Computer Vision and Image Understanding 114 (2010) 85–99

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Synergistic arc-weight estimation for interactive image segmentation using graphs

P.A.V. de Miranda [a,*], A.X. Falcão [a,*], J.K. Udupa [b]

[a] LIV, Institute of Computing, University of Campinas, Av. Albert Einstein 1251, 13084-851 Campinas, SP, Brazil
[b] Medical Image Processing Group (MIPG), 4th Floor Blockley Hall, 423 Guardian Drive, Philadelphia, PA 19104-6021, USA

ARTICLE INFO

Article history:
Received 30 December 2008
Accepted 5 August 2009
Available online 12 August 2009

Keywords:
Image foresting transform
Graph-cut segmentation
Relative-fuzzy connectedness
Watershed transform
Live-wire segmentation
Contour tracking
Interactive segmentation
Graph-search algorithms
κ-Connected segmentation

ABSTRACT

We introduce a framework for synergistic arc-weight estimation, where the user draws markers inside each object (including background), arc weights are estimated from image attributes and object information (pixels under the markers), and a visual feedback guides the user's next action. We demonstrate the method in several graph-based segmentation approaches as a basic step (which should be followed by some proper approach-specific adaptive procedure) and show its advantage over methods that do not exploit object information and over methods that recompute weights during delineation, which make the user to lose control over the segmentation process. We also validate the method using medical data from two imaging modalities (CT and MRI-T1).

1. Introduction

In image processing and computer vision, there are several situations in which user interaction becomes essential in obtaining effective image segmentation. The high-level, application-domain-specific knowledge of the user is often required in medical image analysis [16,8,29,43] because of poorly defined structures, and in the digital matting of natural scenes [2,55], because of their heterogeneous nature.

Image segmentation involves two tightly coupled tasks: *recognition* and *delineation* [16]. Recognition is the task of determining the approximate whereabouts of a desired object in the image, while delineation completes segmentation by precisely defining its spatial extent. Humans usually outperform computers in recognition, but the contrary can be observed in object delineation. While the user can reduce recognition to a simple click of the mouse inside the object, repeatable human delineation is challenging due to human subjectivity. On the other hand, computers can be very precise, even when they are not accurate, but often the absence of high-level object information (location, shape, appearance) makes object recognition a difficult task for computers. In order to overcome some of these shortcomings from both sides, some approaches have combined recognition by the user with delineation by the computer in a synergistic way [22,15,7,45]. This topic forms the central focus of this paper.

In operator-assisted synergistic segmentation, the user usually adds/removes markers (seed pixels, anchor boundary points) for recognition, while subsequent delineation is performed by the computer in interactive time. Accuracy becomes a compromise between the user's patience for verification and correction, and the quality of delineation. The methods usually make direct/indirect use of some image-graph concept, such as arc weight between pixels. The weight may represent different attribute functionals such as similarity, speed function, affinity, cost, distance, etc; depending on different frameworks used, such as watershed, level sets, fuzzy connectedness, graph cuts, etc. The accurate delineation by these methods with minimum user intervention strongly depends on a suitable arc-weight estimation, which usually takes into account image attributes and/or object information often obtained from markers selected by the user during segmentation [7,45]. Object information is very crucial for improving the quality of arc-weight estimation. However, the user's actions need guidance from visual feedback about the quality of the arc weights. Further, the markers used for delineation should never be used to recompute weights. Very often, these markers need to be selected in regions where object and background have similar properties. Weight recomputation followed by delineation based on these markers may destroy other parts of the image where the user was already satisfied with the segmentation results, making the user to lose control over the process.

* Corresponding authors. Fax: +55 19 35215847.
E-mail addresses: paulo.miranda@ic.unicamp.br (P.A.V. de Miranda), afalcao@ic.unicamp.br (A.X. Falcão), jay@mipg.upenn.edu (J.K. Udupa).

1077-3142/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2009.08.001

We propose a synergistic approach for arc-weight estimation, which is separated from the process of interactive image segmentation itself. As a training step, the user selects markers inside each object, where image background is also considered as an object, guided by a visual feedback about the quality of arc-weight estimation. The training markers may be used to start object delineation, but markers selected during segmentation are never allowed to modify arc-weight assignment. We use this approach as a basic step in several image segmentation methods, such as those based on the min-cut/max-flow algorithm [7] and approaches which can be easily implemented by the *image foresting transform* (IFT) [21]. Note that our aim is not to compare segmentation methods, but to show that several of them can benefit from a disciplined, systematic, objective, and effective procedure for arc-weight assignment, followed by some proper approach-specific adaptive procedure, such as the complement of the weights, owing to the nature of the meaning of weights in some methods; or by some tuning procedure (e.g., non-maximal suppression [39], increasing transformations [39], gradient orientation (Section 4.7)). The visualization of the arc weights also allows the user to choose the most appropriate method for a given image. For example, it is desirable in live wire that the arc weights be lower along the object's boundary than in the neighborhood around it [16,22]; the local affinities in relative-fuzzy connectedness [47] be higher inside and outside the object than on its boundary; the gradient values in watershed transforms be higher for pixels on the object's boundary than in its interior and exterior [32,4,15]; the gradient values in tree pruning be higher on the object's boundary than in its interior, and, at least, in a neighborhood in its exterior [3]; and the arc weights in graph-cut segmentation be lower across the object's boundary than in its interior and exterior [7,51,56]. Additionally, energy minimization in [7] using the min-cut/max-flow algorithm from source to sink nodes also requires higher arc weights between source and object pixels, lower arc weights between source and background pixels, lower arc weights between sink and object pixels, and higher arc weights between sink and background pixels. Clearly, the effectiveness of these approaches suffers when the above desirable conditions are not satisfied, and this explains why the visual feedback helps the user to choose the most appropriate method for a given segmentation task.

To outline this paper: Section 2 presents the basic concepts on image graphs and the terminology adopted in this paper. Arc-weight estimation is presented in Section 3 by showing how to exploit image attributes and object information provided by user-selected markers. Section 4 describes several interactive segmentation methods based on the arc-weight assignment of Section 3, and the main advantages of the synergistic approach are demonstrated in Section 5, including evaluation experiments with medical data from two imaging modalities (CT and MRI). Our conclusions are stated in Section 6.

2. Basic concepts on image graphs

An image $\hat{I}$ is a pair $(D_{\hat{I}}, \vec{I})$ where $D_{\hat{I}} \subset Z^n$ is the image domain and $\vec{I}(s)$ assigns a set of m scalars $I_i(s), i = 1, 2, \ldots, m$, to each pixel $s \in D_{\hat{I}}$. This definition applies to multi-dimensional and multi-parametric images. For example, $\{I_1(s), I_2(s), I_3(s)\}$ may be the red, green and blue values of s in a color image $\hat{I}$. The subindex i is dropped for gray images since it becomes awkward when $m = 1$.

An irreflexive *adjacency relation* $\mathscr{A}$ is a binary relation between distinct pixels. We use $t \in \mathscr{A}(s)$ or $(s,t) \in \mathscr{A}$ to indicate that t is adjacent to s. Once $\mathscr{A}$ is fixed, the image $\hat{I}$ can be interpreted as a graph $(D_{\hat{I}}, \mathscr{A})$, whose nodes are the image pixels and whose arcs are the pairs (s,t) in $\mathscr{A}$. For example, one can take $\mathscr{A}$ to consist of all pairs of pixels (s,t) in the Cartesian product $D_{\hat{I}} \times D_{\hat{I}}$ such that $d(s,t) \leqslant \rho$ and $s \neq t$, where $d(s,t)$ denotes the Euclidean distance and $\rho > 0$ is a specified constant (Fig. 1).

A 2D image graph is illustrated in Fig. 2a for $\rho = 1$. This graph topology can be the same for any segmentation method based on optimum paths. The approach based on the min-cut/max-flow algorithm can use this image graph extended by two virtual nodes, source o and sink b, with arcs (o,s) and (s,b) connecting them to each pixel $s \in D_{\hat{I}}$ (Fig. 2b). The arc weights $w(s,t)$ of the image graph are estimated via training, as described next, and the extended arc weights, $w(o,s)$ and $w(s,b)$, are estimated from intermediate results of the training step, as explained in Section 4.5.

3. Synergistic arc-weight estimation

Arc-weight estimation takes into account image attributes and object information in order to enhance the discontinuities between object and background. Let v be an algorithm which extracts attributes (color, gradient, texture) from any pixel $s \in D_{\hat{I}}$ and returns a vector $\vec{v}(s)$. In the simplest case, we may take $\vec{v}(s) = \vec{I}(s)$. However, the best set of attributes depends on each given application.[1] In the segmentation of natural scenes, for example, one may exploit the *Lab* color space [59] and/or compute texture attributes around each pixel from the results of the image convolved with a bank of filters [31,35,41,33,42]. Other options are discussed in Section 3.2. For c objects $l = 1, 2, \ldots, c$, including the background as object numbered c without loss of generality, the weight $w(s,t)$ assigned to each arc $(s,t) \in \mathscr{A}$ is a linear combination of an image-based weight $0 \leqslant w_i(s,t) \leqslant K$ and an object-based weight $0 \leqslant w_o(s,t) \leqslant K$, which takes into account all c objects.

$$w(s,t) = \lambda w_o(s,t) + (1-\lambda) w_i(s,t), \tag{1}$$

where $0 \leqslant \lambda \leqslant 1$. The weights $w_i(s,t)$ exploit only image attributes to capture discontinuities that may exist between homogeneous regions. The weights $w_o(s,t)$ take into account the image attributes for pixels under selected markers, drawn by the user inside each object $l = 1, 2, \ldots c$. They aim to characterize the discontinuities existing between each selected object and the rest of the image. The user can adjust the parameter λ and add/remove markers to recompute the arc weights. The quality of the arc weights is evaluated by visualizing a weight image $\hat{W} = (D_{\hat{I}}, W)$, where

$$W(s) = \max_{\forall t \in \mathscr{A}(s)} \{w(s,t)\} \tag{2}$$

for all $s \in D_{\hat{I}}$. Our aim is to make $w(s,t)$ higher on the desired object boundaries than inside the objects, so $\hat{W}$ must show a suitable boundary enhancement for a given image (see Fig. 3). The complement of $w(s,t)$ may be used depending on the segmentation method. The process of arc-weight assignment stops when the user is satisfied with the boundary enhancement. The image-based weights $w_i(s,t)$ become more important (λ is lower), when nearby objects have similar image properties (Figs. 7 and 14).

The following sections describe how to define object-based and image-based weights and discuss some implementation and customization issues.

3.1. Object-based weight assignment

Let $\hat{d}(s,t) \geqslant 0$ be the distance between the corresponding attribute vectors, $\vec{v}(s)$ and $\vec{v}(t)$, of two pixels s and t. One can use any distance function suitable for the defined attributes. The most common is the vector norm $\|\vec{v}(t) - \vec{v}(s)\|$, which is the one used in this paper, but some image attributes may require special distance algorithms [35,42]. The pair $(\vec{v}, \hat{d})$ then describes how the pixels of

[1] The images used for arc-weight assignment in Figs. 3, 4, 6, 10, and 14 are in the RGB color space.

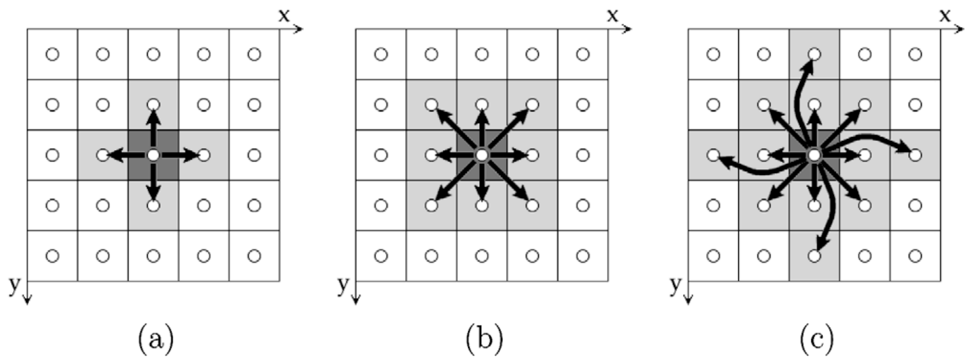

Fig. 1. Euclidean adjacency relations for 2D images: (a) 4-neighborhood ($\rho = 1$), (b) 8-neighborhood $\left(\rho = \sqrt{2}\right)$, and (c) extended adjacency to the 12 closest neighbors ($\rho = 2$).

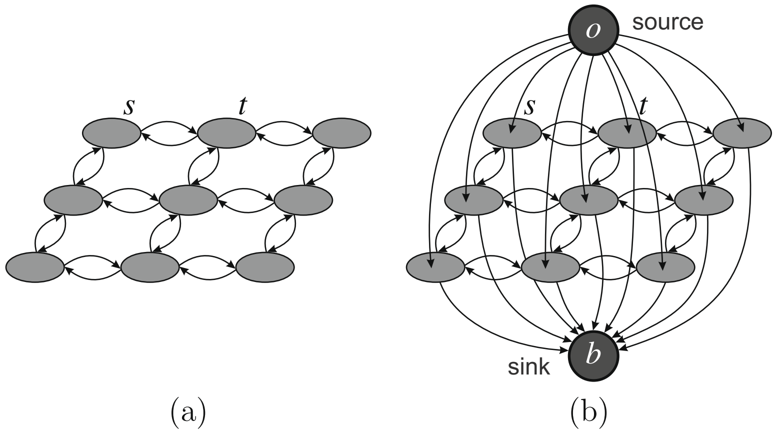

Fig. 2. (a) A 2D image graph with 4-adjacent pixels s and t. (b) An extended graph obtained by adding two terminal nodes (*source o* and *sink b*), which represent object and background, respectively.

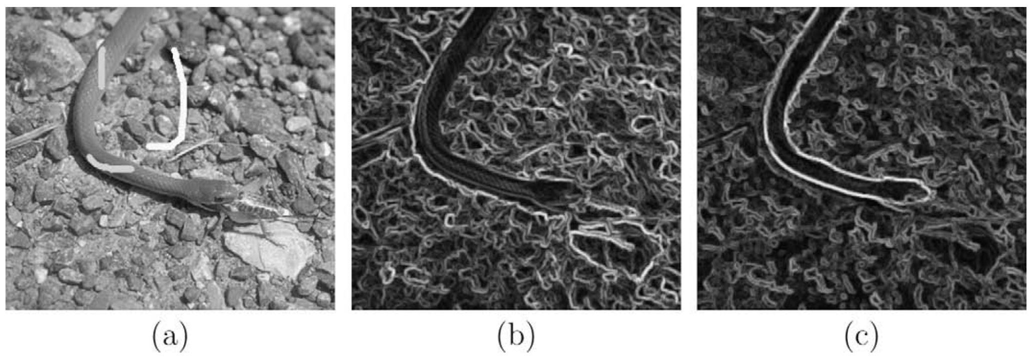

Fig. 3. (a) An image with markers selected inside and outside the object. (b) The weight image $\hat{W}$ considering only the image-based component ($\lambda = 0$), and (c) by combining it with the object-based weight ($\lambda = 0.8$), $c = 2$.

a dataset are distributed in the attribute space and we call it a *descriptor*.

Let $\mathcal{S}_l \subset D_{\hat{I}}$ be the set of representative pixels (markers) selected by the user inside each object $l = 1, 2, \ldots, c$. A suitable descriptor should group pixels of distinct objects in different regions of the attribute space, but the same object may be represented by multiple clusters and pixels of distinct objects may fall in the same cluster. This explains the importance of pixel connectivity for the success of segmentation. We define $\mathcal{A}_{k,l}$ as a special adjacency relation in the attribute space between any pair of pixels (s, t), such that, $s \in D_{\hat{I}}$, and $t \in S_l$ is a k-nearest neighbor of s in the attribute space.

$$t \in \mathcal{A}_{k,l}(s) \quad \text{if } t \in \mathcal{S}_l \text{ is a } k\text{-nearest neighbor of } s \in D_{\hat{I}}. \tag{3}$$

We expect that the mean distance $\bar{d}(s, \mathcal{A}_{k,l}(s))$ between s and its k neighbors in $\mathcal{S}_l$, $l = 1, 2, \ldots, c$, be the smallest for pixels of the same object of s.

$$\bar{d}(s, \mathcal{A}_{k,l}(s)) = \frac{1}{k} \sum_{\forall t \in \mathcal{A}_{k,l}(s)} \hat{d}(s, t). \tag{4}$$

The fuzzy membership value $\mu(l|\vec{v}(s))$ can then be conjectured to be proportional to the total mean distance $\sum \bar{d}(s, \mathscr{A}_{k,i}(s))$ for $i = 1, 2, \ldots, c$, and $i \neq l$.

$$\mu(l|\vec{v}(s)) \approx \left(\frac{K}{c-1}\right) \frac{\sum_{\forall i=1,2,\ldots,c|\ i\neq l} \bar{d}(s, \mathscr{A}_{k,i}(s))}{\sum_{\forall i=1,2,\ldots,c} \bar{d}(s, \mathscr{A}_{k,i}(s))}. \tag{5}$$

These membership values can be viewed through a set of images $\hat{M}_i = (D_I, M_i),\ i = 1, 2, \ldots, c$ such that $M_i(s) = \mu(l = i|\vec{v}(s))$ for all $s \in D_I$. For multiple objects, the user should keep on drawing markers inside the dark regions of object i in each image $\hat{M}_i,\ i = 1, 2, \ldots, c$, until that object becomes brighter than the rest in this membership image. For the sake of simplicity, all examples in this paper use only "object and background" type of situation. In this case, $\mu(l = 1|\vec{v}(s)) = K - \mu(l = 2|\vec{v}(s))$, and then we can, and need to, show only $\hat{M}_1$ with internal and external markers (Fig. 4). As the user adds markers, the estimation improves and the object becomes increasingly distinguished (brighter) from the background (darker).

We could have estimated $\mu(l|\vec{v}(s))$ by Baye's Theorem, by directly computing the posterior probability $\mathscr{P}(l|\vec{v}(s))$ from the markers and the distances between s and its neighbors in $\mathscr{A}_{k,l}(s)$ in the attribute space [14]. A similar approach to compute probability density functions is described in [46]. We compared with that approach and the results were equivalent to those obtained by Eq. (5), which is simpler and more efficient. Note that, according to Eq. (5), for any pixel s the sum of $\mu(l|\vec{v}(s))$ for $l = 1, 2, \ldots, c$ is the maximum value K, as desired for a discrete surrogate of the probability quantized into K levels.

The discontinuities between each object l and the rest of the image can be captured from a gradient vector $\vec{G}_l(s)$, defined for all $s \in D_I$ and computed as follows.

$$\vec{G}_l(s) = \sum_{\forall t \in \mathscr{A}(s)} [\mu(l|\vec{v}(t)) - \mu(l|\vec{v}(s))]\vec{st}, \tag{6}$$

where $\vec{st}$ is the unit vector connecting s to t in the image domain. For a 2D Euclidean adjacency $\mathscr{A}$ with $\rho = \sqrt{2}$, the gradient vector $\vec{G}_l(s)$ is estimated from the vectorial sum of the first derivatives of $\mu(l \mid \vec{v}(s))$ along the 8 directions, rather than from the x and y directions only.

For each arc $(s, t) \in \mathscr{A}$, we compute the magnitude of the mean gradient vector of its pixels and use it as the weight $w_{o,l}(s, t)$ with respect to the object l. The final object-based weight $w_o(s, t)$ is considered to be the maximum of $w_{o,l}(s, t)$ among all objects.

$$w_{o,l}(s,t) = \left| \frac{\vec{G}_l(s) + \vec{G}_l(t)}{2} \right| \tag{7}$$

$$w_o(s,t) = \max_{l=1,2,\ldots,c} \{w_{o,l}(s,t)\}. \tag{8}$$

The orientation of $\vec{G}_l(s) + \vec{G}_l(t)$ can also be exploited to modify arc-weight assignment (Section 4.7).

As the user adds markers, the size of the union set $\mathscr{X} = \bigcup_{l=1,2,\ldots,c} \mathscr{S}_l$ increases and Eq. (5) becomes computationally more expensive to assign membership values to all pixels in D_I. On the other hand, we do not need quantity, but quality, in choosing pixels for $\mathscr{X}$. In order to choose the best representative pixels for each object from the drawn markers, and, at the same time, to estimate the best parameter k, we use supervised learning as described next.

3.1.1. Supervised learning from markers

The main idea is to reduce the size of $\mathscr{X}$ by selecting a subset $\mathscr{X}_1 \subset \mathscr{X}$ of the most representative pixels. These pixels are defined as those that maximize the classification accuracy of the remaining set of pixels $\mathscr{X}_2 = \mathscr{X} \setminus \mathscr{X}_1$, using the maximum $\mu(l|\vec{v}(s))$ as the decision rule for the pixels $s \in \mathscr{X}_2$ with respect to its neighbors $\mathscr{A}_{k,l}(s) \subset \mathscr{X}_1$ by Eq. (5).

The set $\mathscr{X}$ is divided into two subsets, $\mathscr{X}_1$ and $\mathscr{X}_2$, by randomly selecting the same percentage of pixels from each object. Set $\mathscr{X}_1$ has a maximum size (e.g., 100 pixels). When the number of seeds is less than that maximum size, we may divide $\mathscr{X}$ into 50% for $\mathscr{X}_1$ and 50% for $\mathscr{X}_2$. The maximum $\mu(l \mid \vec{v}(s))$ is used to classify the pixels in $\mathscr{X}_2$. This process is repeated for each k from 1 to k_{max} (Eq. (9))

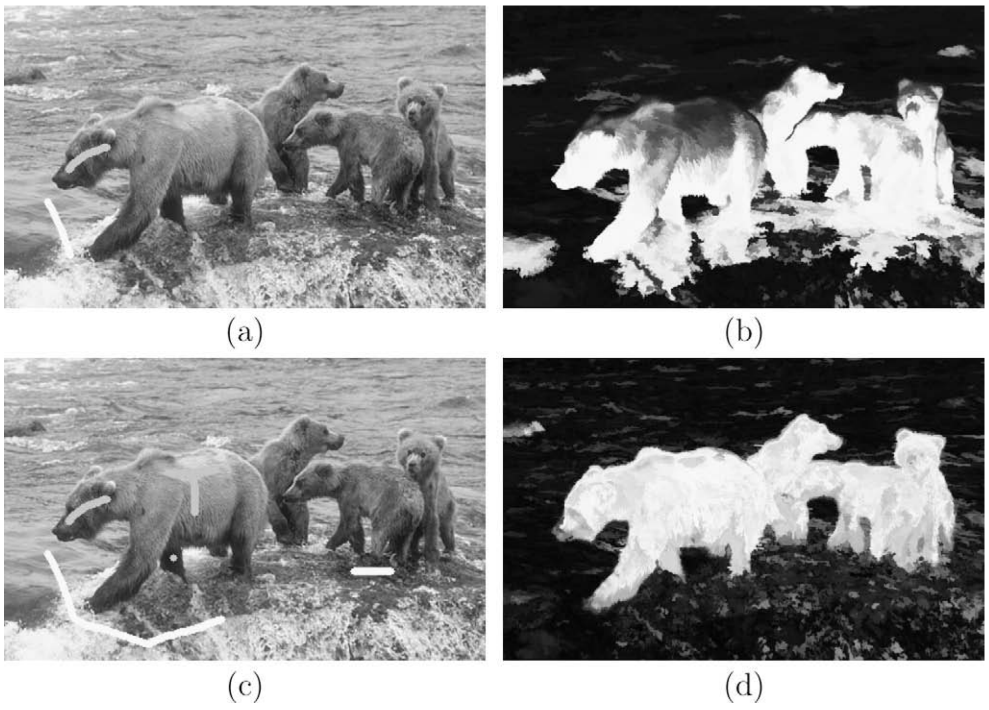

Fig. 4. Image $\hat{M}_1$, where the desired object is a family of bears. (a–b) An initial marker selection and the corresponding membership image $\hat{M}_1$. (c–d) The estimation improves as the user adds internal markers on the dark regions of the object and external markers on the bright regions of the background in $\hat{M}_1$.

in order to obtain the best value of k for the given $\mathscr{Z}_1$. The misclassified pixels with the best k are randomly replaced by pixels of the same object in $\mathscr{Z}_1$. The whole process is repeated over a few iterations T (e.g., $T = 5$) and the pair $(\mathscr{Z}_1, k)$ of maximum accuracy is selected as output (see Algorithm 1).

$$k_{max} = \min_{\forall l=1,2,\ldots,c} \left\{ \frac{|\mathscr{S}_l \cap \mathscr{Z}_1|}{2} \right\}. \tag{9}$$

Algorithm 1. Learning Algorithm

```
INPUT: Initial sets $\mathscr{Z}_1$ and $\mathscr{Z}_2$, number $T$ of iterations, and the
       descriptor $(v, d)$.
OUTPUT: The pair $(\mathscr{Z}_1^*, k^*)$ of maximum accuracy.
AUXILIARY: Arrays $FP$ and $FN$ of sizes $c$ for false positives and false
           negatives, list $M$ of misclassified pixels and its auxiliary
           $M^*$, and array $A$ of size $T$ for the best accuracies.
1: Compute $k_{max}$ by Eq. (9).
2: For each iteration $i = 1, 2, \ldots, T$, do
3:    Set $Acc^* \leftarrow 0$.
4:    For each $k = 1, 2, \ldots, k_{max}$, do
5:       $M \leftarrow \emptyset$.
6:       For each class $l = 1, 2, \ldots, c$, do
7:          $FP(l) \leftarrow 0$ and $FN(l) \leftarrow 0$.
8:       For each sample $s \in Z_2$, do
9:          Classify $s$ with label $1 \leqslant L(s) \leqslant c$, as described
            above.
10:         If $s \in \mathscr{S}_l$ and $L(s) \neq l$, then
11:            $FP(L(s)) \leftarrow FP(L(s)) + 1$.
12:            $FN(l) \leftarrow FN(l) + 1$.
13:            $M \leftarrow M \cup t$.
14:      Compute $Acc$ by Eq. (12).
15:      If $Acc \geqslant Acc^*$, then
16:         $Acc^* \leftarrow Acc, M^* \leftarrow M, \mathscr{Z}_1^* \leftarrow \mathscr{Z}_1$ and $k^* \leftarrow k$.
17:   Save $(\mathscr{Z}_1^*, k^*)$ and set $A(i) \leftarrow Acc^*$.
18:   While $M^* \neq \emptyset$
19:      $M^* \leftarrow M^* \setminus s$
20:      Replace $s$ by a randomly selected pixel of the same
21:      class in $\mathscr{Z}_1$.
22: Select the instance of $(\mathscr{Z}_1^*, k^*)$ with maximum accuracy
    $A(i),\ i = 1, 2, \ldots, T$.
```

Accuracy is measured, as suggested in [44], by taking into account the fact that objects may have different sizes in $\mathscr{Z}_2$. If there are two objects, for example, with very different sizes and the classifier always assigns the label of the largest object, its accuracy will fall drastically due to the high error rate on the smallest object. This accuracy is defined as follows. Let $N_2(l)$ be the number of pixels of $\mathscr{S}_l$ in $\mathscr{Z}_2$. We first define

$$e_{l,1} = \frac{FP(l)}{|\mathscr{Z}_2| - N_2(l)} \quad \text{and} \quad e_{l,2} = \frac{FN(l)}{N_2(l)}, \qquad l = 1, \ldots, c, \tag{10}$$

where $FP(l)$ and $FN(l)$ are the number of false positive and false negative pixels (Lines 11–12), respectively. That is, $FP(l)$ is the number of pixels from other objects that were classified as being from the object l in $\mathscr{Z}_2$, and $FN(l)$ is the number of pixels from the object l that were incorrectly classified as being from other objects in $\mathscr{Z}_2$. The errors $e_{l,1}$ and $e_{l,2}$ are used to define

$$E(l) = e_{l,1} + e_{l,2}, \tag{11}$$

where $E(l)$ is the partial sum error of object l. Finally, the accuracy Acc of classification is expressed as

$$Acc = \frac{2c - \sum_{l=1}^{c} E(l)}{2c} = 1 - \frac{\sum_{l=1}^{c} E(l)}{2c}. \tag{12}$$

3.2. Image-based weight assignment

In general, one may use $0 \leqslant \hat{d}(s,t) \leqslant K$ as the image-based weight $w_i(s,t)$ in Eq. (1). It is also possible to learn the posterior probability of a pixel (or arc) to be on a boundary from local image attributes [36]. We present another interesting option based on image smoothing at several scales.

Multiscale image smoothing can be accomplished by linear convolutions with Gaussians [31] and/or levelings [37,54,49,48]. Except for Fig. 15, the other examples in this paper use sequences of opening by reconstruction and closing by reconstruction, computed over each image band $I_b, b = 1, 2, \ldots, m$, for disks of radii $r = 1, 2, \ldots, S$ (e.g., $S = 4$ pixels). Gaussian filters provide smoother contours than morphological reconstructions, but the latter may be preferable to better conserve the natural shape indentations and profusions. In Fig. 15, we illustrate the contour smoothness obtained by Gaussian filters with means equal to 0 and standard deviations $\sigma = \frac{r}{3}$ for scales $r = 1, 2, \ldots, S = 6$ pixels.

Let $\vec{v}_b(s) = (v_{b,1}(s), v_{b,2}(s), \ldots, v_{b,S}(s))$ be the resulting pixel intensities $v_{b,j}(s), j = 1, 2, \ldots, S$, of the multiscale smoothing on the image band $I_b, b = 1, 2, \ldots, m$. We compute a gradient vector $\vec{G}_b(s)$ for each $s \in D_{\hat{I}}$ and band $b = 1, 2, \ldots, m$. The idea is the same as in Eq. (6), where $\mathscr{A}$ may be Euclidean with $\rho = \sqrt{2}$.

$$\vec{G}_b(s) = \sum_{j=1}^{S} \sum_{\forall t \in \mathscr{A}(s)} [v_{b,j}(t) - v_{b,j}(s)] \vec{st} \tag{13}$$

$$w_i(s,t) = \max_{b=1,2,\ldots,m} \left\{ \left| \frac{\vec{G}_b(s) + \vec{G}_b(t)}{2} \right| \right\}. \tag{14}$$

Note that, the gradients $\vec{G}_b(s)$ are filtered vectors, the gradient orientation of the mean vector of maximum magnitude may be used to modify arc-weight assignment (Section 4.7), and the best choice of attributes for a given image should be learned from the selected markers and a database of descriptors. One can select, for example, the descriptor which maximizes the accuracy in Algorithm 1.

4. Interactive segmentation methods

A segmentation result is represented by a label image $\hat{L} = (D_{\hat{I}}, L)$, in which each label $1 \leqslant L(s) \leqslant c$ assigns a pixel $s \in D_{\hat{I}}$ to one object out of c objects, including background. For the sake of simplicity, we have considered the case of $c = 2$ in all examples of this paper. All methods presented in this section have been well published, so we will present only a short description with their graph parameters customized as a function of $w(s,t)$ and $\mu(l|\vec{v}(s))$, although for other methods, different adaptive procedures may be required. The methods based on optimum paths are described by using the image foresting transform (IFT) [21]. We also describe the graph-cut approach based on the min-cut/max-flow algorithm of [7]. What is novel in this section is the way these methods are used in combination as tools in an interactive segmentation paradigm.

4.1. Image foresting transform

The *image foresting transform* (IFT) is a tool for the design, implementation, and evaluation of image processing operators based on connectivity values among pixels [21].

In a given image graph $(D_{\hat{I}}, \mathscr{A})$, a path $\pi_t = \langle t_1, t_2, \ldots, t \rangle$ is a sequence of two or more adjacent pixels with the terminus at a pixel $t \in D_{\hat{I}}$, $\pi_t = \langle t \rangle$ being considered a trivial path. A path π_t is *optimum* under a *path-value function* $f(\pi_t)$, when $f(\pi_t) \leqslant f(\tau_t)$ for any other path τ_t. The IFT computes an *optimum-path forest P* by minimizing (or maximizing) Eq. (15) for every $t \in D_{\hat{I}}$.

$$V(t) = \min_{\forall \pi_t in (D_{\hat{I}}, \mathcal{A})} \{f(\pi_t)\}, \tag{15}$$

where $V(t)$ is the value of the optimum path with terminus t. The initial pixels of the optimum paths are called *roots* of the forest. By starting with trivial paths $\pi_t = \langle t \rangle$ for all pixels $t \in D_{\hat{I}}$, the IFT algorithm (a generalized Dijkstra's algorithm [11], which in practice, executes in linear time in most cases) first identifies the forest roots (minima/maxima of V) and then propagates optimum paths to their adjacent pixels, continuing from these nodes to their neighbors, and following a non-decreasing (non-increasing) order of path values, according to the path-propagation rule below.

$$\text{if } f(\pi_s \cdot \langle s, t \rangle) < f(\pi_t) \quad \text{then } \pi_t \leftarrow \pi_s \cdot \langle s, t \rangle, \tag{16}$$

where $\pi_s \cdot \langle s, t \rangle$ indicates the extension of a path π_s by an arc $(s, t) \in \mathcal{A}$ (Fig. 5a). These paths are represented in backwards, where $P(t)$ indicates the predecessor node of t in the path π_t and $R(t)$ is its root pixel for which $P(R(t)) = nil$ (Fig. 5b). An optimum-path forest P is a function which takes every pixel to *nil* in a finite number of iterations, such that all paths are optimum (Fig. 5c).

The path-value functions define different IFT-based image operators, which are reduced to a local processing operation on one or more of the output maps V, P, and R [19,18,46,44,3,52]. The IFT algorithm is an optimum region (path) growing process from the roots of the forest (Fig. 6). Variants can also gather on-the-fly other information, such as a root label for each pixel [32,15], the propagation order of the pixels [40], the area of the wavefronts of same path value [40], and a graph-cut measure for the border of the growing regions [20].

Particularly, the image segmentation methods described in the following sections adopt a minimization of path-value functions f_1 and f_2, and some of their variants, such that the roots of the forest are constrained into the union set $\mathcal{X} = \bigcup_{l=1,2,\dots,c} \mathcal{S}_l$ of selected markers. We note that, these IFT algorithms run in linear time independently of $|\mathcal{X}|$.

$$f_1(\langle t \rangle) = \begin{cases} H(t) & \text{if } t \in \mathcal{X} \\ +\infty & \text{otherwise.} \end{cases}$$
$$f_1(\pi_s \cdot \langle s, t \rangle) = \max\{f_1(\pi_s), w(s, t)\} \tag{17}$$
$$f_2(\langle t \rangle) = \begin{cases} H(t) & \text{if } t \in \mathcal{X} \\ +\infty & \text{otherwise.} \end{cases}$$
$$f_2(\pi_s \cdot \langle s, t \rangle) = f_2(\pi_s) + w(s, t), \tag{18}$$

where $0 \leqslant H(t) < \infty$ is a handicap value and $0 \leqslant w(s, t) \leqslant K$ is the fixed arc weight, as described in Section 3. Function $f_1(\pi_t)$ computes the maximum arc weight along π_t and $f_2(\pi_t)$ computes the sum of the arc weights along π_t.

4.2. Segmentation by differential IFT (DIFT)

Multiple objects can be obtained by competition among markers in $\mathcal{S}_l, l = 1, 2, \dots, c$. By assigning higher arc weights across the desired boundaries, the IFT with f_1 (e.g., $H(t) = 0$) tends to propagate optimum paths inside the objects before they meet paths from seeds of other objects at the image boundaries (Fig. 6). Additional seeds are required when this condition is not fully satisfied. Each seed $r \in \mathcal{X}$ defines an influence zone (optimum-path tree rooted at r) composed of the pixels that are more strongly connected to r than to any other seed. Each object l is then defined by the union of the influence zones with that label in L. This essentially incorporates approaches, such as the watershed transform from markers [4,32] and iterative relative-fuzzy connectedness [9]. The formal relation that exists between these approaches is studied in [39,1]. The same strategy with f_2 (e.g., $H(t) = 0$) would be a segmentation by weighted distance transform [2,45].

In any case, the user may want to add/remove markers to correct the segmentation results (Fig. 7). Instead of computing one IFT from the beginning for each new instance of seeds, the DIFT algorithm allows us to recompute the optimum-path forest in time proportional to the number of pixels in the modified regions [15] (sublinear time in practice).

4.3. Segmentation by κ-connected components

User involvement can be reduced when we exploit other properties of the optimum paths during the IFT algorithm [40]. The IFT with f_1 (with $H(t) = 0$) propagates wavefronts $\mathcal{W}_u(s)$ of iso-optimum-path value u around each seed s, following an increasing order of values $u = 0, 1, \dots, K$. The maximal extent of a seed inside an object is defined by a κ_s value as $\bigcup_{u=0,1,\dots,\kappa_s} \mathcal{W}_u(s)$ (Fig. 8a). When the competition with external seeds fails (or there is no external seeds) and an optimum path from s invades the background, it usually crosses the boundary through its weakest link (arc with the lowest weight or *leaking arc*), ramifies and conquers a large region of surrounding pixels with the same path value $\kappa_s + 1$ (Fig. 8b). This background invasion is characterized by a considerable increase of $|\mathcal{W}_u(s)|$, which can be observed by displaying a curve of the total area $\sum_{\forall s \in \mathcal{S}_l} |\mathcal{W}_u(s)|$ for $u = 0, 1, \dots, K$ during propagation (Fig. 8c). A single area threshold $0\% < T < 100\%$ on the size $|\mathcal{W}_u(s)|$ can be used to detect κ_s for all seeds $s \in \mathcal{S}_l$ and forbid the leaking by stopping the region growing from s. The object is defined as the subset of pixels which are more strongly κ-connected to its internal seeds than to any other (Fig. 8d). As discussed in Ref. [39], a non-maximal suppression to make the weights in the object's boundary thinner is an adequate preprocessing that should be adopted in these κ-connected methods.

Note that, since the internal seeds also compete among themselves, with distinct κ_s values, the method can work even when

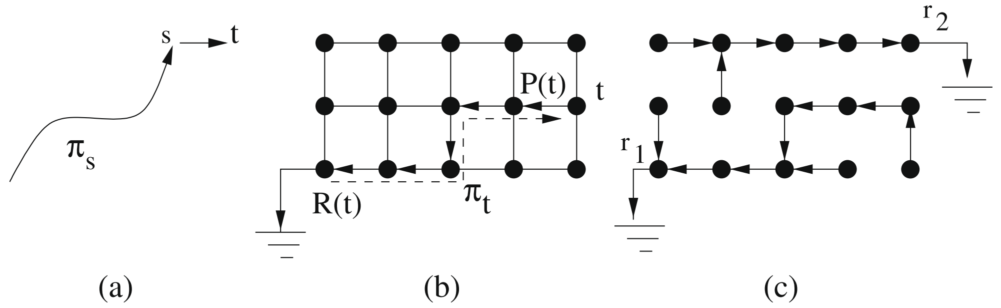

Fig. 5. (a) Path $\pi_t = \pi_s \cdot \langle s, t \rangle$ indicates the extension of path π_s by an arc $(s, t) \in \mathcal{A}$. (b) A 4-neighborhood graph showing a path π_t (dashed line) represented in backwards, where $P(t)$ is the predecessor node of t and $R(t)$ is the root pixel. (c) A forest P with two root nodes r_1 and r_2.

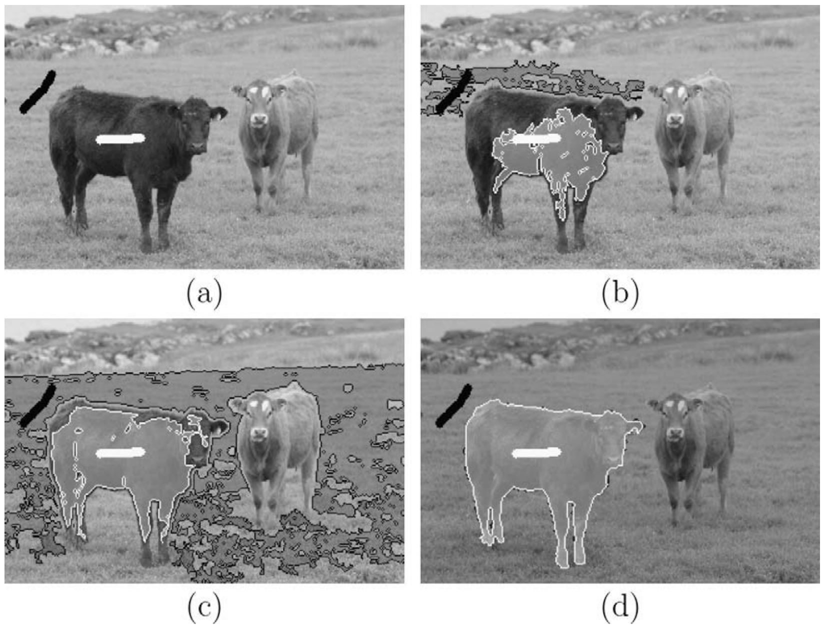

Fig. 6. (a) An initial marker selection for segmentation. (b–d) The IFT region growing. (d) The regions meet each other at the object's boundary.

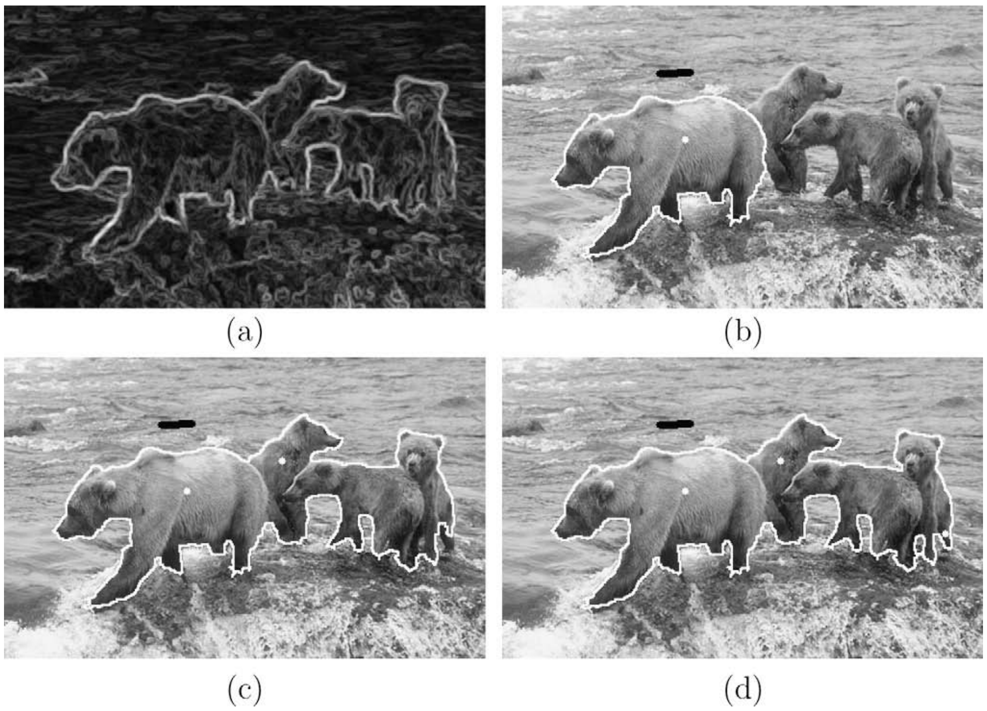

Fig. 7. (a) A weight image $\hat{W}$ obtained from Fig. 4 using $\lambda = 0.4$. (b) An initial marker selection and segmentation. (c) An additional seed is inserted in order to include the other bears. (d) The final segmentation after some small corrections.

leaking occurs before the object is fully segmented (Fig. 8b). The method usually reduces the number of external seeds required to complete segmentation [40], and it is equivalent to the segmentation by differential IFT when we increase the number of external seeds. That is, it is more general than the previous approach.

4.4. Segmentation by tree pruning

Another idea to reduce/eliminate external seeds using the IFT with f_1 (with $H(t) = 0$) has been proposed in [17,3]. In both approaches, the idea is to let the object and the background get connected through the leaking arcs by computing the IFT from internal seeds. The leaking arcs can be then detected interactively [17] or automatically [3]. By removing their subtrees from the forest P, the remaining forest defines the object. The first approach can handle multiple objects, but we will discuss here only the second approach.

In [3], there is no competition with external markers. They are called an external set $\mathscr{B}$, which is used to detect all leaking arcs automatically. In most cases the set $\mathscr{B}$ is the image's border, but the user can also add external pixels to $\mathscr{B}$ or internal seeds, if needed. The optimum paths that leak to the background are called *leaking paths*. They can be enhanced by displaying the number of descendants that each forest node has in $\mathscr{B}$ (Fig. 9a). Since the leaking paths are ramified after leaking, there is a considerable decrease in the descendant number after the leaking arcs. The method can detect this variation, remove the leaking arcs and output the object (Fig. 9b). It has been shown that segmentation by tree pruning is less sensitive to the heterogeneity of the background than the watershed transform from markers [3].

4.5. Segmentation by graph cut

Approaches for graph-cut segmentation are based on objective functions that measure some global property of the object's boundary using the arc weights. The idea is to assign weights to the arcs such that the minimum of this objective function corresponds to the desired segmentation (i.e., a *cut boundary* whose arcs connect the nodes between object and background).

Wu and Leahy [58] were the first to introduce a solution for graph cut using as measure the sum of the arc weights in the cut boundary. Their cut measure had a bias toward small boundaries, and subsequently, other objective functions, such as average cut [12], mean cut [56], average association [50], normalized cut [51], ratio cut [57], and energy functions [7] have been proposed to circumvent this problem.

Interactive segmentation using the min-cut/max-flow algorithm [7,29] uses extended image graphs (Fig. 2), where two terminal nodes o and b (*source* and *sink*) represent object $(l = 1)$ and background $(l = 2)$, respectively, directly connected to all pixels $s \in D_I$ by arcs (o, s) and (s, b). A variant of the min-cut/max-flow algorithm from source to sink [24,5] is then used to speed up computation of the minimum-cut boundary according to the following equation:

$$E(\hat{L}) = \sum_{\forall (s,t)\in\mathscr{A} |\ L(s)=1, L(t)=2} K - w(s,t) + \sum_{\forall s\in D_I |\ L(s)=1} w(s,b) + \sum_{\forall s\in D_I |\ L(s)=2} w(o,s), \tag{19}$$

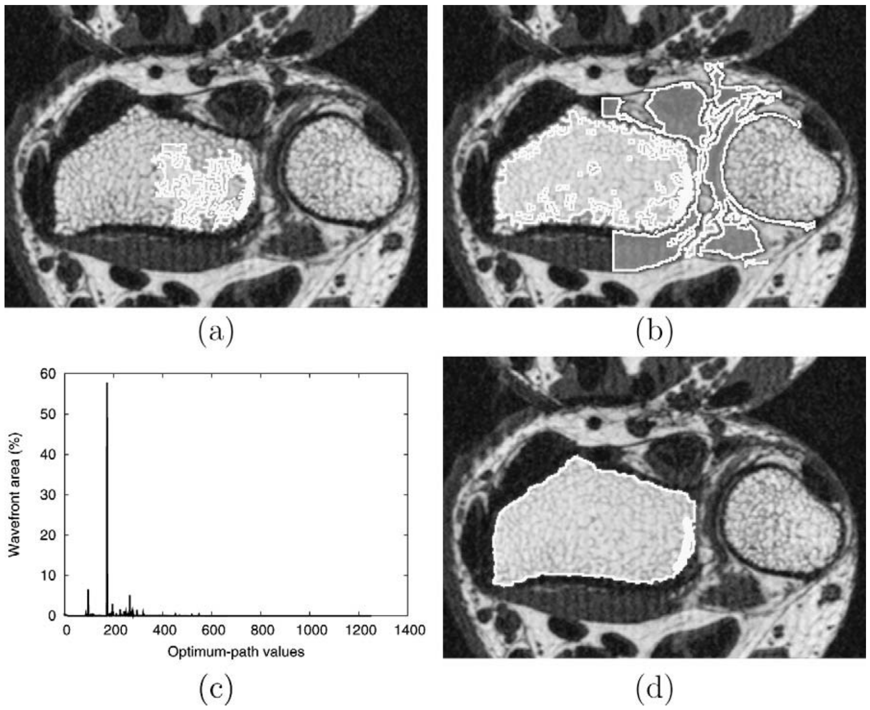

Fig. 8. (a and b) The IFT region growing from internal seeds. There is a burst in the size of the wavefront when an optimum path reaches the background. (c) The total wavefront area for each optimum-path value $u = 1, 2, \ldots, K$ during propagation. (d) The resulting segmentation with κ-connected components.

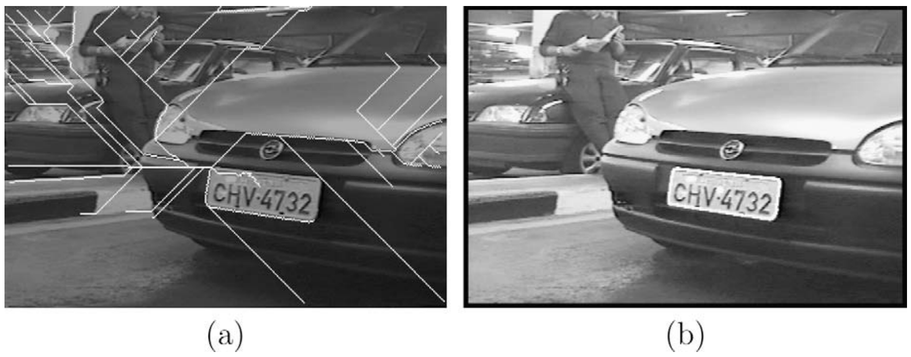

Fig. 9. Example of license plate segmentation. (a) The original image overlaid by the number of descendants in the background set $\mathscr{B}$. (b) The resulting segmentation with tree pruning.

where $w(s,b)$ and $w(o,s)$ can be computed based on the membership values given in Eq. (5):

$$w(o,s) = \alpha \cdot \mu(l = 1|\vec{v}(s)) \tag{20}$$

$$w(s,b) = \alpha \cdot \mu(l = 2|\vec{v}(s)). \tag{21}$$

Here, $\alpha \geqslant 0$ specifies the relative importance of the arcs with the virtual nodes versus the arcs between pixels (Fig. 10). As discussed in Ref. [39], an interesting adaptive procedure to improve this method is to penalize arcs between pixels with high complemented weights by applying some increasing transformation (e.g., power functions, the exponential function [30]). This is especially important for low α values since in this case Eq. (19) becomes almost the same as in Wu and Leahy [58] and it helps to circumvent the undesirable bias.

If the algorithm fails in delineating the desired boundary, the user forces arc weights with source and sink by adding markers inside and outside the object [7]. The problems related to the simultaneous segmentation of multiple objects are discussed in [6].

4.6. Segmentation by IFT with graph cut

If the arc weights are higher on the desired boundary than inside the objects, then the borders of the growing regions from internal seeds must merge and fit to the desired boundary during the IFT propagation with f_1 (with $H(t) = 0$). Such borders work as cut boundaries and different cut measures may be computed on-the-fly for every instant (propagation order of each pixel) during region growing (Fig. 11a). Within this considerably reduced search space, the minimum cut is expected to occur on the object's boundary (Fig. 11b). The method has been evaluated for normalized cut, mean cut and energy functions [20]. When the weight condition is not fully satisfied, the desired cut is not a global minimum even within this reduced search space, but the user can add more internal seeds. The reduction of the search space represents a considerable efficiency gain with respect to some graph-cut approaches [51,57].

We note that, a non-maximal suppression to tune the weights can help this method by allowing a better fit to the boundaries during the internal region growing [39].

4.7. Segmentation by live wire

In order to segment the object with live wire [22], the user selects a starting point on the object's boundary (point s_1 in Fig. 12a), and, for any subsequent position of the mouse cursor, the method computes an optimum path from s_1 to that position in real time. As the user moves the cursor close to the boundary, the optimum segment snaps on to it. The user can quickly verify the longest segment, as the one with terminus at point s_2 in Fig. 12b, and deposit the mouse cursor at that position. The process is then repeated from s_2 until the user decides to close the contour (Fig. 12c and d).

The closed contour is an optimum curve that is constrained to pass through a sequence $\langle \mathscr{S}^{(1)}, \mathscr{S}^{(2)}, \ldots, \mathscr{S}^{(N)} \rangle$ of N anchor points (seeds) on the object's boundary, in that order, starting from $\mathscr{S}^{(1)}$ and ending in $\mathscr{S}^{(N)}$, where each set $\mathscr{S}^{(i)}, i = 1, 2, \ldots, N$, has a single pixel s_i and $s_1 = s_N$. The optimum curve that satisfies those constraints consists of $N - 1$ segments $\pi_{s_2}, \pi_{s_3}, \ldots, \pi_{s_N}$, where each π_{s_i} is an optimum path connecting s_{i-1} to s_i. Therefore, we can solve this problem by $N - 1$ executions of the IFT and the optimum contour can be obtained from the predecessor map P after the last execution. For $i = 2, 3, \ldots, N$, the IFT is computed using the initial point $s_{i-1} \in \mathscr{S}^{(i-1)}$ as seed, 8-adjacency relation and path-value function f_3 (a variant of f_2).

$$f_3(\langle t \rangle) = \begin{cases} V(t) & \text{if } t \in \pi_{s_2} \cup \ldots \cup \pi_{s_{i-1}} \\ +\infty & \text{otherwise} \end{cases} \tag{22}$$

$$f_3(\pi_s \cdot \langle s,t \rangle) = f_3(\pi_s) + (K - \max\{\vec{G}(s,t) \cdot \vec{\eta}(s,t), 0\})^a, \tag{23}$$

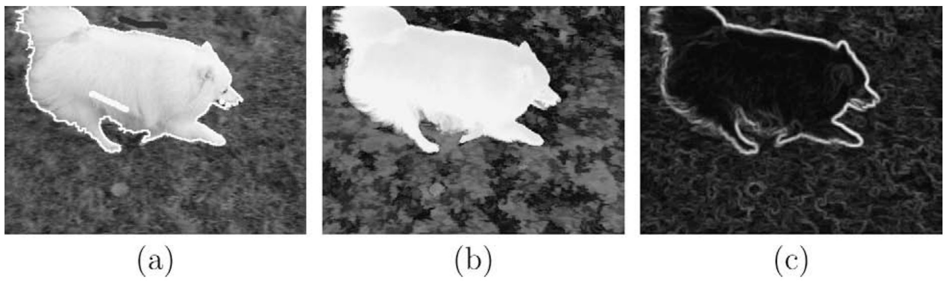

Fig. 10. Graph-cut segmentation with $\alpha = 20$ and $\lambda = 0.5$. (a) Marker selection for training and the result of segmentation. (b) The membership image $\hat{M}_1$ used in $w(o,s)$ and $w(s,b)$. (c) The weight image $\hat{W}$ that reflects $w(s,t)$.

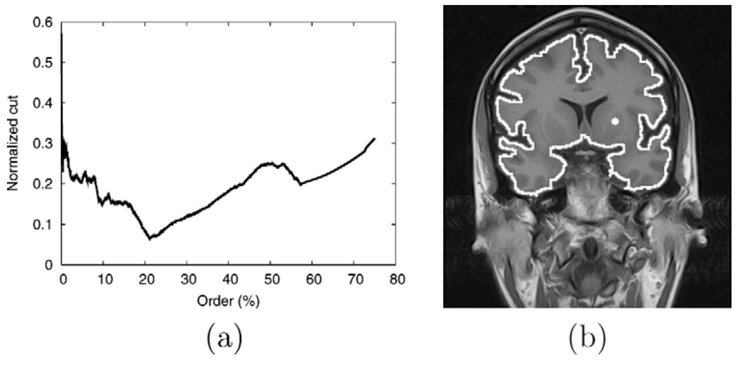

Fig. 11. Segmentation example of a MR-brain image. (a) The normalized cut versus the pixel propagation order. (b) The respective segmentation.

where $V(t)$ is the optimum path value of the previous executions, $a > 0$ (e.g., $a = 1.5$), $0 \leqslant |\vec{G}(s,t)| \leqslant K$ is the gradient vector estimated at the midpoint of arc (s,t), and $\vec{\eta}(s,t)$ is the unit vector $\vec{st}$ rotated 90 degrees counter-clockwise. This formulation favors segmentation on a single orientation, but allows longer boundary segments. We use $\vec{G}(s,t) = \frac{\vec{G}_l(s)+\vec{G}_l(t)}{2}$ obtained from the membership map, but the image gradient $\vec{G}(s,t) = \frac{\vec{G}_b(s)+\vec{G}_b(t)}{2}$ of maximum magnitude for $b = 1,2,\ldots,m$ is also an option, when there are no training markers. The initial path value $V(s_1) = f_3(\langle s_1 \rangle) = 0$ and we make $V(s_1 = s_i) = +\infty$ to compute the last segment.

We note that, other variants of live wire can also take advantage of the proposed arc-weight assignment [21,27,23,26,34,25].

5. Experiments and results

The examples in the previous sections have shown that the proposed process for arc-weight assignment is useful in several image segmentation methods. The synergism between the user and the computer offers some important advantages as well. Object information is incorporated into arc-weight estimation under user supervision and control. In traditional segmentation methods [28,4,51,56], arc-weight estimation is usually treated as a simple embedded process, disregarding, in many cases, the user interventions. For example, the watershed from markers over the weight image (Fig. 13a) can be drastically improved by incorporating object information as presented in Section 3.1 (Fig. 13b).

The arc-weight assignment process and computation without visual inspection by the user makes it difficult to understand what part is contributing more to the final segmentation result: arc-weight estimation or the segmentation algorithm. Hence, only a common base strategy for arc-weight assignment, with the proper adaptive procedures, allows fair comparisons among methods. For instance, it is easy to see that any approach based on the weights of Fig. 3b will be at a clear disadvantage when compared with those based on the weights of Fig. 3c.

Other approaches also incorporate object information into arc-weight estimation [7,2,45]. However, the absence of weight visualization and the use of segmentation markers for both arc-weight esti-

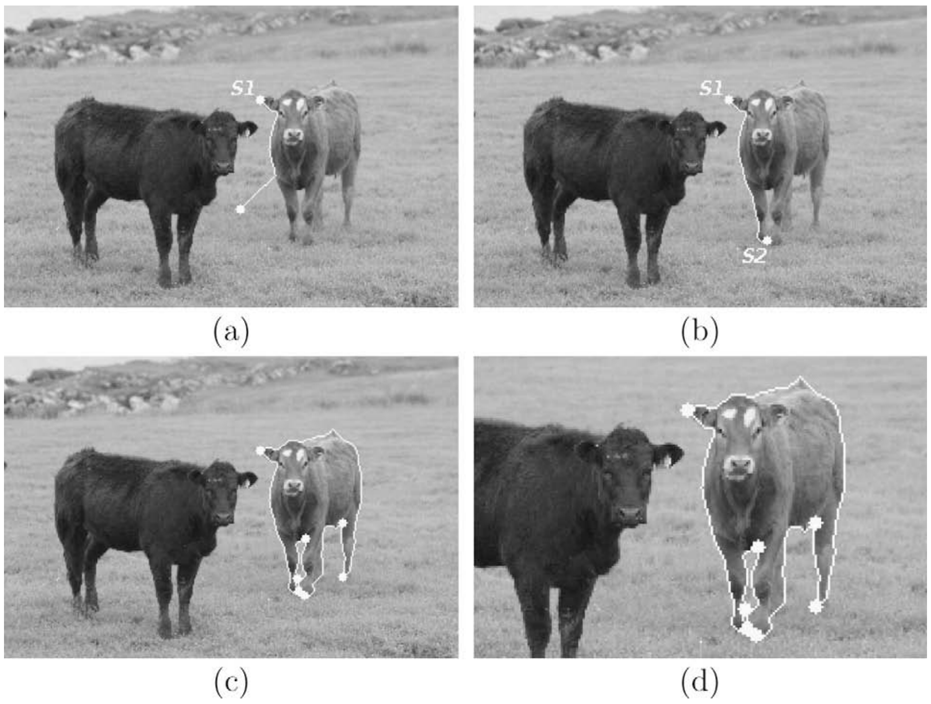

Fig. 12. Contour tracking with live wire. (a) Initial point s_1 is selected on the boundary and the user moves the mouse. (b) A second point s_2 is selected on the boundary. (c and d) Final contour with 7 segments.

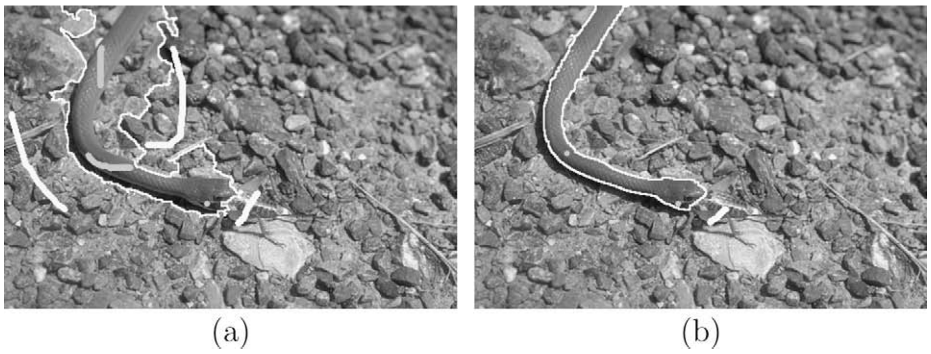

Fig. 13. (a) Segmentation result of the watershed transform from markers considering only the traditional image-based component (Fig. 3b). (b) A better result is obtained with less seeds, by the use of object-based weights (Fig. 3c).

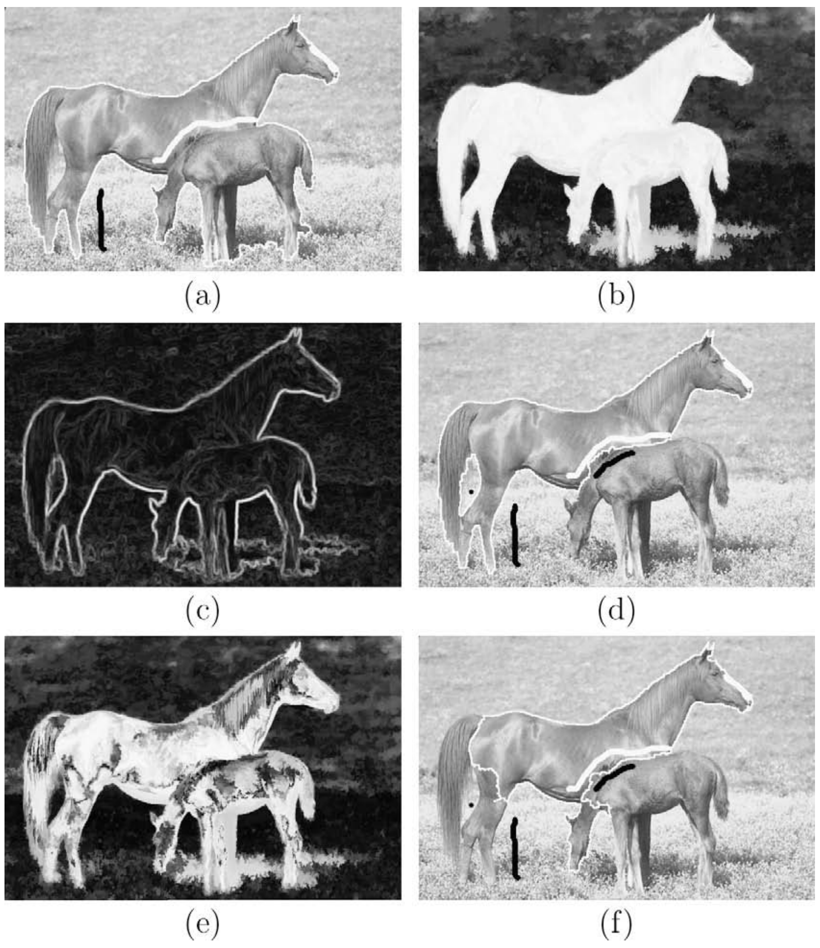

Fig. 14. (a) DIFT segmentation using the same markers for training and delineation. (b and c) The respective membership image $\hat{M}_1$ and weight image $\hat{W}$ of the training ($\lambda = 0.5$). (d) The correct segmentation is obtained with additional markers, which should never be used to recompute weights. (e) The membership image $\hat{M}_1$ is destroyed if we recompute weights from the additional markers, affecting (f) the result of segmentation.

mation and delineation make the user lose control over the segmentation process, when there exist ambiguities between object and background properties. Fig. 14, in which the bigger horse is the object of interest, provides an illustration of this phenomenon. Fig. 14a shows the DIFT segmentation from the same markers used for training and delineation. The corresponding membership image $\hat{M}_1$ and weight image $\hat{W}$ used for arc-weight assignment are shown in Fig. 14b and c ($\lambda = 0.5$). Note that the segmentation fails owing to the weak boundary between the bigger and the smaller horses, which have similar image properties. Additional markers can correct segmentation in a differential way (Fig. 14d). However, arc weights should never be recomputed from the new markers. If we do that, the membership image $\hat{M}_1$ gets destroyed (Fig. 14e) and the segmentation results would not be correct (Fig. 14f). This explains the importance of having arc-weight estimation as a separated training step from image segmentation. During training, the user should select the most representative and distinguishable parts of the objects, and leave corrections to the interactive segmentation session, in order to avoid arc-weight estimation based on exceptions.

The visual feedback during training also assists the user in choosing the image segmentation method which is likely to require less markers. Fig. 15a illustrates the DIFT segmentation of the left caudate nucleus in an MR-image, using the same markers for training and delineation. The corresponding membership image $\hat{M}_1$ (Fig. 15b) and weight image $\hat{W}$ (Fig. 15c) for $\lambda = 0.5$ indicate that the arc weights on the object's boundary are not strictly higher than inside and outside the boundary. However, when they indeed are greater, only one internal seed and one external seed are enough to complete segmentation by DIFT. Since this is not the case, segmentation would fail if we remove the external marker on the lateral ventricle (dark part) and additional markers are actually needed to refine the results shown in Fig. 15a. However, arc weights seem to be higher on the object's boundary than inside. This favors other methods such as segmentation by κ-connected components (Fig. 15d), which provides the desired segmentation with only one internal seed. The nearby boundaries with similar properties would make more than two seeds required to complete segmentation with live wire (Fig. 15e). Graph-cut segmentation fails because object and background have similar properties (Fig. 15f). Correction in this case is impractical.

In order to validate the synergistic arc-weight estimation method, we need to show that it can really improve accuracy and efficiency (in terms of the amount of user help required) of a given segmentation method as compared to the direct approach based only on w_i, which does not require user assistance. We demonstrate this for one of the methods, namely DIFT, described in Section 4.2, instead of evaluating all methods in this manner since the latter is not likely to generate new insight. We used DIFT with ($\lambda = 0.5$) and without $w_o(\lambda = 0)$ to segment different objects in 100 MRI and CT slice images. Among these, 40 slice images came from MRI-T1 acquisition simulations of phantoms (available at the BrainWeb site[2] [10]), 40 slice images were selected from CT cervical

[2] URL: http://www.bic.mni.mcgill.ca/brainweb/

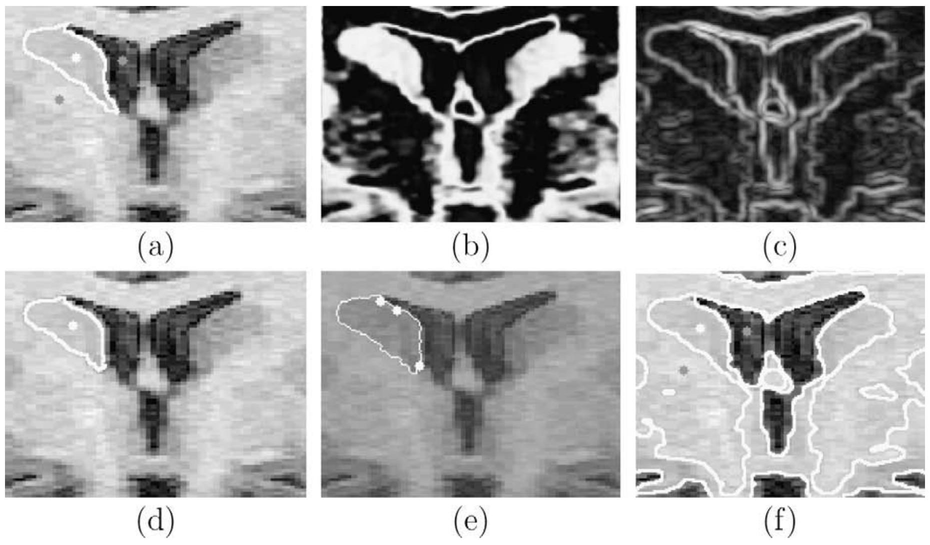

Fig. 15. (a) The same training markers are used to delineate the left caudate nucleus by the DIFT algorithm. Additional markers are needed to refine segmentation. (b and c) The corresponding membership image $\hat{M}_1$ and weight image $\hat{W}$ for $\lambda = 0.5$. (d–f) Segmentations by κ-connected component, live wire and graph cut.

spine studies of 10 subjects, and 20 slice images came from CT thoracic studies of 10 subjects. This gave us a total of 100 2D-segmentations for each method as shown in Table 1. Sample objects are shown in Fig. 16.

The ground-truth segmentations were available for each object and they were used to compute the following accuracy measures: the normalized number of false positive pixels $m1 = FP/(FP + TN)$, where $FP + TN$ is the sum of the number of false positive and true negative pixels, the normalized number of false negative pixels $m2 = FN/(FN + TP)$, where $FN + TP$ is the sum of the number of false negative and true positive pixels [53], and Dice similarity [13]. Efficiency can be measured by the number of markers (marking actions) required by the user to complete segmentation. The method with less markers requires less user involvement. Note that, in interactive segmentation, accuracy depends on the user's patience for verification and correction. In practice, the user tends to stop the corrective actions when the efforts needed to improve the results increase too much relative to the returned improvement in accuracy. Therefore, high accuracy with less number of markers is highly desirable. The mean and standard deviation of the accuracy and efficiency measures estimated in our experiments are presented for the DIFT with and without w_o in Tables 2 and 3, respectively.

In order to simulate a real environment, the similarity measure should be maximized as much as possible until the corrections become almost manual and impractical (i.e., until only small differences distributed along the boundary remain). In the case of O1 with w_o, about 5–10 markers are needed to achieve more than 90% of Dice similarity. This result is already equivalent to the best obtained without w_o (line 1 of Table 3), and can be further improved by adding more markers for small corrections (line 1 of Table 2). For objects O2 and O3, in general, only 4–5 markers are sufficient to obtain more than 95% of Dice when using w_o, while three times more interactions are required without w_o to reach 90% similarity with the true segmentation. As we keep adding more markers, the results converge to the values listed in Tables 2 and 3. In the case of O4, the results with and without w_o are very similar. We already have more than 91% of Dice similarity with 3 markers and about 96% with 6 markers.

The results indicate that the use of w_o usually provides higher accuracy and higher efficiency. By considering the standard deviation, the DIFT with w_o presented better performance than the DIFT

Table 1
Description, imaging modality and number of slice images for each object used in the experiments. Objects O2 and O3 use sample slices from phantoms with different degrees of noise (N) and inhomogeneity (INU).

Object	Description	Modality	# Images
O1	Spinal-vertebra	CT	40
O2	White matter ($N = 3\%$ and $INU = 20\%$)	MRI-T1	20
O3	White matter ($N = 5\%$ and $INU = 40\%$)	MRI-T1	20
O4	Liver	CT	20

without w_o in all cases, except in the case of object O4, where both were equivalent. In all cases, it was enough to consider only two markers to compute w_o, and these markers were also used to start delineation (i.e., they are counted in Table 2). The use of w_o enhances the desired boundaries and suppresses unwanted borders (see Fig. 17). This explains the reduction of markers required to complete segmentation, as verified in all cases (Tables 2 and 3). Since the remaining errors are distributed along the boundary, the similarity measure for each object will vary according with the perimeter/area ratio. Note that objects with complex shapes (O1) produce lower similarity values as compared to simple shapes (O4).

Although we used the DIFT for the experiments, there may be more adequate methods depending on the application. For example, the visual feedback during the training of object O1 (Fig. 18a-b) indicates that the graph-cut approach is likely to require less markers. Note that object O1 contains several background parts (holes) inside it, and at least one background seed at each hole will be required by the DIFT. Indeed, results similar to those of line 1 in Table 2 can be obtained with the graph-cut approach by using only two markers for training (Fig. 18a) and four markers to remove the background bones during delineation (Fig. 18c). Note, however, that the use of w_o is imperative in the graph-cut approach. By choosing $\alpha = 40$, the method becomes practically a threshold on the membership map (Fig. 18b) followed by corrections. This gives another strong indication of the importance of w_o.

6. Conclusion

We have presented an interactive method for arc-weight estimation, which can be employed effectively by several graph-based segmentation approaches as we demonstrated. Our method ex-

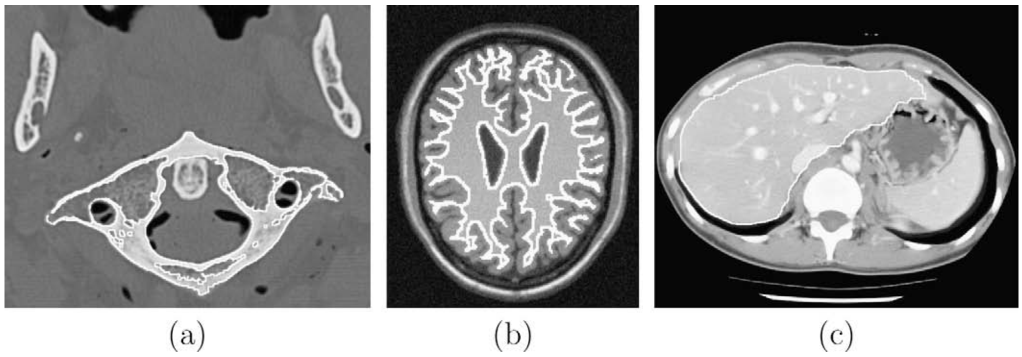

Fig. 16. Sample slice images showing objects O1, O3 and O4. Object O2 is the same as object O3, but on images with less noise and inhomogeneity.

Table 2
Segmentation results using DIFT with $w_o(\lambda = 0.5)$.

	Dice		$m1$		$m2$		# Markers	
	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
O1	0.9329	0.0115	0.00277	0.00073	0.0830	0.0256	17.8	5.5
O2	0.9767	0.0020	0.00732	0.00096	0.0205	0.0038	13.6	4.6
O3	0.9663	0.0028	0.01059	0.00186	0.0297	0.0039	18.7	4.9
O4	0.9841	0.0027	0.00293	0.00118	0.0180	0.0045	11.1	3.0

Table 3
Segmentation results using DIFT without w_o $(\lambda = 0)$.

	Dice		$m1$		$m2$		# Markers	
	Mean	Std dev	Mean	Std dev	Mean	Std dev	Mean	Std dev
O1	0.9058	0.0109	0.00347	0.00054	0.1219	0.0196	31.6	7.4
O2	0.9621	0.0039	0.01836	0.00316	0.0128	0.0030	26.0	9.5
O3	0.9562	0.0041	0.02064	0.00295	0.0168	0.0034	28.9	9.8
O4	0.9862	0.0024	0.00261	0.00081	0.0152	0.0034	13.4	3.5

ploits in a synergistic way the human abilities for recognition and the computer abilities for delineation. While the user draws markers inside each object (including background), arc weights are estimated from image attributes and object information (pixels under the markers), and a visual feedback guides the user's next action toward improving accuracy. Markers should be drawn on the most representative and distinguishable parts of the objects in order to make arc-weight estimation effective. The training markers can be used to start delineation and additional markers selected on similar parts of the objects can correct segmentation, but they should never be used to recompute weights that are estimated in the training step.

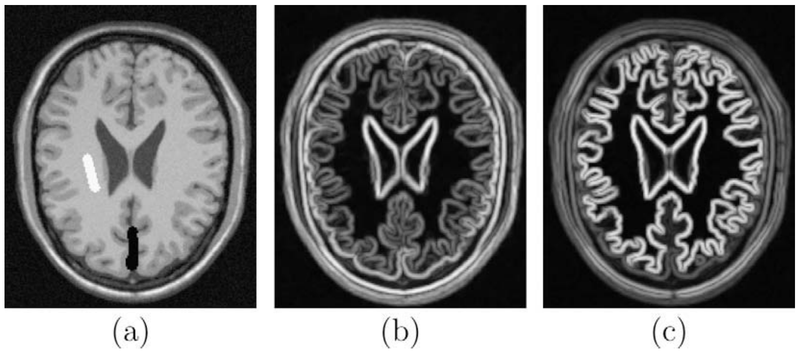

Fig. 17. (a) Sample slice of object O2 with training markers. (b and c) The corresponding weight images $\hat{W}$ for $\lambda = 0$ and $\lambda = 0.5$, respectively. By using w_o a suitable boundary enhancement is obtained for the white matter.

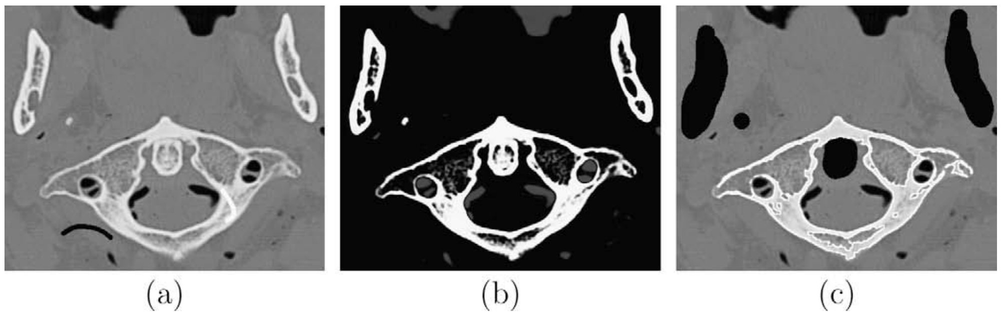

Fig. 18. (a) Sample slice of object O1 with training markers. (b) The corresponding membership image $\hat{M}_1$. (c) Graph-cut segmentation with $\alpha = 40$ and $\lambda = 0.5$. The background bones are easily removed using 4 markers with a larger brush.

We validated the method by showing that the combination of object-based weights with image-based weights usually improves accuracy and efficiency in interactive segmentation, as compared to the same method with only image-based weights. More recently, this approach for arc-weight estimation together with object delineation by DIFT was also successfully used to improve object tracking in video, handling partial occlusion, camera motion, and deformable objects [38]. In this approach, after interactive image segmentation in a first frame, the method combines motion estimation with automatic segmentation of the remaining frames using the proposed framework. In interactive segmentation, the importance of weight visualization to choose the most suitable segmentation approach was also evident from the examples presented in the paper. The selection of the best image attributes, however, requires further investigation. These attributes can be learned from the drawn markers. Another area that requires further work, which is a current limitation of the method, is the user action of how to draw the markers in an effective manner. This is at present somewhat of an art. Our future work will focus on these directions.

Acknowledgments

The authors thank FAPESP and CNPq for the financial support. The images of Figs. 3, 4, 6, 10, and 14 have been obtained from http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping.

References

[1] R. Audigier, R.A. Lotufo, Seed-relative segmentation robustness of watershed and fuzzy connectedness approaches, in: XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), Belo Horizonte, MG, October 2007, IEEE CPS, pp. 61–68.
[2] Xue Bai, Guillermo Sapiro, Distance cut: interactive segmentation and matting of images and videos, in: IEEE International Conference on Image Processing (ICIP), vol. 2, San Antonio, TX, 2007, pp. II-249–II-252.
[3] F.P.G. Bergo, A.X. Falcão, P.A.V. Miranda, L.M. Rocha, Automatic image segmentation by tree pruning, Journal of Mathematical Imaging and Vision 29 (2–3) (2007) 141–162.
[4] S. Beucher, F. Meyer, The morphological approach to segmentation: the watershed transformation, in: Mathematical Morphology in Image Processing, Marcel Dekker, 1993, pp. 433–481 (Chapter 12).
[5] Y. Boykov, V. Kolmogorov, An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (9) (2004) 1124–1137.
[6] Y. Boykov, O. Veksler, R. Zabih, Fast approximate energy minimization via graph cuts, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (11) (2001) 1222–1239.
[7] Y.Y. Boykov, M.-P. Jolly, Interactive graph cuts for optimal boundary and region segmentation of objects in N–D images, in: International Conference on Computer Vision (ICCV), vol. 1, 2001, pp. 105–112.
[8] G. Bueno, O. Musse, F. Heitz, J.P. Armspach, Three-dimensional segmentation of anatomical structures in MR images on large data bases, Magnetic Resonance Imaging 19 (2001) 73–88.
[9] K.C. Ciesielski, J.K. Udupa, P.K. Saha, Y. Zhuge, Iterative relative fuzzy connectedness for multiple objects with multiple seeds, Computer Vision and Image Understanding 107 (3) (2007) 160–182, doi:10.1016/j.cviu.2006.10.005.
[10] D.L. Collins, A.P. Zijdenbos, V. Kollokian, J.G. Sled, N.J. Kabani, C.J. Holmes, A.C. Evans, Design and construction of a realistic digital brain phantom, IEEE Transactions on Medical Imaging 17 (3) (1998) 463–468.
[11] T. Cormen, C. Leiserson, R. Rivest, Introduction to Algorithms, MIT, 1990.
[12] I.J. Cox, S.B. Rao, Y. Zhong, Ratio regions: a technique for image segmentation, in: International Conference on Computer Vision and Pattern Recognition (CVPR), 1996, pp. 557–564.
[13] L.R. Dice, Measures of the amount of ecologic association between species, Ecology 26 (1945) 297–302.
[14] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, 2nd ed., Wiley-Interscience, 2001.
[15] A.X. Falcão, F.P.G. Bergo, Interactive volume segmentation with differential image foresting transforms, IEEE Transactions on Medical Imaging 23 (9) (2004) 1100–1108.
[16] A.X. Falcão, J.K. Udupa, S. Samarasekera, S. Sharma, B.E. Hirsch, R.A. Lotufo, User-steered image segmentation paradigms: live-wire and live-lane, Graphical Models and Image Processing 60 (4) (1998) 233–260.
[17] A.X. Falcão, F.P.G. Bergo, P.A.V. Miranda, Image segmentation by tree pruning, in: Proceedings of the XVII Brazillian Symposium on Computer Graphics and Image Processing, IEEE, October 2004, pp. 65–71.
[18] A.X. Falcão, L.F. Costa, B.S. da Cunha, Multiscale skeletons by image foresting transform and its applications to neuromorphometry, Pattern Recognition 35 (7) (2002) 1571–1582.
[19] A.X. Falcão, B.S. da Cunha, R.A. Lotufo, Design of connected operators using the image foresting transform, in: Proceedings of SPIE on Medical Imaging, vol. 4322, February 2001, pp. 468–479.
[20] A.X. Falcão, P.A.V. Miranda, A. Rocha, A linear-time approach for image segmentation using graph-cut measures, in: Eigth International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS), vol. LNCS 4179, Springer, Antwerp, Belgium, 2006, pp. 138–149.
[21] A.X. Falcão, J. Stolfi, R.A. Lotufo, The image foresting transform: theory, algorithms, and applications, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (1) (2004) 19–29.
[22] A.X. Falcão, J.K. Udupa, F.K. Miyazawa, An ultra-fast user-steered image segmentation paradigm: live-wire-on-the-fly, IEEE Transactions on Medical Imaging 19 (1) (2000) 55–62.
[23] Matthias Farber, Jan Ehrhardt, Heinz Handels, Live-wire-based segmentation using similarities between corresponding image structures, Computerized Medical Imaging and Graphics 31 (7) (2007) 549–560.
[24] L. Ford, D. Fulkerson, Flows in Networks, Princeton University Press, 1962.
[25] H.G. He, J. Tian, Y. Lin, K. Lu, A new interactive segmentation scheme based on fuzzy affinity and live-wire, in: Fuzzy Systems and Knowledge Discovery, vol. 3613, 2005, pp. 436–443.
[26] Hyung W. Kang, G-wire: a livewire segmentation algorithm based on a generalized graph formulation, Pattern Recognition Letters 26 (13) (2005) 2042–2051.
[27] Hyung Woo Kang, Sung Yong Shin, Enhanced lane: interactive image segmentation by incremental path map construction, Graphical Models 64 (5) (2002) 282–303.
[28] Michael Kass, Andrew Witkin, Demetri Terzopoulos, Snakes: active contour models, International Journal of Computer Vision 1 (1987) 321–331.
[29] K. Li, X. Wu, D.Z. Chen, M. Sonka, Optimal surface segmentation in volumetric images: a graph-theoretic approach, IEEE Transactions on Pattern Analysis Machine Intelligence 28 (1) (2006) 119–134.
[30] L. Liang, K. Rehm, R.P. Woods, D.A. Rottenberg, Automatic segmentation of left and right cerebral hemispheres from MRI brain volumes using the graph cuts algorithm, NeuroImage 34 (3) (2007) 1160–1170.
[31] T. Lindeberg, Scale-space theory: a basic tool for analysing structures at different scales, Journal of Applied Statistics 21 (1994) 224–270.
[32] R.A. Lotufo, A.X. Falcão, The ordered queue and the optimality of the watershed approaches, in: Mathematical Morphology and its Applications to Image and Signal Processing, vol. 18, Kluwer, June 2000, pp. 341–350.
[33] D. Lowe, Distinctive image features from scale-invariant keypoints, in: Proceedings of the International Journal of Computer Vision, vol. 20, 2003, pp. 91–110.
[34] F. Malmberg, E. Vidholm, I. Nystrom, A 3D live-wire segmentation method for volume images using haptic interaction, in: Discrete Geometry for Computer Imagery, vol. 4245, 2006, pp. 663–673.
[35] B.S. Manjunath, W.Y. Ma, Texture features for browsing and retrieval of image data, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI – Special issue on Digital Libraries) 18 (8) (1996) 837–842.
[36] D.R. Martin, C.C. Fowlkes, J. Malik, Learning to detect natural image boundaries using local brightness, color, and texture cues, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (5) (2004) 530–549.
[37] F. Meyer, Levelings, image simplification filters for segmentation, Journal of Mathematical Imaging and Vision 20 (1-2) (2004) 59–72.
[38] R. Minetto, J.P. Papa, T.V. Spina, A.X. Falcão, N.J. Leite, J. Stolfi, Fast and robust object tracking using image foresting transform, in: Proceedings of the 16th International Conference on Systems, Signals, and Image Processing, Chalkida, Greece, 2009.
[39] P.A.V. Miranda, A.X. Falcão, Links between image segmentation based on optimum-path forest and minimum cut in graph, Journal of Mathematical Imaging and Vision (2009), doi:10.1007/s10851-009-0159-9.
[40] P.A.V. Miranda, A.X. Falcão, A. Rocha, F.P.G. Bergo, Object delineation by κ-connected components, EURASIP Journal on Advances in Signal Processing (2008) 1–14, doi:10.1155/2008/467928.
[41] N. Mittal, D.P. Mital, Kap Luk Chan, Features for texture segmentation using gabor filters, in: Image Processing and its Applications, in: Seventh International Conference on (Conf. Publ. No. 465), vol. 1, 1999, pp. 353–357.
[42] J.A. Montoya-Zegarra, J.P. Papa, N.J. Leite, R.S. Torres, A.X. Falcão, Learning how to extract rotation-invariant and scale-invariant features from texture images, EURASIP Journal on Advances in Signal Processing (2008) 1–15, doi:10.1155/2008/691924.
[43] S.D. Olabarriaga, A.W.M. Smeulders, Interaction in the segmentation of medical images: a survey, Medical Image Analysis 5 (2) (2001) 127–142.
[44] J.P. Papa, A.X. Falcão, C.T.N. Suzuki, N.D.A. Mascarenhas, A discrete approach for supervised pattern recognition, in: Proceedings of the 12th International Workshop on Combinatorial Image Analysis, vol. LNCS 4958, Buffalo, NY, USA, April 7th–9th 2008, Springer, pp. 136–147.
[45] Alexis Protiere, Guillermo Sapiro, Interactive image segmentation via adaptive weighted distances, IEEE Transactions on Image Processing 16 (4) (2007) 1046–1057.

[46] L.M. Rocha, A.X. Falcão, L.G.P. Meloni, A robust extension of the mean shift algorithm using optimum path forest, in: Proceedings of the 12th International Workshop on Combinatorial Image Analysis, Buffalo, NY, USA, April 7th–9th 2008, RPS, pp. 29–38.
[47] P.K. Saha, J.K. Udupa, Relative fuzzy connectedness among multiple objects: theory, algorithms, and applications in image segmentation, Computer Vision and Image Understanding 82 (2001) 42–56.
[48] P. Salembier, A. Oliveras, L. Guarrido, Antiextensive connected operators for image and sequence processing, IEEE Transactions on Image Processing 7 (4) (1998) 555–570.
[49] P. Salembier, J. Serra, Flat zones filtering, connected operators, and filters by reconstruction, IEEE Transactions on Image Processing 4 (8) (1995) 1153–1160.
[50] S. Sarkar, K.L. Boyer, Quantitative measures of change based on feature organization: eigenvalues and eigenvectors, in: International Conference on Computer Vision and Pattern Recognition (CVPR), 1996, pp. 478–483.
[51] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8) (2000) 888–905.
[52] R.S. Torres, A.X. Falcão, Contour salience descriptors for effective image retrieval and analysis, Image and Vision Computing 25 (1) (2007) 3–13.
[53] J.K. Udupa, V.R. LeBlanc, Y. Zhuge, C. Imielinska, H. Schmidt, L.M. Currie, B.E. Hirsch, J. Woodburn, A framework for evaluating image segmentation algorithms, Computerized Medical Imaging and Graphics 30 (2) (2006) 75–87.
[54] L. Vincent, Morphological grayscale reconstruction in image analysis, IEEE Transactions on Image Processing 2 (2) (2003) 176–201.
[55] Jue Wang, Michael F. Cohen, An iterative optimization approach for unified image segmentation and matting, in: ICCV '05: Proceedings of the 10th IEEE International Conference on Computer Vision, Washington, DC, USA, 2005, IEEE Computer Society, pp. 936–943.
[56] S. Wang, J.M. Siskind, Image segmentation with minimum mean cut, in: International Conference on Computer Vision (ICCV), vol. 1, July 2001, pp. 517–525.
[57] Song Wang, Jeffrey Mark Sinkind, Image segmentation with ratio cut, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (6) (2003) 675–690.
[58] Z. Wu, R. Leahy, An optimal graph theoretic approach to data clustering: theory and its applications to image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (11) (1993) 1101–1113.
[59] G. Wyszecki, W. Stiles, Color Science: Concepts and Methods, Quantitative Data and Formulas, J. Wiley and Sons, 1982.