A comparative study among pattern classifiers in interactive image segmentation

Thiago V. Spina, Javier A. Montoya-Zegarra, Fábio Andrijauskas, Fábio A. Faria,
Carlos E. A. Zampieri, Sheila M. Pinto-Cáceres, Tiago J. de Carvalho and Alexandre X. Falcão

Institute of Computing – University of Campinas (UNICAMP),
C.P. 6176, 13084-971, Campinas, SP, Brazil.
afalcao@ic.unicamp.br

***Abstract*—Edition of natural images usually asks for considerable user involvement, being segmentation one of the main challenges. This paper describes a unified graph-based framework for fast, precise and accurate interactive image segmentation. The method divides segmentation into object recognition, enhancement and extraction. Recognition is done by the user when markers are selected inside and outside the object. Enhancement increases the dissimilarities between object and background and extraction separates them. Enhancement is done by a fuzzy pixel classifier and it has a great impact in the number of markers required for extraction. In view of minimizing user involvement, we focus this paper on a comparative study among popular classifiers for enhancement, conducting experiments with several natural images and seven users.**

***Keywords*-Pattern classifiers, graph-based image segmentation, image foresting transform, fuzzy classification**

I. Introduction

Automatic image segmentation is a very difficult and challenging task. When segmenting natural scenes such as photos and videos, the lack of global information (shape of the object, position, etc.) creates a gap that can only be fulfilled by the user's knowledge. The segmentation task should be accurate, precise, and fast, aiming to minimize user's involvement and the total time needed for segmentation. As noted by [1], the interactice image segmentation process can be divided into three main tasks: object recognition, enhancement and extraction.

Recognition is the task that identifies the whereabouts of an object in the image. It determines useful image properties for enhancement and extraction (object delineation). Enhancement increases the dissimilarities between object and background in order to improve extraction. Extraction finally separates the object from the background. Traditionally, computers outperform humans in the delineation task, whilst the opposite is true for recognition. Although it is important to exploit a synergism between them [2], the only interactive task should be recognition.

In [1], the authors present an unified framework for interactive image segmentation which attends to all aforementioned requirements. Recognition is done by the user through the selection of markers inside and outside the object for enhancement (Figure 1a) and extraction (Figure 1d). Extraction and enhancement are done by exploiting the optimum-path forest computed by the *image foresting transform* (IFT) — a tool for the design of image processing operators based on *connectivity functions* in graphs derived from the image [3]. Furthermore, the extraction process uses the sublinear-time IFT (differential IFT — DIFT [4]) on a gradient image (Figure 1c). This gradient results from image properties and object enhancement. Enhancement uses a pixel classifier based on a variant of the optimum-path forest classifier (OPF) [5], by assigning a fuzzy-membership value to each pixel regarding its resemblance to the object (Figure 1b).

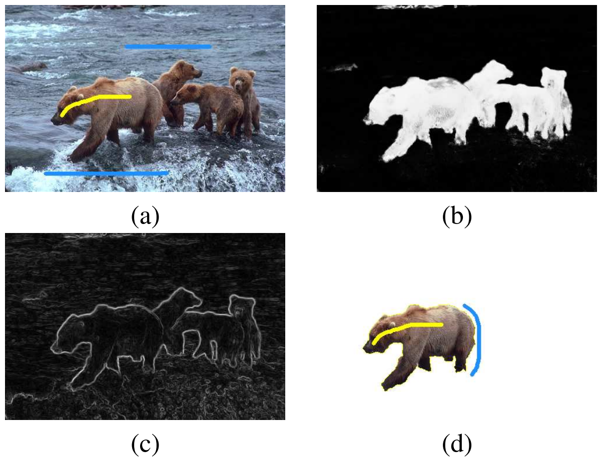
(a) (b) (c) (d)

Figure 1. (a) Markers are selected for enhancement. (b) Object membership map. (c) The weight image resulting from enhancement. (d) Segmentation results using three markers.

Given that object enhancement has great impact in the number of markers required for extraction and we aim to minimize user involvement in interactive segmentation, we present in this paper a comparative study among pixel classifiers for enhancement. For comparison, we have chosen a fuzzy version of some popular classifiers, Support Vector Machines (SVM), Artificial Neural Networks (ANN), k-nearest neighbors (kNN) and a decision tree (DT), against the fuzzy version of the optimum-path forest classifier (OPF), as originally proposed in [1]. The experiments involved several natural images and seven users who segmented these images using the DIFT and at least two distinct methods for enhancement. The results include computational time, segmentation accuracy, and efficiency with respect to the number of required markers. The framework itself was shown to be effective in the segmentation task when compared to other graph-based methods in [1].

In Section II, we review some important concepts about the IFT, which will be used for the OPF classifier and object extraction. Section III details object enhancement and Section IV explains object extraction by DIFT. In Section V we present the classifiers considered in our study, whilst experimental results are given in Section VI. Finally, conclusions are drawn in Section VII.

II. Image Foresting Transform

An IFT-based image operator requires to derive a graph from the image, compute an optimum-path forest according to a connectivity function, and apply a local processing to one or more of its attributes.

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. For example, the triplet $(I_1(t), I_2(t), I_3(t))$ may denote the red, green, and blue values of a pixel t in the RGB color space. In the case of gray images, the subindex b is suppressed and $I(t)$ is adopted. Multiscale feature extraction essentially transforms an image $\hat{I} = (D_{\hat{I}}, \vec{I})$ into another image $\hat{F} = (D_{\hat{I}}, \vec{F})$ where $\vec{F}(t) = (F_1(t), F_2(t), \ldots, F_m(t))$ is a feature vector assigned to t.

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 $s \in \mathcal{N}$.

B. Optimum-path forest

A *path* $\pi_t = \langle t_1, t_2, \ldots, t \rangle$ in a graph is a sequence of adjacent nodes with terminus at a node t, being $\pi_t = \langle t \rangle$ a *trivial path*. A connectivity function f assigns to any path π_t a value $f(\pi_t)$. In this work we are interested in function $f_{\max}$:

$$f_{\max}(\langle t \rangle) = H(t) \quad (1)$$

$$f_{\max}(\langle t_1, \ldots, t_n \rangle) = \max_{i=1,\ldots,n-1} \{H(t_1), w(t_i, t_{i+1})\} \quad (2)$$

where $H(t)$ is a handicap value, which is finite only to root candidates (i.e., *seed pixels*), and $w(t_i, t_{i+1}) \geq 0$ is an arc weight, both computed from $\hat{F}$ in different ways, depending on the operator.

Considering all possible paths with terminus at each node t, the optimum connectivity value map is $V(t) = \min_{\forall \pi_t \, in \, (\mathcal{N},\mathcal{A})} \{f(\pi_t)\}$. The IFT solves this minimization 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 with terminus t 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). The IFT algorithm is presented below for function $f_{\max}$. The root $R(t)$ of each pixel t can be obtained by following its optimum path backwards in P. However, it is more efficient to propagate them on-the-fly, creating a root map R.

Algorithm 1: – IFT Algorithm for $f_{\max}$

```
INPUT:      Graph $(\mathcal{N}, \mathcal{A})$
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.      $P(t) \leftarrow nil$, $R(t) \leftarrow t$ and $V(t) \leftarrow H(t)$.
3.      If $V(t) \neq +\infty$, then insert $t$ in $Q$.
4.  While $Q \neq \emptyset$, do
5.      Remove $s$ from $Q$ such that $V(s)$ is minimum.
6.      For each $t \in \mathcal{A}(s)$, such that $V(t) > V(s)$, do
7.          Compute $tmp \leftarrow \max\{V(s), w(s,t)\}$.
8.          If $tmp < V(t)$, then
9.              If $V(t) \neq +\infty$, then remove $t$ from $Q$.
10.             $P(t) \leftarrow s$, $R(t) \leftarrow R(s)$, $V(t) \leftarrow tmp$.
11.             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. The next sections show how to use this framework for object enhancement and extraction.

III. Object Enhancement

Enhancement consists of feature extraction, 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 $f_{\max}$ from only two marker pixels, one inside and one outside it. However, perfect arc-weight assigment is usually not possible, asking for more user involvement (marker selection).

For feature extraction, we convert the image from RGB to Lab color space and use a cosine-based low-pass filter $\hat{L}$ applied to each of the Lab components in three different scales [6], followed by a leveling operation [7] to avoid border shifting. Considering the original Lab values and the additional filtered values, each pixel is represented by

12 features, i.e. $\vec{F}(t) = (F_1(t), F_2(t), \ldots, F_{12}(t))$ for every pixel $t \in D_{\hat{I}}$.

Let $\mathcal{M}$ be a set of markers for enhancement, selected on parts where object and background have distinct properties. Note that, markers selected in Figure 1d for extraction should never be used for enhancement. The pixels in $\mathcal{M}$ are used as training samples for each classifier compared in this study. The algorithms use the marked pixels feature vectors to "fuzzy-classify" them in such way that an object membership map M_o is created (Figure 1b). Actually, for each method it was developed a special way to compute these fuzzy values, but in all of them brighter pixels (i.e. higher values) have greater probability to belong to the object than to the background. The map values were normalized in the range $[0, 1000]$.

Now, consider the image graph $(D_{\hat{I}}, \mathcal{A})$ where $t \in \mathcal{A}(s)$ if $t \neq s$ is 8-neighbor of s. Enhancement is represented by a weight image $\hat{W} = (D_{\hat{I}}, W)$ (Figure 1c).

$$W(s) \quad = \quad \gamma W_o(s) + (1-\gamma) W_f(s), \tag{3}$$

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. Note that both arc weights must be normalized to be in the same range for combination in Equation 3.

Given that $\vec{F}$ stores filtered maps F_b, $b = 1, 2, \ldots, 12$, for each Lab component in different scales, let $\vec{G}_b(s)$ be the gradient for every scale b defined at each pixel s as

$$\vec{G}_b(s) \quad = \sum_{\forall t \in \mathcal{A}(s)} (F_b(t) - F_b(s))\, \vec{st}, \tag{4}$$

where $\vec{st}$ is the unit vector from s to t. We estimate $W_f(s) = \max_{b=1,2,\ldots,12} \|\vec{G}_b(s)\|$ as the maximum magnitude of the feature-based gradients $\vec{G}_b(s)$. Note that this feature-based weight is computed in a different way than the one used in [1], but it represented an improvement in the overall result of segmentation. The weight $W_o(s) = \|\vec{G}_o(s)\|$ is estimated in a similar way as the magnitude of an object-based gradient

$$\vec{G}_o(s) \quad = \sum_{\forall t \in \mathcal{A}(s)} (M_o(t) - M_o(s))\, \vec{st}. \tag{5}$$

IV. Object Extraction by DIFT

After enhancement, we have the image graph $(D_{\hat{I}}, \mathcal{A})$, whose arcs are defined between 8-neighbor pixels with weights $w(s,t) = \frac{W(s)+W(t)}{2}$ in Eq. 2 and $H(t)$ by

$$H(t) \quad = \quad \begin{cases} 0 & \text{if } t \in \mathcal{M}, \\ \infty & \text{otherwise.} \end{cases} \tag{6}$$

It is expected that the optimum-path forest P computed on $(D_{\hat{I}}, \mathcal{A})$ for $f_{\max}$ in Eq. 2 with $\mathcal{M}$ given by the set of selected markers (e.g., the same markers drawn for enhancement in Figure 1a) 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 $\lambda(R(t)) \in \{0, 1\}$ 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 (lines in Figure 1d) 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) [4]. 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.

The DIFT requires a variant of Algorithm 1 (see [4] for details). Its important to note that the minimization of the path value $V(t)$, using either Algorithm 1 or the DIFT algorithm with $f_{\max}$, 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 [1] (Figure 1d).

V. Classifiers

The following subsections describe each pixel classifier used in our study. For each classifier it is explained how the object membership map M_o is estimated and how its training is done by considering the set of selected markers $\mathcal{M}$.

A. *Optimum-path forest*

The training of this classifier begins by randomly dividing the labeled markers in $\mathcal{M} = \mathcal{T} \cup \mathcal{E}$ into a training set $\mathcal{T}$ and an evaluation set $\mathcal{E}$, with the same proportion of object and background pixels. Note that, for efficiency purpose, we take $|\mathcal{T}| = 400$ if $|\mathcal{M}| > 400$ and $|\mathcal{T}| = \frac{|\mathcal{M}|}{2}$ otherwise. Set $\mathcal{T} = \mathcal{T}_b \cup \mathcal{T}_o$ is further divided into object markers in $\mathcal{T}_o$ and background markers in $\mathcal{T}_b$.

Now, consider a complete graph $(\mathcal{T}, \mathcal{A})$, where $t \in \mathcal{A}(s)$ for all $t \neq s$, 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 $(\mathcal{T}, \mathcal{A})$ define the closest nodes between object and background in the feature space. They represent key elements to protect each class, object and background, as seeds in an optimum-path forest classifier [8]. 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 $f_{\max}$ on a complete graph $(\mathcal{T}_o, \mathcal{A})$ and one optimum-path forest P_b for $f_{\max}$ on a complete graph $(\mathcal{T}_b, \mathcal{A})$. 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.

$$V_o(t) = \min\{\max\{V_o(s), w(s,t)\}\}, \ \forall s \in \mathcal{T}_o, \quad (7)$$
$$V_b(t) = \min\{\max\{V_b(s), w(s,t)\}\}, \ \forall s \in \mathcal{T}_b. \quad (8)$$

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:

$$M_o(t) = \frac{V_b(t)}{V_o(t) + V_b(t)} \quad (9)$$

B. Support Vector Machines

Support Vector Machine (SVM) is a useful technique for data classification [9]. Given a training set of instance-label pairs ($\mathbf{x}_i$; $\mathbf{y}_i$), i = 1,...,l where $\mathbf{x}_i \in \mathbb{R}^n$ and $\mathbf{y} \in \{1, -1\}^l$, the SVM solves the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \left\{ \frac{1}{2}\mathbf{w}^T\mathbf{w} + \mathbf{C}\sum_{i=1}^{l} \xi_i \right\} \quad (10)$$

$$\text{such that} \qquad \mathbf{y}_i \left(\mathbf{w}^T \phi(\mathbf{x}_i) + b \geq 1 - \xi_i \right), \quad \xi \geq 0.$$

Here training vectors $\mathbf{x}_i$ are mapped into a higher dimensional space by function ϕ. Then, SVM finds a linear separating hyperplane with maximal margin in the referred dimensional space. In Eq. 10, $\mathbf{C} > 0$ is the penalty parameter of the error term. Furthermore, $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is called the kernel function and can be of four basic types:

- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$;
- Polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma\mathbf{x}_i^T\mathbf{x}_j + r)^d, \gamma > 0$;
- Radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2), \gamma > 0$;
- Sigmoide: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma\mathbf{x}_i^T\mathbf{x}_j + r)$;

where γ, r, and d are kernel parameters.

In this work, we used the LIBSVM toolbox [10]. First, the feature vectors $\vec{F}(s)$ were normalized to range between $[0, 1]$. Then, they were used by the classifier as $\mathbf{x}_s = \vec{F}(s)$, with the proper label $\mathbf{y}_i$ (object or background) assigned, to train using all sample pixels $s \in \mathcal{M}$. The initial parameters were set to their default values, then the variables were selected through cross-validation (N-fold) and the RBF kernel — which is a gaussian function — had its parameters optimized.

Once the SVM training is finished, a sample is classified according to its relative position to the computed hyperplane, which is given by the sign function [10]

$$\mathbf{sgn}\left(\sum_{i=1}^{l} \mathbf{y}_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b \right). \quad (11)$$

However, the SVM classification only determines a class label $L(s)$ for a sample s and not a fuzzy-value as needed by the object enhancement. The work presented in [10] shows a way to extend the SVM to compute a probability associated with each classified sample, that was used to build the object membership map M_o. Given c classes of data, for any $\mathbf{x}$, the goal is to estimate

$$p_k = p(y = k \mid \mathbf{x}), k = 1, ..., c. \quad (12)$$

Following the setting of the one-against-one approach for multi-class classification, we first estimated pairwise class probabilities

$$r_{kj} \approx p(y = k \mid y = k \, or \, j, \mathbf{x}) \quad (13)$$

using an improved implementation as shown in [10]

$$r_{kj} \approx \frac{1}{1 + e^{A\hat{f}+B}}, \quad (14)$$

where A and B are estimated by minimizing the negative log-likelihood function using known training data and their decision values $\hat{f}$. Let $k = 1$ denote the object class for $c = 2$ (i.e. two classes: object and background), the value of every pixel t in the object membership image can finally be assigned as

$$M_o(t) = p(y = 1 \mid \mathbf{x}_t) \quad (15)$$

where $\mathbf{x}_t$ is the feature vector $\vec{F}(t)$ of the pixel t.

C. Artificial Neural Network

An Artificial Neural Network is a machine designed to model the way in which the brain performs a particular task or function of interest [11]. This machine acts as a highly parallel and distributed classifier. Multilayer Perceptron Network was used in this study because of its typical and important features such as backpropagation. This class of network has several layers of neurons connected in a feedforward manner (a neuron on a layer can only have connections to the next-layer neurons) and allows the recognition of more complex decision regions according to the universal approximation theorem [11]. This theorem states that every continuous function that maps real number intervals as input to any real number interval as output, can be arbitrarily approximated with precision by a multilayer perceptron with only one hidden layer. Considering this factor, several possible topology configurations were tested until establishing the 12, 25 and 2 neurons in the input, hidden and output layers respectively [12].

Let $\mathbf{x}_j = \vec{F}(j)$ be the feature vector from sample pixel j. The training process consists in presenting a randomly chosen $\mathbf{x}_j$ (to avoid network memorization) to the input layer of a network for learning.

This input is forward propagated through the network and the result is compared with the expected value (values for object and background classes in this case) to get an error measure. With this measure, a backpropagation process recomputes the connection weights from the last to the first layer, and this process is repeated for each element $j \in \mathcal{M}$. $\mathcal{M}$ must represent the regions and details of an image in equal proportions to avoid memorization. Thus, it needs to receive preferably the same number of samples for each class. If $\mathcal{M}$ has a different number of samples in each class, this difference is avoided by repeating some randomly chosen samples of the least represented class.

The weight of each connection is initially set to a random factor between $[0, 1]$. The output of a neuron in the forward process will be $\phi(\xi)$, where ϕ is an activation function and $\xi = \Sigma_j \; w_j \; \mathbf{x}_j$ is the weighted sum of all connections (from the previous layer) to the specific neuron. The activation function must be differentiable, so it was used the sigmoidal function in Eq. 16.

$$\phi(\xi) \quad = \quad \frac{1}{1+e^{\xi}} \tag{16}$$

In order to adjust the network weights in a backpropagation process, a δ value is needed. This δ value is computed according

$$\delta \quad = \quad [g(\mathbf{x}_j) - \mathcal{I}(w, L(t))] \; g(\mathbf{x}_j) \; [1 - g(\mathbf{x})] \tag{17}$$

where $g(\mathbf{x}_j)$ is the output of a network for an input $\mathbf{x}_j$, and $\mathcal{I}$ represents the desired value which is 1 if the label $L(t)$ of a sample t for the class w is 1 and 0 otherwise.

To compute δ in the previous layers, the weighted δ of each neuron from the subsequent layer is used as shown in Eq. 18, where v is the output of the current neuron and n is the number of neurons from the next layer that have already computed δ.

$$\delta \quad = \quad (\Sigma_{i=1}^{n} \delta_i \; p_i) \; v \; (1 - v) \tag{18}$$

Finally, each weight p can be updated as in Eq. 19.

$$\Delta p \quad = \quad \eta \; \delta \; v, \tag{19}$$

$$p_k \quad = \quad p_{k-1} - \Delta p \tag{20}$$

where η represents the learning rate and k is the index of each $t \in \mathcal{M}$. Thus, the connection between the current neuron and the previous one is updated. If η is too small, the algorithm will take a long time to converge but will probably get a more precise Δp. Conversely, if it is too large the learning will be faster but rather unstable. As the training set is generally small when compared to the whole set, the learning rate $\eta = 0.1$ was chosen to increase precision in the classification process.

Artificial Neural Network is an unstable method for classification due to several factors, such as the initial random weight choice for neuron connections. For this reason, it was used a modified *Bagging* technique which increases the stability over a collection of unstable classifiers. Therefore, the training process is improved by using not only one but a collection of neural network classifiers $D_j, j = 1, 2, ...L$ (in this study $L = 51$), each one receiving a different randomly selected subset of $\mathcal{M}$ samples to be classified as previously explained.

After the training process, the set $\mathcal{M}$ is presented to each network D_j once more. This set is propagated through every network and an estimated output is obtained. Thus, a reliability measure A_j, required for classification, that represents the percentage of correctly labeled elements in $\mathcal{M}$ can be computed according to the known output.

The fuzzy pixel classification process consists of the weighted voting of each network for every pixel in the image. Let $G_{ij}(t)$ be the output of classifier D_j for a sample t, and $i = 0, 1$ one of the two classes available (background and object) represented by the output neuron i. The object membership M_o can be computed by considering $i = 1$ as the object class in

$$M_o(t) \quad = \quad \frac{\Sigma_{j=1}^{L} \; A_j \; G_{1j}(t)}{\Sigma_{i=0}^{1} \Sigma_{j=1}^{L} \; A_j \; G_{ij}(t)}. \tag{21}$$

D. k-nearest neighbors

We use here a training process similar to the one described in subsection V-A for the computation of $M_o(p)$. Let $\mathcal{S}_o, \mathcal{S}_b \in \mathcal{M}$ be the subsets of pixels from the object and the background respectively, such that $\mathcal{S}_o \cup \mathcal{S}_b = \mathcal{M}$. The union set $\mathcal{S}_o \cup \mathcal{S}_b$ is divided into a training set $\mathcal{T}$ and an evaluation set $\mathcal{E}$ with the same proportion of object and background seeds in each one. For efficiency purpose we take $|\mathcal{T}| = 200$ if $|\mathcal{M}| > 200$ and $|\mathcal{T}| = \frac{|\mathcal{M}|}{2}$ otherwise.

Let $d(p, q)$ be the distance $\sum_{b=1}^{12}(F_b(q) - F_b(p))^2$ between the corresponding filtered values of p and q. Let $\mathcal{A}_{k,b}(p)$ and $\mathcal{A}_{k,o}(p)$ be sets with the k-closest pixels of p in $\mathcal{T} \cap \mathcal{S}_b$ and $\mathcal{T} \cap \mathcal{S}_o$, respectively, according to distance d. Let $\bar{d}(p, \mathcal{A}_{k,b}(p))$ and $\bar{d}(p, \mathcal{A}_{k,o}(p))$ be the average distances between p and the pixels in $\mathcal{A}_{k,b}(p)$ and $\mathcal{A}_{k,o}(p)$, respectively. A binary classifier assigns to each pixel $p \in \mathcal{E}$ the object label, if $\bar{d}(p, \mathcal{A}_{k,b}(p)) > \bar{d}(p, \mathcal{A}_{k,o}(p))$, and the background label, otherwise. The best value of k and the best subset $\mathcal{T}$ are those which minimize the number of misclassified pixels in $\mathcal{E}$. A procedure in [13] learns these parameters by replacing misclassified pixels in $\mathcal{E}$ by pixels of $\mathcal{T}$ in a few iterations of the classification process. Once, $\mathcal{T}$ and k are fixed, an object membership map is created by

$$M_o(p) \quad = \quad \frac{\bar{d}(p, \mathcal{A}_{k,b}(p))}{\bar{d}(p, \mathcal{A}_{k,o}(p)) + \bar{d}(p, \mathcal{A}_{k,b}(p))}, \tag{22}$$

for all $p \in D_{(p)}$.

E. Decision Tree

Problems of pattern recognition always come to a decision point. Decision trees map questions to define which class will be the one chosen for pixels in the training set $\mathcal{M}$. The decision tree is constructed recursively starting with the root and grows down by successively splitting the set $\mathcal{M}$ with questions regarding each feature F_b. Each split is done according to an impurity criterion which defines that the leaf nodes are always purer than their parents, and the growth of that branch stops when one predefined condition is met. Let n_i be the ith node of a tree d, the mapped question for n_i is a threshold t_j chosen for one specific feature F_b. Multiple thresholds t_j are tested for each component $F_b \in \vec{F}$ and the pair (F_b, t_j) that can best split n_i into two new nodes n_{i+1}, n_{i+2} purer than n_i is chosen as this node's question. In this work it was used the Gini impurity as the criterion of splitting, defined as

$$i(t) \quad = \quad 1 - \sum_{j=1}^{c} P_j^2 \tag{23}$$

where P is the probability associated with a class c. When $i(t) = 0$ we have a pure node containing only one class, and nodes with $i(t) = \frac{c-1}{c}$ have the greatest heterogeneity. The tree growth is stopped by the the following conditions:

- A predefined maximum depth value was reached (in our case 30 nodes);
- There are not enough samples to determine a new node split (less than 20 samples);
- All samples belong to the same class, i.e. the node is completely pure;
- The new split does not provide improvement for the tree.

If all samples of each class c are distinguishable, then it can be constructed a tree that does not misclassify any evaluation samples (i.e. it is super-trained). This embeds a natural unstability in decision trees that is usually controlled by tree prunning.

We used the OpenCV library [14] implementation of the decision tree algorithm descrived above, with cross-validation to determine the best prune. Once the training is completed, an evaluation pixel s can be classified by starting at the root of the tree and going down to a leaf, through a path determined by the question (F_b, t_j) of each node n_i. To output the object membership map M_o needed by the enhancement, instead of using a single tree d the majority vote of a set $\mathcal{D}$ with 10 decision trees was considered. First, object and background pixels in set $\mathcal{M}$ were interleaved and the result was divided equally among training sets for each tree in $\mathcal{D}$. Then, let $\mathcal{D}_o(p) \subseteq \mathcal{D}$ be a subset of trees $d_t \in \mathcal{D}$ that classified a pixel p as object and $\mathcal{D}_b(p) = \mathcal{D} \setminus \mathcal{D}_o(p)$ be the ones that voted for background, the object membership can finally be defined as

$$M_o(p) \quad = \quad \frac{|\mathcal{D}_o(p)|}{|\mathcal{D}_o(p)| + |\mathcal{D}_b(p)|} \tag{24}$$

VI. Experiments and Results

In our experiments, a dataset with 20 natural images with ground-truths was used from [15], [16] (Figure 2 shows some examples). 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.

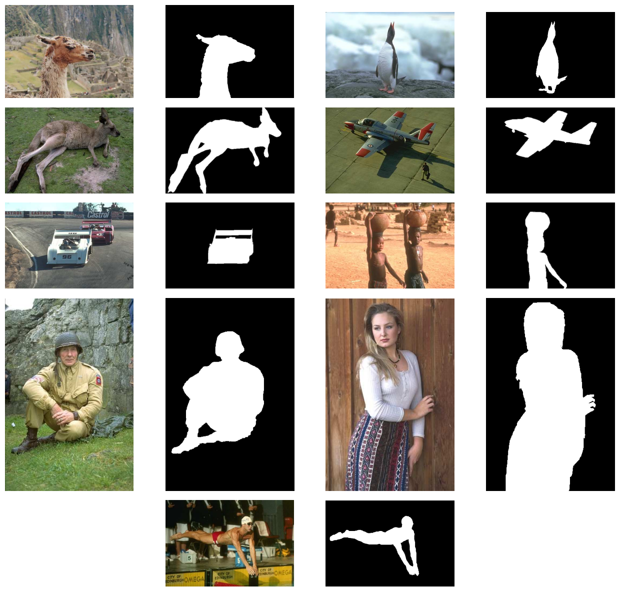
Figure 2. Image samples used in the experiments with their segmentation ground-truths.

To determine how each classifier performs in enhancement tasks and which is its impact in the whole interactive segmentation process, three different users were asked to use each of the 5 classifiers to segment a database image. Therefore, the total number of segmentations per classifier is $60 = 20 \times 3$, whilst the total number of segmentations considered in our study is $300 = 60 \times 5$. Note that, to homogenize our experiments, the same set of markers were used during object enhancement.

From those series of experiments, three different measures were considered to determine the suitability of each classifier in interactive segmentation tasks: (1) F-measure [17], (2) Number of markers, and (3) Total computational time. The F-measure represents the accuracy of each single method and was estimated as the average between the segmentation results, with respect to the ground-truths of the database images. The standard deviation of these measures indicates *precision*. The results of the whole 300 segmentations are presented in Table I. Furthermore, for notation purposes the optimum path forest, Support Vector Machine, Artificial Neural Network, k-nearest neighbors, and Decision Tree methods will be denoted respectively as OPF, SVM, ANN, kNN, and DT.

Table I
AVERAGE RESULTS OF F-MEASURE, NUMBER OF MARKERS AND TOTAL COMPUTATIONAL TIME, WITH THEIR RESPECTIVE STANDARD DEVIATIONS USING OPF, SVM, ANN, kNN, AND DT.

	F-measure	num. of markers	tot. comp. time (sec)
OPF	98.07 $\pm$ 0.73	8.47 $\pm$ 3.83	10.48 $\pm$ 4.92
SVM	97.74 $\pm$ 0.88	7.58 $\pm$ 2.75	18.30 $\pm$ 15.71
ANN	97.85 $\pm$ 0.91	13.73 $\pm$ 8.45	82.81 $\pm$ 22.84
kNN	98.07 $\pm$ 0.75	8.60 $\pm$ 4.19	12.80 $\pm$ 6.91
DT	97.64 $\pm$ 1.00	13.67 $\pm$ 9.93	1.38 $\pm$ 0.54

First, from the accuracy and precision perspective, we can conclude that all five classifiers achieve equivalent results, since there are slight differences in the corresponding average F-measures. Though, we can see that the highest accuracy and precision was achieved by the OPF method (OPF), followed respectively by k-nearest neighbors (kNN) and by Artificial Neural Network (ANN), SVM (SVM), and Decision Tree (DT).

Second, methods OPF, SVM, and kNN can reduce the number of required markers (user interaction) with respect to ANN and DT. Furthermore, note that, for methods ANN and DT there's a great variation in precision. Finally, the total computational time varies drastically for each method. We can observe that the fastest method is the Decision Tree (DT), followed by OPF (OPF), k-nearest neighbors (kNN), SVM (SVM), and Artificial Neural Network (ANN). As one can observe, the duration of the training process in supervised methods has an impact in the computational time.

VII. CONCLUSIONS

We presented a comparative study among fuzzy pixel classifiers designed from popular methods, SVM, ANN, kNN, DT and OPF, for the purpose of object enhancement in interactive image segmentation. We also described a segmentation framework where the user places markers inside and outside the object for recognition and the computer performs object enhancement and extraction. The extraction technique, DIFT, was fixed, as well as the markers for enhancement, being the result of segmentation dependent of the user and the classification method used for enhancement. The experiments involved 20 natural images and 7 users, and the statistics of accuracy (F-measure), user involvement (number of markers for enhancement and extraction), and efficiency (total computational time) in segmentation were obtained. Given that they average the results from multiple users using the same classification method, we can compare the classifiers by analysis of these segmentation results.

From our results we can see that, whilst the classification methods in study achieve similar accuracies, there

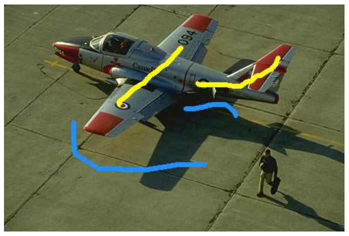

Figure 3. Enhancement markers used to create the object membership maps in figure 4.

are notorious differences in the number of markers and total computational time for segmentation. For instance, the fuzzy classification with SVM was really good, which was reflected in a reduced number of markers (third column in Figure 4). However, SVM is quite dependent of the marker set size and can take a considerable amount of time to train, thus decreasing its interactive efficiency. Multiple ANN classifiers revealed good enhancement (fourth column in Figure 4), but the number of required classifiers had a negative impact in the overall computational time. The fuzzy classifier based on DT failed sometimes and the task was completed only due to the feature-based weights. Also, we have noted that the ANN and DT classifiers perform better when a more balanced and greater amount of object and background pixels is marked for enhancement, to avoid memorization. However, the decision tree is unstable to the point of completely inverting the object membership map, i.e. background regions become brighter than object ones (fifth column in Figure 4). Finally, OPF and kNN provided the best overall results (first and second columns in Figure 4).

VIII. ACKNOWLEDGEMENTS

The authors thank Guilherme Junqueira, Tiago Primini, Lucas Bueno, Carolina Simões Gomes, André Körbes, and Felipe Valio for collaboration; the teams involved in the projects [15], [16] for the images used in this paper; CNPq, FAPESP, and CAPES for financial support.

REFERENCES

[1] 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 (International Conference on Digital Signal Processing)*. Santorini, Greece: IEEE, Jul 2009.

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

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

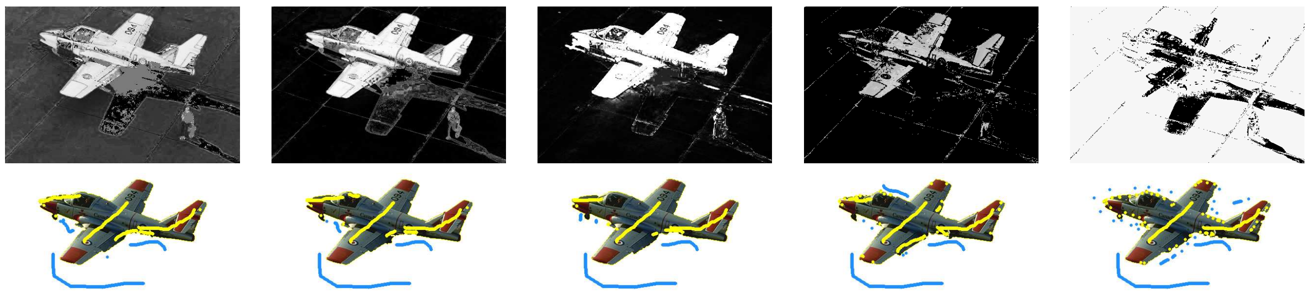

Figure 4. The images on the first row display the object membership maps, and the ones in the second row the segmentation results using the OPF, kNN, SVM, ANN and DT, respectively, for enhancement. All methods used the markers in Figure 3 for enhancement. Note that the segmentation markers in the second row include the enhancement markers.

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

[5] J. P. Papa, A. X. Falcão, C. T. N. Suzuki, and N. D. A. Mascarenhas, "A discrete approach for supervised pattern recognition," in *12th International Workshop on Combinatorial Image Analysis*, vol. 4958. LNCS Springer Berlin/Heidelberg, 2008, pp. 136–147.

[6] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Intl. Journal of Computer Vision*, vol. 40, no. 1, pp. 49–70, 2000.

[7] F. Meyer, "Levelings, image simplification filters for segmentation," *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, pp. 59–72, 2004.

[8] 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*, vol. LNCS 4958. Buffalo, NY, USA: Springer, Apr 7th-9th 2008, pp. 136–147.

[9] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Department of Computer Science, National Taiwan University, Tech. Rep., 2009. [Online]. Available: http://www.csie.ntu.edu.tw/˜cjlin/papers/guide/guide.pdf

[10] C. C. Chang and C. J. Lin, *LIBSVM: a library for support vector machines*, 2001. [Online]. Available: http://www.csie.ntu.edu.tw/˜cjlin/libsvm

[11] S. Haykin, *Neural Networks, A Comprehensive foundation*. Prenticer Hall, Inc, 1999.

[12] R. Hecht-Nielsen, *Neurocomputing*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[13] 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*, pp. 1–14, 2008, doi: 10.1155/2008/467928.

[14] G. Bradski, "The OpenCV Library," *Dr. Dobbs Journal of Software Tools*, 2000.

[15] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "Berkeley segmentation dataset and benchmark." [Online]. Available: http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping

[16] C. Rother, V. Kolmogorov, A. Blake, and M. Brown, "Image and video editing: Grabcut." [Online]. Available: http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/ segmentation/grabcut.htm

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