathcal{P} ;
 - 6: **while** #vertices(G) > 1 **do**
 - 7: Compute edge costs (define merging order);
 - 8: Construct Minimum Spanning Tree (MST) T_m on G ;
 - 9: Apply watershed transform on T_m to build new partition \mathcal{P}' ;
 - 10: Compute region model of all regions of \mathcal{P}' ;
 - 11: Establish child-parent relations between regions in T at the previous level and regions in \mathcal{P}' ;
 - 12: Compute RAG G' of the new partition.
 - 13: **end while**
-

belonging to the initial partition, but merely the most useful merging steps. Thus, both the *merging order* and the *region model* upon which the tree-construction process relies [15], have to be carefully chosen.

The region model is given in our method by the LE mean (see Eq. (5)) of the vectors belonging to a region. During the merging process, the mean vector of the resulting region, corresponding to a parent node, is updated to reflect the union of the regions spanned by its child nodes. The merging order prioritizes regions which are both similar and homogeneous, and it is determined by *Hotelling's* two-sample, T-square statistic, Eq. (7). As a direct generalization of Student's t-value to multivariate data, this measure is useful to test the differences between the multivariate means of different populations. Therefore, we use it to quantify the dissimilarity between sets of tensors belonging to different regions, thus defining the order in which regions are merged by the waterfall algorithm, see below. When the region model is updated during the merging process, the merging order is also recomputed.

The tree representation employed by our method builds upon the *waterfall method* [25], extended to DT images. Specifically, the construction of the watershed tree T is shown in Algorithm 1. Our algorithm is largely based on the fast,

greedy method in [32] for computing waterfalls. In the first step of the algorithm (line 1), the gradient-magnitude image E is computed using Eq. (4). The initial partition \mathcal{P} is given by the regions computed through the watershed transform [25] of the gradient-magnitude image E . We use a specific algorithm based on a hierarchical queue [26], which can be regarded as Dijkstra's algorithm on the (image) graph, modified to allow multiple sources. These sources are located at the regional minima of E , corresponding to extrema of the input DT image F . Initially, the regions of \mathcal{P} constitute the leaves of T . After the Region Adjacency Graph (RAG) G of \mathcal{P} is constructed, the waterfall algorithm [32] is employed, to construct T in a bottom-up manner. Figure 3 illustrates the steps of the waterfall algorithm, corresponding to the **while** loop in Algorithm 1.

Essentially, this specific waterfall algorithm repeatedly performs watershed transforms on the Minimum Spanning Trees (MST) T_m of G , line 9. In our method, edge costs are given by Hotelling's T-square statistic from Eq. (7), thus defining the merging order of our representation. A *minimum edge* e_i , from which the watershed propagation on T_m is started, is defined as an edge surrounded by direct edges which all have costs larger then or equal to that of e_i , see also Fig. 3. The newly-obtained regions (shown colored in Fig. 3) represent parent nodes of the corresponding regions at the previous tree depth. Thus, parent-child relations are now established, line 11. The algorithm continues until the number of vertices in G (corresponding to image regions) equals one. The last region, spanning the whole image domain, is assigned to the root node of the tree.

Once the tree representation has been created, the input DT image can be discarded, since in our framework any interactive segmentation and visualization task can be accomplished using solely information available in the tree nodes, *i.e.*, *average tensors* computed by the $\text{Exp}(F)$ mapping in section III-A (last pre-processing step in Fig. 1). In addition to average tensors, each node in the tree stores an *attribute value* describing (the size or shape of) the region it represents, see section III-C.

C. Watershed-Tree Filtering

In general, there are two approaches for simplifying a hierarchical tree representation: pruning and non-pruning strategies, see [15], [16]. Both strategies have been used in previous work to *reconstruct* filtered scalar images from simplified tree representations [14]–[16]. Here, however, we employ them to *simplify the views* of the data aggregated into image regions, represented by the tree, to facilitate easier user manipulation and interaction through these views.

One way to simplify the views of the data is by *filtering* the watershed tree. In our method, filtering is governed by a *criterion* involving shape or size *attributes* of the regions represented by the tree nodes. Attributes describing region homogeneity or similarity among regions are superfluous, because such region characteristics are used in our representation in its very construction process. Furthermore, filtering is only based on the shapes or sizes of the regions represented by tree nodes and not on tensor values. For more complex criteria, which go beyond the scope of this paper (e.g., texture and motion), yet pertinent to scalar images, see [15], [16].

Tree filtering is a pruning strategy, if the filtering criterion is *increasing* [16]. However, since most (shape) attributes are non-increasing, filtering becomes a non-pruning strategy. Thus, in the latter case it makes sense to speak about *filtering rules*, see also [15], [16]. In the filtering step, the attribute of each tree node is assessed against a filtering criterion, i.e., each node attribute's value is compared to a given threshold τ . If its attribute value is smaller than τ , the node does not meet the criterion and is labeled accordingly. In the final tree traversal, branches are pruned from leaves up to the first ancestor that has been preserved by the filtering step. Technically, this corresponds to the *max filtering rule* in [15]. Currently, only one attribute at a time is used for filtering. However, the extension to simultaneous filtering criteria (vector attributes) is straightforward, and can be done by taking simple boolean combinations of the outcomes of the different criteria. Our method supports the following size and shape attributes of a region R : volume, non-compactness, elongation, flatness, and sparseness; see [27] and [33] for definitions and details on their *incremental* computation.

In general, it is not possible to select a threshold value τ fully automatically before the segmentation step, since the structure of the neural fibers is not *a priori* known. In fact, identifying fiber structures represents the very purpose of the segmentation task. If however, e.g., an atlas describing the fiber structures were available, one could arguably infer appropriate (yet probably error prone) threshold values for certain attributes, e.g., elongation. Instead, in our method, whenever the user changes the threshold of a certain attribute, the tree is quickly filtered and its views are updated.

D. Interactive Segmentation

Our watershed tree constitutes a flexible, hierarchical representation of the input DTI image, allowing various strategies for tree simplification to support the interactive segmentation task. The general work flow of interactive segmentation is depicted in Fig. 4. There are two main components: a Tree

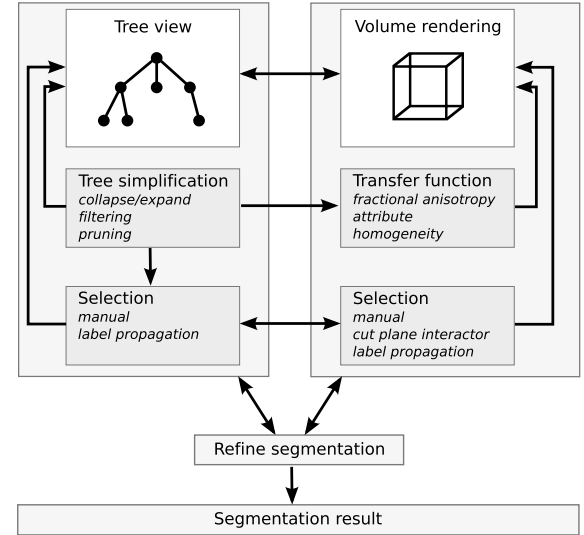


Fig. 4. Interactive segmentation work flow.

view that provides a visualization of the watershed tree itself and the Volume rendering that provides a visualization of the simplified DTI data. These two visualizations are linked, and the user can interact with both components in various ways. In the supplementary material, we provide more details about the user interface and interactions that our implementation supports. In this section, we describe our approach to interactive segmentation, which involves simplification of the tree in various ways, and performing segmentation by different selection methods. The process is carried out in a loop, during which the segmentation is refined until the desired final segmentation result is obtained.

1) *Visualization*: The *Volume rendering* component shows a visualization of the simplified DTI data, and is based on a standard graphics hardware accelerated approach [34]. We use a color scheme that maps the principal eigenvector \mathbf{V}_1 to color [2]: $(r, g, b) = (|\mathbf{V}_1 \cdot (1, 0, 0)^T|, |\mathbf{V}_1 \cdot (0, 1, 0)^T|, |\mathbf{V}_1 \cdot (0, 0, 1)^T|)$. To support the viewer in interpreting this color scheme, we display the color axes in the visualization. The transparency of the colors is derived from the fractional anisotropy (FA): the higher the FA, the more opaque. The FA is calculated from the three eigenvalues of the diffusion tensor, and it is a measure for the spatial distribution of the diffusion rates over the major axes.

The *Tree view* component shows a visualization of the watershed tree itself. The nodes are positioned by a radial layout algorithm [35], which puts the root node in the center, and arranges the other nodes on concentric circles around the root. The depth of the tree determines the number of circles required to lay out the entire tree, and each subtree is allocated a sector of the circle proportional to its size. A node is sized proportionally (on a logarithmic scale) to the volume of the region associated with it. The color of a node depends on the average orientation of the tensors in the corresponding region. The color scheme is as described above.

2) *Tree Browsing*: In our method, *Tree browsing* is a pruning strategy which performs a cut at the same depth d along each path from leaf to root. All nodes on the

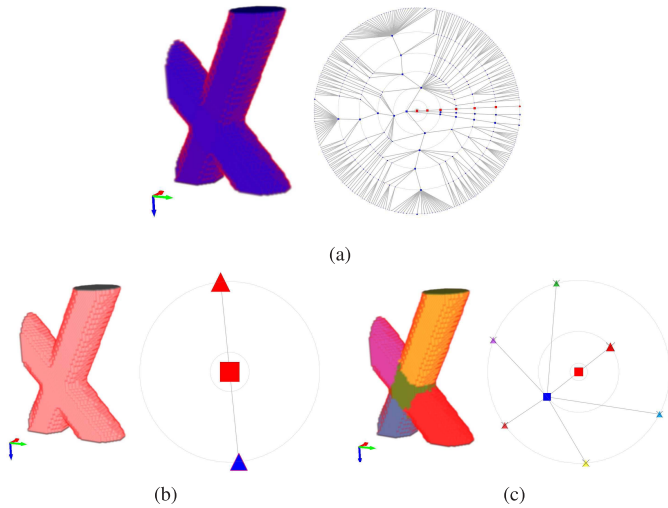


Fig. 5. (a) Synthetic data set representing two crossing fibers. The full watershed tree (at depth D) is shown on the right. (b) Tree collapsed to depth $D - 1$. The corresponding regions in the DTI volume have been merged, for details, see text. One leaf (red triangle) now corresponds to the background, and the other leaf (blue triangle) to the fibers. (c) The blue leaf node is expanded one level deeper to depth $D - 2$. Now the individual components of the fibers can be separated. For clarity, each has been assigned a different color manually.

leaf-side of the cut are *collapsed* onto the last surviving ancestors along the root-side of the cut. At any depth d , Tree browsing can be regarded as a region-based clustering method, using a simultaneous region similarity and homogeneity criterion. Due to the fact that the support region of any node at a depth d is included into that of a node at depth $d + 1$ (its parent), the effect of Tree browsing at depth d results in simplified views as d increases from 0 (leaves) to the maximum D (root).

Figure 5 shows an example of Tree browsing on a synthetic data set of two crossing fibers. The full tree and corresponding volume rendering are shown in Fig. 5(a). In Fig. 5(b) the tree has been collapsed to depth $D - 1$. There are now two leaves: one represents the background (the red triangle), and the other (the blue triangle) represents the fibers. In Fig. 5(c), the blue leaf has been expanded to depth $D - 2$, which reveals all components of the crossing fibers as separate leaves. The color assignment has been done manually.

3) *Label Propagation*: Our method relies on a simple yet powerful and interactive segmentation mechanism, through the so-called morphological marker (label) propagation approach [15], [25]; for more involved approaches, see [19]. Label propagation works in three main steps:

- Interactive label assignment (object of interest or background) to some leaf nodes in the tree;
- Bottom-up propagation of labels to parents, in the absence of conflicts between labels (*i.e.*, as long as parent nodes can be labeled as either object or background regions);
- Top-down propagation of labels, such that sub-trees rooted in the children of the nodes with labeling conflicts have the same label as their root node.

Note that label propagation does not necessarily assign labels to all leaves. Indeed, only those leaves of the sub-trees rooted

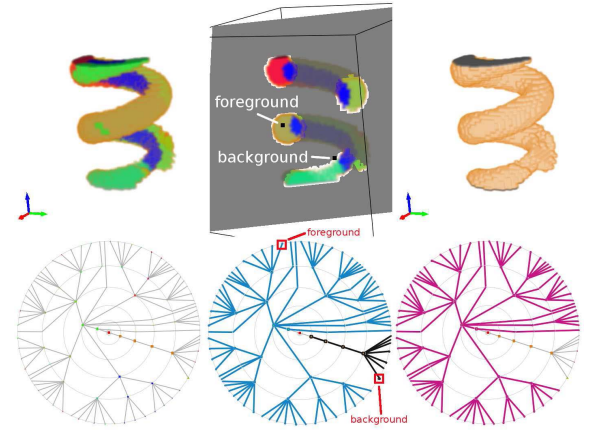


Fig. 6. Label propagation on a synthetic input data set. *Left*: volume rendering of input data and corresponding tree. *Center*: label propagation using the cut-plane interactor for seed placement; one background seed and one foreground seed are placed at the shown locations (top). Label propagation is initiated from the two leaves highlighted by red squares (bottom). The result after label propagation is shown in the tree, where blue corresponds to foreground and black to background. *Right*: final segmentation and corresponding tree.

in children of nodes with labeling conflicts are assigned a label. Thus, label assignment is controlled by the similarity between regions, marked as either object or background by the user. Finally, the user can refine the segmentation, by further assigning labels to leaves and then propagating them and *e.g.*, by Tree browsing. Both views are updated in real-time to reflect the current segmentation. An example for a synthetic data set is shown in Fig. 6. A volume rendering of the input data and the corresponding tree are shown on the left. In the center, a foreground label is assigned to a part on the inside of the helix structure, and a background label to a point outside the object. Label assignment is done by a cut plane, see the top center image. The leaves corresponding to the marked points in this plane are highlighted by a red square in the tree visualization (see the center image on the bottom row). In this example, label propagation can be done up to the level directly under the root, before reaching a conflict. The subtree containing the helix is then labelled foreground (blue), and the other subtree is labelled background (black). The right image shows the foreground object in a selected state.

4) *Direct (Manual) Selection of Nodes*: Manual segmentation of the regions defined by the watershed tree can be performed by selecting nodes in the Tree view, and assigning them a color via the user interface. The corresponding nodes become part of the segmentation, which is indicated by a small cross drawn over the node. An example of manual color assignment is shown in Fig. 5(c).

IV. RESULTS

In this section we present results for interactive segmentation by our method, on both synthetic and real DTI data sets, and we compare our approach to the graph cuts method by [11]. The evaluation is qualitative in terms of efficiency and visual inspection of the segmentation results. Note that, because no repository of real DTI data sets, including ground-truth segmentation exists, a quantitative evaluation is not feasible. All experiments were performed on a machine equipped

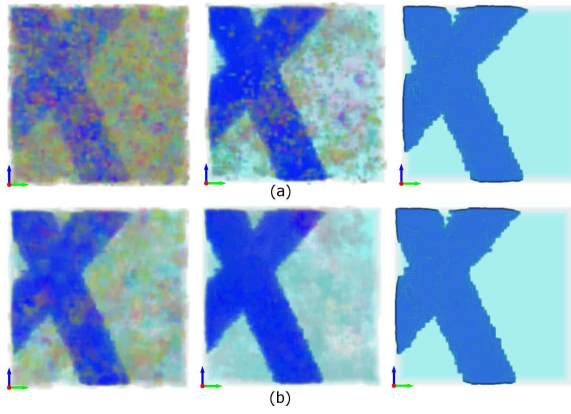


Fig. 7. Segmenting the Crossing Fibers data set. *Left-to-right*: direct volume renderings of the segmentation results. (a) Tree browsing outcomes at depths $d = 0, 5, 10$, where $d = 0$ corresponds to the leaves and $d = D = 11$ to the root node. (b) Volume filtering results with $\tau = 50, 1,000, 43,000$.

with an Intel Core 2 Quad CPU at 2.4 GHz, 6 GB RAM and a GeForce GTX 280 GPU. We refer to reader also to the video (see supplementary material), which demonstrates how to obtain some of these results with our prototype.

A. Synthetic Data

The tensors of the synthetic data sets follow the shapes of the geometrical figures, such that the more twisted the shape is, the larger the tensor variability becomes, and thus the more difficult it is for the watershed tree to represent the whole object by a single region. Furthermore, all synthetic data sets used in our experiments were corrupted by Gaussian noise, generated as suggested in [10]. The volume renderer uses the standard DTI colors as described in Section III-D1.

1) *The Crossing Fibers Data Set*: In the first experiment we use a synthetic data set depicting the crossing of two fiber bundles. Within the bundles, in the noise-free data set, the fractional anisotropy decreases slightly with the distance from their center-lines. In the crossing region itself, the tensors are disk shaped, whereas in the background they have spherical shapes. As in [10], Gaussian noise with zero mean tensor and (Frobenius) norm of the covariance matrix, $\|C\|_F = 0.028$, was used to corrupt the data.

The results of Tree browsing this data set are shown in Fig. 7(a). The watershed tree has a maximum depth of 12 levels, 17,971 leaves and an average branching factor of 2.5. Note that even though the data set was corrupted by noise, at tree depth $d = 10$, almost all noise is removed. The same could be achieved after volume-filtering the watershed tree (using $\tau = 43,000$), see Fig. 7(b). In both cases, the desired segmentation could easily be obtained by manual selection of nodes or label propagation. As has been shown by [10] on a similar (Y-shaped) synthetic data set, level-set methods have a performance that is comparable to our approach.

2) *The Torus Data Set*: Next we consider another example, where the tensors follow the tangent of the center-line of a torus, so that the orientation variability is larger. This data set was corrupted by Gaussian noise at the same levels as

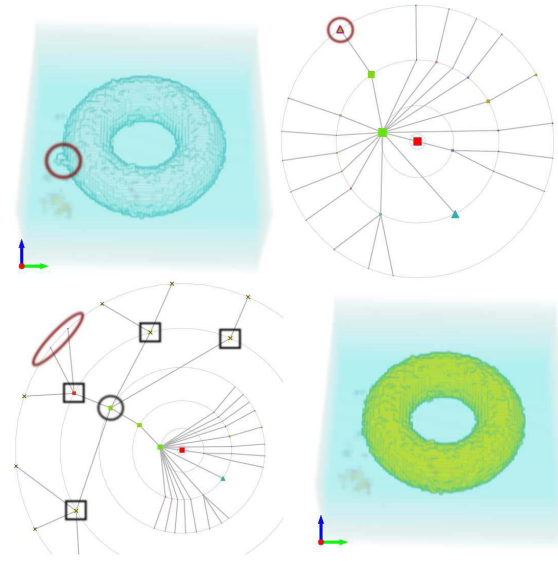


Fig. 8. *Left-to-right, top-to-bottom*: Initial segmentation by label propagation at $d = 8$ with noise structures (highlighted) attached to the torus object; tree view showing a highlighted node corresponding to noise structures; after expanding the selected node by two levels and performing manual selection of the expanded nodes (marked with squares), the object is nicely segmented.

the Crossing Fibers DT image. Segmentation can be easily performed by Tree browsing in conjunction with manual selection, or by label propagation, see Fig. 8. This data set has proven to be challenging to level-set approaches, where the surface evolution can get trapped in a local minimum if a Euclidian probability metric is used [10].

3) *The Helix Data Set*: In the last synthetic example, we consider a helicoidal object, such that the tensors of the clean Helix data set are also oriented along the tangent of its center-line. The fractional anisotropy of the tensors within the object varies around each spire, and also, their orientation spans a broader variability range than for the Torus data set. The noise level was as in the previous experiments. The watershed tree had a maximum depth of 12 levels, 10,156 leaves and an average branching factor of 2.3.

As argued in [10], it is highly desirable for any segmentation method to succeed on such data sets, since this tensor-variation pattern is fairly realistic with respect to real DT images. However, in [10] it is shown that the only (level-set) method that correctly manages to segment a similar data set uses statistical estimates computed within the computationally expensive, full Riemannian metric. The other (level-set) methods, based on estimates in the Euclidean and J-Divergence probability metrics [9], both fail to correctly segment the object. As shown in Fig. 9(a), in our method segmentation can be again easily performed, *e.g.*, by Tree browsing at depth $d = 7$ and manual selection. Moreover, filtering the watershed tree using elongation or sparseness attributes also delivers very good results, see Fig. 9(b).

B. Real DTI Data

The last experiment was performed on a $148 \times 190 \times 160$ DT image of a human brain. The voxel size of the data set

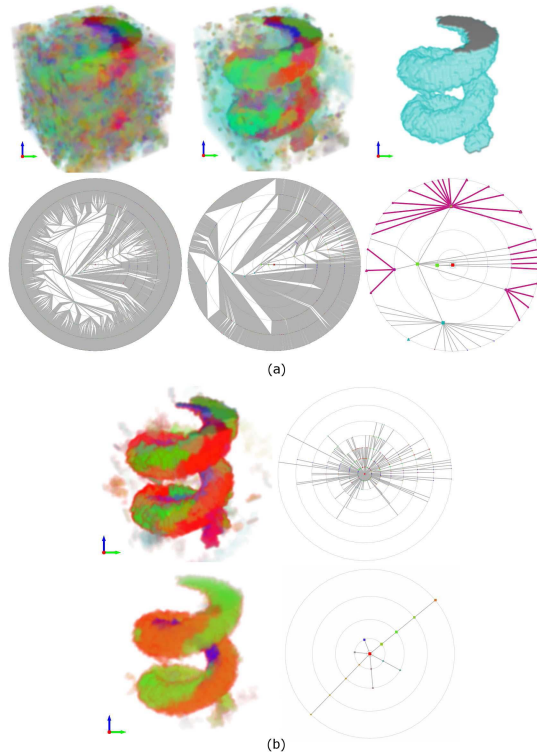


Fig. 9. Segmenting the noisy Helix data set. (a) Tree browsing and manual selection. *Left-to-right, first row*: tree browsing (and segmentation) at depths $d = 0, 3, 7$, *second row*: corresponding views of the watershed tree. (b) Attribute filtering using elongation (*first row*) with $\tau = 5.2$, and sparseness (*second row*) with $\tau = 4.2$.

is $1 \times 1 \times 1$ mm. Due to noise and the fact that only 6 diffusion-weighted images were used for acquisition, not all samples have positive eigenvalues; this is tackled by setting such eigenvalues to small positive values. The watershed tree has a depth $D = 14$ and 60,524 leaves, and the average branching factor is about 2.9.

We now show how to segment the corpus callosum, corona radiata, and the projection fiber. To accomplish this task the following (interaction) strategy was used. First, we quickly previewed the data set by Tree browsing. We also employed volume filtering with $\tau = 400$, to remove very small regions. Then, we started selecting nodes in the tree view, at depth $d = 8$, where from a total of 846 leaves, about 230 remained after volume filtering, see the first image of Fig. 10(a). By selecting several large nodes at higher levels, and observing the selections in the rendering window, it was immediately clear that the desired objects were part of the regions corresponding to these compound nodes. After refining the selection a bit, and assigning different colors to the identified structures, we obtained the segmentation shown in the third image of Fig. 10(a). Next, we used label propagation at depth $d = 8$ to ‘grow’ the corona radiata from the segmented corpus callosum. The final segmentation is shown in Fig. 10(b). To add context to the visualization of the segmentation result, fiber tracts were generated by tracing streamlines seeded uniformly in the segmented regions.

This example demonstrates the versatility of our approach, and shows that it can be used to segment different structures

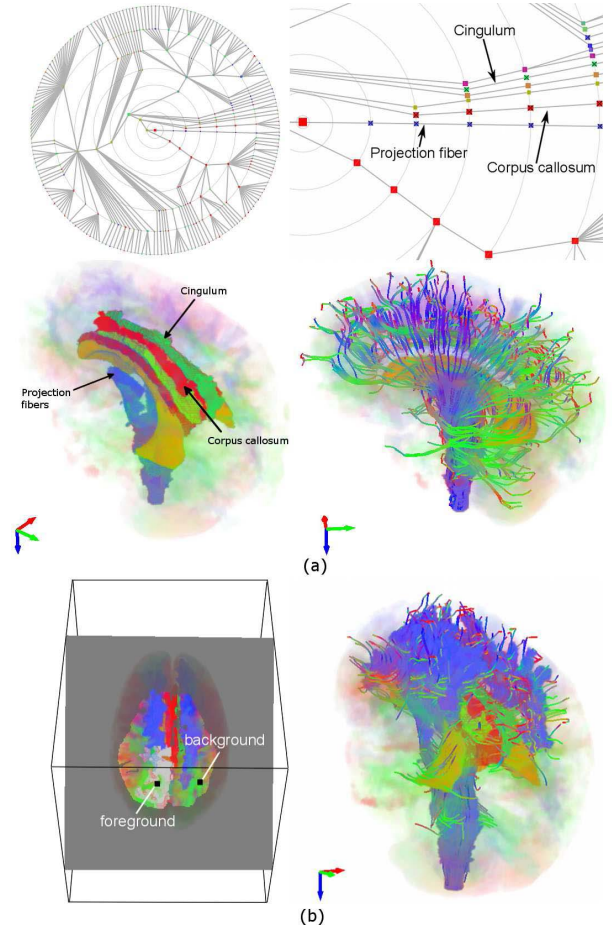


Fig. 10. Segmenting brain structures in a real DT image. (a) Segmentation of the corpus callosum, cingulum and projection fibers. *First row*: Tree browsing at depth $d = 8$; zoom-in showing sub-trees corresponding to the segmented structures; *second row*: Segmented structures and streamlines. (b) ‘Growing’ the corona radiata from the corpus callosum by label propagation. *Left*: label propagation at $d = 8$, *right*: final segmentation combined with streamlines.

in the brain. In related work, segmentation results on real data are only provided for large structures, such as the cardiac wall [11] and corpus callosum [6], [7], [9], [11]. Using the Riemannian framework, only [10] show a segmentation of the corona radiata, which took considerable effort to obtain.

C. Efficiency

Our interactive segmentation method is computationally very efficient. For example, the time required during pre-processing to construct the watershed-tree representation of the DT brain image of size $148 \times 190 \times 160$ was 40 seconds. Moreover, once the tree representation has been computed, it can be saved to external memory, for later usage.

The (interaction) time required to obtain the segmentation in Fig. 10(a) was about 5 minutes, using the strategy described above. Further, obtaining its refined version from Fig. 10(b) required only a few seconds extra. For comparison, the level set method of [10] took about 10 minutes to obtain a simpler, one-region segmentation, on a (brain) DT image four times smaller than ours. If multiple regions have to be segmented, or if a segmentation has to be changed/refined, the whole

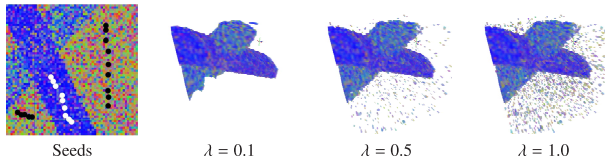


Fig. 11. Segmentation of the crossing fibers by the graph-cuts method of [11]; *left-to-right*: initialization (foreground–white and background–black seeds) and segmentation results for different weights λ , see text.

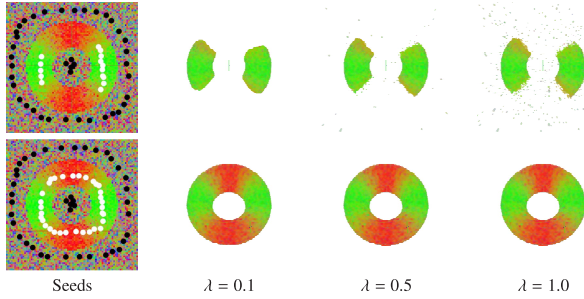


Fig. 12. Segmentation of the noisy torus by graph cuts with two different initializations.

initialization and level-set propagation process has to be repeated. Moreover, if the result after propagation does not meet certain requirements parameter adjustment has to be done, and the whole process has to be repeated. By contrast, in our method one can just continue assigning new segments (or refine existing ones), at the same or another tree depth, by filtering, manual selection, or label propagation, see Fig. 4.

D. Comparison to Graph Cuts

As mentioned in Section II, segmentation methods based on graph cuts represent popular alternatives to level-set approaches. The standard technique optimizes an energy functional over the segmentation (binary labeling) of image voxels regarded as nodes of the image graph. Typically, the energy functional is the weighted sum of two terms, representing (i) the likelihood of a node to belong to foreground or background (region term) and (ii) the penalty for two neighboring nodes for having different labels (boundary term).

Here, we compare DT-MRI segmentation results by our method to those of the graph-cuts method by [11]. We chose this method for comparison since it is (i) based on globally-optimal graph-cuts, (ii) semi-automatic, and (iii) faster than standard level-set approaches, see [11] for details. Furthermore, we used the same (noisy synthetic) data sets as input. The results of this method are shown in Figs. 11, 12 and 13; the value of the weighting parameter λ , specifying the relative importance of the region term versus the boundary term, is indicated in each figure.

Figure 11 shows that the graph-cuts method can successfully segment the crossing fibers (with $\lambda = 0.1$), although the initialization (manual placement of seeds) was performed in only one slice, and moreover, foreground seeds were located in only one part of one of the crossing fiber bundles. Note though that the parameter λ needs to be chosen carefully, to avoid noise in the final segmentation result.

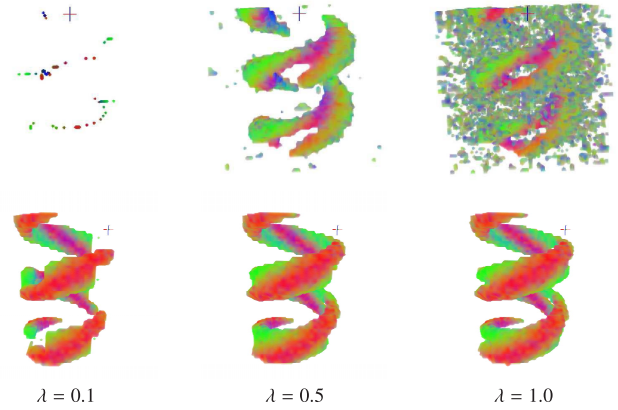


Fig. 13. Segmentation of the noisy helix by graph cuts; *First row*: results when seeds were placed only in a few slices, *second row*: results with seeds placed in each slice along the whole centerline of the helix.

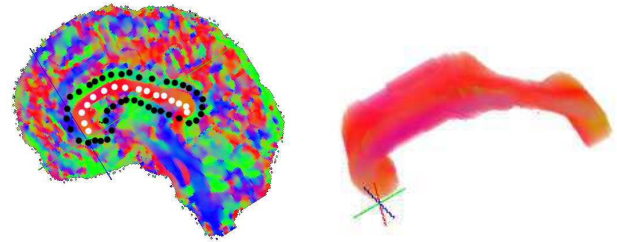


Fig. 14. Segmentation of the Brain data set by graph cuts. *Left*: seeds placed along the whole corpus callosum structure and around it in the background, in one slice. *Right*: segmentation result.

For the noisy Torus data set, a similar type of initialization results in only a partial segmentation of the torus object, see the top row in Fig. 12. The quality of the result in this case was independent of the parameter value λ . To obtain the whole object, we had to place foreground seeds inside both the red and green colored regions of the torus; the corresponding segmentation result is shown in Fig. 12 on the bottom row.

The Helix data set proved to be problematic for the graph-cuts method, see Fig. 13 and also our discussion in Section IV-A3. The first row shows segmentation results obtained by placing seeds in only a few slices. In this case, the result consists of only regions near the seed points (small λ), only some parts of the helix (medium λ), or a large amount of noise (high λ). If foreground seeds are placed in every slice and along the centerline of the entire helix a better segmentation can be obtained as shown in Fig. 13 on the bottom row. However, even then the helix object is not obtained completely. Such laborious initializations are not realistic for real data sets, because the sought structures are not *a priori* known. However, if an atlas is available, the initial seed placement can be performed using a registration-like approach, which could potentially improve the final segmentation results for graph-cuts (and level-set) methods.

Figure 14 shows a segmentation of the corpus callosum in the Brain data set. Note that this structure is recovered only partially. To obtain this result, we had to carefully place seeds and tune the parameter λ . Therefore, we had to fully re-run the method several times, which was very time consuming as each

run required about 5 minutes. In contrast, using our method, we were able to extract much more detail from this data set (see Fig. 10), in about the same time.

E. Discussion

The results on the Crossing Fibers and Torus data sets show that level sets, graph cuts, and our method perform similarly. However, even for these simple data sets, the parameters for both the level sets and graph cuts approaches have to be chosen carefully to obtain the desired result. In our approach, this is not very critical, and the segmentations were easy to obtain.

The Helix data set, representing more realistic data, could not be recovered completely by the graph cuts approach, despite an elaborate initialization. The level-set approach only segmented this data set successfully in the full Riemannian metric [10]. As demonstrated, segmentation of this data set was straightforward with our approach by simplifying the data by tree browsing or attribute filtering, followed by label propagation or manual selection of nodes in the tree.

On the Brain data set, it was possible to recover the corpus callosum partially with the graph cuts approach. This is also possible with level sets, but smaller structures could only be obtained in the Riemannian framework [10]. We showed that such smaller structures are easily obtained by our method. The results on this data set demonstrate that the interaction mechanisms, simplification strategies, fast filtering and response to parameter changes result in an effective and efficient, interactive segmentation method.

V. CONCLUSIONS

We have presented an efficient, interactive method for segmenting DT images. Our method relies on a hierarchical representation, the watershed tree, obtained by merging watershed regions, through a waterfall algorithm. The region model is given by average tensors computed within the regions, whereas the merging order prioritizes homogeneous and similar regions. We have demonstrated the performance of our approach on (noisy) synthetic and real DTI data, and by a comparison to other (semi-automatic) segmentation approaches.

The current method has a number of shortcomings. First, very small structures cannot properly be recovered, due to our region-based representation. This is inherent to the approach and can only be solved by applying other techniques. Secondly, errors introduced early, at low depth levels, in the merging process propagate up in the tree at subsequent levels. In such cases, automatic label propagation may fail (depending on the initialization), and in general, more user interaction is required to, *e.g.*, expand regions and seek for the desired structures. Such merging errors may be corrected during the interactive segmentation by using graph cuts as proposed by [19]. A further improvement consists in replacing the sample mean and covariance estimators, defining the region model and merging order, by robust measures of location and dispersion, *e.g.*, median and Minimum Volume Ellipsoid, respectively.

REFERENCES

- [1] P. J. Basser, J. Mattiello, and D. LeBihan, "Estimation of the effective self-diffusion tensor from the NMR spin echo," *J. Magn. Reson.*, B, vol. 103, no. 3, pp. 247–254, 1994.
- [2] A. Vilanova, S. Zhang, G. Kindlmann, and D. Laidlaw, "An introduction to visualization of diffusion tensor imaging and its applications," in *Visualization and Processing of Tensor Fields* (Mathematics and Visualization), J. Weickert and H. Hagen, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 121–153.
- [3] B. Moberts, A. Vilanova, and J. J. van Wijk, "Evaluation of fiber clustering methods for diffusion tensor imaging," in *Proc. IEEE Vis.*, Oct. 2005, pp. 65–72.
- [4] A. Brun, H. Knutsson, H.-J. Park, M. E. Shenton, and C.-F. Westin, "Clustering fiber traces using normalized cuts," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI* (Lecture Notes in Computer Science), vol. 3216. Berlin, Germany: Springer-Verlag, 2004, pp. 368–375.
- [5] W. Chen *et al.*, "A novel interface for interactive exploration of DTI fibers," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 6, pp. 1433–1440, Nov./Dec. 2009.
- [6] L. Zhukov, K. Museth, D. Breen, A. H. Barr, and R. Whitaker, "Level set modeling and segmentation of diffusion tensor magnetic resonance imaging brain data," *J. Electron. Imag.*, vol. 12, no. 1, pp. 125–133, 2003.
- [7] M. Rousson, C. Lenglet, and R. Deriche, "Level set and region based surface propagation for diffusion tensor MRI segmentation," in *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis* (Lecture Notes in Computer Science), vol. 3117, M. Sonka, I. A. Kakadiaris, and J. Kybic, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 123–134.
- [8] C. Lenglet, M. Rousson, R. Deriche, O. Faugeras, S. Lehericy, and K. Ugurbil, "A Riemannian approach to diffusion tensor images segmentation," in *Information Processing in Medical Imaging* (Lecture Notes in Computer Science), vol. 3565, G. E. Christensen and M. Sonka, Eds. Berlin, Germany: Springer-Verlag, 2005, pp. 591–602.
- [9] Z. Wang and B. C. Vemuri, "DTI segmentation using an information theoretic tensor dissimilarity measure," *IEEE Trans. Med. Imag.*, vol. 24, no. 10, pp. 1267–1277, Oct. 2005.
- [10] C. Lenglet, M. Rousson, and R. Deriche, "DTI segmentation by statistical surface evolution," *IEEE Trans. Med. Imag.*, vol. 25, no. 6, pp. 685–700, Jun. 2006.
- [11] Y. T. Weldeelassie and G. Hamarneh, "DT-MRI segmentation using graph cuts," *Proc. SPIE, Med. Imag., Image Process.*, vol. 6512, p. 65121K, Mar. 2007.
- [12] S. P. Awate, H. Zhang, and J. C. Gee, "A fuzzy, nonparametric segmentation framework for DTI and MRI analysis: With applications to DTI-tract extraction," *IEEE Trans. Med. Imag.*, vol. 26, no. 11, pp. 1525–1536, Nov. 2007.
- [13] E. J. Breen and R. Jones, "Attribute openings, thinnings, and granulometries," *Comput. Vis. Image Understand.*, vol. 64, no. 3, pp. 377–389, 1996.
- [14] P. Salembier, A. Oliveras, and L. Garrido, "Antiextensive connected operators for image and sequence processing," *IEEE Trans. Image Process.*, vol. 7, no. 4, pp. 555–570, Apr. 1998.
- [15] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *IEEE Trans. Image Process.*, vol. 9, no. 4, pp. 561–576, Apr. 2000.
- [16] P. Salembier and M. H. F. Wilkinson, "Connected operators," *IEEE Signal Process. Mag.*, vol. 26, no. 6, pp. 136–157, Nov. 2009.
- [17] C. Feddern, J. Weickert, and B. Burgeth, "Level-set methods for tensor-valued images," in *Proc. 2nd IEEE Workshop Geometric Level Set Methods Comput. Vis.*, Oct. 2003, pp. 65–72.
- [18] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," *Int. J. Comput. Vis.*, vol. 70, no. 2, pp. 109–131, 2006.
- [19] C. J. Armstrong, B. L. Price, and W. A. Barrett, "Interactive segmentation of image volumes with live surface," *Comput. Graph.*, vol. 31, no. 2, pp. 212–229, 2007.
- [20] J. Malcolm, Y. Rathi, and A. Tannenbaum, "A graph cut approach to image segmentation in tensor space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2007, pp. 1–8.

- [21] T. Schultz, H. Theisel, and H.-P. Seidel, "Segmentation of DT-MRI anisotropy isosurfaces," in *Proc. EuroVis, Joint Eurograph.-IEEE VGTC Symp. Vis. (VisSym/EUROVIS)*, May 2007, pp. 187–194.
- [22] L. Rittner and R. de Alencar Lotufo, "Segmentation of DTI based on tensorial morphological gradient," *Proc. SPIE, Med. Imag., Image Process.*, vol. 7259, pp. 72591E-1–72591E-12, Mar. 2009.
- [23] P. R. Rodrigues, A. C. Jalba, P. Fillard, A. Vilanova, and B. M. ter Haar Romeny, "A multi-resolution watershed-based approach for the segmentation of diffusion tensor images," in *Proc. MICCAI Workshop Diffusion Modelling*, London, U.K., 2009, pp. 161–172.
- [24] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Thinnings, shortest path forests, and topological watersheds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 925–939, May 2010.
- [25] S. Beucher and F. Meyer, "The morphological approach to segmentation: The watershed transformation," in *Mathematical Morphology in Image Processing*, E. R. Dougherty, Ed. New York, NY, USA: Marcel Dekker Inc., 1993, pp. 433–481.
- [26] J. B. T. M. Roerdink and A. Meijster, "The watershed transform: Definitions, algorithms and parallelization strategies," *Fundam. Informat.*, vol. 41, nos. 1–2, pp. 187–228, Mar. 2000.
- [27] M. A. Westenberg, J. B. T. M. Roerdink, and M. H. F. Wilkinson, "Volumetric attribute filtering and interactive visualization using the max-tree representation," *IEEE Trans. Image Process.*, vol. 16, no. 12, pp. 2943–2952, Dec. 2007.
- [28] P. Thomas Fletcher and S. Joshi, "Principal geodesic analysis on symmetric spaces: Statistics of diffusion tensors," in *Computer Vision and Mathematical Methods in Medical and Biomedical Image Analysis (Lecture Notes in Computer Science)*, vol. 3117, M. Sonka, I. A. Kakadiaris, and J. Kybic, Eds. Berlin, Germany: Springer-Verlag, 2004, pp. 87–98.
- [29] X. Pennec, P. Fillard, and N. Ayache, "A Riemannian framework for tensor computing," *Int. J. Comput. Vis.*, vol. 66, no. 1, pp. 41–66, 2006.
- [30] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Log-Euclidean metrics for fast and simple calculus on diffusion tensors," *Magn. Reson. Med.*, vol. 56, no. 2, pp. 411–421, Aug. 2006.
- [31] H. Hotelling, "Multivariate quality control," in *Techniques of Statistical Analysis*, C. Eisenhart, M. W. Hastay, and W. A. Wallis, Eds. New York, NY, USA: McGraw-Hill, 1947.
- [32] B. Marcotegui and S. Beucher, "Fast implementation of waterfall based on graphs," in *Proc. 7th Int. Conf. ISMM*, 2005, pp. 177–186.
- [33] A. C. Jalba and M. A. Westenberg, "A comparison of two tree representations for data-driven volumetric image filtering," in *Mathematical Morphology and Its Applications to Image and Signal Processing (Lecture Notes in Computer Science)*, vol. 6671, P. Soille, M. Pesaresi, and G. Ouzounis, Eds. Berlin, Germany: Springer-Verlag, 2011, pp. 405–416.
- [34] F. Dacheille, K. Kreeger, B. Chen, I. Bitter, and A. Kaufman, "High-quality volume rendering using texture mapping hardware," in *Proc. EUROGRAPHICS/SIGGRAPH Workshop Graph. Hardw.*, 1998, pp. 69–76.
- [35] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1999.



computing.

Andrei C. Jalba received the B.Sc. and M.Sc. degrees in applied electronics and information engineering from the Politehnica University of Bucharest, Bucharest, Romania, in 1998 and 1999, respectively, and the Ph.D. degree from the Institute for Mathematics and Computing Science, University of Groningen, Groningen, The Netherlands, in 2004. He is currently an Assistant Professor with the Eindhoven University of Technology, Eindhoven, The Netherlands. His research interests include computer graphics and vision, shape processing, and parallel



the Institute for Visualization and Interactive Systems, University of Stuttgart, Stuttgart, Germany. He returned as a Post-Doctoral Researcher with the Institute for Mathematics and Computing Science, University of Groningen, in 2006. Since 2008, he has been with the Department of Mathematics and Computer Science, Eindhoven University of Technology. His research interests include biological and biomedical data visualization and image processing.

Michel A. Westenberg is currently an Assistant Professor in visualization with the Eindhoven University of Technology, Eindhoven, The Netherlands. He received the Ph.D. degree in mathematics and natural sciences and the M.Sc. degree in computing science from the University of Groningen, Groningen, The Netherlands, in 2001 and 1996, respectively. He was a recipient of a Humboldt Research Fellowship by the Alexander von Humboldt Foundation in 2004, which funded a post-doctoral position with



was involved in image processing and tomographic reconstruction. He was appointed as an Associate Professor and a Full Professor with the Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, Groningen, The Netherlands, in 1992 and 2003, where he is currently the Chair of Scientific Visualization and Computer Graphics. His research interests include mathematical morphology, biomedical visualization, neuroimaging, and bio/neuroinformatics.

Jos B. T. M. Roerdink (SM'03) received the M.Sc. degree in theoretical physics from the University of Nijmegen, Nijmegen, The Netherlands, in 1979, and the Ph.D. degree from the University of Utrecht, Utrecht, The Netherlands, in 1983. He was a Post-Doctoral Fellow with the University of California at San Diego, San Diego, CA, USA, in 1983 and 1985, where he was involved in the area of stochastic processes. He was with the Centre for Mathematics and Computer Science, Amsterdam, The Netherlands, from 1986 to 1992, where he