Interactive Volume Segmentation with Differential Image Foresting Transforms

Alexandre X. Falcão and Felipe P. G. Bergo

Abstract— **The absence of object information very often asks for considerable human assistance in medical image segmentation. Many interactive 2D and 3D segmentation methods have been proposed, but their response time to user's actions should be considerably reduced to make them viable from the practical point of view. We circumvent this problem in the framework of the *image foresting transform* (IFT)— a general tool for the design of image operators based on connectivity— by introducing a new algorithm (DIFT) to compute sequences of IFTs in a differential way. We instantiate the DIFT algorithm for watershed-based and fuzzy-connected segmentations under two paradigms (single-object and multiple-object) and evaluate the efficiency gains of both approaches with respect to their linear-time implementation based on the non-differential IFT. We show that the DIFT algorithm provides efficiency gains from 10 to 17, reducing the user's waiting time for segmentation with 3D visualization on a common PC from 19–36 seconds to 2–3 seconds. We also show that the multiple-object approach is more efficient than the single-object paradigm for both segmentation methods.**

Index Terms— **User-assisted image segmentation, medical imaging, 3D visualization, image analysis, image foresting transform, watershed transform, fuzzy connectedness, graph-search algorithms, minimum-cost path forest, Dijkstra's algorithm.**

I. Introduction

Despite of the progress over the last two decades, image segmentation remains one of the main challenges in medical image analysis [1]. Volume segmentation and visualization at interactive speeds are highly desirable for routine use in clinical settings [2], [3]. On the other hand, accurate segmentation methods are likely to be complex, computationally expensive, application-dependent, and often require considerable human interaction [4], [5], [6], [7]. These aspects make very important to reduce as much as possible the total time spent by the user for interactive volume segmentation [8], [9], [10], [11], [12]. In this work, we circumvent the problem in the framework of the *image foresting transform* (IFT)— a general tool for the design, implementation, and evaluation of image processing operators based on connectivity [13]— by computing sequences of IFTs in a differential way. The new algorithm is called *differential image foresting transform* (DIFT).

Given a sequence of tomographic slices forming a data volume in the 3D discrete space (a *scene*), the IFT defines a *minimum-cost path forest* in a graph whose nodes are the scene voxels and whose arcs are defined by an *adjacency relation* between voxels. The cost of a path in this graph is determined by an application-specific *path-cost function*, which usually depends on local scene properties along the path— such as brightness, gradient, and voxel position. The roots of the forest are drawn from a given set of *seed voxels*. For suitable path-cost functions, the IFT considers all possible paths from the seed set to each voxel and assigns the voxel to the tree whose root reaches it through a path of minimum-cost, in such a way that the union of all paths forms an oriented forest, spanning the whole scene. It outputs three attributes for each voxel: its predecessor in the optimum path, the cost of that path, and the corresponding root (or some label associated with it). A great variety of powerful image operators can be implemented by simple local processing of these attributes [14], [15], [16], [17], [18].

For given segmentation task, the user assigns a distinct label to each object and selects at least one seed voxel per object (including background). The IFT outputs an optimum-path forest where each object is represented by a set of trees rooted at seeds with the same label. Very often the user needs to correct segmentation by adding new seeds and/or removing roots (trees), forming a sequence of IFTs. Instead of computing one IFT from the beginning for each instance of seed set, the DIFT algorithm updates the segmentation results in a differential way, taking time proportional to the number of voxels in the modified regions of the scene (see Figure 1).

Classical approaches for image segmentation, such as those based on the watershed transform [19], [20], [21] and fuzzy connectivity [22], [23], [7], [24], can be more efficiently implemented by simple choice of path-cost function. For example, linear-time implementations of the watershed transform [14], [25], [19] take about 33 seconds per iteration to process 256^3 voxels, running on an 1.5-GHz Pentium-4 PC. This performance obviously compromises the immediacy of response to the user's actions, making interactive segmentation extremely tedious, and thus impractical. We have proposed DIFT algorithms that reduce the computational time of the watershed transform by factors from 2 to 13 [26], [27]. The present paper is an extended version of the work reported in [27], which includes more details about the method, a path-cost function for fuzzy-connected segmentation, and several experiments involving the watershed-based and fuzzy-

Manuscript received October 31, 2003; revised February 10, 2004. *Asterisk indicates corresponding author.*

*A.X. Falcão is with the Institute of Computing, University of Campinas, Av. Albert Einstein, 1251, CEP 13084-851, Campinas, SP, Brazil (e-mail: afalcao@ic.unicamp.br).

F.P.G. Bergo is with the Institute of Computing, University of Campinas, Campinas, SP, Brazil.

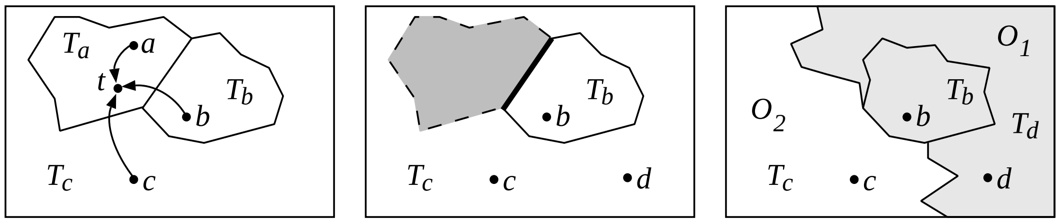

Fig. 1. Image segmentation by DIFT. (a) An optimum-path forest with trees $\{T_a, T_b, T_c\}$ rooted at pixels a, b, and c. Pixel t is assigned to T_a since a is its "closest" seed. (b) T_a is marked for removal and d is added to the seed set. The dispute will involve b, c, and d, where b and c are represented by the frontier pixels between T_a and T_b (bold line— pixels in T_b) and between T_a and T_c (dashed line— pixels in T_c), respectively. (c) The resulting forest $\{T_b, T_c, T_d\}$ where T_b and T_d have been assigned to object O_1 and T_c to object O_2.

connected approaches.

Section II presents the basic concepts and a formal definition of the IFT. The DIFT and its algorithm are described in Section III. We instantiate the IFT in Section IV for watershed-based and fuzzy-connected segmentations. We have also developed an *interactive volume segmentation* tool, called IVS, which provides both approaches in the differential way with fast 3D visualization during segmentation [1]. The main features of the IVS software tool are described in Section V. Section VI reports the results of several experiments involving interactive 3D segmentation with visualization for the watershed-based and fuzzy-connected approaches. The conclusions and our current work on IFT-based image segmentation are presented in Section VII.

II. Background

A. Scene and adjacency relations

We call *scene* any sequence of tomographic slices that forms a data volume in $\mathbb{Z}^3$. More formally, a scene $\mathbf{I}$ is a pair $(\mathcal{I}, I)$ consisting of a finite set $\mathcal{I}$ of *voxels* (points in $\mathbb{Z}^3$), and a mapping I that assigns to each voxel t in $\mathcal{I}$ a value $I(t)$ in some arbitrary value space.

An *adjacency relation* $\mathcal{A}$ is an irreflexive binary relation between voxels of $\mathcal{I}$, which usually depends only on the relative position $t - s$ of the voxels, where $(s,t) \in \mathcal{A}$. Once the adjacency relation $\mathcal{A}$ has been fixed, the scene can be interpreted as a graph whose nodes are the scene voxels and whose arcs are the voxel pairs in $\mathcal{A}$. For example, one can take $\mathcal{A}$ to consist of all pairs of distinct voxels $(s,t) \in \mathcal{I} \times \mathcal{I}$ such that $d(s,t) \leq \rho$, where $d(s,t)$ denotes the Euclidean distance and ρ is a specified constant. More generally, we can take a finite subset $\mathcal{B}$ of $\mathbb{Z}^3 \setminus \{(0,0,0)\}$, for example $\mathcal{B} = \{(1,-1,2), (-1,1,-2), (-3,1,1)\}$, and define $\mathcal{A}$ as all pairs of voxels (s,t) where $t - s \in \mathcal{B}$.

B. Path, path costs, and optimum paths

A *path* is a sequence of voxels $\pi = \langle t_1, t_2, \ldots, t_k \rangle$ where $(t_i, t_{i+1}) \in \mathcal{A}$ for $1 \leq i \leq k-1$. The path is *trivial* if $k = 1$. We denote the *origin* t_1 and the *destination* t_k of π by $\mathrm{org}(\pi)$ and $\mathrm{dst}(\pi)$, respectively. If π and τ are paths such that $\mathrm{dst}(\pi) = \mathrm{org}(\tau) = t$, we denote by $\pi \cdot \tau$ the concatenation of the two paths, with the two joining instances of t merged into one.

[1]The IVS software tool is available at `http://www.lni.hc.unicamp.br/app/ivs`

We assume given an application-dependent function f that assigns to each path π a *path cost* $f(\pi)$ in some totally ordered set $\mathcal{V}$ of cost values, with maximum element denoted by $+\infty$. Usually, the path cost depends on local properties of the scene $\mathbf{I}$—such as brightness, gradient, and voxel position—along the path. A path π is *optimum* if $f(\pi) \leq f(\pi')$ for any other path π' with $\mathrm{dst}(\pi') = \mathrm{dst}(\pi)$, irrespective of its starting voxel. In that case, $f(\pi)$ is by definition the *cost* of voxel $t = \mathrm{dst}(\pi)$, denoted by $\hat{f}(t)$. Note that a trivial path is not necessarily optimum, since $f(\langle s \rangle)$ and $f(\langle t \rangle)$ may be such that a non-trivial path from s to t is cheaper than the trivial path $\langle t \rangle$.

The IFT requires that path-cost functions be *smooth*. That is, for any voxel $t \in \mathcal{I}$, there must exist an optimum path π ending at t which either is trivial, or has the form $\tau \cdot \langle s, t \rangle$ where (C1) $f(\tau) \leq f(\pi)$; (C2) τ is optimum; and (C3) for any optimum path τ' ending at s, $f(\tau' \cdot \langle s, t \rangle) = f(\pi)$. Observe that conditions (C1)–(C3) are not required to hold for *all* paths ending at t, but only for *some* path π that is optimum.

C. Spanning forests

A *predecessor map* is a function P that assigns to each voxel t in $\mathcal{I}$ either some other voxel in $\mathcal{I}$, or a distinctive marker *nil* not in $\mathcal{I}$ — in which case t is said to be a *root* of the map. A *spanning forest* is a predecessor map which contains no cycles. For every voxel $t \in \mathcal{I}$, a spanning forest P defines a path $P^*(t)$ recursively as $\langle t \rangle$ if $P(t) = \mathit{nil}$, and $P^*(s) \cdot \langle s, t \rangle$ if $P(t) = s \neq \mathit{nil}$. We denote by $P^0(t)$ the initial voxel of $P^*(t)$ (see Figure 2).

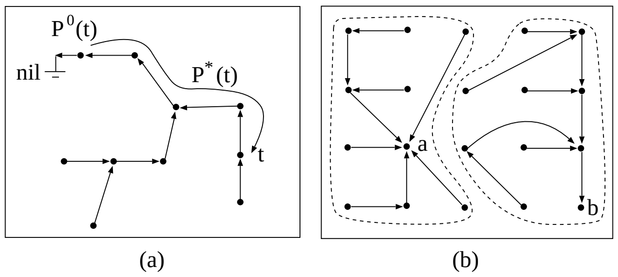

Fig. 2. (a) The main elements of a spanning forest. (b) A spanning forest with two trees, rooted at a and b.

D. The Image Foresting Transform

The *image foresting transform* (IFT) takes a scene $\mathbf{I}$, a smooth path-cost function f and an adjacency relation $\mathcal{A}$; and returns an *optimum-path forest*— a spanning forest P such that $P^*(t)$ is optimum for every voxel t. During this process, the IFT builds a *cost map* $C : \mathcal{I} \rightarrow \mathcal{V}$ such that $C(t) = f(P^*(t))$ and a *root map* $L : \mathcal{I} \rightarrow \mathcal{I}$ such that $L(t) = P^0(t)$. Once the IFT has computed the optimum path forest, $C(t) = \hat{f}(t)$ and $L(t) = P^0(t)$ for every $t \in \mathcal{I}$.

Figure 3 illustrates an example of the detection of the *regional minima* of an image. A *regional minimum* of an image $\mathbf{I} = (\mathcal{I}, I)$ is a maximal connected set $\mathcal{X} \subseteq \mathcal{I}$ of same gray level, where $I(s) < I(t)$ for any $s \in \mathcal{X}$, $t \notin \mathcal{X}$ and $(s,t) \in \mathcal{A}$ [28]. It can be detected as the roots of the forest with a path-cost function f_{ini}, defined as

$$\begin{aligned} f_{\mathrm{ini}}(\langle t \rangle) &= I(t), \text{ for all } t \in \mathcal{I}, \\ f_{\mathrm{ini}}(\pi \cdot \langle s,t \rangle) &= \begin{cases} f_{\mathrm{ini}}(\pi), & \text{if } I(s) \le I(t), \\ +\infty, & \text{otherwise.} \end{cases} \end{aligned} \quad (1)$$

E. Seed voxels

We wish to predefine a smooth path-cost function f but constrain the search to paths that start in a given set $\mathcal{S} \subseteq \mathcal{I}$ of *seed voxels*. This is equivalent to define a new path-cost function $f^{\mathcal{S}}(\pi)$, which is equal to $f(\pi)$ when $\mathrm{org}(\pi) \in \mathcal{S}$, and $+\infty$ otherwise. Unfortunately, $f^{\mathcal{S}}(\pi)$ is not necessarily smooth [13]. In order to guarantee smoothness for arbitrary seed sets in $\mathcal{I}$, we shall consider only a subclass of smooth cost functions, called *monotonic incremental* (MI), which satisfy

$$\begin{aligned} f(\langle t \rangle) &= h(t), \\ f(\pi \cdot \langle s,t \rangle) &= f(\pi) \odot (s,t), \end{aligned} \quad (2)$$

where $h(t)$ is an arbitrary handicap cost, π is any path ending at s, and $\odot : \mathcal{V} \times \mathcal{A} \rightarrow \mathcal{V}$ is a binary operation that satisfies (M1) $x' \ge x \Rightarrow x' \odot (s,t) \ge x \odot (s,t)$ and (M2) $x \odot (s,t) \ge x$, for any $x, x' \in \mathcal{V}$ and any $(s,t) \in \mathcal{A}$. The seed set restriction is equivalent to setting $h(t) = +\infty$ for voxels $t \notin \mathcal{S}$. Examples are the *additive* f_{sum} and the *max-arc* $f_{\max}$ path-cost functions:

$$\begin{aligned} f_{\mathrm{sum}}(\langle t \rangle) &= h(t), \\ f_{\mathrm{sum}}(\pi \cdot \langle s,t \rangle) &= f_{\mathrm{sum}}(\pi) + w(s,t), \quad (3) \\ f_{\max}(\langle t \rangle) &= h(t), \\ f_{\max}(\pi \cdot \langle s,t \rangle) &= \max\{f_{\max}(\pi), w(s,t)\}, \quad (4) \end{aligned}$$

where $h(t)$ is a fixed *handicap cost* for any paths starting at voxel t and $w(s,t)$ is a fixed weight (non-negative in the case of f_{sum}) assigned to the arc (s,t).

Note that, since a non-trivial path from s to t may be cheaper than the trivial path $\langle t \rangle$, there is no guarantee that a seed t will become a root of the forest. In such a case t becomes ineffective.

III. The Differential Image Foresting Transform

The key idea in the DIFT is to allow seed addition and removal of optimum-path trees in a differential way. When a voxel is added to the seed set, it may define a new optimum-path tree by invading the influence zones of other roots. When an optimum-path tree is removed from the forest, its voxels become available for a new dispute among the remaining roots. Tree removal is done by marking any of its voxels. We call $\mathcal{M} \subset \mathcal{I}$ the set of *marking voxels*, which belong to the optimum-path trees to be removed. Figure 4 illustrates an example of image segmentation using the DIFT.

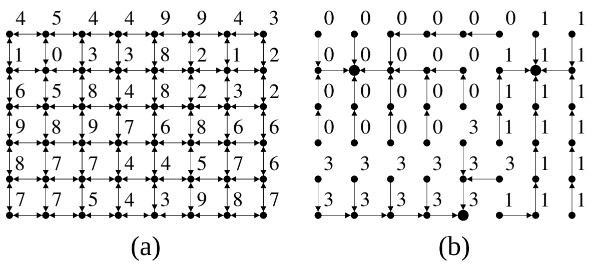

Fig. 3. (a) An image graph with 4-connected adjacency, where the integers are the image values $I(t)$. (b) An optimum-path forest for the path-cost function f_{ini}, where the roots of the forest are represented by bigger dots and the integers are the cost values.

Initially $\mathcal{M}$ is empty, $C(t) = +\infty$, $L(t) = t$, and $P(t) = nil$, $\forall t \in \mathcal{I}$. The DIFT algorithm is described below.

Algorithm 1: DIFT

```
INPUT:      Scene I, cost map C, root map L, predecessor map
            P, MI path-cost function f, adjacency relation A,
            set S' of new seed voxels, and set M of marking
            voxels;
OUTPUT:     C, L and P;
AUXILIARY:  Priority queue Q.
1.  Set $Q \leftarrow \emptyset$;
2.  $(C, P, \mathcal{F}) \leftarrow$DIFT-TreeRemoval$(C, L, P, \mathcal{A}, \mathcal{M})$;
3.  $\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{S}'$;
4.  While $\mathcal{S}'$ is not empty, do
5.      Remove any $t$ from $\mathcal{S}'$;
6.      If $f(\langle t \rangle) < C(t)$ then set $C(t) \leftarrow f(\langle t \rangle)$, $L(t) \leftarrow t$,
        $P(t) \leftarrow nil$, and $\mathcal{F} \leftarrow \mathcal{F} \cup \{t\}$;
7.  While $\mathcal{F}$ is not empty, do
8.      Remove any $t$ from $\mathcal{F}$ and insert $t$ in $Q$;
9.  While $Q$ is not empty, do
10.     Remove a voxel $s$ from $Q$, such that $C(s)$ is minimum;
11.     For each $t$ such that $(s,t) \in \mathcal{A}$, do
12.         Compute $cost \leftarrow f(P^*(s) \cdot \langle s,t \rangle)$;
13.         If $cost < C(t)$ or $P(t) = s$ then
14.             If $t \in Q$ then remove $t$ from $Q$;
15.             Set $P(t) \leftarrow s$, $C(t) \leftarrow cost$, $L(t) \leftarrow L(s)$, and
                insert $t$ in $Q$;
```

Algorithm 2: DIFT-TreeRemoval

```
INPUT:      Cost map C, root map L, predecessor map P,
            adjacency relation A, set M of marking voxels;
OUTPUT:     C, P, and set F of frontier voxels;
AUXILIARY:  FIFO Queue T and set R of roots of trees marked
            for removal.
1.  Set $\mathcal{R} \leftarrow \emptyset$, and $\mathcal{F} \leftarrow \emptyset$;
2.  While $\mathcal{M}$ is not empty, do
3.      Remove any $t$ from $\mathcal{M}$;
4.      Set $r \leftarrow L(t)$, $\mathcal{R} \leftarrow \mathcal{R} \cup \{r\}$;
5.      If $C(r) \neq +\infty$ then
6.          insert $r$ in $T$, set $C(r) \leftarrow +\infty$, $P(r) \leftarrow nil$;
7.  While $T$ is not empty, do
8.      Remove $s$ from $T$;
```

```
9.    For each $t$ such that $(s,t) \in \mathcal{A}$, do
10.       If $P(t) = s$ then
11.           set $C(t) \leftarrow +\infty$, set $P(t) \leftarrow nil$, insert $t$ in $T$;
12.       Else if $L(t) \notin \mathcal{R}$ then $\mathcal{F} \leftarrow \mathcal{F} \cup \{t\}$;
```

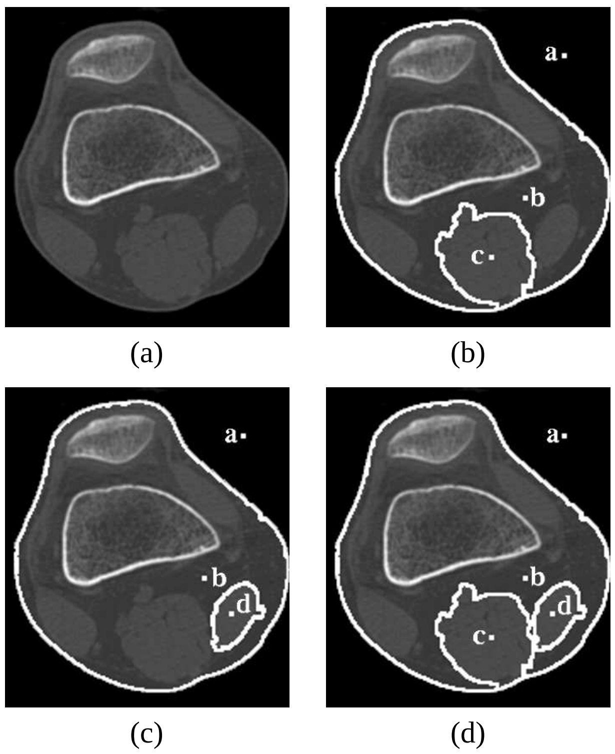

Fig. 4. An example of image segmentation by DIFT with $f_{\max}$. (a) CT image of a knee. (b) Initial forest with 3 roots, $\{a, b, c\}$. (c) State after differential computation with addition of seed d and removal of c (DIFT with $\mathcal{S}' = \{d\}$ and $\mathcal{M} = \{c\}$). (d) After the re-addition of c (DIFT with $\mathcal{S}' = \{c\}$ and $\mathcal{M} = \emptyset$).

Algorithm 1 above is essentially Dijkstra's algorithm [29], [13], slightly modified for multiple sources and to allow seed addition and tree removal in a differential way. An important difference between Algorithm 1 and the IFT algorithm reported in [13] is the *predecessor test* $P(t) = s$ in line 13. It guarantees that when a new seed is included in $\mathcal{S}'$, all voxels $t \in \mathcal{I}$ reachable from it by some optimum-path $\pi = P^*(s) \cdot \langle s, t \rangle$ with cost $f(\pi) \leq C(t)$ will be reevaluated (but when $f(\pi) = C(t)$, lines 14–15 are executed only if $P(t) = s$). Figure 5 illustrates the problem solved by the predecessor test. Let $\langle a, .., s, t, .., u \rangle$ be an optimum path from a to u in an optimum-path forest with two roots, a and b (Figure 5a). Consider in the next iteration a new seed c (Figure 5b), such that there is an optimum path $\langle c, .., s \rangle$ with $f(\langle c, .., s \rangle) < C(s)$ and $f(\langle c, .., s, t \rangle) = C(t)$. Without the predecessor test, t will not be inserted in Q when it is visited from s and the pixels along the whole suffix $\langle t, .., u \rangle$ may not be reevaluated at all, if they are not visited by other pixels rooted at c (Figures 5c–d). In such a case, even if $\langle c, .., s, t, .., u \rangle$ is an optimum path from c to u, the root map along the whole suffix will be pointing to the old root a instead of c, and the predecessor test corrects this problem. It is worth noticing that the predecessor test does not make Algorithm 1 fail when f is smooth. For $\langle c, .., s \rangle$ optimum (with $f(\langle c, .., s \rangle) < f(\langle a, .., s \rangle)$) and $f(\langle c, .., s, t \rangle) > f(\langle a, .., s, t \rangle)$, the predecessor test will cause the algorithm to consider $\langle c, .., s, t \rangle$ temporarily. However, due to the smoothness condition (C2), there must exist another $s' \neq s$ such that $\langle c, .., s', t \rangle$ is optimum (i.e. $f(\langle c, .., s', t \rangle) < f(\langle a, .., s, t \rangle)$). Otherwise f would not be smooth, because $\langle a, .., s, t \rangle$ would be an optimum path with a non-optimum prefix $\langle a, .., s \rangle$. When f is MI, condition (M1) ensures that, if $f(\langle c, .., s \rangle) < f(\langle a, .., s \rangle)$ then $f(\langle c, .., s, t \rangle) \leq f(\langle a, .., s, t \rangle)$.

Algorithm 2 resets cost and predecessor values of all voxels that belong to trees marked by voxels in $\mathcal{M}$ (thus removing those trees from the forest), and adds to the set $\mathcal{F}$ all voxels of non-removed trees which are adjacent to removed voxels. That is, all voxels $t \in \mathcal{I}$ such that $\exists r \in \mathcal{M}$, $L(r) = L(t)$, have their cost and predecessor values reinitialized to $+\infty$ and nil, respectively; and all voxels $t \in \mathcal{I}$, such that $(s, t) \in \mathcal{A}$, the tree of s was removed but the tree of t wasn't, are added to the set $\mathcal{F}$. Therefore, $\mathcal{F}$ contains non-reset voxels of the frontier between removed and non-removed trees. Since the frontier voxels have not been reset, they will only represent the propagation front of their roots in the dispute for the available voxels; i.e. they will not become new roots as illustrated in Figure 1. Some voxels in $\mathcal{F}$ may also have been selected as seeds, and therefore must be removed from $\mathcal{F}$ (line 3 of Algorithm 1).

In general, there may be many paths of minimum cost leading to a given voxel, and many optimum-path forests; only the voxel costs $\hat{f}(t)$ are uniquely defined. Some of these ambiguities are already resolved in Algorithm 1 by selecting the optimum-path which reaches a given voxel first and by using a tie-breaking policy to remove voxels from Q. Usually, the most convenient choice for image segmentation is to pick the voxel that entered in Q first, i.e. ties are broken by using the *first-in first-out* (FIFO) policy [13]. However, the previous relative position of the frontier voxels with the same cost in Q is not recovered in Algorithm 1. It is also possible to implement this in line 8, when a frontier voxel is inserted in Q, with additional information about the number of arcs in the optimum path to each frontier voxel. Apart of these ambiguities, Algorithm 1 provides seed addition and tree removal in a revertible way (an *undo* feature). All optimum-path forest are valid and the possible differences in the segmentation results seem not to be relevant in practice.

Efficient implementations of Dijkstra's algorithm, and so of the IFT algorithm, require computation time $O(|\mathcal{I}|)$ for sparse graphs ($|\mathcal{A}| \ll |\mathcal{I}|$) and integer path costs with limited increments [29], [13]. Since Algorithm 1 is differential, its running time will be proportional to the number of voxels in the modified regions of the scene— usually, a number much less than $|\mathcal{I}|$.

In order to use Algorithm 1 for image segmentation, we select seed voxels inside each object of interest and assign a same label to each object in such a way that each object is defined by an optimum-path sub-forest whose trees are rooted at voxels with the same label. For image segmentation, it only makes sense to use symmetric adjacency relations, since there is no established hierarchy relations among voxels to restrict the flow of propagation to any particular direction.

Object labeling can be efficiently computed in time $O(|\mathcal{I}|)$

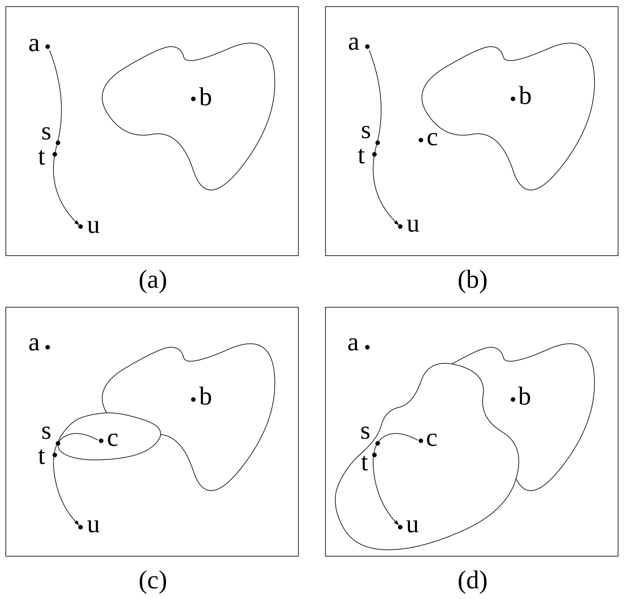

Fig. 5. (a) A valid optimum-path forest with two trees, rooted at pixels a and b, where $\langle a, .., s, t, .., u\rangle$ is an optimum path. (b) Seed c is added. (c) During the DIFT computation $\langle c, .., s\rangle$ is found as an optimum-path cheaper than $\langle a, .., s\rangle$, but $C(t) = f(\langle c, .., s, t\rangle)$. The predecessor test guarantees that all pixels whose optimum paths contained s, as for example the suffix $\langle t, .., u\rangle$, will be re-evaluated and assigned to root c. (d) A possible outcome of the DIFT in this example.

after each iteration of Algorithm 1, by assigning to each voxel $t \in \mathcal{I}$ the label $\lambda(L(t))$ of its root $L(t)$. A second alternative is to propagate $\lambda(t)$ of each seed $t \in \mathcal{S}'$ forming a label map, instead of the root map L. This on the other hand requires the use of the predecessor map P to find the root r of each voxel $t \in \mathcal{M}$ in lines 4 and 12 of Algorithm 2. A third and the most efficient solution, also the most memory costly, is to propagate $\lambda(t)$ in a separate label map L'. Since we wish to provide volume visualization after each iteration of Algorithm 1 (see Section V), and this already takes time $O(|\mathcal{I}|)$, we have chosen the first alternative where voxel labeling and splatting are computed at same time.

IV. Application to Volume Segmentation

Novel and classical segmentation operators can be created by defining different path-cost functions. We demonstrate the DIFT using two classical approaches: a *watershed-based segmentation* [19], [20] and a *fuzzy-connected segmentation* [22], [23].

A. Watershed-based segmentation

There is no unique and precise definition for a watershed transform in the literature [20], [19], [21]. It is informally described as a flooding process on an image surface, with a source of water at each seed pixel; a barrier (*watershed line*) is erected wherever two bodies of water coming from distinct sources meet. However, the position of the watershed lines is not precisely defined in many situations, such as on plateaus and when two or more bodies of water overflow into an adjacent basin at the same time. In the IFT approach, the position of the barrier is determined by the FIFO tie-breaking policy in Q. The FIFO policy usually leads to a fair (if not symmetrical) partition of plateaus and flooded basins among competing sources [13].

The watershed-based segmentation requires a gradient-like image where the objects of interest are delimited by ridges which are higher than any other hills inside or outside the objects. The water out-coming from a seed pixel of a basin will overflow into a neighboring basin when it reaches the minimum altitude between them, which is the maximum altitude along the water path to reach any pixel below that height in the neighboring basin. The extension of this approach to the three-dimensional space is straightforward and it can be modeled as a particular case of $f_{\max}$ (Equation 4) with at least one seed voxel at each object.

$$\begin{aligned} f_{\text{peak}}(\langle t\rangle) &= h(t), \\ f_{\text{peak}}(\pi \cdot \langle s, t\rangle) &= \max\{f_{\text{peak}}(\pi), I(t)\}, \end{aligned} \tag{5}$$

where $I(t)$ is the intensity of voxel t in a gradient-like scene $\mathbf{I}$, and $h(t) < I(t)$ if $t \in \mathcal{S}$, and $+\infty$ otherwise. Algorithm 1 will output in L the catchment basins of the watershed transform of I [20], [14], [13]. If we use $h(t) > I(t)$, then Algorithm 1 will output in L the catchment basins of the watershed transform of C, where C is the *superior morphological reconstruction* [28] of I from the *marker function* h. In this case, the IFT-based approach provides pre-filtering and segmentation simultaneously.

Note that there are many ways to create a gradient-like scene $\mathbf{I}$ by preprocessing the original scene [30]. We wish gradient-like scenes where the objects are represented by a minimal set of basins in order to make the segmentation more effective and to minimize the user involvement, by reducing the number of seeds required to complete segmentation. In this work, we have used a very simple and yet effective preprocessing approach based on the morphological gradient (see Section V).

B. Fuzzy-connected segmentation

For a given scene with objects of interest, the relative fuzzy-connected segmentation [23] considers the *strength of connectedness* between the voxels and each object, and assigns each voxel to the object whose strength of connectedness is maximum. Each object may be represented by at least one seed voxel o and each pair (s, t) of adjacent voxels in the scene defines a *local affinity* $\alpha_o(s, t)$ with that object. For example:

$$\alpha_o(s,t) = \exp\left(\frac{-(\frac{I(s)+I(t)}{2} - \mu_o)^2}{2\sigma_o^2}\right), \tag{6}$$

where $I(t)$ is the intensity of voxel t in the *input* scene, and μ_o and σ_o are the estimated mean and standard deviation for voxel intensities within the object represented by o. The effectiveness of the method requires an input scene with homogeneous distribution of brightness (Section V). The theoretical results are valid as long as these local affinities with respect to each object are combined into a *single affinity* $\alpha(s, t)$ per adjacent voxels [23]:

$$\alpha(s,t) = \max_{\forall o \in \mathcal{S}}\{\alpha_o(s,t)\}, \tag{7}$$

where $\mathcal{S} = \{o_1, o_2, \ldots, o_m\}$ represents a set of m objects with properties μ_{o_i} and σ_{o_i}.

The strength of connectedness of a path is defined as the minimum value among the affinities of all pairs of adjacent voxels along the path. That is, the weakest link between the origin voxel and the destination voxel of the path. The strength of connectedness between an object, represented by a seed o, and a voxel t is the maximum value among the strength of connectedness of all possible paths between o and t in the scene. Clearly, the fuzzy-connected segmentation can be modeled as an instance of the IFT, where each object may be represented by a subset of seed voxels and the cost function evaluates the complement of the strength of connectedness of a path. Therefore, we can define the cost function f_{fuzz} as a particular case of $f_{\max}$ (Equation 4) with at least one seed voxel at each object.

$$\begin{aligned} f_{\text{fuzz}}(\langle t \rangle) &= 0, \text{ if } t \in \mathcal{S}, \text{ or } +\infty \text{ otherwise.} \\ f_{\text{fuzz}}(\pi \cdot \langle s,t \rangle) &= \max\{f_{\text{fuzz}}(\pi), w(s,t)\}, \end{aligned} \tag{8}$$

where $w(s,t) = K * (1 - \alpha(s,t))$ is the complement of the affinity between the adjacent voxels s and t, and K is the maximum voxel intensity in the scene.

Note that, the watershed-based and fuzzy-connected segmentations use particular cases of $f_{\max}$. In this sense, the IFT allows a better understanding of the relation between these approaches.

V. Implementation

Volume visualization provides important feedback about the form and shape of the objects. Usually, volume visualization is computed after segmentation when it is too late for the user to correct eventual segmentation mistakes. We illustrate the breakthrough of the DIFT by providing in the IVS software tool the watershed-based and fuzzy-connected approaches with fast 3D visualization during segmentation. Verification and corrections can also be done based on cut-plane views of the scene, as the users are used to do.

We have not concentrated on improving preprocessing yet, since our aim for the time being is to present a general framework for efficient interactive volume segmentation. However, it must be noted that preprocessing is paramount to make both methods more effective. For example, the use of isotropic scenes increase effectiveness. We also adopt the following sequence to create a gradient-like scene for the watershed approach. A Gaussian stretching is applied to the original scene $\mathbf{I}$ to increase the contrast between the interior and the exterior of the object. It outputs a scene $\mathbf{J} = (\mathcal{I}, J)$ with

$$J(t) = K \exp\left(-\frac{(I(t)-\mu)^2}{2\sigma^2}\right), \tag{9}$$

where μ and σ are the estimated mean and standard deviation for voxel intensities inside the object, and K is the maximum voxel intensity in the scene. A smoothing filter is sometimes used to reduce noise in $\mathbf{J}$, and a morphological gradient is computed to create the input scene $\mathbf{G} = (\mathcal{I}, G)$ for the watershed as follows.

$$G(s) = \max_{\forall t \in \mathcal{A}(s)} \{J(t)\} - \min_{\forall t \in \mathcal{A}(s)} \{J(t)\}, \tag{10}$$

where $\mathcal{A}(s)$ is the set of voxels adjacent to a voxel s, including s. Typically $\mathcal{A}$ defines arcs to the 6 closest neighbors in 3D.

It seems that the most indicated preprocessing for fuzzy-connected segmentation is the one proposed in [31]. Unfortunately, it is very time consuming and we could not get good results with some simplifications of the original method. Therefore, we usually select the parameters (μ_o, σ_o) for each object with no preprocessing, and sometimes, we use a median filter to increase the homogeneity inside the objects.

In both approaches, the user selects seed voxels in each object (including background) by clicking or dragging the mouse over saggital, coronal or transversal slices of the scene viewed in orthogonal projection (Figure 6a). The result of segmentation iterations can be verified in the orthogonal projections of the scene (Figure 6b), in surface rendering (Figure 6c), and in 3D views that combine both scene and anatomic information (Figure 6d). Each object is identified by a distinct color (label) in the user interface, but here we show only the gray-shaded renditions. The root map L allows direct access to the color of the corresponding object from any voxel in the scene. Fast 3D visualization requires efficient normal vector estimation, object labeling and orthogonal voxel splatting on a viewing plane. Orthogonal voxel coloring and splatting is computed in *front-to-back* with respect to the viewing direction. The user may add new seeds and/or select trees to be removed by clicking or dragging the mouse in any of the three ways of visualization (e.g. Figure 6e), and the DIFT algorithm corrects the segmentation in a few seconds (Figure 6f). The correction process can be repeated as many times until the user is satisfied with the segmentation result.

VI. Experiments and results

We selected 10 MR scenes of the head from 10 subjects with no known anomalies (controls) for our experiments. The scenes were acquired with voxel size $0.98 \times 0.98 \times 1.00$ mm and T1 modality with 12-bit quantization. All scenes were linearly interpolated to an isotropic voxel size of $0.98 \times 0.98 \times 0.98$ mm and clipped to exclude non-interest regions for our segmentation purposes (empty regions above the head and on the sides of the head, and the region below the cerebellum). Table I presents the resulting clipped scenes used in all experiments.

All experiments were performed on a 1100 MHz Athlon PC with 384 MB RAM, running the Linux operating system. The experiments were divided in two categories: single-object segmentation and multiple-object segmentation. We first evaluated the efficiency gain of Algorithm 1 with respect to the non-differential IFT algorithm [13] for single-object segmentation, using the watershed-based and fuzzy-connected approaches. In some situations, multiple-object segmentation is possible with the same preprocessing procedure. We then evaluated the efficiency gain of multiple-object segmentation with respect to single-object segmentation.

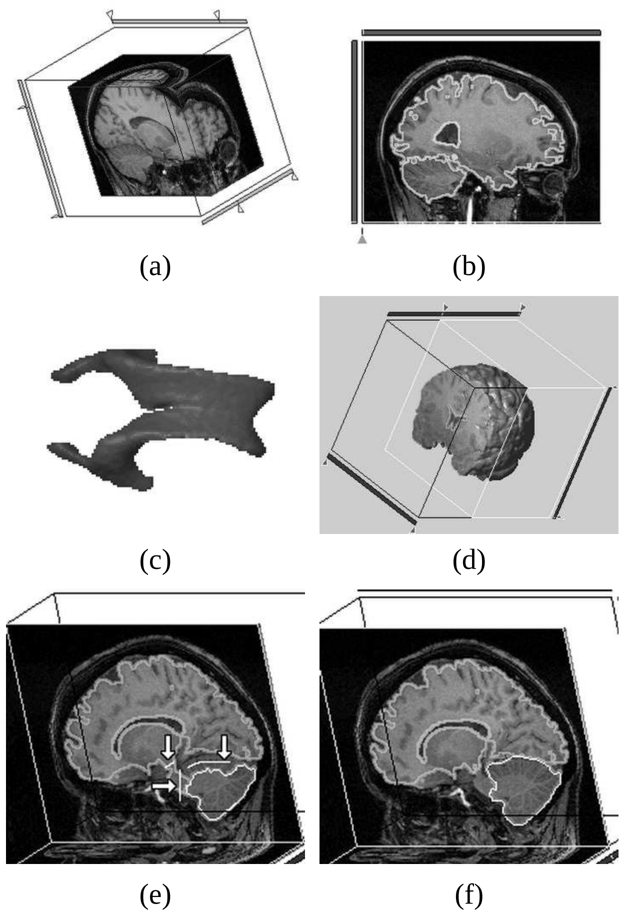

Fig. 6. Examples of the views available to the user in the IVS software tool. (a) Orthogonal cut-plane view before segmentation. Segmentation state in (b) cut-plane view, (c) surface rendering, and (d) combination of cut-plane view and surface rendering. (e) Example of seed selection. (f) Resulting segmentation with the seeds selected in (e).

TABLE I
DIMENSIONS OF THE DATA SETS USED IN THE EXPERIMENTS.

Subject	Dimensions	# of Voxels
1	$253\times161\times161$	6,558,013
2	$225\times163\times166$	6,088,050
3	$228\times175\times159$	6,344,100
4	$225\times175\times173$	6,811,875
5	$230\times162\times159$	5,924,340
6	$230\times168\times165$	6,375,600
7	$229\times181\times217$	8,994,433
8	$234\times183\times195$	8,350,290
9	$216\times161\times152$	5,285,952
10	$213\times164\times160$	5,589,120

TABLE II
RESULTS OF THE EXPERIMENTS WITH SINGLE-OBJECT SEGMENTATION (PART 1).

Object-method	Mean number of corrections	Mean time for the first DIFT	Mean time for each correction	Mean time for response
LV-DFC	22.4	32.64s	1.88s	2.94s
CN-DFC	44.9	34.78s	2.42s	3.46s
CBL-DWS	20.4	17.86s	1.79s	2.85s
BRN-DWS	18.4	20.03s	1.84s	2.95s

A. Single-object segmentation

We selected 4 objects for segmentation: lateral ventricles (LV), the caudate nuclei (CN), the cerebellum (CBL) and the brain (BRN).

The LV were segmented using the *differential fuzzy-connected approach* (DFC) and with no preprocessing, which is possible due to the high contrast found between the ventricles and the surrounding brain tissues. The CN were also segmented with the DFC, but the clipped scenes were preprocessed with a $3\times3\times3$ median filter. The CBL and the BRN were segmented with the *differential watershed approach* (DWS). The preprocessing sequence described in Section V was used for both objects, but we did not use the smoothing filter for the BRN. Tables II and III report the results of the experiments for each object and segmentation method. The time measurements presented in both tables are averaged over the time measurements obtained during the segmentation of the 10 clipped scenes.

Table II presents the average number of corrections, the mean CPU time for computing the first DIFT, the mean CPU time for computing each correction (i.e., each subsequent DIFT), and the mean response time for each correction (i.e., the user's waiting time on average, which includes the CPU time for correction and 3D visualization). The computation of the first DIFT takes more time in the DFC approach than in the DWS approach (3rd column), due to the cost of computing Equations 6 and 7. This difference becomes much less noticeable in the subsequent differential corrections (4th column), since the DIFT takes time proportional to the number of voxels in the modified regions of the scene and most corrections modify only small regions. The DIFT considerably reduces the time for corrections to about 2 seconds (4th column). The user's waiting time for correction and 3D visualization is also reduced to about 3 seconds (5th column). Only 4 out of the 388 corrections during the CBL-DWS and BRN-DWS sessions and 16 out of the 673 corrections during the LV-DFC and CN-DFC sessions had response times above 5 seconds.

If we divide the 3rd column by the 4th column, we may conclude that Algorithm 1 performs corrections from 9.98 to 17.36 times faster than the IFT algorithm [13] (see the 2nd column of Table III), being the efficiency gain higher for the DFC approach than for the DWS approach. Table III also shows the mean total CPU time to complete segmentation (3rd column), which includes the total CPU time for preprocessing, corrections and renditions, and the mean total time to complete segmentation (4th column), which includes the total CPU time and the time spent by the user choosing preprocessing parameters, doing verification and making corrections. Note that the user spends most of the time required to complete segmentation doing verification (68% to 75% of the total segmentation time). Assuming the user time is the same for the IFT and the DIFT approaches, the total time to complete segmentation with the DIFT becomes from 1.61 to 2.42 times faster than with the IFT (5th column). One may say that this efficiency gain in total time weakens the advantages of the DIFT over the IFT, but the user would never wait 19–36 seconds to get the results of each correction (3rd column

TABLE III
RESULTS OF THE EXPERIMENTS WITH SINGLE-OBJECT SEGMENTATION (PART 2).

Object-method	Efficiency gain in CPU time for corrections	Mean total CPU time for segmentation	Mean total time for segmentation	Efficiency gain in total time
LV-DFC	17.36	2min 58s	10min 53s	2.06
CN-DFC	14.37	5min 24s	17min 03s	2.42
CBL-DWS	9.98	2min 40s	8min 59s	1.61
BRN-DWS	10.89	2min 19s	9min 12s	1.61

of Table II plus the mean rendition time, which is about 1 second). Therefore, we may conclude that the DIFT really makes interactive volume segmentation viable from the user's practical point of view.

It is important to note that the mean total time to complete segmentation of a single object varies from 8 minutes and 59 seconds to 17 minutes and 3 seconds (Table III, 4th column). This is certainly an advantage as compared to the manual segmentation of scenes with 152–217 slices (Table I). Examples of the resulting segmentations are shown in Figure 7.

B. Multiple-object segmentation

We concluded from the previous experiments that most of the segmentation time is spent by the user doing verification. Multiple-object segmentation allows verification and correction of many objects at the same time, so it is expected to reduce the mean total time for segmentation.

We selected the lateral ventricles (LV), the cerebellum (CBL), and the brain without lateral ventricles (BRN) to be segmented with the DWS approach. The LV and the caudate nuclei (CN) were selected for segmentation with the DFC approach. We used the preprocessing sequence described in Section V for the DWS and no preprocessing for the DFC.

Table IV shows the results of the experiments with multiple-object segmentation. The mean total time to segment each object was reduced from about 9 minutes (Table III, 4th column) to 6 minutes (Table IV, 3rd column) in the DWS approach. In the DFC approach, the mean total time to segment the LV remained about the same, and the mean total time to segment the CN was reduced from 17 to 11 minutes (see 4th column of Table III and 3rd column of Table IV).

The efficiency gain of the multiple-object approach over the single-object approach (Table IV, 4th column) can be calculated dividing the total time required to segment all objects individually (Table III, 4th column) by the total time required to segment all objects simultaneously. We can conclude that multiple-object segmentation becomes more efficient as the number of objects segmented simultaneously increases, and even the segmentation of only 2 objects is already 30% faster than the equivalent single-object segmentation.

VII. CONCLUSIONS

We described a general framework for interactive volume segmentation, where novel and classical approaches can be efficiently implemented by simple choice of parameters of the IFT. We introduced a variant of the IFT algorithm which computes sequences of image foresting transforms in a differential way. The DIFT algorithm was instantiated for watershed-based (DWS) and fuzzy-connected (DFC) segmentations and we showed several experiments with both approaches.

The DIFT algorithm performed segmentation corrections from 10 to 17 times faster than the IFT algorithm, reducing the response time for interactive segmentation with 3D visualization from 19–36 seconds to 2–3 seconds. This makes interactive volume segmentation in common PCs viable from the user's practical point of view. This is certainly a remarkable result, especially in the case of the fuzzy-connected segmentation where the IFT-based algorithm is already considerably more efficient than the one presented in [23], since similar advantage of a linear-time implementation was shown in [24] with respect to the absolute fuzzy-connected segmentation [22]. We also evaluated the mean total time for segmentation using DWS and DFC under two paradigms, single-object and multiple-object. Since most of the segmentation time is spent by the user doing verification, the multiple-object approach was from 30% to 53% more efficient than the single-object approach.

Clearly, user verification is a time-consuming task with respect to the total time for segmentation, but necessary for accuracy in medical imaging. We are currently developing a more user-friendly interface to reduce the verification time. We are also trying to improve preprocessing for MR images of the head and investigating methods to generate seeds automatically. We believe that a more adequate preprocessing and automatic seed selection will significantly reduce the number of corrections, and then minimize the user's involvement during segmentation.

TABLE IV
RESULTS OF THE EXPERIMENTS WITH MULTIPLE-OBJECT SEGMENTATION.

Objects/ Method	Mean total CPU time per object	Mean total time per object	Mean efficiency gain over single-object approach
LV-CN/DFC	3min 13s	10min 42s	1.30
LV-CBL-BRN/DWS	1min 56s	6min 19s	1.53

ACKNOWLEDGMENTS

We thank the Dept. of Neurology, University of Campinas for the medical images used in the experiments. This work was partially supported by CNPq (Procs. 302966/02-1) and CAPES.

REFERENCES

[1] J. S. Duncan and N. Ayache, "Medical image analysis: Progress over two decades and the challenges ahead," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 85–105, 2000.

[2] K. H. Höhne and W. A. Hanson, "Interactive 3D-segmentation of MRI and CT volumes using morphological operations," *Journal of Computer Assisted Tomography*, vol. 16, no. 2, pp. 285–294, 1992.

[3] G. Kühne, C. Poliwoda, C. Reinhart, T. Günther, J. Hesser, and R. Männer, "Interactive segmentation and visualization of volume data sets," in *IEEE Proceedings of Visualization 97, Late Breaking Hot Topics*, Phoenix, AZ, 1997, pp. 9–12.

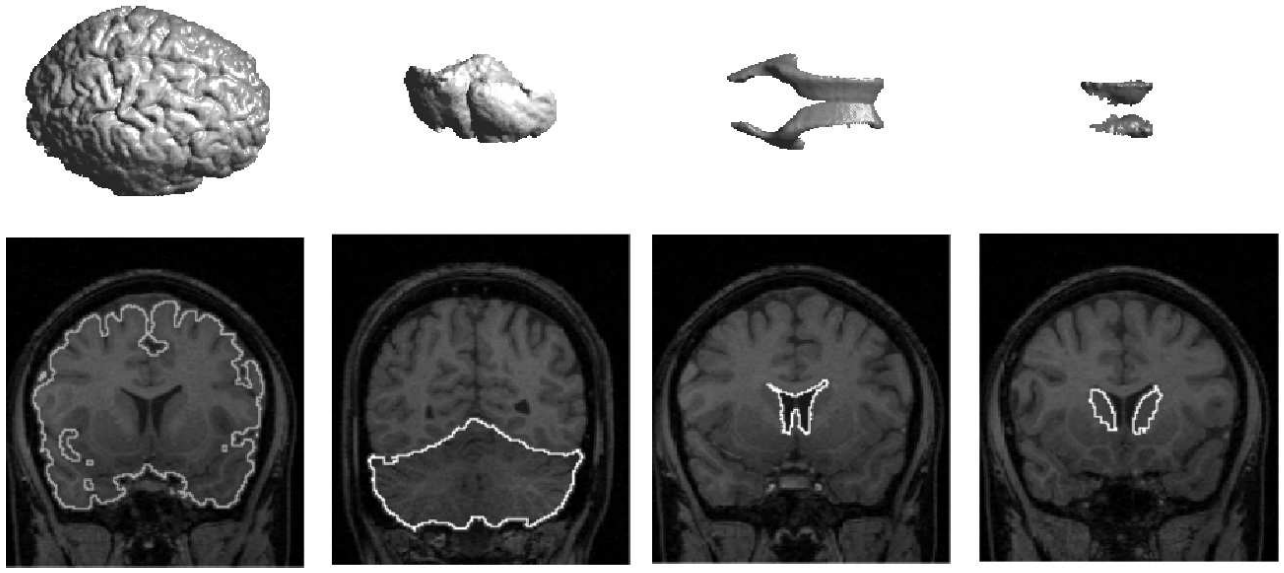

Fig. 7. Example of segmentation obtained in the single-object experiments. Left to right: BRN-DWS, CBL-DWS, LV-DFC and CN-DFC. Top row: surface renditions of the segmented objects. Bottom row: sample coronal outlines of each object.

[4] A. X. Falcão, J. K. Udupa, S. Samarasekera, S. Sharma, B. E. Hirsch, and R. A. Lotufo, "User-steered image segmentation paradigms: Live-wire and live-lane," *Graphical Models and Image Processing*, vol. 60, no. 4, pp. 233–260, Jul 1998.

[5] T. McInerney and D. Terzopoulos, "T-snakes: Topology adaptive snakes," *Medical Image Analysis*, vol. 4, pp. 73–91, 2000.

[6] H. G. Schnack, H. E. HulshoffPol, W. F. C. Baaré, M. A. Viergever, and R. S. Kahn, "Automatic segmentation of the ventricular system from MRI images of the human brain," *NeuroImage*, vol. 14, pp. 95–104, 2001.

[7] T. Lei, J. K. Udupa, P. K. Saha, and D. Odhner, "Artery-vein separation via MRA - An image processing approach," *IEEE Trans. on Medical Imaging*, vol. 20, no. 8, 2001.

[8] A. X. Falcão, J. K. Udupa, and F. K. Miyazawa, "An ultra-fast user-steered image segmentation paradigm: Live-wire-on-the-fly," *IEEE Trans. on Medical Imaging*, vol. 19, no. 1, pp. 55–62, Jan 2000.

[9] A. X. Falcão and J. K. Udupa, "A 3D generalization of user-steered live wire segmentation," *Medical Imaging Analysis*, vol. 4, no. 4, pp. 389–402, Dec 2000.

[10] W. E. Higgins and E. J. Ojard, "Interactive morphological watershed analysis for 3D medical images," *Computerized Medical Imaging and Graphics*, vol. 17, no. 4/5, pp. 387–395, 1993.

[11] G. Bueno, O. Musse, F. Heitz, and J. P. Armspach, "Three-dimensional segmentation of anatomical structures in MR images on large data bases," *Magnetic Resonance Imaging*, vol. 19, pp. 73–88, 2001.

[12] A. Schenk, G. Prause, and H. O. Peitgen, "Efficient semiautomatic segmentation of 3D objects in medical images," *Lecture Notes in Computer Science*, vol. 1935, pp. 186–195, 2000.

[13] A. X. Falcão, J. Stolfi, and R. A. Lotufo, "The image foresting transform: Theory, algorithms and applications," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 19–29, Jan 2004.

[14] R. A. Lotufo and A. X. Falcão, "The ordered queue and the optimality of the watershed approaches," in *Mathematical Morphology and its Applications to Image and Signal Processing*. Kluwer, Jun 2000, vol. 18, pp. 341–350.

[15] A. X. Falcão, B. S. da Cunha, and R. A. Lotufo, "Design of connected operators using the image foresting transform," in *Proc. of SPIE on Medical Imaging*, vol. 4322, Feb 2001, pp. 468–479.

[16] A. X. Falcão, L. F. Costa, and B. S. da Cunha, "Multiscale skeletons by image foresting transform and its applications to neuromorphometry," *Pattern Recognition*, vol. 35, no. 7, pp. 1571–1582, Apr 2002.

[17] R. A. Lotufo, A. X. Falcão, and F. Zampirolli, "IFT-Watershed from gray-scale marker," in *Proc. of XV Brazilian Symp. on Computer Graphics and Image Processing*. IEEE, Oct 2002, pp. 146–152.

[18] R. S. Torres, A. X. Falcão, and L. F. Costa, "A graph-based approach for multiscale shape analysis," *Pattern Recognition*, 2004, to appear.

[19] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, Jun 1991.

[20] S. Beucher and F. Meyer, "The morphological approach to segmentation: The watershed transformation," in *Mathematical Morphology in Image Processing*. Marcel Dekker, 1993, ch. 12, pp. 433–481.

[21] J. B. T. M. Roerdink and A. Meijster, "The watershed transform: Definitions, algorithms and parallelization strategies," *Fundamenta Informaticae*, vol. 41, pp. 187–228, 2000.

[22] J. K. Udupa and S. Samarasekera, "Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation," *Graphical Models and Image Processing*, vol. 58, pp. 246–261, 1996.

[23] P. K. Saha and J. K. Udupa, "Relative fuzzy connectedness among multiple objects: theory, algorithms, and applications in image segmentation," *Computer Vision and Image Understanding*, vol. 82, pp. 42–56, 2001.

[24] L. G. Nyúl, A. X. Falcão, and J. K. Udupa, "Fuzzy-connected 3D image segmentation at interactive speeds," *Graphical Models*, vol. 64, no. 5, pp. 259–281, 2002.

[25] P. Felkel, M. Bruckschwaiger, and R. Wegenkittl, "Implementation and complexity of the watershed-from-markers algorithm computed as a minimal cost forest," *Computer Graphics Forum (Eurographics)*, vol. 20, no. 3, pp. C26–C35, 2001.

[26] A. X. Falcão and F. P. G. Bergo, "The iterative image foresting transform and its application to user-steered 3D segmentation," in *Proc. of SPIE on Medical Imaging*, vol. 5032, Feb 2003, pp. 1464–1475.

[27] F. P. G. Bergo and A. X. Falcão, "Interactive 3D segmentation of brain MRI with differential watersheds," Institute of Computing, University of Campinas, Tech. Rep. IC-03-16, July 2003, available from http://www.ic.unicamp.br/ic-tr.

[28] L. Vincent, "Morphological grayscale reconstruction in image analysis," *IEEE Trans. on Image Processing*, vol. 2, no. 2, pp. 176–201, Apr 1993.

[29] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, 1993.

[30] J. M. Gauch, "Image segmentation and analysis via multiscale gradient watershed hierarchies," *IEEE Trans. on Image Processing*, vol. 8, no. 1, pp. 69–79, Jan 1999.

[31] P. Saha and J. K. Udupa, "Scale-based fuzzy connected image segmentation: Theory, algorithms and validation," *Computer Vision and Image Understanding*, vol. 77, no. 2, pp. 145–174, 2000.