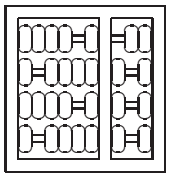


INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Synergistic Arc-Weight Estimation for Interactive Image Segmentation using Graphs

P.A.V. de Miranda *A.X. Falcão*
J.K. Udupa

Technical Report - IC-08-21 - Relatório Técnico

September - 2008 - Setembro

Synergistic Arc-Weight Estimation for Interactive Image Segmentation using Graphs

P.A.V. de Miranda
LIV – Institute of Computing – Unicamp
CP 6176, 13084-971, Campinas, SP, Brazil

A.X. Falcão
LIV – Institute of Computing – Unicamp
CP 6176, 13084-971, Campinas, SP, Brazil

J.K. Udupa
MIPG, Department of Radiology, University of Pennsylvania

Abstract

We introduce a framework for synergistic arc-weight estimation and show its application to several interactive segmentation approaches using graphs. The method is a dynamic training process, 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 next user's action. We demonstrate the advantages of the proposed framework with respect to methods that do not exploit object information for arc-weight estimation and methods that recompute weights during delineation, making the user to loose control over the segmentation process.

1 Introduction

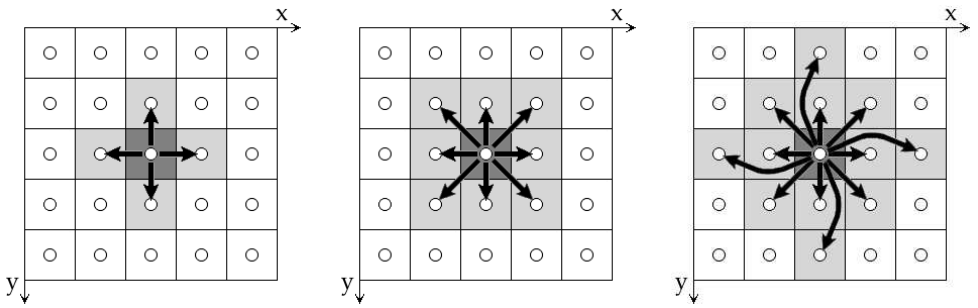
In image processing and computer vision, there are several situations in which user interaction is required to obtain effective image segmentation. The high-level, application-domain-specific knowledge of the user is often required in medical image analysis [13, 8, 25, 36] because of poorly defined structures, and in the digital matting of natural scenes [3, 48], because of their heterogeneous nature.

Image segmentation involves two tightly coupled tasks: *recognition* and *delineation* [13]. 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 [18, 11, 7, 38].

In operator-assisted synergistic segmentation, the user usually adds/removes markers (seed pixels) 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. It may represent different core concepts such as similarity, speed function, affinity, cost, distance, etc; depending on different frameworks used such as watershed, level sets, fuzzy connectedness, graph cuts, etc. Their accurate delineation with minimum user intervention strongly depends on a suitable arc-weight estimation, which usually takes into account image attributes and/or object information from markers selected by the user during segmentation [7, 38]. 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 the same as those used to recompute weights, because the user loses control over segmentation when local interventions for correction destroy other parts where the user was already satisfied with the segmentation results.

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 seeds selected during segmentation are never used to modify/destroy arc-weight assignment. We demonstrate the advantages of this approach using 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) [17]. Note that our aim is not to compare segmentation methods, but to show that they can all benefit from the same procedure of arc-weight assignment, in some cases followed by the complement of the weights. 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 [13, 18]; the local affinities in relative-fuzzy connectedness [40] 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 [27, 5, 11]; 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 [4]; and the arc weights in graph-cut segmentation be lower across the object's boundary than in its interior and exterior [7, 44, 49]. 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 is affected 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.

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



(a) (b) (c)

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

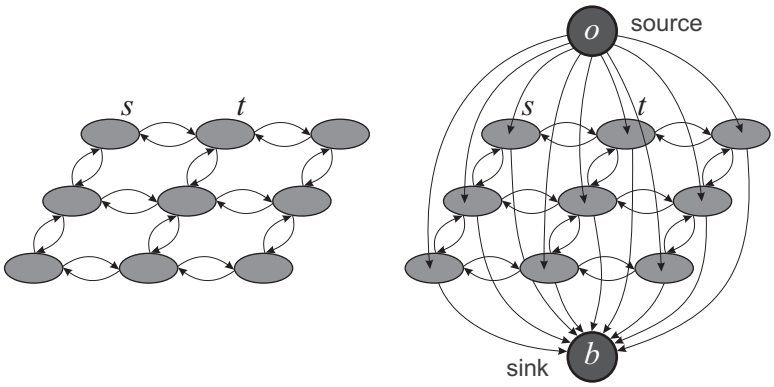
attributes and object information provided by user-selected markers. Section 4 describes several interactive segmentation methods based on that arc-weight assignment, and the main advantages of the synergistic approach are demonstrated in Section 5. Our conclusions are stated in Section 6.

2 Basic Concepts on Image Graphs

An image $\hat{I}$ is a pair $(D_I, \vec{I})$ where $D_I \subset Z^n$ is the image domain and $\vec{I}(s)$ assigns a set of m scalars $I_b(s)$, $b = 1, 2, \ldots, m$, to each pixel $s \in D_I$. 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 b is removed for gray images.

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

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



(a) (b)

Figure 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.

3 Synergistic arc-weight estimation

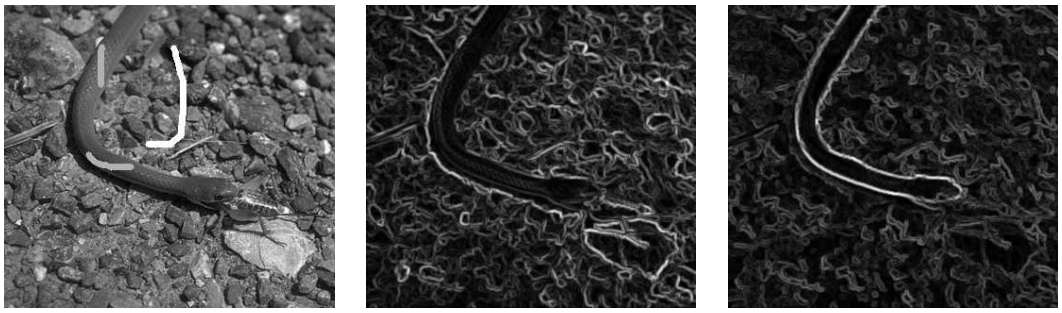
Arc-weight estimation takes into account image attributes and object information in order to capture the discontinuities between object and background. Let v be an algorithm which extracts attributes (color, gradient, texture) from any pixel $s \in D_I$ and returns a vector $\vec{v}(s)$. In the simplest 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 [52] and/or compute texture attributes around each pixel from the results of the image convolved with a bank of filters [26, 30, 34, 28, 35]. Other options are discussed in Section 3.2. For c objects $l = 1, 2, \ldots, c$, including the background as object c without loss of generality, the weight $w(s,t)$ assigned to each arc $(s,t) \in \mathcal{A}$ is a linear combination of an image-based weight $0 \leq w_i(s,t) \leq K$ and an object-based weight $0 \leq w_o(s,t) \leq K$, which takes into account all c objects.

$$w(s,t) = \lambda w_o(s,t) + (1-\lambda) w_i(s,t) \quad (1)$$

where $0 \leq \lambda \leq 1$. The weights $w_i(s,t)$ exploit only image attributes to capture discontinuities 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 capture discontinuities 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 $\hat{W} = (D_I, W)$, where

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

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



(a) (b) (c)

Figure 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 ($\lambda = 0.8$).

for all $s \in D_I$. 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 (Figure 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 (Figures 7 and 14).

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

3.1 Object-based weight assignment

Let $d(s,t) \geq 0$ 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 extracted 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 [30, 35]. A pair (v, d) then describes how the pixels of a dataset are distributed in the attribute space and we call it a *descriptor*.

Let $\mathcal{S}_l \subset D_I$ 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, $s \in D_I$, and $t \in S_l$ is a k-nearest neighbor of s in the attribute space.

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

We expect that the mean distance $\bar{d}(s, \mathcal{A}_{k,l}(s))$ between s and its k adjacents in $\mathcal{S}_l$, $l =$

$1, 2, \ldots, c$, be the smallest for pixels of the same object of s.

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

The posterior probability $\mathcal{P}(l \mid \vec{v}(s))$ can then be expected to be proportional to the total mean distance $\sum \bar{d}(s, \mathcal{A}_{k,i}(s))$ for $i = 1, 2, \ldots, c$, and $i \neq l$.

$$\mathcal{P}(l \mid \vec{v}(s)) \approx K \frac{\sum_{\forall i=1,2,\ldots,c \mid\ i \neq l} \bar{d}(s, \mathcal{A}_{k,i}(s))}{\sum_{\forall i=1,2,\ldots,c} \bar{d}(s, \mathcal{A}_{k,i}(s))}. \quad (5)$$

For multiple objects, the user should keep on drawing markers inside the dark regions of object i in each image $\mathcal{P}(l = i \mid \vec{v}(s))$, $i = 1, 2, \ldots, c$, until that object becomes brighter than the rest in the probability image. For the sake of simplicity, all examples in this paper use only "object and background" type of situation. In this case, $\mathcal{P}(l = 1 \mid \vec{v}(s)) = K - \mathcal{P}(l = 2 \mid \vec{v}(s))$, and then we can, and need to, show only $\mathcal{P}(l = 1 \mid \vec{v}(s))$ with internal and external markers (Figure 4). As the user adds markers, the estimation improves and the object becomes increasingly distinguished (brighter) from the background (darker).

We could have estimated $\mathcal{P}(l \mid \vec{v}(s))$ by Baye's Theorem, by directly computing $\mathcal{P}(l \mid \vec{v}(s))$ from the markers and the distances between s and its adjacents in $\mathcal{A}_{k,l}(s)$ in the attribute space. A similar approach to compute probability density functions is described in [39]. We compared with that approach and the results were equivalent to those obtained by Equation 5, which is simpler and more efficient.

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.

$$\vec{G}_l(s) = \sum_{\forall t \in \mathcal{A}(s)} [\mathcal{P}(l \mid \vec{v}(t)) - \mathcal{P}(l \mid \vec{v}(s))]\vec{st} \quad (6)$$

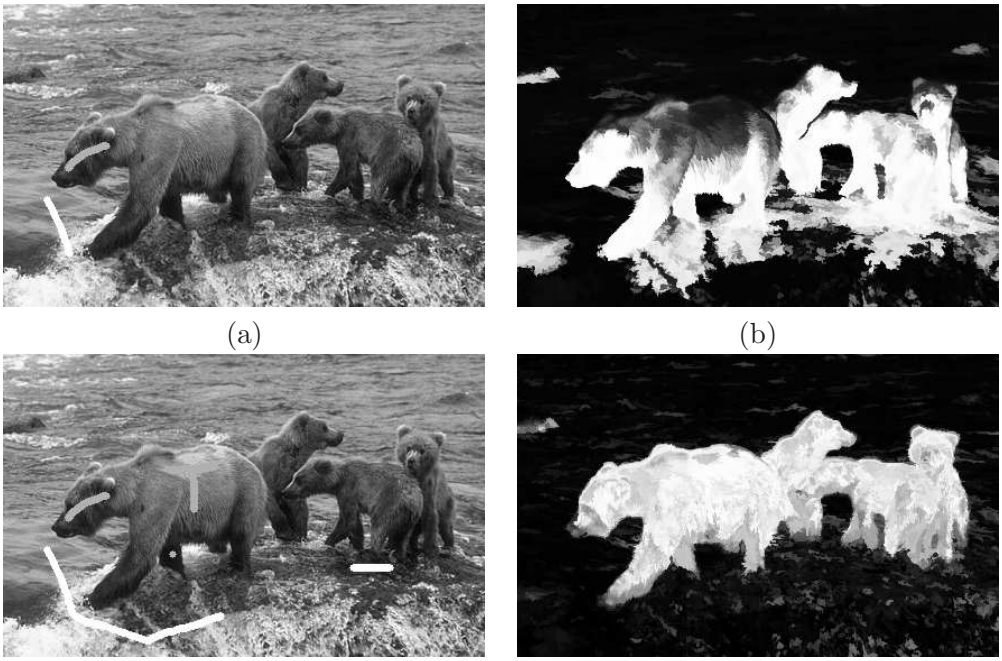
where $\vec{st}$ is the unit vector connecting s to t in the image domain. For a 2D Euclidean adjacency $\mathcal{A}$ with $\rho = \sqrt{2}$, the gradient vector $\vec{G}_l(s)$ is estimated from the vectorial sum of the first derivatives of $\mathcal{P}(l \mid \vec{v}(s))$ along the 8 directions, rather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

For each arc $(s,t) \in \mathcal{A}$, 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.

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

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

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



(a) (b) (c) (d)

Figure 4: Image $\mathcal{P}(l = 1 \mid \vec{v}(s))$, where the desired object is a family of bears. (a-b) An initial marker selection and the corresponding image $\mathcal{P}(l = 1 \mid \vec{v}(s))$. (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 $\mathcal{P}(l = 1 \mid \vec{v}(s))$.

As the user adds markers, the size of the union set $\mathcal{Z} = \bigcup_{l=1,2,\ldots,c} \mathcal{S}_l$ increases and Equation 5 becomes computationally more expensive to assign probabilities to all pixels in D_I. On the other hand, we do not need quantity, but quality, in choosing pixels for $\mathcal{Z}$. 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 $\mathcal{Z}$ by selecting a subset $\mathcal{Z}_1 \subset \mathcal{Z}$ of the most representative pixels. These pixels are defined as those that maximize the classification accuracy of the remaining set of pixels $\mathcal{Z}_2 = \mathcal{Z} \backslash \mathcal{Z}_1$, using the maximum $\mathcal{P}(l \mid \vec{v}(s))$ as the decision rule for the pixels $s \in \mathcal{Z}_2$ with respect to its adjacents $\mathcal{A}_{k,l}(s) \subset \mathcal{Z}_1$ by Equation 5.

The set $\mathcal{Z}$ is divided into two subsets, $\mathcal{Z}_1$ and $\mathcal{Z}_2$, by randomly selecting the same percentage of pixels from each object. Set $\mathcal{Z}_1$ has a maximum size (e.g., 100 pixels). When the number of seeds is less than that maximum size, we may divide $\mathcal{Z}$ into 50% for $\mathcal{Z}_1$ and 50% for $\mathcal{Z}_2$. The maximum $\mathcal{P}(l \mid \vec{v}(s))$ is used to classify the pixels in $\mathcal{Z}_2$. This process is repeated for each k from 1 to k_{max} (Equation 9) in order to obtain the best value of k for the given $\mathcal{Z}_1$. The misclassified pixels with the best k are randomly replaced by pixels of the same object in $\mathcal{Z}_1$. The whole process is repeated over a few iterations T (e.g., $T = 5$) and the pair $(\mathcal{Z}_1, k)$ of maximum accuracy is selected as output (see Algorithm 1).

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

Algorithm 1 – Learning Algorithm

Input:	Initial sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$, number T of iterations, and the descriptor (v, d).
Output:	The pair $(\mathcal{Z}_1^*, k^*)$ 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 Equation 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 $1 \leq L(s) \leq c$, *as described above.*
10. If $s \in \mathcal{S}_l$ *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 Equation 12.*
15. If $Acc \geq Acc^*$, *then*

16. *$Acc^* \leftarrow Acc$, $M^* \leftarrow M$, $\mathcal{Z}_1^* \leftarrow \mathcal{Z}_1$ and $k^* \leftarrow k$.*
17. *Save $(\mathcal{Z}_1^*, k^*)$ and set $A(i) \leftarrow Acc^*$.*
18. *While $M^* \neq \emptyset$*
19. *$M^* \leftarrow M^* \backslash s$*
20. *Replace s by a randomly selected pixel of the same*
21. *class in $\mathcal{Z}_1$.*
22. *Select the instance of $(\mathcal{Z}_1^*, k^*)$ with maximum accuracy $A(i)$, $i = 1, 2, \ldots, T$.*

Accuracy is measured as suggested in [37], by taking into account the fact that objects may have different sizes in $\mathcal{Z}_2$. 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 computed as follows. Let $N_2(l)$ be the number of pixels of $\mathcal{S}_l$ in $\mathcal{Z}_2$. We first define

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

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 $\mathcal{Z}_2$, and $FN(l)$ is the number of pixels from the object l that were incorrectly classified as being from other objects in $\mathcal{Z}_2$. 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 written 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 $0 \leq d(s,t) \leq K$ as image-based weight $w_i(s,t)$ in Equation 1. It is also possible to learn the posterior probability of a pixel (or arc) to be on a boundary from local image attributes [31]. We present another interesting option based on image smoothing at several scales.

Multscale image smoothing can be accomplished by linear convolutions with Gaussians [26] and/or levelings [32, 47, 42, 41]. Except for Figure 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., $S = 4$). Gaussian filters provide smoother contours than morphological reconstructions, but the latter may be preferable to better conserve the natural shape indentations and profusions. In Figure 15, we illustrate the contour smoothness obtained by Gaussian filters with means equal to 0 and standard deviations $\sigma = \frac{r}{3}$ for scales $r = 1, 2, \ldots, S = 6$.

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 $s \in D_I$ and band $b = 1, 2, \ldots, m$. The idea is the same as in Equation 6, where $\mathcal{A}$ may be Euclidean with $\rho = \sqrt{2}$.

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

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

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 $\hat{L} = (D_I, L)$, in which each label $1 \leq L(s) \leq c$ assigns a pixel $s \in D_I$ 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 awkward, so we will present only a short description with their graph parameters as a function of $w(s,t)$ and $\mathcal{P}(l \mid \vec{v}(s))$. The methods based on optimum paths are described by using the image foresting transform (IFT) [17]. 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 between pixels [17].

In a given image graph $(D_I, \mathcal{A})$, 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 $t \in D_I$, $\pi_t = \langle t \rangle$ being a trivial path. A path π_t is *optimum* under a *path-value function* $f(\pi_t)$, when $f(\pi_t) \leq f(\tau_t)$ for any other path τ_t. The IFT computes an *optimum-path forest* P by minimizing (or maximizing) Equation 15 for every $t \in D_I$.

$$V(t) = \min_{\forall \pi_t in (D_I, \mathcal{A})} \{f(\pi_t)\} \quad (15)$$

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 $t \in D_I$, the IFT algorithm (a generalized Dijkstra's algorithm [9], implemented in linear time) 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 $(s,t) \in \mathcal{A}$ (Figure 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 $P(R(t)) = nil$ (Figure 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 (Figure 5c).

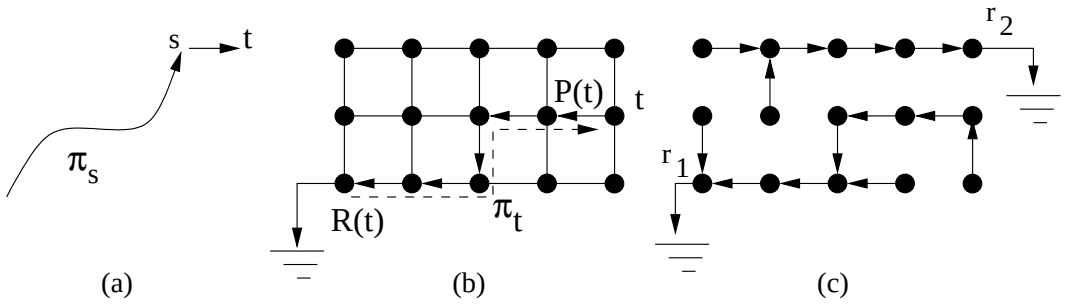


Figure 5: (a) Path $\pi_t = \pi_s \cdot \langle s,t \rangle$ indicates the extension of path π_s by an arc $(s,t) \in \mathcal{A}$. (b) A 4-neighborhood graph showing a path π_t (dashed line) represented in backwards, where $P(t)$ is the predecessor node of t and $R(t)$ is the root pixel. (c) A forest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

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 [15, 14, 39, 37, 4, 45]. The IFT algorithm is an optimum region (path) growing process from the roots of the forest (Figure 6). Variants can also compute on-the-fly other informations, such as a root label for each pixel [27, 11], the propagation order of the pixels [33], the area of the wavefronts of same path value [33], and a graph-cut measure for the border of the growing regions [16].

Particularly, the image segmentation methods described in the next sections adopt the 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 $\mathcal{Z} = \bigcup_{l=1,2,\ldots,c} \mathcal{S}_l$ of selected markers. We note that, the IFT algorithm runs in linear time independently of $|\mathcal{Z}|$.

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

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

where $0 \leq H(t) < \infty$ is a handicap value and $0 \leq w(s,t) \leq K$ 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.

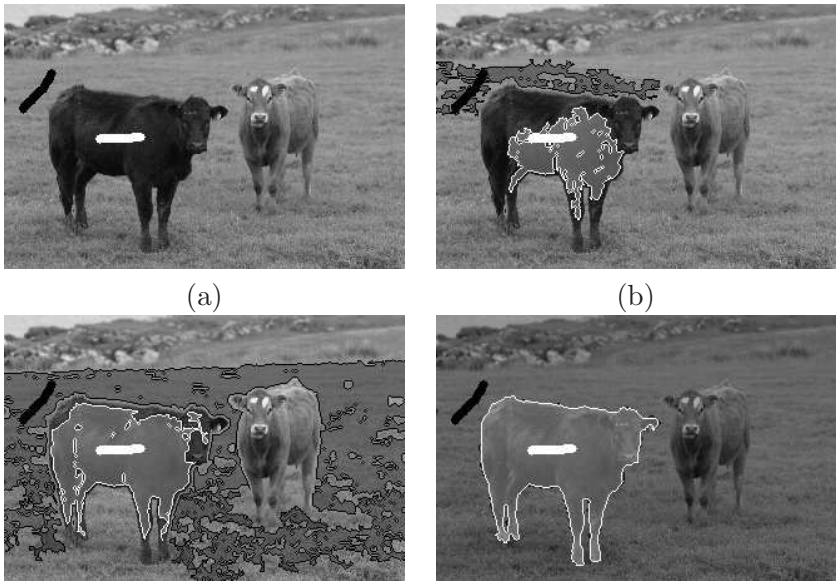


Figure 6: (a) An initial marker selection for segmentation. (b-d) The IFT region growing. (d) The regions meet each other at the object's boundary.

4.2 Segmentation by differential IFT (DIFT)

Multiple objects can be obtained by competition among markers in $\mathcal{S}_l$, $l = 1, 2, \ldots, c$. 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 (Figure 6). Additional seeds are required when this condition is not fully satisfied. Each seed $r \in \mathcal{Z}$ 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 [5, 27] and relative-fuzzy connectedness [40, 46]. The formal relation that exists between these approaches is studied in [2]. The same strategy with f_2 (e.g., $H(t) = 0$) would be a segmentation by weighted distance transform [3, 38].

In any case, the user may want to add/remove markers to correct the segmentation results (Figure 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 [11] (sublinear time in practice).

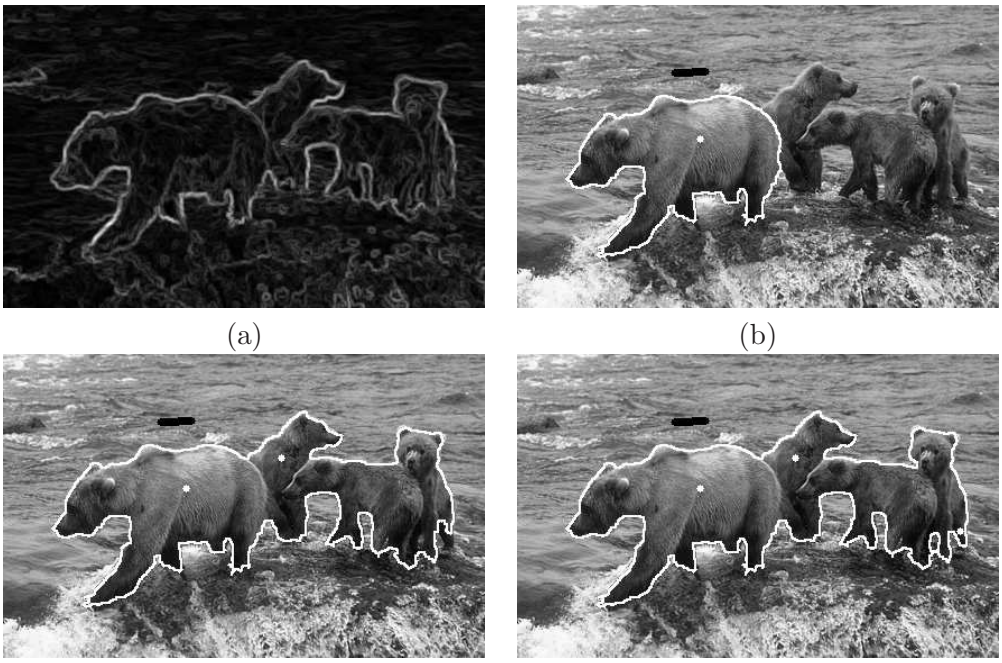


Figure 7: (a) A weight image $\hat{W}$ obtained from Figure 4 using $\lambda = 0.4$. (b) An initial marker selection and segmentation. (c) An additional seed is inserted in order to include the other bears. (d) The final segmentation after some small corrections.

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 [33]. The IFT with f_1 (with $H(t) = 0$) propagates wavefronts $\mathcal{W}_u(s)$ of same optimum-path value u around each seed s, following an increasing order of values $u = 0, 1, \ldots, K$. The maximal extent of a seed inside an object is defined by a κ_s value as $\bigcup_{u=0,1,\ldots,\kappa_s} \mathcal{W}_u(s)$ (Figure 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 $\kappa_s + 1$ (Figure 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 $u = 0, 1, \ldots, K$ during propagation (Figure 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 (Figure 8d).

Note that, since the internal seeds also compete among themselves, with distinct κ_s values, the method can work even when leaking occurs before the object is fully segmented (Figure 8b). The method usually reduces the number of external seeds required to complete

segmentation [33], 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.

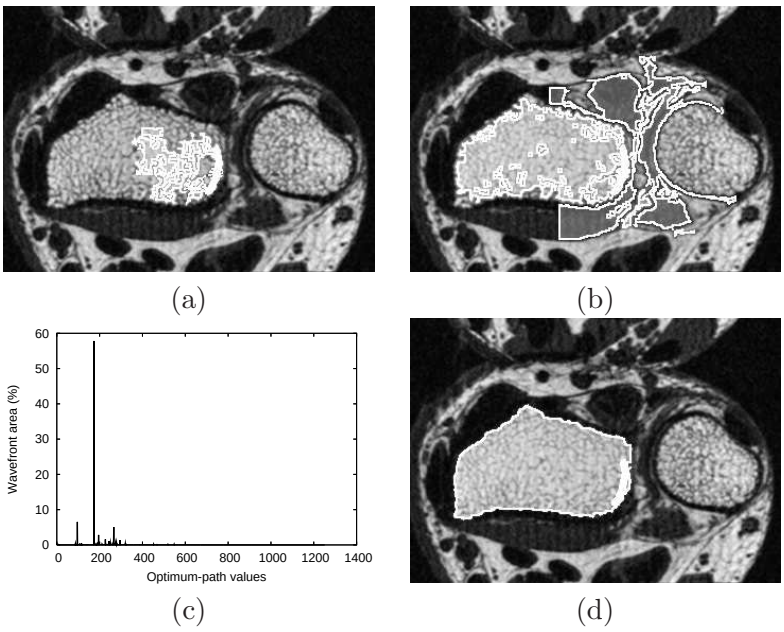


Figure 8: (a-b) The IFT region growing from internal seeds. There is a burst in the size of the wavefront when an optimum path reaches the background. (c) The total wavefront area for each optimum-path value $u = 1, 2, \ldots, K$ during propagation. (d) The resulting segmentation with κ-connected components.

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 [12, 4]. 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 [12] or automatically [4]. 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 [4], there is no competition with external markers. They are called an external set $\mathcal{B}$, which is used to detect all leaking arcs automatically. In most cases the set $\mathcal{B}$ is the image's border, but the user can also add external pixels to $\mathcal{B}$ or internal seeds, if it is 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 $\mathcal{B}$ (Figure 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 (Figure 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 [4].

(a) (b)

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

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 [51] 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 has a bias toward small boundaries, and subsequently, other objective functions, such as average cut [10], mean cut [49], average association [43], normalized cut [44], ratio cut [50], and energy functions [7] have been proposed to circumvent this problem.

Interactive segmentation using the min-cut/max-flow algorithm [7, 25] uses extended image graphs (Figure 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 [20, 6] is then used to speed up computation of the minimum-cut boundary according to the following equation:

$$\begin{aligned} E(\hat{L}) &= \sum_{\forall (s,t)\in\mathcal{A} \mid L(s)=1, L(t)=2} K - w(s,t) \\ &+ \sum_{\forall s\in D_I \mid L(s)=1} w(s,b) + \sum_{\forall s\in D_I \mid L(s)=2} w(o,s) \end{aligned} \tag{19}$$

where $w(s, b)$ and $w(o, s)$ can be computed based on the probabilities given in Equation 5:

$$w(o,s) = \alpha \cdot \mathcal{P}(l=1 \mid \vec{v}(s)) \quad (20)$$
$$w(s,b) = \alpha \cdot \mathcal{P}(l=2 \mid \vec{v}(s)) \quad (21)$$

Here, $\alpha \geq 0$ specifies the relative importance of the arcs with the virtual nodes versus the arcs between pixels (Figure 10).

(a) (b) (c)

Figure 10: Graph-cut segmentation with $\alpha = 20$ and $\lambda = 0.5$. (a) Marker selection for training and the result of segmentation. (b) The probability image $\mathcal{P}(l=1 \mid \vec{v}(s)) = \bar{\mathcal{P}}(l=2 \mid \vec{v}(s))$ used in $w(o,s)$ and $w(s,b)$. (c) The weight image $\hat{W}$ that reflects $w(s,t)$.

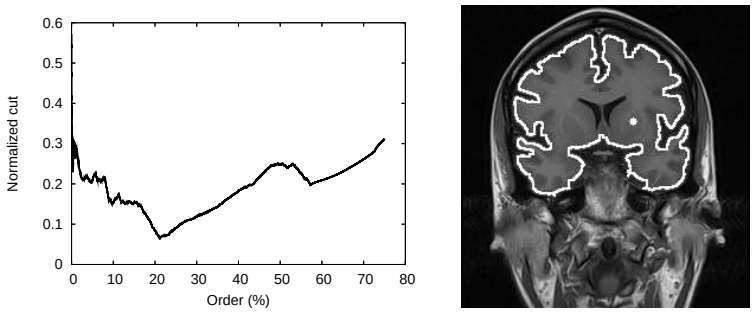
If the algorithm fails in delineating the desired boundary, the user forces arc weights with source and sink by adding markers inside and outside the object [7]. The problems related to the simultaneous segmentation of multiple objects are discussed in [1].

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 (Figure 11a). Within this considerably reduced search space, the minimum cut is expected to occur on the object's boundary (Figure 11b). The method has been evaluated for normalized cut, mean cut and energy functions [16]. 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 internal seeds. External seeds make the method equivalent to the segmentation by differential IFT. The reduction of the search space represents a considerable efficiency gain with respect to some graph-cut approaches [44, 50].

4.7 Segmentation by live wire

In order to segment the object with live wire [18], the user selects a starting point on the object's boundary (point s_1 in Figure 12a) and, for any subsequent position of the mouse, the method computes an optimum path from s_1 to that position in real time. As the user



(a) (b)

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

moves the mouse 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 Figure 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 (Figures 12c-d).

The closed contour is an optimum curve that is constrained to pass through a sequence $\langle \mathcal{S}^{(1)}, \mathcal{S}^{(2)}, \ldots, \mathcal{S}^{(N)} \rangle$ of N landmarks (seeds) on the object's boundary, in that order, starting from $\mathcal{S}^{(1)}$ and ending in $\mathcal{S}^{(N)}$, where each set $\mathcal{S}^{(i)}$, $i = 1, 2, \ldots, N$, has a single pixel s_i and $s_1 = s_i$. 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 $s_{i-1} \in \mathcal{S}^{(i-1)}$ as seed, 8-adjacency relation and path-value function f_3 (a variant of f_2).

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

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

where $V(t)$ is the optimum path value of the previous executions, $a > 0$ (e.g., $a = 1.5$), $0 \leq |\vec{G}(s,t)| \leq K$ 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 probability 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 [17, 23, 19, 22, 29, 21].

5 Results and Discussion

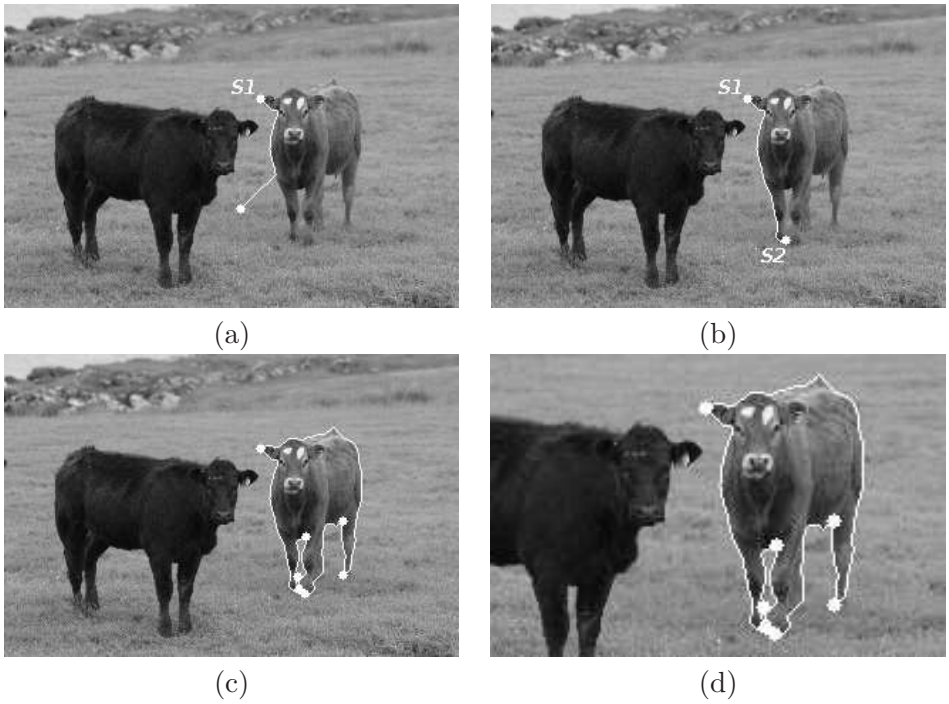
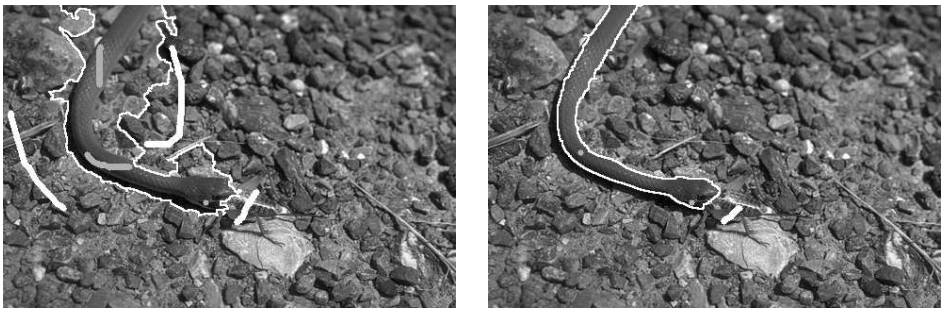


Figure 12: Contour tracking with live wire. (a) Initial point s_1 is selected on the boundary and the user moves the mouse. (b) A second point s_2 is selected on the boundary. (c-d) Final contour with 7 segments.

The examples in the previous sections have shown that the same 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 [24, 5, 44, 49], 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 (Figure 13a) can be drastically improved by incorporating object information as presented in Section 3.1 (Figure 13b).

The arc-weight assignment 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 arc-weight assignment strategy allows fair comparisons among methods. For instance, it is easy to see that any approach based on the weights of Figure 3b will be at a clear disadvantage when compared with those based on the weights of Figure 3c.

Other approaches also incorporate object information into arc-weight estimation [7, 3, 38]. However, the absence of the weight visualization and the use of segmentation markers



(a) (b)

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

for both, arc-weight estimation and delineation, make the user to lose control over the segmentation process, when there exist ambiguities between object and background properties. Figure 14, in which the bigger horse is the object of interest, provides an illustration of this phenomenon. Figure 14a shows the DIFT segmentation from the same markers used for training and delineation. The corresponding probability image $\mathcal{P}(l = 1 \mid \vec{v}(s))$ and weight image $\hat{W}$ used for arc-weight assignment are shown in Figures 14b and 14c ($\lambda = 0.5$). Note that segmentation fails owing to the weak boundary between the bigger and smaller horses, which have similar image properties. Additional markers can correct segmentation in a differential way (Figure 14d). However, arc weights should never be recomputed from the new markers. If we do that, the probability image $\mathcal{P}(l = 1 \mid \vec{v}(s))$ gets destroyed (Figure 14e) and the segmentation results would not be correct (Figure 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. Figure 15a illustrates the DIFT segmentation of the left caudate nucleus in an MR-image, using the same markers for training and delineation. The corresponding probability image $\mathcal{P}(l = 1 \mid \vec{v}(s))$ (Figure 15b) and weight image $\hat{W}$ (Figure 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 Figure 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 (Figure 15d), which provides the desired segmentation with only one internal seed. The nearby boundaries with similar properties would make more than two

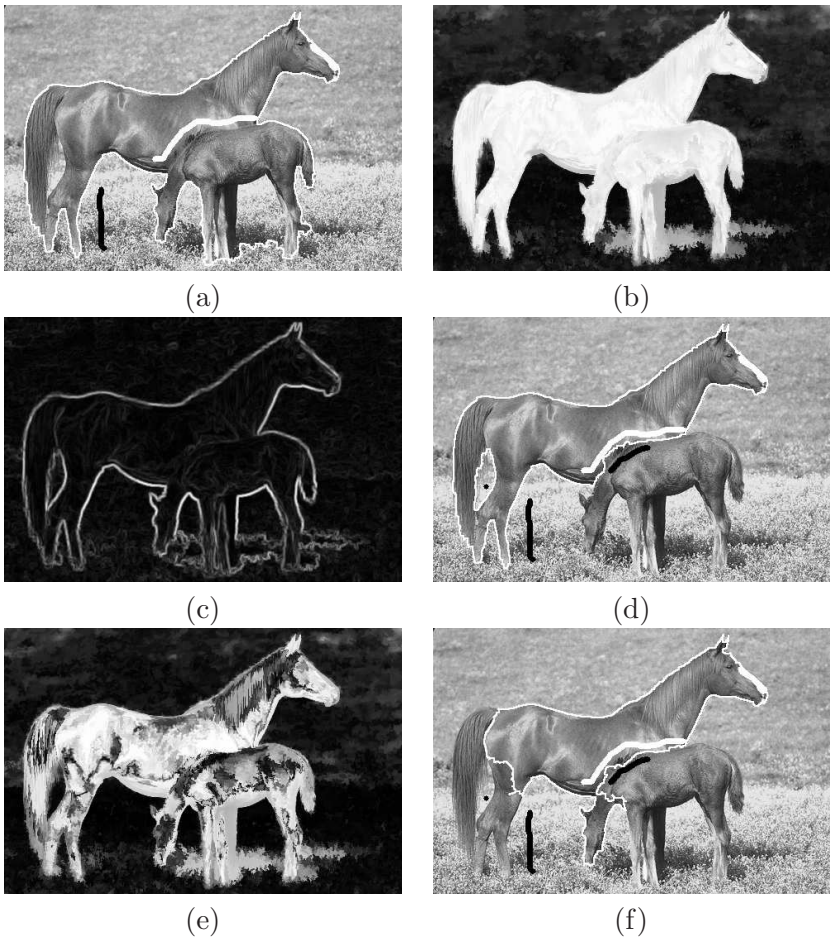


Figure 14: (a) DIFT segmentation using the same markers for training and delineation. (b-c) The respective probability image $\mathcal{P}(l = 1 \mid \vec{v}(s))$ and weight image $\hat{W}$ of the training ($\lambda = 0.5$). (d) The correct segmentation is obtained with additional markers, which should never be used to recompute weights. (e) The probability image $\mathcal{P}(l = 1 \mid \vec{v}(s))$ is destroyed if we recompute weights from the additional markers, affecting (f) the result of segmentation.

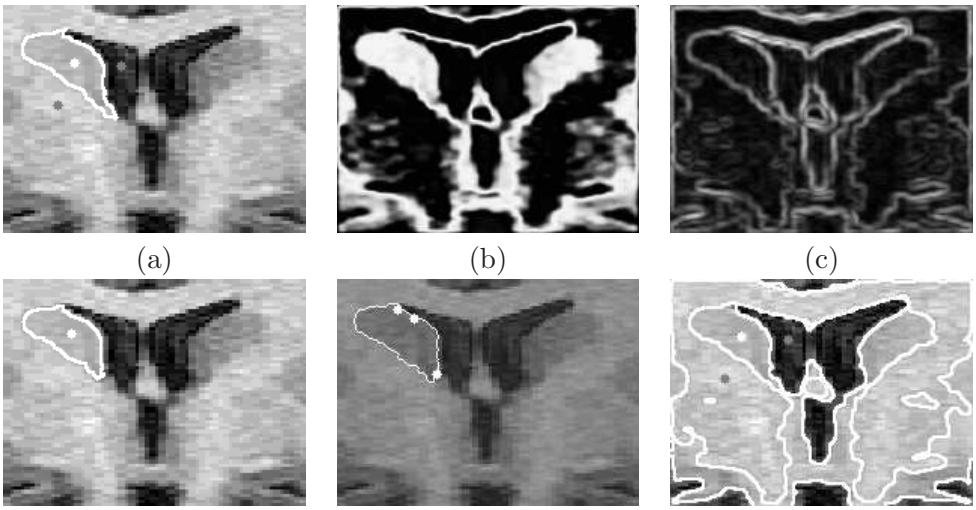


Figure 15: (a) The same training markers are used to delineate the left caudate nucleus by the DIFT algorithm. Additional markers are needed to refine segmentation. (b-c) The corresponding probability image $\mathcal{P}(l = 1 \mid \vec{v}(s))$ and weight image $\hat{W}$ for $\lambda = 0.5$. (d-f) Segmentations by κ-connected component, live wire and graph cut.

seeds required to complete segmentation with live wire (Figure 15e). Graph-cut segmentation fails because object and background have similar properties (Figure 15f). Correction in this case is impractical.

6 Conclusion

We have presented a method for synergistic arc-weight estimation, which can be used in a variety of interactive segmentation scenarios that employ the graph framework. 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. 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. We have also shown the advantages of object-based weights in methods that were designed with only image-based weights and the importance of weight visualization to choose the most suitable segmentation approach.

As a common procedure to assign arc weights and select markers, the proposed framework can be used to find the most suitable segmentation method for a given application. 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

is the user action of drawing markers. 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 Figures 3, 4, 6, 10, and 14 have been obtained from http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping

References

[1] C. Allène, J.Y. Audibert, M. Couprie, J. Cousty, and R. Keriven. Some links between min-cuts, optimal spanning forests and watersheds. In *Mathematical Morphology and its Applications to Signal and Image Processing (ISMM)*, pages 253–264. MCT/INPE, 2007.

[2] R. Audigier and R.A. Lotufo. Seed-relative segmentation robustness of watershed and fuzzy connectedness approaches. In *XX Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 61–68, Belo Horizonte, MG, Oct 2007. IEEE CPS.

[3] Xue Bai and Guillermo Sapiro. Distance cut: interactive segmentation and matting of images and videos. In *IEEE Intl. Conf. on Image Processing (ICIP)*, volume 2, pages II – 249–II – 252, San Antonio, Texas, 2007.

[4] F.P.G. Bergo, A.X. Falcão, P.A.V. Miranda, and L.M. Rocha. Automatic image segmentation by tree pruning. *Journal of Mathematical Imaging and Vision*, 29(2–3):141–162, Nov 2007.

[5] S. Beucher and F. Meyer. The morphological approach to segmentation: The watershed transformation. In *Mathematical Morphology in Image Processing*, chapter 12, pages 433–481. Marcel Dekker, 1993.

[6] Y. Boykov and 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):1124–1137, Sep 2004.

[7] Y.Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 105–112, 2001.

[8] 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*, 19:73–88, 2001.

[9] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT, 1990.

[10] I. J. Cox, S. B. Rao, and Y. Zhong. Ratio regions: a technique for image segmentation. In *Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 557–564, 1996.

[11] A. X. Falcão and F. P. G. Bergo. Interactive volume segmentation with differential image foresting transforms. *IEEE Trans. on Medical Imaging*, 23(9):1100–1108, 2004.

[12] A. X. Falcão, F. P. G. Bergo, and P. A. V. Miranda. Image segmentation by tree pruning. In *XVII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 65–71. IEEE, Oct 2004.

[13] 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*, 60(4):233–260, 1998.

[14] A.X. Falcão, L.F. Costa, and B.S. da Cunha. Multiscale skeletons by image foresting transform and its applications to neuromorphometry. *Pattern Recognition*, 35(7):1571–1582, 2002.

[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*, volume 4322, pages 468–479, Feb 2001.

[16] A.X. Falcão, P.A.V. Miranda, and A. Rocha. A linear-time approach for image segmentation using graph-cut measures. In *8th Intl. Conf. on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, volume LNCS 4179, pages 138–149, Antwerp, Belgium, 2006. Springer.

[17] 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*, 26(1):19–29, 2004.

[18] 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*, 19(1):55–62, 2000.

[19] Matthias Farber, Jan Ehrhardt, and Heinz Handels. Live-wire-based segmentation using similarities between corresponding image structures. *Computerized Medical Imaging and Graphics*, 31(7):549–560, Oct 2007.

[20] L. Ford and D. Fulkerson. *Flows in networks*. Princeton University Press, 1962.

[21] H.G. He, J. Tian, Y. Lin, and K. Lu. A new interactive segmentation scheme based on fuzzy affinity and live-wire. In *FUZZY SYSTEMS AND KNOWLEDGE DISCOVERY*, volume 3613, pages 436–443, 2005.

[22] Hyung W. Kang. G-wire: A livewire segmentation algorithm based on a generalized graph formulation. *Pattern Recognition Letters*, 26(13):2042–2051, Oct 2005.

[23] Hyung Woo Kang and Sung Yong Shin. Enhanced lane: interactive image segmentation by incremental path map construction. *Graphical Models*, 64(5):282–303, Sep 2002.

[24] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1:321–331, 1987.

[25] K. Li, X. Wu, D.Z. Chen, and M. Sonka. Optimal surface segmentation in volumetric images: A graph-theoretic approach. *IEEE Trans. Pattern Analysis Machine Intelligence*, 28(1):119–134, 2006.

[26] T. Lindeberg. Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21:224–270, 1994.

[27] 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*, volume 18, pages 341–350. Kluwer, Jun 2000.

[28] D. Lowe. Distinctive image features from scale-invariant keypoints. In *Proc. of the International Journal of Computer Vision*, volume 20, pages 91–110, 2003.

[29] F. Malmberg, E. Vidholm, and I. Nystrom. A 3D live-wire segmentation method for volume images using haptic interaction. In *DISCRETE GEOMETRY FOR COMPUTER IMAGERY*, volume 4245, pages 663–673, 2006.

[30] B. S. Manjunath and 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):837–42, Aug 1996.

[31] D.R. Martin, C.C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(1), Jan 2004.

[32] F. Meyer. Levelings, image simplification filters for segmentation. *Journal of Mathematical Imaging and Vision*, 20(1-2):59–72, 2004.

[33] P.A.V. Miranda, A.X. Falcão, A. Rocha, and F.P.G. Bergo. Object delineation by κ-connected components. *EURASIP Journal on Advances in Signal Processing*, 2008. to appear.

[34] N. Mittal, D.P. Mital, and Kap Luk Chan. Features for texture segmentation using gabor filters. In *Image Processing And Its Applications. Seventh International Conference on (Conf. Publ. No. 465)*, volume 1, pages 353–357, 1999.

[35] J.A. Montoya-Zegarra, J.P. Papa, N.J. Leite, R.S. Torres, and A.X. Falcão. Rotation-invariant and scale-invariant texture recognition. EURASIP Journal on Advances in Signal Processing*, publisher =*.

[36] S. D. Olabarriaga and A. W. M. Smeulders. Interaction in the segmentation of medical images: A survey. *Medical Image Analysis*, 5(2):127–142, Jun 2001.

[37] J.P. Papa, A.X. Falcão, C.T.N. Suzuki, and N.D.A. Mascarenhas. A discrete approach for supervised pattern recognition. In *Proc. of the 12th Intl. Workshop on Combinatorial Image Analysis*, volume LNCS 4958, pages 136–147, Buffalo, NY, USA, Apr 7th-9th 2008. Springer.

[38] Alexis Protiere and Guillermo Sapiro. Interactive image segmentation via adaptive weighted distances. *IEEE Transactions on Image Processing*, 16(4):1046–1057, Apr 2007.

[39] L.M. Rocha, A.X. Falcão, and L.G.P. Meloni. A robust extension of the mean shift algorithm using optimum path forest. In *Proc. of the 12th Intl. Workshop on Combinatorial Image Analysis*, pages 29–38, Buffalo, NY, USA, Apr 7th-9th 2008. RPS.

[40] 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*, 82:42–56, 2001.

[41] P. Salembier, A. Oliveras, and L. Guarrido. Antiextensive connected operators for image and sequence processing. *IEEE Trans. on Image Processing*, 7(4):555–570, Apr 1998.

[42] P. Salembier and J. Serra. Flat zones filtering, connected operators, and filters by reconstruction. *IEEE Trans. on Image Processing*, 4(8):1153–1160, Aug 1995.

[43] S. Sarkar and K. L. Boyer. Quantitative measures of change based on feature organization: eigenvalues and eigenvectors. In *Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 478–483, 1996.

[44] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, Aug 2000.

[45] R.S. Torres and A.X. Falcão. Contour salience descriptors for effective image retrieval and analysis. *Image and Vision Computing*, 25(1):3–13, Jan 2007.

[46] J.K. Udupa, P.K. Saha, and R.A. Lotufo. Relative fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24:1485–1500, 2002.

[47] L. Vincent. Morphological grayscale reconstruction in image analysis. *IEEE Trans. on Image Processing*, 2(2):176–201, Apr 1993.

[48] Jue Wang and Michael F. Cohen. An iterative optimization approach for unified image segmentation and matting. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 936–943, Washington, DC, USA, 2005. IEEE Computer Society.

[49] S. Wang and J.M. Siskind. Image segmentation with minimum mean cut. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 517–525, Jul 2001.

[50] Song Wang and Jeffrey Mark Sinkind. Image segmentation with ratio cut. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(6):675–690, Jun 2003.

[51] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its applications to image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(11):1101–1113, Nov 1993.

[52] G. Wyszecki and W. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulas*. J. Wiley and Sons, 1982.