Intelligent understanding of user input applied to arc-weight estimation for graph-based foreground segmentation

Thiago Vallin Spina, Alexandre Xavier Falcão
Institute of Computing – University of Campinas (UNICAMP),
13083-852, Campinas, SP, Brazil.
tvspina@liv.ic.unicamp.br, afalcao@ic.unicamp.br

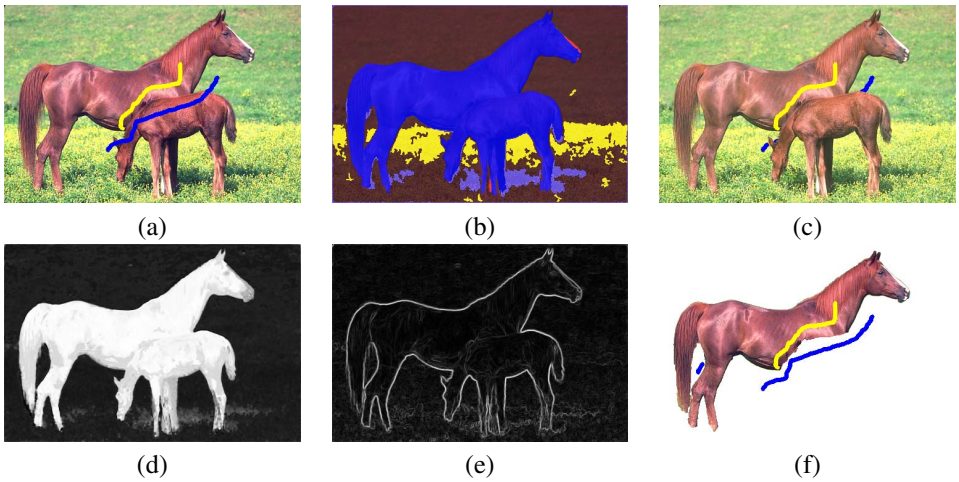

Figure 1. (a) First set of markers created by user inside and outside the object. (b) Clustering based on features extracted from the image. (c) Marker pixels automatically selected on distinct foreground and background regions to learn object information. (d) Object membership map obtained through fuzzy classification. (e) Weight image where brighter regions represent higher arc weights. (f) Final segmentation results by differential IFT, with one more added marker.

***Abstract*—We present an intelligent approach for understanding user interaction to simplify the interface of graph-based image segmentation. The user draws a set of markers (strokes) over the object and the background, and our method automatically determines a subset of these pixels with dissimilar image properties for arc-weight estimation. Arc-weight estimation combines object information learned from the set of selected pixels with image information, to make object delineation more effective. Our method differs from approaches that recompute the arc weights carelessly during delineation, by further interpreting user interaction to determine when and where to recompute them. Furthermore, we build our framework around the *image foresting transform* (IFT), by taking advantage of its operators for supervised fuzzy classification, clustering, and object delineation. We evaluate our framework using a dataset with 50 natural images and by comparing it against another recent IFT-based method, which computes arc weights in a separated step of user interaction for more effective segmentation.**

***Keywords*-Graph-based image segmentation, intelligent arc-weight estimation, image foresting transform, fuzzy supervised classification, clustering, multiscale image filtering.**

I. Introduction

Automatic object segmentation on natural scenes is still an open problem in image processing and computer vision. The lack of information regarding the object's location, shape, and appearance makes user interaction necessary to obtain an effective result. Thus, a challenge is to minimize user involvement and the time required for segmentation without compromising accuracy and precision. As noted by [1] the image segmentation task can be divided into two tightly coupled tasks: recognition and delineation.

The whereabouts of the object in the image and the verification of its delineation are examples of recognition tasks. Delineation defines the object's spatial extent. For a user, determining the object's position is a simple matter of selecting markers (strokes) using mouse clicks, but manual delineation is a burdensome task with hard-to-reproduce results. Therefore, a great number of works have exploited the synergism between the human high-level knowledge about the object and the computer's capability of precise, even when not accurate, delineation [2], [3], [4], [5], [6], [7]. In such approaches, the user selects markers to locate the object, which is subsequently delineated by the

computer. The user then verifies the result, and performs corrections when necessary by adding/removing markers. Thus, the segmentation accuracy becomes a compromise between the user's patience for verification and correction, and the quality of delineation.

According to [6], most segmentation methods use direct/indirect concepts of image-graph, such as arc weights between pixels. The weight may represent different attribute functionals such as similarity, speed function, affinity, cost, and distance; depending on different frameworks used, such as watershed, level sets, fuzzy connectedness, and graph cuts. The effectiveness of segmentation is due to the quality of such arc-weight estimation. Therefore, to improve the accuracy while decreasing the need for user intervention, the authors in [6] further divide the delineation task into two separated steps: arc-weight estimation and object extraction. Arc-weight estimation aims at computing values that best represent the relation between pixels, by combining image properties with object information learned from strokes drawn by the user. Object extraction uses the arc-weights computed by the previous step, with a new set of user-drawn markers, to finally define the spatial extent of the object. A good arc-weight estimation relies on the choice of object and background markers on regions with dissimilar image properties, while object extraction markers should be drawn wherever necessary. Markers used for extraction should never be used for arc-weight estimation, although the opposite can be true. This is important to prevent the recomputation of the arc weights, which could potentially destroy regions already accepted as correct [6]. However, most methods in the literature do not treat the delineation task as two separated steps and cannot easily cope with the aforementioned problem [3], [4], [2].

We propose an intelligent solution to automatically determine, from user input, if and where the arc weights can be recomputed, such that the accuracy of segmentation is not (or is minimally) affected. Also, the foreground and background of some images are too similar for learning object properties, and our method is able to decide not to do it. This allows a much simpler and intuitive user interface in which a single common set of markers is required for segmentation, while the user's control over segmentation is preserved by keeping the division of the delineation process. Moreover, at anytime the user is allowed to stop the object information relearning process.

The above strategy was accomplished using the *image foresting transform* (IFT) — a tool for the design of image processing operators based on *connectivity functions* in graphs derived from the image [8]. Although segmentation methods based on the max-flow/min-cut algorithm have been actively pursued [2], it is proven that the procedures commonly adopted to circumvent problems with this algorithm actually lead to approximations of the IFT-based segmentation with competition between internal and external markers [9]. Thus, we use this approach with the differential IFT algorithm for object delineation [7], given that it also allows segmentation in linear time and corrections in sublinear time.

Although segmentation of multiple objects by IFT is quite straightforward, we are only interested on the binary case. First, multiscale color and texture features are extracted from the image. Those features are then used to *cluster* the pixels into some regions, using an IFT-based classifier [10], such that a same label is assigned to similar pixels (Figure 1b). When the user draws object and background markers, only the pixels that belong to certain dissimilar clusters are selected for arc-weight estimation (Figure 1c). Object information is learned by a procedure similar to [11], which uses the selected pixels on a variant of an IFT-based supervised classifier [12]. This procedure attributes fuzzy values to each pixel composing an *object membership map*, in which brighter regions are more similar to the foreground (Figure 1d). Arc-weight estimation is represented by a weight image obtained from the linear combination of a gradient computed on the object membership map, with another one calculated from the multiscale image features (Figure 1e). The object is finally extracted by partitioning an 8-neighbor image-graph through the differential IFT [7], using the computed arc-weights and every marked pixel as a seed. The result may be improved by adding/removing markers (Figure 1f). The newly added markers are evaluated using the object membership map, to determine if they might bring new object information. If so, the arc weights are recomputed with caution, in order to preserve the segmented parts already accepted as correct. Then, extraction is repeated once more, and the process continues until a satisfactory result is achieved.

The first contribution of this paper was the development of a graph-based framework for interactive image segmentation, which exploits the IFT algorithm for selection of marker pixels for arc-weight estimation (using clustering); arc-weight estimation (using supervised classification), and for object extraction (the DIFT approach). Our main contribution with respect to other IFT-based approaches, concerns the automatic selection of marker pixels for arc-weight estimation, and the detection of when to do it, by understanding user input. Finally, our last contribution was to compare our framework against the approach in [11], which requires separate user recognition for arc-weight estimation and object extraction as proposed by [6].

Figure 2 depicts the overall scheme of a regular execution of our framework. Given that the marker selection, arc-weight estimation, and object extraction use different IFT-based operators, Section II reviews some important concepts about the image foresting transform. Our segmentation method starts with user interaction by selecting markers on object and background for segmentation. Then, marker pixel selection executes as aforementioned (Section III) to provide

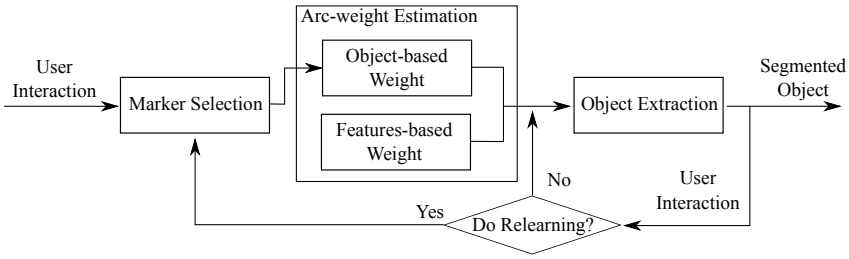

Figure 2. Overall scheme of the framework.

a good input for the arc-weight estimation (Section IV). Object extraction uses all marked pixels and is done afterwards to define the object's spatial extent (Section V). We show some results and evaluate our method in Section VI, by comparing it with the approach in [11] using a dataset with 50 natural images. We state our conclusions in Section VII.

II. Image Foresting Transform

The image foresting transform is a generalization of the Dijkstra's algorithm, that works for multiple sources and smooth path-cost functions [8]. An IFT-based operator requires to first derive a graph from the image. Then, an optimum-path forest is computed according to a connectivity function, followed by a local processing to one or more of its attributes [8].

A. Graphs from images

An image $\hat{I}$ is a pair $(D_{\hat{I}}, \vec{I})$, where $D_{\hat{I}} \subset Z^n$ corresponds to the image domain and $\vec{I}(t)$ assigns a set of m scalars $I_b(t)$, $b = 1, 2, \ldots, m$, to each pixel $t \in D_{\hat{I}}$. We will consider the cases where $n = 2$ and $m = 1, 3$ or 12. In this paper, the triplet $(I_1(t), I_2(t), I_3(t))$ for natural images denotes the L, a, and b values of a pixel t in the *Lab* color space. Multiscale feature extraction essentially transforms an image $\hat{I} = (D_{\hat{I}}, \vec{I})$ into the pair $\hat{F} = (D_{\hat{I}}, \vec{F})$, where $\vec{F}(t) = (F_1(t), F_2(t), \ldots, F_m(t))$ is a feature vector assigned to t. In this work, we apply a Gaussian filter to each Lab component with four different values of σ. The filtered values assign to each pixel a vector with 12 features (i.e., $m = 12$). The smoothing of the pixel values allows a better segmentation of objects on noisy images.

In our approach, a graph $(\mathcal{N}, \mathcal{A})$ may be defined by taking a set $\mathcal{N} \subseteq D_{\hat{I}}$ of pixels as nodes and an *adjacency relation* $\mathcal{A}$ between nodes of $\mathcal{N}$ to form the arcs. We use $t \in \mathcal{A}(s)$ or $(s, t) \in \mathcal{A}$ to indicate that a node $t \in \mathcal{N}$ is adjacent to a node $s \in \mathcal{N}$. Our method uses the following types of adjacency relations, depending on the operator

$$t \in \mathcal{A}_1(s) \quad \text{if} \quad t \neq s, \forall t \in \mathcal{N}, \tag{1}$$

$$t \in \mathcal{A}_2(s) \quad \text{if} \quad \|t - s\| \leq \sqrt{2}, \forall t \in D_{\hat{I}}, \tag{2}$$

$$t \in \mathcal{A}_3(s) \quad \text{if} \quad t \text{ is a } k\text{-nearest neighbor of } s \text{ in the feature space.} \tag{3}$$

Adjacency relation $\mathcal{A}_1$ is used for object learning during the arc-weight estimation, and determines a complete graph on $\mathcal{N}$. $\mathcal{A}_2$ is used for object extraction with $\mathcal{N} = D_{\hat{I}}$, and implies that every 8-neighbor pixel on the image domain is adjacent. $\mathcal{A}_3$ is used by the clustering algorithm and will be detailed later. The arc weights $w(s, t)$ for all $(s, t) \in \mathcal{A}$ are computed from the feature image $\hat{F}$ and selected markers, but in different ways for each IFT-based operator. Note that, given the differences between those three adjacency relations, we use an implicit graph approach in our method.

B. Connectivity functions

A *path* $\pi_t = \langle t_1, t_2, \ldots, t \rangle$ is a sequence of adjacent nodes with terminus at a node t. A path $\pi_t = \pi_s \cdot \langle s, t \rangle$ indicates the extension of a path π_s by an arc (s, t) and a path $\pi_t = \langle t \rangle$ is said *trivial*. A connectivity function f assigns to any path π_t a value $f(\pi_t)$. Examples of such functions are

$$\begin{aligned} f_1(\langle t \rangle) &= H(t) \\ f_1(\pi_s \cdot \langle s, t \rangle) &= \max\{f_1(\pi_s), w(s, t)\}, \end{aligned} \tag{4}$$

$$\begin{aligned} f_2(\langle t \rangle) &= H(t) \\ f_2(\pi_s \cdot \langle s, t \rangle) &= \min\{f_2(\pi_s), w(s, t)\}, \end{aligned} \tag{5}$$

where $H(t)$ is a *handicap* value specific to each IFT-based operator, and $w(s, t)$ represents a fixed cost to the arc (s, t). The functions f_1 and f_2 are dual and propagate, respectively, the maximum and minimum arc weight value along the path. Other connectivity functions are discussed in [8] for several image operators.

C. Optimum-path forest

A path π_t is optimum if $f(\pi_t) \leq f(\tau_t)$ for any other path τ_t in $(\mathcal{N}, \mathcal{A})$. Considering all possible paths with terminus at each node $t \in \mathcal{N}$, an optimum value $V(t)$ is

$$V(t) = \min_{\forall \pi_t \, in \, (\mathcal{N}, \mathcal{A})} \{f(\pi_t)\}, \text{ or} \tag{6}$$

$$V(t) = \max_{\forall \pi_t \, in \, (\mathcal{N}, \mathcal{A})} \{f(\pi_t)\}. \tag{7}$$

The IFT solves this minimization (or maximization) problem by computing an *optimum-path forest* — a function P which contains no cycles and assigns to each node $t \in \mathcal{N}$ either its predecessor node $P(t) \in \mathcal{N}$ in the optimum path

or a distinctive marker $P(t) = nil \notin \mathcal{N}$, when $\langle t \rangle$ is optimum (i.e., t is said *root* of the forest). Figure 3 depicts an optimum-path forest, computed by forcing the roots to be in a set of seeds $\mathcal{S}$ using $f_1(\langle t \rangle)$ with $H(t) = 0$, if $t \in \mathcal{S}$, or $H(t) = +\infty$, otherwise.

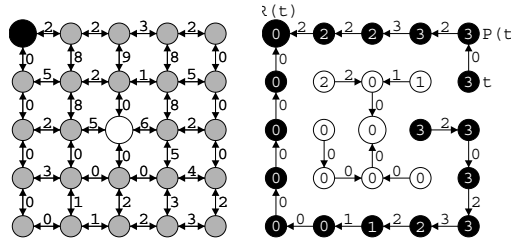

Figure 3. (a) A 4-neighborhood graph with $\mathcal{S} = \{(0,0),(2,2)\}$, where the top left is the origin $(0,0)$ of the (x,y) pixel coordinates, and the numbers are the arc weights. (b) An optimum-path forest P for f_1 with the two roots in $\mathcal{S}$. The numbers inside the nodes indicate V(t) by Eq. 6 and the arrows indicate $P(t)$.

We present below the general IFT algorithm for solving the minization problem of Eq. 6, for a connectivity funcion f that satisfies certain conditions required for only optimum paths [8] (such as f_1 and f_2). The root $R(t)$ of each pixel t can be obtained by following its optimum path backwards in P (Figure 3b). However, it is more efficient to propagate them on-the-fly, creating a root map R.

Algorithm 1: – GENERAL IFT ALGORITHM

INPUT: *Graph $(\mathcal{N}, \mathcal{A})$ and connectivity function f.*
OUTPUT: *Optimum-path forest P, its connectivity value map V and its root map R.*
AUXILIARY: *Priority queue Q and variable tmp.*

1. **For each** $t \in \mathcal{N}$, **do**
2. $\quad P(t) \leftarrow nil$, $R(t) \leftarrow t$ and $V(t) \leftarrow f(\langle t \rangle)$.
3. $\quad$ **If** $V(t) \neq +\infty$, **then** insert t in Q.
4. **While** $Q \neq \emptyset$, **do**
5. $\quad$ Remove s from Q such that $V(s)$ is minimum.
6. $\quad$ **For each** $t \in \mathcal{A}(s)$, such that $V(t) > V(s)$, **do**
7. $\quad\quad$ Compute $tmp \leftarrow f(\pi_s \cdot \langle s,t \rangle)$.
8. $\quad\quad$ **If** $tmp < V(t)$, **then**
9. $\quad\quad\quad$ **If** $V(t) \neq +\infty$, **then** remove t from Q.
10. $\quad\quad\quad$ Set $P(t) \leftarrow s$, $R(t) \leftarrow R(s)$, $V(t) \leftarrow tmp$.
11. $\quad\quad\quad$ Insert t in Q.

Lines 1–3 initialize maps for trivial paths. The minima of the initial map V compete with each other and some of them become roots of the forest. They are pixels with optimum trivial-path values, which are inserted in queue Q. The main loop computes an optimum path from the roots to every node s in a non-decreasing order of value (Lines 4–11). At each iteration, a path of minimum value $V(s)$ is obtained in P when we remove its last pixel s from Q (Line 5). Ties are broken in Q using first-in-first-out policy. The remaining lines evaluate if the path that reaches an adjacent pixel t through s is cheaper than the current path with terminus t and update Q, $V(t)$, $R(t)$ and $P(t)$ accordingly.

III. UNDERSTANDING USER INPUT

By letting the user draw markers freely, we face the challenge of determining what is the meaning of such interactions. On the first iteration, we need to analyse the markers to see whether the object-based weight can be computed. That is, we need to verify if the pixels were selected on foreground and background regions with dissimilar image features. Afterwards, user interaction will be primarily for corrections on the segmentation result, but for certain images the new markers might bring useful information to improve the arc-weight estimation.

Similarity between pixels is established by clustering the image according to its multiscale properties, and assigning to every pixel $t \in D_{\hat{I}}$ the corresponding cluster label $C(t)$. If the user selects object and background markers on pixels with the same cluster label, we can choose to discard some of them from the supervised fuzzy classification during the arc-weight estimation. If every foreground and/or background pixel is discarded by this procedure, then those two classes are too similar and the object-based weight should not be considered for extraction. Note that, if the object-based weight is not computed, then the marker pixel selection is executed every time that more strokes are added, in an attempt of eventually improving the arc-weight estimation. Therefore, this is a special case of the segmentation algorithm and was not depicted on the scheme of Figure 2.

When new object (background) markers are added by the user on regions where the object membership map values are low (high), they denote pixels that were misclassified on the previous iteration by the fuzzy classification. Therefore, this situation indicates when the marker pixel selection followed by fuzzy classification should be recomputed to improve the arc-weight estimation. Otherwise, only the object extraction is executed (condition on Figure 2).

We explain next the clustering procedure and the criterion devised to determine which pixels to select. Section IV-C further describes the condition for arc-weight recomputation, and how it works in order to preserve user-control.

A. Clustering by optimum-path forest

Let $\mathcal{S} \subset D_{\hat{I}}$ be a set of pixels sampled from the image using a uniform rectangular grid, such that $|\mathcal{S}| \approx 600$. The pixels in $\mathcal{S}$ are used as samples for clustering using an optimum-path forest classifier [10], along with the corresponding feature vectors in $\vec{F}$. The prototypes (key elements) of this classifier are found on regions with high concentration of samples. Those regions indicate the presence of data clustering. Each prototype becomes the root of an optimum-path tree (cluster), partitioning $\mathcal{S}$ into a forest.

Clustering is done using a graph $(\mathcal{S}, \mathcal{A}_3)$, with arc-weights (s,t) corresponding to the euclidean distance $d(s,t) = \|\vec{F}(s) - \vec{F}(t)\|$ between the feature vectors $\vec{F}(s)$ and $\vec{F}(t)$. The arcs of the adjacency relation $(s,t) \in \mathcal{A}_3$ are defined by the k nearest samples t of s using the function $d(s,t)$.

The nodes of the graph are also weighted by computing the probability density function (pdf) given by

$$\rho(s) = \frac{1}{\sqrt{2\pi\sigma^2}|\mathcal{A}_3(s)|} \sum_{t \in \mathcal{A}_3(s)} \exp\left(\frac{-d^2(s,t)}{2\sigma^2}\right) \quad (8)$$

where $\sigma = \max_{\forall (s,t) \in \mathcal{A}_3} \left\{\frac{d(s,t)}{3}\right\}$.

If a region contains a high concentration of samples, the distances between the k nearest neighbors will be smaller, resulting in pdf maxima. Considering that the lowest density along a path in the graph represents its value, and that the optimum path of a pdf maximum to a sample is the one with maximum value, the pdf maxima will compete for samples $s \in \mathcal{S}$ and each maximum will define an influence zone (optimum-path tree) composed by the samples most strongly connected to it. For such purpose, f_2 is used with $H(t) = \rho(t)$, if $t \in \mathcal{R}$, or $H(t) = h(t)$, otherwise, where $h(t) < \rho(t)$ for every $t \in \mathcal{S}$ and $\mathcal{R} \subset \mathcal{S}$ is the set of prototypes (roots). The maximization of f_2 using the IFT algorithm automatically defines the prototypes $s \in \mathcal{R}$, which are identified as samples t removed from Q with $P(t) = nil$. Then, the handicap value $h(t)$ is set as $h(t) = \rho(t) - \delta$, with $\delta = \min_{(s,t) \in \mathcal{A}_3 | \rho(t) \neq \rho(s)} |\rho(t) - \rho(s)|$, to guarantee that all maxima will be preserved, each one with a prototype.

The unsupervised learning of the OPF classifier consists of finding a suitable value for k of the adjacency relation $\mathcal{A}_3$, and the corresponding forest. If a pdf maximum is located on a plateau, it is necessary to make sure that only one prototype conquers the remaining samples of this maximum. This cannot be solely assured by the initialization of f_2, because $\mathcal{A}_3$ is not a symmetric adjacency relation. Therefore, we have to extend that $\mathcal{A}_3$ to guarantee that plateau samples be interconnected. This is accomplished by defining $\mathcal{A}_4$ such that if $t \in \mathcal{A}_3(s)$, $s \notin \mathcal{A}_3(t)$, and $\rho(s) = \rho(t)$ then

$$\mathcal{A}_4(t) \leftarrow \mathcal{A}_3(t) \cup \{s\}. \quad (9)$$

The maximization version of the IFT Algorithm 1 needs little modification to perform the clustering. All that needs to be added is a condition between Lines 4 and 5 that does $V(s) = \rho(s), C(s) = c$, and $c \leftarrow c+1$ if $P(s) = nil$, where $c = 1, 2, \ldots, |\mathcal{R}|$ is a variable that assigns a unique label c to each clusters. Also, the cluster labels are propagated from the prototypes to the neighbors by doing $C(t) \leftarrow C(s)$ on Line 10. The choice of the most suitable k is done by computing the result of the algorithm on the graph $(\mathcal{S}, \mathcal{A}_4)$, for $k = 5, 6, \ldots, 50$, and selecting the one that minimizes the normalized cut [13] along with the corresponding forest. A sample $t \in D_{\hat{I}} \backslash \mathcal{S}$ is classified in one of the clusters by identifying which root would offer it an optimum path. By considering the adjacent samples $s \in \mathcal{A}_4(t) \subset \mathcal{S}$, we compute ρ by Eq. 8, evaluate the paths $\pi_s \cdot \langle s, t \rangle$, and select the one that satisfies

$$V(t) = \max_{\forall (s,t) \in \mathcal{A}_4} \{\min\{V(s), \rho(t)\}\}. \quad (10)$$

Let the node $s^* \in \mathcal{S}$ be the one that satisfies Eq. 10. The classification simply assigns $C(s^*)$ as the cluster of t.

B. Marker pixel selection criterion

Let $\mathcal{M}$ be a set of object and background marker pixels selected by the user at any given place. If $\lambda(t) \in \{0, 1\}$ is, respectively, the background or object label of every $t \in \mathcal{M}$, then we can define the concept of object/background clusters. Let $\mathcal{J}_c$ be the set of pixels $t \in \mathcal{M}$ where $C(t) = c$, and $\mathcal{J}_c^o$ be the set of pixels $s \in \mathcal{J}_c$ with $\lambda(s) = 1$. We say that $\mathcal{J}_c$ is an object cluster if it follows the "purity" criterion

$$\frac{|\mathcal{J}_c^o|}{|\mathcal{J}_c|} \geq \tau, \quad (11)$$

where $0 \leq \tau \leq 1$ is a threshold parameter. Similar condition is used to define background clusters by substituting $\mathcal{J}_c^b = \mathcal{J}_c \backslash \mathcal{J}_c^o$ on the upper term of Eq. 11.

By analysing object and background clusters, we are able to choose the important pixels for fuzzy classification. Empirically, we have noted that nearly every object marker pixel is important to learn the object's image properties. Therefore, we create at first a set $\mathcal{Z}$ with every pixels $t \in \mathcal{M}$ where $\lambda(t) = 1$. Then, we remove from $\mathcal{Z}$ pixels $s \in \mathcal{Z}$ that are part of any background cluster $\mathcal{J}_c^b$. That is, we remove from $\mathcal{Z}$ object pixels s that belong to "pure" background clusters, because those pixels tend to be outliers and will not contribute to arc-weight estimation. Finally, only the background pixels $t \in \mathcal{M}$, with $\lambda(t) = 0$, that belong to background clusters are added to $\mathcal{Z}$.

The threshold parameter τ plays an important role in automatically selecting the marker pixels, but we experimentally noticed that setting $\tau = 0.96$ yielded good results. Also, keeping only the object pixels from mixed clusters is important because some images have similar, but separable, foreground and background in which the fuzzy classification can help the extraction. The only reason not to perform the object learning is when no background clusters are computed. Those cases happen in images with extremely similar foreground and background, whose object's boundary is usually weak. Thus, only the knowledge from the image features should be used for object extraction.

IV. Arc-weight estimation

Arc-weight estimation consists of fuzzy classification and arc-weight assignment, aiming higher weights to arcs on the object's boundary than elsewhere. Under this condition, the object can be extracted using f_1 from only two marker pixels, one inside and one outside it. However, perfect arc-weight assignment is usually not possible, asking for more user involvement (marker selection).

A. Fuzzy classification by OPF

Let $\mathcal{Z} \subseteq \mathcal{M}$ be a set of object and background pixels selected for learning of object properties (e.g. Figure 1c).

We first randomly divide the labeled pixels in $\mathcal{Z} = \mathcal{T} \cup \mathcal{E}$ into a training set $\mathcal{T}$ and an evaluation set $\mathcal{E}$, with the same proportion of object and background markers. Set $\mathcal{T} = \mathcal{T}_b \cup \mathcal{T}_o$ is further divided into object pixels in $\mathcal{T}_o$ and background pixels in $\mathcal{T}_b$.

Now, consider a complete graph $(\mathcal{T}, \mathcal{A}_1)$, using $\mathcal{N} = \mathcal{T}$ and $\mathcal{A} = \mathcal{A}_1$, with arc weights $w(s,t) = \|\vec{F}(t) - \vec{F}(s)\|$. The arcs (s,t), $s \in \mathcal{T}_o$ and $t \in \mathcal{T}_b$, or vice-versa, in a minimum-spanning tree of $(\mathcal{T}, \mathcal{A}_1)$ define the closest nodes between object and background in the feature space. They represent key elements (prototypes) to protect each class, object and background, as seeds in an optimum-path forest classifier [12] (OPF). We use here a variant, by inserting s in a set $\mathcal{S}_o \subset \mathcal{T}_o$, t in a set $\mathcal{S}_b \subset \mathcal{T}_b$, and computing one optimum-path forest P_o for f_1 on a complete graph $(\mathcal{T}_o, \mathcal{A}_1)$ and one optimum-path forest P_b for f_1 on a complete graph $(\mathcal{T}_b, \mathcal{A}_1)$. In the first case, the handicap $H(t) = 0$, if $t \in \mathcal{S}_o$, or $H(t) = \infty$, otherwise. The second case is similar, changing $\mathcal{S}_o$ by $\mathcal{S}_b$.

A local processing operation can compute the optimum connectivity values $V_o(t)$ and $V_b(t)$ for any remaining pixel $t \in D_{\hat{I}} \backslash \mathcal{T}$ incrementally, as though t were part of the original graphs.

$$V_o(t) = \min\{\max\{V_o(s), w(s,t)\}\}, \ \forall s \in \mathcal{T}_o, \quad (12)$$

$$V_b(t) = \min\{\max\{V_b(s), w(s,t)\}\}, \ \forall s \in \mathcal{T}_b. \quad (13)$$

This allows fast propagation of the optimum connectivity values from $\mathcal{S}_o$ and $\mathcal{S}_b$ to the remaining image pixels.

An *object membership value* $M_o(t)$ (Figure 1b) can finally be assigned to each pixel $t \in D_{\hat{I}}$ as:

$$M_o(t) = \frac{V_b(t)}{V_o(t) + V_b(t)} \quad (14)$$

A binary classification of pixels $t \in \mathcal{E}$ is also possible by assigning to them the label $l(t) \in \{0,1\}$ of background or object ($l(t) = 1$ if $V_o(t) < V_b(t)$, and $l(t) = 0$ otherwise). If $\lambda(t) \in \{0,1\}$ is the correct label of $t \in \mathcal{E}$, then an error occurs when $\lambda(t) \neq l(t)$. Aiming to minimize the number of misclassifications, we can select better training nodes by replacing misclassified nodes in $\mathcal{E}$ with randomly selected nodes in $\mathcal{T} \backslash \mathcal{S}_o \cup \mathcal{S}_b$ [12].

B. Arc-weight assignment

After the fuzzy classification we have the image-graph $(D_{\hat{I}}, \mathcal{A}_2)$, where $\mathcal{N} = D_{\hat{I}}$ and $\mathcal{A} = \mathcal{A}_2$. Arc-weight assignment is represented by a weight image $\hat{W} = (D_{\hat{I}}, W)$ (Figure 1c).

$$W(s) = \gamma W_o(s) + (1-\gamma) W_f(s), \quad (15)$$

where $W_o(s)$ is an *object-based weight*, $W_f(s)$ is a *feature-based weight*, and $0 \leq \gamma \leq 1$ represents the importance of the object membership map in this estimation.

Given that $\vec{F}$ stores filtered maps F_b, $b = 1, 2, \ldots, 12$, we estimate $W_f(s) = \max_{b=1,2,\ldots,12}\{\|\vec{G}_b(s)\|\}$ as the maximum gradient magnitude among the feature-based gradients $\vec{G}_b(s) = \sum_{\forall t \in \mathcal{A}_2(s)} (F_b(t) - F_b(s))\, \vec{st}$, t. The weight $W_o(s) = \|\vec{G}_o(s)\|$ is estimated in a similar way as the magnitude of an object-based gradient $\vec{G}_o(s) = \sum_{\forall t \in \mathcal{A}_2(s)} (M_o(t) - M_o(s))\, \vec{st}$.

C. Arc-weight recomputation

Let $\mathcal{H}$ represent one single stroke drawn by the user (a set of labeled pixels in a same connected component). When $\mathcal{H}$ is an object marker ($\lambda(t) = 1, \forall t \in \mathcal{H}$), we can say that every pixel $s \in \mathcal{H}$ with $M_o(s) < 0.5$ was misclassified by the OPF fuzzy classifier, and the user may be trying to correct that. The same reasoning is valid for a background marker ($\lambda(t) = 0, \forall t \in \mathcal{H}$), if pixels $s \in \mathcal{H}$ have $M_o(s) >= 0.5$. Therefore, if a great percentage of the pixels in at least one object/background marker $\mathcal{H}$ were misclassified (in our case 70%), we can perform the marker pixel selection and fuzzy classification in an attempt to improve the arc-weight assignment.

However, to preserve user control the recomputation of the weight image values $W(t)$ does not always occur for all pixels. Let $L(t) \in \{0,1\}$, $\forall t \in D_{\hat{I}}$, be the final object extraction label computed on the previous iteration. If at least one object marker satisfies the relearning requirement previously stated, then only the pixels $t \in D_{\hat{I}}$ with $L(t) = 0$ are going to have their $W(t)$ value recomputed. Conversely, if at least one background marker satisfies the requirement, $W(t)$ will be recomputed only for the pixels $t \in D_{\hat{I}}$ with $L(t) = 1$.

V. Object Extraction

After arc-weight estimation, we use the 8-neighbor image-graph $(D_{\hat{I}}, \mathcal{A}_2)$ with arc weights $w(s,t) = \frac{\hat{W}(s)+W(t)}{2}$ in Eq. 4 and $H(t) = 0$, if $t \in \mathcal{M}$, or $H(t) = +\infty$, otherwise. Note that $\mathcal{M}$ is the whole set of marked pixels (Figure 1a).

It is expected that the optimum-path forest P computed on $(D_{\hat{I}}, \mathcal{A}_2)$ for $f_{\max}$ in Eq. 4 with $\mathcal{M}$ extracts the object as the union of trees rooted at the object pixels in $\mathcal{M}$. The object is identified as the pixels 1 after a local operation, which assigns the correct label $L(t) = \lambda(R(t)) \in \{0,1\}$ of the root to each pixel $t \in D_{\hat{I}}$. That is, object and background seeds will compete with each other to conquer their most strongly connected pixels, hopefully from the same label, but segmentation errors may occur.

The user may draw new markers (Figure 1f) and/or remove markers by clicking on them, and the optimum-path forest P can be recomputed in a differential way, taking time proportional to the number of pixels in the modified image regions (sublinear time in practice), if $w(s,t)$ is normalized within an integer range of numbers. This approach is known as differential image foresting transform (DIFT) [7]. That is, each marker pixel added to $\mathcal{M}$ may define a new optimum-path tree by invading the trees of other roots. The removal of a marker eliminates all optimum-path trees

rooted at it, making their pixels available for a new dispute among the remaining roots. Note that care must be taken when using DIFT in our approach, because the arc-weight recomputation requires that all marked pixels be used once again to compute the optimum-path forest.

The DIFT requires a variant of Algorithm 1 (see [7] for details). It is important to note that the minimization of the path value $V(t)$, using either Algorithm 1 or the DIFT algorithm with f_1, makes optimum paths from object roots to avoid as much as possible higher weight arcs inside the object (usually in noisy regions) and meet optimum paths from background roots at the lower arc weights on the object's boundary. After that meeting, paths from the background are blocked and the noisy pixels inside the object tend to be conquered by object roots. Therefore, markers should be selected around those weaker parts of the boundary in order to make the segmentation more effective [11].

VI. Experiments and results

We have compared the approach proposed in [11] (M1), which separates user recognition for arc-weight estimation and object extraction, against our method for intelligent understanding of user input (M2). Both methods were used to segment a dataset with 50 natural images with ground-truths, obtained from [4]. The pixels in the ground-truths may represent three different regions: background, object, and mixed areas. Since we are only interested in background and object components, the mixed areas were considered as object. For a fair comparison, we used $\gamma = 0.5$ for the segmentation of every image.

Accuracy was estimated as the average of the F-measures [14] computed between the segmentation results and the ground truths of the 50 images. The standard deviation of these measures indicates *precision*. The total time for segmentation may take from 3 seconds to 3 minutes per image, depending mostly on the user's patience and experience to select method and markers suitable for a given image. Due to the difficulty in measuring the duration of this process, we measured only the total computation time of both methods. However, M2 should take much less of the user's time than M1 for non-expert users. The experiments were executed on a computer with a Core i5-430M processor running at 2.27 GHz and using 4GB of RAM. The supervised fuzzy OPF classification was optimized using SSE 3 instructions. Table I shows our experimental results.

Table I
Average results of F-measure, number of markers and total computational time, with their respective standard deviations, over the 50 images, using M1 and M2.

	F-measure	num. of markers	tot. comp. time in sec
M1	0.98 ± 0.01	5.6 ± 3.3	1.1 ± 0.4
M2	0.98 ± 0.01	4.8 ± 3.2	2.7 ± 1.2

From Table I, both methods performed equally well in accuracy, 0.98, and computational time, less than 3 seconds, using a few markers, because the user was an expert. Even though, M2 required lower number of markers than M1, and this advantage tends to considerably increase with non-expert users.

The second and third columns of Figure 4 show object membership maps computed without and with marker pixel selection from the markers drawn on the first column. If the map of the second column were used for arc-weight estimation, object extraction would certainly require more markers for correction due to its poorer quality. The fourth column shows the result of M2, using the map of the third column and markers of the first column. An example of arc-weight recomputation during segmentation with M2 is also shown in Figure 5.

VII. Conclusions

We presented an IFT-based framework which implements an intelligent method for understanding user interaction applied to image segmentation. The user selects object and background markers, and the method selects automatically the best pixels for learning object image properties during arc-weight estimation. The object is finally extracted using all marked pixels and the computed arc weights for an image-graph. If more markers are added, the method is able to further understand when and where the newly selected pixels can improve arc-weight estimation, while preserving user control.

We compared our method against an approach proposed in [11], which requires separated user interaction for arc-weight estimation and object extraction. Our experiments demonstrated that our method achieves high segmentation accuracy and preserves the user's control with lower number of markers.

Future work includes extending the framework for soft segmentation and experiments with non-expert users, also measuring the user's time. We also intend to exploit the regions of clustered pixels as a visual cue to aid marker selection.

We would like to thank FAPESP (Proc. 2009/11908-8 and Proc. 2007/52015-0) and CNPq (Proc. 481556/2009-5 and Proc. 302617/2007-8) for financial support.

References

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

[2] 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*, vol. 1, 2001, pp. 105–112.

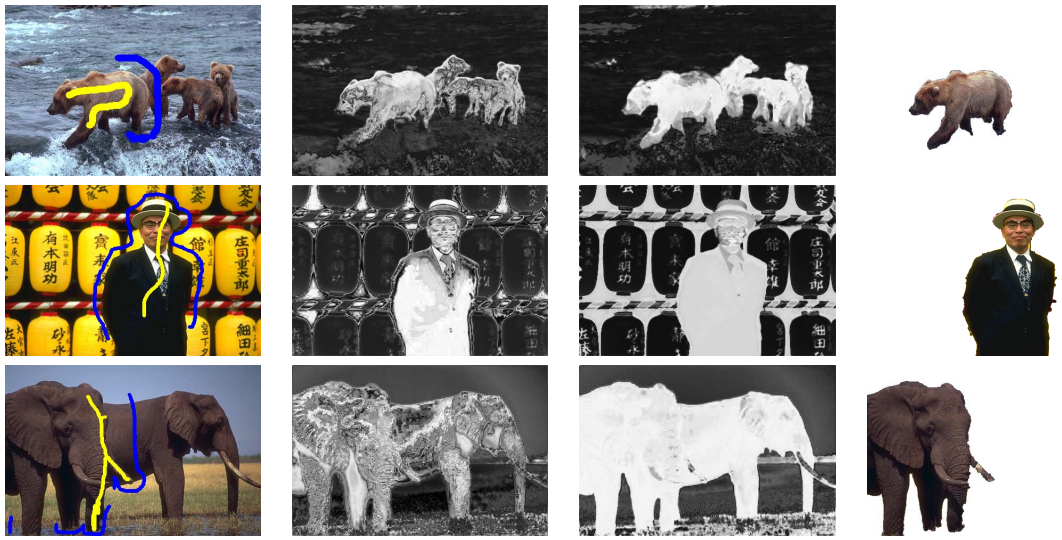

Figure 4. Images on the first column show user-drawn markers. The second and third columns present the object membership maps computed, respectively, without and with marker pixel selection. Segmentation results of M2 are shown on the last column, using the object membership map from the third column and the markers from the first column.

Figure 5. Two consecutive iterations of the segmentation process with M2. The arc-weights for the second iteration (last two images) were recomputed only for object pixels.

[3] A. Protiere and G. Sapiro, "Interactive image segmentation via adaptive weighted distances," *IEEE Transactions on Image Processing*, vol. 16, no. 4, pp. 1046–1057, Apr 2007.

[4] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": interactive foreground extraction using iterated graph cuts," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 309–314, 2004.

[5] L. Grady, "Random walks for image segmentation," *IEEE Transactions Pattern Anaysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, 2006.

[6] P. A. V. Miranda, A. X. Falcão, and J. K. Udupa, "Synergistic Arc-Weight Estimation for Interactive Image Segmentation using Graphs," *Computer Vision and Image Understanding*, vol. 114, no. 1, pp. 85–99, Jan 2010.

[7] A. X. Falcão and F. P. G. Bergo, "Interactive volume segmentation with differential image foresting transforms," *IEEE Transactions on Medical Imaging*, vol. 23, no. 9, pp. 1100–1108, 2004.

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

[9] P. A. V. Miranda and A. X. Falcão, "Links between image segmentation based on optimum-path forest and minimum cut in graph," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 128–142, October 2009.

[10] L. M. Rocha, F. A. M. Cappabianco, and A. X. Falcão, "Data clustering as an optimum-path forest problem with applications in image analysis," *International Journal of Imaging Systems and Technology*, vol. 19, no. 2, pp. 50–68, 2009.

[11] T. V. Spina, J. A. Montoya-Zegarra, A. X. Falcão, and P. A. V. Miranda, "Fast interactive segmentation of natural images using the image foresting transform," in *DSP'09: Proceedings of the 16th international conference on Digital Signal Processing*. Santorini, Greece: IEEE Press, 2009, pp. 998–1005.

[12] J. P. Papa, A. X. Falcão, and C. T. N. Suzuki, "Supervised pattern classification based on optimum-path forest," *International Journal of Imaging Systems Technology*, vol. 19, no. 2, pp. 120–131, June 2009.

[13] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug 2000.

[14] C. van Rijsbergen, *Information retrieval*, 2nd ed. London: Wiley Inter-science, 1979.