New hierarchy-based segmentation layer: towards automatic marker proposal

Gabriel Barbosa da Fonseca*†, Romain Negrel‡, Benjamin Perret†, Jean Cousty† and Silvio Jamil F. Guimarães*
*Laboratory of Image and Multimedia Data Science
Pontifical Catholic University of Minas Gerais, Brazil, 31980–110
Email: sjamil@pucminas.br
†LIGM, Université Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallée
Emails: {gabriel.fonseca,benjamin.perret,jean.cousty}@esiee.fr
‡ESIEE Paris, Université Gustave Eiffel
Email: romain.negrel@esiee.fr

***Abstract*—Image segmentation is an ill-posed problem by definition, as it is not always possible to automatically select which object appearing in an image is the object of interest. To deal with this issue, prior knowledge in the form of human-given markers can be included in the segmentation pipeline. Even though user interaction can drastically improve segmentation results, it is an expensive resource, and finding ways to reduce human effort on an interactive segmentation loop is of great interest. In this work, we propose a new segmentation layer to be used with deep neural networks, which allows us to create and train in an end-to-end fashion a marker creation network. To train the network, we propose a loss function composed of: a segmentation loss using the proposed differentiable segmentation layer; and a set of regularization functions that enforce the desired characteristics on the produced markers. We showed that by using the proposed layer and loss function, we can train the network to automatically generate markers that recover a good segmentation and have desirable shape characteristics. This behavior is observed on the training dataset, as well as on four unseen datasets.**

I. Introduction

The recent boom in learning by deep neural networks constitutes a disruptive advance in computer vision [1] making it possible to effectively solve many problems such as object detection [2], localization [3], and classification [4]. In contrast, the general problem of image segmentation remains difficult despite the significant improvements achieved thanks to deep learning. One of the issues related to image segmentation is that it is by definition an ill-posed problem. For instance, in a binary segmentation scenario in which we segment the image into foreground and background instances, it is not possible to define in an unsupervised way which object appearing in an image is the object of interest for a specific user. Hence, in many cases, user supervision is still required to produce satisfactory segmentations.

One way to introduce user-given knowledge in the segmentation process is to provide cues indicating the location of objects of interest in the image. These cues are called *markers* (or *seeds*) and can take different shapes and forms, including scribbles [5]–[7], bounding boxes [8], points [9], and image regions [10], [11]. Also, many works study different approaches for producing marker-based segmentations, ranging from the classic graph-based works relying on watersheds [6], [12], graph cuts [8], [13], random walks [7], geodesics [5] and shortest paths [14], to the more recent works based on convolutional neural networks [9], [15], [16].

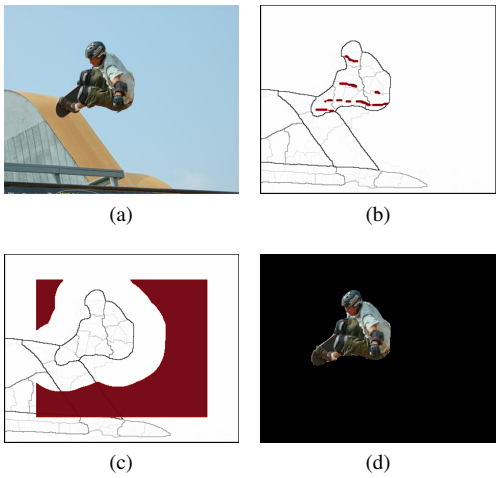


Fig. 1: Example of an automatically generated marker. (a) Image. (b) The object markers generated by the marker creation network. (c) A background marker created from the background segmentation ground-truth. (d) The marker-based segmentation with respect to the markers on (b) and (c).

In general, an interactive procedure allows the user to iteratively refine or edit the markers until a satisfactory segmentation result is obtained [17], [18]. In complex images, *i.e.*, with a large number of objects or with objects that are not easily distinguishable from the background, a user might need to input many cues to produce an acceptable segmentation. In this context, studying ways to reduce the effort requested from a user during an interactive segmentation procedure is of great interest.

In this work, our main contribution is a novel method to train a network for generating markers, using a segmentation layer based on an extension of the widely used marker-based segmentation method presented in [10]. In the original method, markers are propagated to the closest regions of the image,

according to a distance associated to a hierarchy of partitions of the given image. Other works proposed similar strategies to produce a marker-based segmentation, such as the ones based on shortest paths [14] and fuzzy connectedness [19]. In the extension, named fuzzy-marker-based segmentation [20], a fuzzy connection value is computed from the pixels of the image to each marker. Then, each pixel receives the label from the "closest" marker, according to the fuzzy connection values.

By using the proposed segmentation layer, we are able to train the network in an end-to-end fashion, meaning that the segmentation method is embedded as a layer of the deep network architecture. A similar approach is used in [21], where a Random Walker segmentation is used in the process of training a network to learn edge weights of a graph, which are subsequently used to produce a segmentation. In this work, for a given image and a cue about an object of interest, we want our network to propose a set of object and background markers that can be characterized as good markers. As defining optimal markers is complex, we select the following desirable features of good markers: (i) being able to recover the correct segmentation when used as input in the chosen marker-based segmentation method, and; (ii) being easily editable by a user, which could either select, discard or modify parts of a marker during an interaction phase. Based on these features, we propose a loss function composed of a combination of a segmentation loss with a set of regularization functions applied on the markers.

In Section II, we describe the proposed marker learning architecture. In Section III we define the segmentation layer and describe its properties. The segmentation loss and regularization functions are detailed on Section IV. The experimental setup and results are presented in Section V. Finally, in Section VI we draw our conclusions and discuss future works.

II. Marker creation network

In this section we present the proposed architecture for learning markers, combining neural networks and the new segmentation layer based on the fuzzy-marker-based image segmentation method [20].

The main goal of our network is to produce a pair of markers (one for the object and one for the background) for a given image and an object location cue. The object location cue is a binary map with the same size as the input image, indicating where the object of interest is located. We set the object location cue as a part of the input since it would not be possible for the network to properly decide which is the object of interest that should be marked on an image containing multiple appearing objects.

The output of the network consists of an object marker and a background marker. The markers proposed by the network are maps with the same size as the input image, containing values between 0 and 1. Locations that belong to the marker contain values close to 1, *i.e.*, the generated object marker contains values close to 1 on locations where the object of interest appears. It is desired that the markers proposed by our network are able to produce a robust object segmentation and, moreover, that the markers can also be easily editable by a user during an interactive process.

To generate such markers, our goal is to train a deep convolutional neural network $f(I;\theta)$, parametrized by θ. The symbol I denotes the input of the network, which is a pair of an image and an object location cue. To learn the parameters θ, we propose to minimize a loss function composed of two main parts: (i) a segmentation loss L_{Seg} computed over a marker-based segmentation $S(f(I;\theta))$ produced using the markers proposed by the network, and (ii) a set of regularization functions $L_i, ..., L_k$ computed over the markers, to enforce some desirable shape characteristics on the produced markers. The total loss to be minimized, named L_{marker}, is defined by:

$$L_{marker}(f(I;\theta)) = L_{Seg}(S(f(I;\theta)), \hat{S}) + \sum_{i \in k} \lambda_i L_i(f(I;\theta)), \tag{1}$$

where $\hat{S}$ is a ground-truth segmentation for the input image, and λ_i is a weighting factor for the regularization L_i.

To train our marker creation network in an end-to-end fashion, the segmentation module $S(f(I;\theta))$ must be differentiable with respect to the output of the network f. To achieve this, we propose a novel segmentation layer based on the fuzzy-marker-based segmentation method [20], which is an extension of the classical segmentation method proposed by Salembier and Garrido in [10].

An overview of the proposed architecture can be seen in Fig. 2. We can observe that we have the image concatenated to the object location cue as input of the marker creation network. From the given input, a pair of object and background markers is generated. Then, with the output markers, a fuzzy-marker-based segmentation is computed, using our proposed segmentation layer. Also, a set of regularization functions is computed over the proposed markers. In other words, during the training of the network, the loss is based on a combination of the capacity of the markers of producing a good segmentation, and the quality of the marker shape.

The total loss is minimized when the segmentation produced from the generated markers is precisely the ground-truth segmentation and when the markers match desired characteristics enforced by the regularization functions. The desirable marker characteristics include having a small size, being composed by a low number of connected components, being smooth, and being localized far from the boundary of the object. The regularization functions used in this work are defined and explained in Section IV.

III. Segmentation layer

When training our network to generate markers, we want that the created markers produce good segmentations when used on a marker-based segmentation method. Since we do not have reference markers for evaluating their qualities, we propose a novel segmentation layer, which allows us to evaluate the capacity of the markers to produce a good segmentation, needing only a reference segmentation. Reference segmentations in the other hand are easier to define, and are widely available on image segmentation datasets.

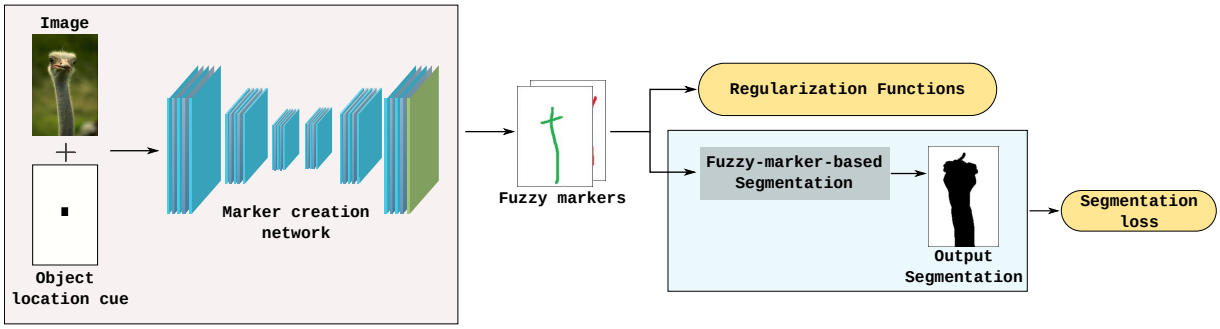


Fig. 2: Overview of the marker creation network training pipeline. The network takes as input the images concatenated with an object location cue, and generates a pair of markers. These markers are then used to produce a fuzzy-marker-based segmentation. The total loss is a combination of a segmentation loss and a set of regularization functions computed over the proposed markers.

The segmentation layer is based on the fuzzy-marker-based segmentation (FMBS) method [20], which is an extension of the classical marker-based segmentation proposed in [10]. In the remainder of this section we define the fuzzy-marker-based segmentation layer, and we start by giving some preliminary notions.

A. Preliminary notions

As in the original marker-based segmentation method proposed in [10], the FMBS produces a segmentation based on an indexed hierarchy produced from the input image. We define the image domain as a finite nonempty set, denoted by V. We define a *hierarchy* $\mathcal{H}$ (on V) as a set of subsets of V such that: (i) V is an element of $\mathcal{H}$; (ii) for every element x of V, the singleton $\{x\}$ belongs to $\mathcal{H}$; and (iii) for any two elements X and Y of $\mathcal{H}$ if the intersection of X and Y is nonempty, then X either includes Y or is included in Y.

Let $\mathcal{H}$ be any hierarchy, any element of $\mathcal{H}$ is called a *region of* $\mathcal{H}$. An *indexed hierarchy (on V)* is a pair $(\mathcal{H}, \omega)$, where $\mathcal{H}$ is a hierarchy and where ω is a function from $\mathcal{H}$ to $\mathbb{R}^+$ such that: (i) $\omega(X) = 0$ if and only if X is a singleton; and (ii) for any two regions X and Y of $\mathcal{H}$, if X is included in Y, then we have $\omega(X) < \omega(Y)$.

Let $\mathcal{H}$ be a hierarchy and let X and Y be two regions of $\mathcal{H}$. The region X is a *parent of* Y and Y is a *child of* X if Y is included in X and if any region of $\mathcal{H}$ which is proper superset of Y is also a superset of X. A region R of $\mathcal{H}$ is called a *leaf (resp. root) of* $\mathcal{H}$ if it is not the parent (resp. child) of any region of $\mathcal{H}$. It can be observed that V is the only root of $\mathcal{H}$ and that the set of leaves of $\mathcal{H}$ is precisely the set of all singletons on V. It can also be noticed that any non-root region X of $\mathcal{H}$ has a unique parent, which is denoted by $par(X)$ in the following. A hierarchy $\mathcal{H}$ is considered a *binary hierarchy* if any non-leaf region has exactly two distinct children.

Hereafter, the pair $(\mathcal{H}, \omega)$ denotes an indexed binary hierarchy, that is an indexed hierarchy such that $\mathcal{H}$ is a binary hierarchy.

In [20], the proposed method relies on ultrametric distances for producing a segmentation. The ultrametric distance associated to an indexed hierarchy $(\mathcal{H}, \omega)$, denoted by $d_{\mathcal{H}}$, is defined as a function from $V \times V$ to $\mathbb{R}^+$ such that, for any two elements x and y of V, the value $d_{\mathcal{H}}(x, y)$ is the index of the smallest region of $\mathcal{H}$ which contains both x and y:

$$d_{\mathcal{H}}(x, y) = \min \{\omega(X) \mid X \in \mathcal{H}, x \in X, y \in X\}. \quad (2)$$

B. Fuzzy-marker-based segmentation

The marker-based segmentation defined in [10] relies on crisp markers to produce a segmentation. In the FMBS, markers are fuzzy, and can indicate the location of an object with different intensities on distinct locations on the image. In the FMBS, the fuzzy markers are represented by fuzzy sets.

A *fuzzy set* (on V) is defined as a function from V to the real interval $[0, 1]$. In [20], the FMBS if defined based on *fuzzy connection values* of elements in V to fuzzy markers. Let O be a fuzzy marker, and let x be an element of V. The *fuzzy connection value of x to O* (for $(\mathcal{H}, \omega)$), denoted by $\mathcal{C}^f_{\mathcal{H}}(x, O)$, is defined by:

$$\mathcal{C}^f_{\mathcal{H}}(x, O) = \min\{\alpha(O(y))(1 - O(y) + d_{\mathcal{H}}(x, y)) \mid y \in V\}, \quad (3)$$

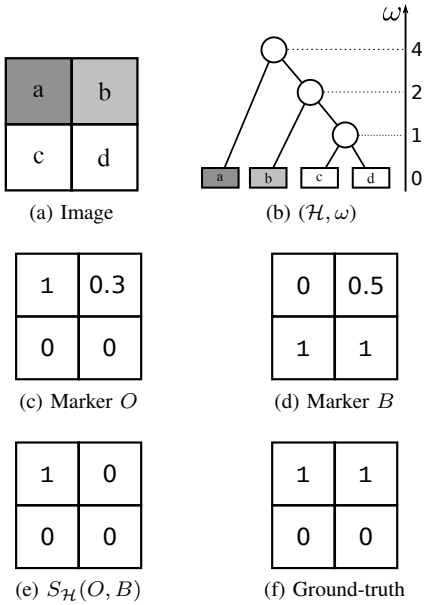
in which α is a decreasing function such that $\alpha(1) = 1$ and $\alpha(0)$ is strictly greater than the maximal value of $d_{\mathcal{H}}$, *i.e.*, $\alpha(0) > \max\{d_{\mathcal{H}}(x, y) \mid x \in V, y \in V\}$.

Finally, the FMBS segmentation with respect to two fuzzy markers, denoted here by $S_{\mathcal{H}}(O, B)$ is given by:

$$S_{\mathcal{H}}(O, B) = \left\{x \in V \mid \mathcal{C}^f_{\mathcal{H}}(x, O) < \mathcal{C}^f_{\mathcal{H}}(x, B)\right\}. \quad (4)$$

In other words, the fuzzy-marker-based segmentation of V for (O, B) is the set that contains every element of V with a fuzzy connection value to O smaller than to B. The intuition behind this definition is that the elements that belong to the segmentation are closer to the object marker than to the background marker.

In [20], an efficient algorithm for computing the FMBS is proposed. The algorithm relies on the structure of a tree representation of the indexed hierarchy. It is shown that with one pass over the tree from leaves to root, followed by one pass from root to leaves, followed by one pass over the elements of V, we can compute the fuzzy connection values for every element of V to a fuzzy marker in linear time with respect to the size of V.



(a) Image (b) $(\mathcal{H}, \omega)$ (c) Marker O (d) Marker B (e) $S_{\mathcal{H}}(O, B)$ (f) Ground-truth

Fig. 3: Toy example of the FMBS. Image on (a) and an indexed hierarchy $(\mathcal{H}, \omega)$ associated to the image on (b). Fuzzy object marker O on (c) and fuzzy background marker B on (d). On (e), the resulting FMBS with respect to markers O and B. Finally, the ground-truth segmentation on (f). In the segmentation and in the ground-truth, values equal to 1 represent elements that belong to the object segmentation. We set $\alpha(x) = 6 - x$ for this example.

C. Differentiable segmentation layer

To train the marker creation network in an end-to-end fashion using traditional gradient descent methods, we must be able to back-propagate the segmentation error through our model. For that, the segmentation layer, highlighted in blue in Fig. 2, must be differentiable with respect to the markers proposed by the network. Here we explain how we create such layer using the FMBS method.

To compute the FMBS of an image with respect a given hierarchy and a pair of markers (object and background), we must get the fuzzy connection value of each element of the image to the object marker and to the background marker, using Equation 3.

Let O be a fuzzy marker, and let $(\mathcal{H}, \omega)$ be an indexed hierarchy associated to V. We can observe in Equation 3 that for the fuzzy connection value of an element $x \in V$ to a marker O, there is an element $y \in V$ that minimizes the equation. We call this element y the closest element to x for a marker O, and we denote it by x^*_O. Intuitively, if we do a small modification on the value of the closest element $O(x^*_O)$, this will lead to a small modification on the fuzzy connection value of x to the marker O. If an element $x \in V$ is mislabeled after a FMBS with respect to two markers O and B, we know which are the closest elements x^*_O and x^*_B that should have their values altered on O and B to fix the labeling of x.

A toy example of a FMBS is illustrated on Fig. 3. We can observe that the element b of the toy image shown in Fig. 3a is mislabeled in the segmentation shown in Fig. 3e, according to the ground-truth illustrated in Fig. 3f. With Equation 3, we can get the value $\mathcal{C}^f_{\mathcal{H}}(b, O)$, as well as the closest element $b^*_O \in V$ that minimizes $\mathcal{C}^f_{\mathcal{H}}(b, O)$. For the example illustrated in Fig. 3, the closest element b^*_O is the element $b \in V$, and its fuzzy connection value to O is $\mathcal{C}^f_{\mathcal{H}}(b, O) = 3.99$. For marker B, we have that the closest element b^*_B is also the element $b \in V$, and $\mathcal{C}^f_{\mathcal{H}}(b, B) = 2.75$. So, to correct the segmentation of the element b, we must modify the values of $O(b)$ and $B(b)$.

In practice, for a pair of markers O and B we can get the closest elements x^*_O and x^*_B for every $x \in V$. Then, we compute the fuzzy connection value of the elements $x \in V$ to a marker O directly by:

$$\mathcal{C}^f_{\mathcal{H}}(x, O) = \alpha(O(x^*_O))(1 - O(x^*_O) + d_{\mathcal{H}}(x, x^*_O)). \quad (5)$$

Having the fuzzy connection values of the elements in V to the markers O and B, instead of producing a crisp segmentation with Equation 4, we use a soft-min to get a soft segmentation differentiable with respect to the fuzzy connection values:

$$S^f_{\mathcal{H}}(O, B) = \frac{e^{-\beta \mathcal{C}^f_{\mathcal{H}}(x,O)}}{e^{-\beta \mathcal{C}^f_{\mathcal{H}}(x,O)} + e^{-\beta \mathcal{C}^f_{\mathcal{H}}(x,B)}}, \quad \forall x \in V,$$

where β is a positive real number. As β tends towards infinity, the soft-min tends to the traditional minimum.

The output of the segmentation layer is given by $S^f_{\mathcal{H}}(O, B)$. We can finally compute a segmentation loss with respect to $S^f_{\mathcal{H}}(O, B)$, get the gradients with automatic differentiation, and minimize the segmentation loss with standard gradient descent algorithms.

Additionally, for computing the gradients efficiently, we can use the algorithm proposed in [20] to get the closest vertices for every element of V. With two passes over the hierarchy followed by one pass over V, we can compute the closest vertex x^*_O for every $x \in V$ (for a marker O) in linear time with respect to the size of V.

IV. Loss and regularization functions

To measure the quality of the generated markers during the training of our network, we combine the segmentation loss with a series of regularization functions. These regularization functions allow us to favor some desirable features on the produced markers. Here, we define the regularization functions that will be used in this work.

For the same image, a network can propose numerous distinct markers that are able to recover a good segmentation. In a scenario where the user can continue to interact after the network generates markers, we want these markers to be simple and easily editable. An intuitive way to delineate the characteristics of such good markers is by thinking about how humans usually interact with an interactive segmentation system. Simple human-given markers are usually represented by a few points and simple strokes, generally located near the center of the objects of interest. Based on this form of human-given markers we define some desirable marker characteristics.

We then propose four regularization functions that allow us to achieve markers that have such characteristics, and these functions are described in the following.

The first characteristic we want to have in a marker is to be small. If we consider the object segmentation ground-truth as a marker, it would be capable of retrieving the ground-truth segmentation when used with a marker-based segmentation method. Although, it does not resembles a human marker in any sense, and editing the ground-truth mask for correcting a segmentation can be quite laborious. To avoid markers that are too large, we define our first regularization function, denoted by L_{size}.

For the remainder of this section, let O be a marker proposed by our network, let $\hat{S_O}$ be the ground-truth segmentation for the object represented by the marker O, and let V be the image domain. The first regularization function, called *size regularization*, is given by:

$$L_{size}(O, \hat{S_O}) = \frac{\sum_{x \in V} O(x)}{\sum_{x \in V} \hat{S_O}(x)}. \quad (6)$$

In other words, this function is minimized when the marker is small compared to the total size of the object.

We also wish that the generated markers are not placed near or over the borders of the object of interest. Users tend to concentrate their initial interactions in the center of the object, and to add markers near the borders for necessary corrections [16]. Based on this, we propose the *distance map regularization*, which is defined by:

$$L_{Dmap}(O, Dmap_O, \hat{S_O}) = \frac{\sum_{x \in V} O(x) Dmap_O(x)}{\sum_{x \in V} \hat{S_O}(x)}, \quad (7)$$

in which for a marker O, $Dmap_O$ is computed based on the euclidean distance map (EDT) of the ground-truth of O. The value $Dmap_O(x)$ for an element x in V is given by:

$$Dmap_O(x) = 1 - \frac{EDT_{\hat{S_O}}(x)}{max(EDT_{\hat{S_O}}(x))},$$

where $EDT_{\hat{S_O}}(x)$ is the euclidean distance from x to the closest point that does not belong to the object represented by $\hat{S_O}$. In other words, this regularization function is minimized when there is no element with high value near the border of the object.

Additionally, we also want our marker to contain a low number of connected components. In order to minimize the number of connected components on the generated markers, we propose to use a third regularization function, which is a topological loss function based on persistent homology [22]. With the use of persistent homology, we can compute the robustness and presence of various topological features at different scales of the proposed markers.

To get the persistent homology of a marker O, we must consider the upper-level sets of O at various scales. The upper-level set of O at a scale p, denoted by $T_O(p)$, comprises the set of elements $x \in V$ such that $O(x) > p$. Beginning with p equal to 1, as we decrease the value of p, more elements are included in $T_O(p)$, and different topological features are formed and destroyed by the inclusion of such elements. The persistent homology consists in counting the different topological features (such as connected components, holes, and hollow voids) found on each $T_O(p)$ for different levels of p.

The only topological feature we are interested in this case is the number of connected components. When analysing the features at different scales, we know at which value p a connected component appears, and at which value it disappears. We call these values the birth time and death time of each connected component, denoted by b and d, respectively. To minimize the number of connected components, use the *component regularization function*, defined by:

$$L_{comp}(O; l) = \sum_{i=1}^{i \leq l} (1 - |b_i^O - d_i^O|^2) + \sum_{i=l+1}^{i \leq k} |b_i^O - d_i^O|^2, \quad (8)$$

in which l is the desired number of connected components, and k is the total number of features found for O. For more details on the topological loss, the reader may refer to [22]. Furthermore, in this work we use an efficient implementation of this topological loss based on component trees [23].

Finally, to avoid noisy markers and to produce smooth ones, we apply a total variation regularization. The markers considered in this work are 2D maps of width W and height H, and every element x in a marker O has a pair (j, k) associated to it, representing its vertical and horizontal position. The third regularization is a *total variation regularization*, defined by:

$$TV(O) = \sum_{\substack{j \in W \\ k \in H}} |O(x_{j,k}) - O(x_{j+1,k})| + |O(x_{j,k}) - O(x_{j,k+1})|$$

$$L_{TV}(O, \hat{S_O}) = \frac{TV(O)}{\sqrt{\sum_{x \in \hat{S_O}} \hat{S_O}(x)}} \quad (9)$$

The total variation regularization is minimized when neighbouring pixels have similar values across the marker.

The terms in the denominator of the total variation, distance map and size regularizations are included to make these losses invariant to the size of the object of interest.

V. Experiments

In this section we describe our experimental setup, the proposed evaluation metrics, the results computed over different datasets, and a quantitative and qualitative analysis of our method.

A. Experimental setup

Datasets. In this work we conducted experiments over five distinct datasets. For training, we use the MSRA10K saliency dataset [24]. This dataset is composed of 10.000 samples. Each sample consists in an RGB image, along with a mask representing its segmentation ground-truth. We split the dataset into two subsets, with 9000 images for training and 1000

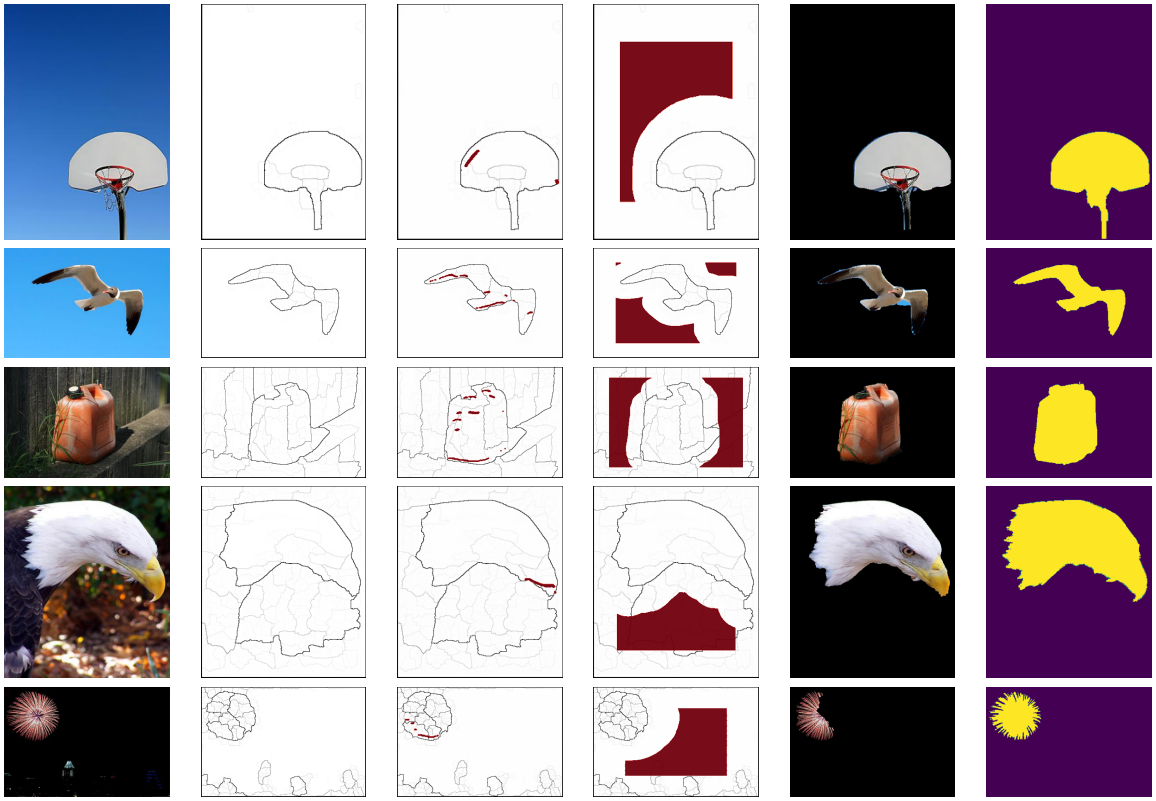


Fig. 4: Results for samples of the MSRA10K validation set. From left to right: input image; saliency map representation of the hierarchy constructed for the input image; object markers proposed by the network superimposed over the saliency map; background markers produced from the ground-truth segmentation, superimposed over the saliency map; resulting fuzzy-marker-based segmentation; and segmentation ground-truth.

images for validation. Additionally, during training we also make use of data augmentation to enrich the dataset.

Along with the MSRA10K, we evaluate our method over four unseen datasets, which are described in the following. We use the Weizmann single object [25], composed of 100 images, and with three ground-truth segmentations proposed by different users for each image. During our experiments we only use the first ground-truth. We also use the Weizmann Horses [26], which contains 328 images of horses, with a segmentation mask for each image. We also use the GSCSEQ dataset [27], which is composed 151 images taken from the GrabCut [8], Pascal VOC [28] and Alpha matting [29] datasets. The GSCSEQ contains the ground-truth mask for each image and also contains scribble information, although in our experiments we only consider the images and ground-truths. Finally, we also test our method on the Pascal VOC 2012 segmentation dataset [28]. This dataset contains 1464 images for training and 1449 images for validation. Each image can contain more than 1 object (of the same or different class), but we only consider 1 object per image, and treat the rest of the image as background.

Marker proposal network. In this work we use a U-net network [30], with a Resnet18 [31] encoder. The input of the network is the RGB image concatenated to a binary object cue mask, ending in a four-channel 320×320 input. For the encoder, with the exception of the first layer, we use the pre-trained weights on the ImageNet dataset. For our network, as for the pre-trained weights, we use the available model implementation from the *Segmentation models pytorch* library [32]. We use a sigmoid activation function in the output, which consists of a 1-channel 320×320 mask, representing the object marker.

In the experiments presented in this work, we only learn the object marker, and use an euclidean distance transform of the background segmentation as a fixed background marker for each image. Additionally, we use the segmentation ground-truth as the object location cue. The goal of the presented experiments is to show that we can learn good markers with the proposed marker creation network, without using optimal markers as reference. In future works, we aim to use weaker cues indicating the object location, as well as learning both the object and background markers simultaneously.

Optimization. We use the Adam optimizer with a learning rate of 5×10^{-4}. We set the batch size equal to 20, and we train the network for 2250 steps. During training, we use a soft Dice loss [33] as the L_{Seg} term. The values of each weight λ for the total loss were manually tuned, and set to: 20 for the L_{seg}, 10 for R_{size}, 3 for R_{Dmap}, 0.01 for R_{TV}, and 0.1 for R_{comp}.

Finally, the value of β is set to 25.

Hierarchy computation. The FMBS produces a segmentation based on a pair of markers and a hierarchy produced over the input image. The hierarchies in this work are computed with a watershed by area hierarchy [34] created over a 4-adjacency graph weighted by the structured edge detector (SED) gradient [35].

B. Evaluation metrics

The evaluation procedure and metrics used to evaluate the markers generated using our method are described in this section.

During training, we resize the images to a fixed size of 320×320, and the network output is a fuzzy marker, *i.e.*, taking values in $[0, 1]$, also with size of 320×320. During our evaluation process, we compute the markers using our network, and resize them to the original image size. For computing some of the evaluation metrics regarding the geometric properties of the proposed markers, the markers must be binary. So, we binarize the markers generated by the network by using Otsu's threshold [36], and the produced background markers by taking the values higher than a threshold, set at 0.5. We finally compute a dilation with a disk structuring element of radius 4 followed by an erosion with a disk structuring element of radius 3, to connect nearby components that have small gaps between them.

Let $\mathbb{O}$ and $\mathbb{B}$ be binarized markers, and let $S_i(\mathbb{O}, \mathbb{B})$ be the segmentation with respect to the markers $\mathbb{O}$ and $\mathbb{B}$ for an image V_i. Let $\hat{S}_i$ be the segmentation ground-truth for V_i. The first evaluation metric used is the vastly used IoU segmentation metric, defined as follows:

$$IoU(S_i(\mathbb{O}, \mathbb{B}), \hat{S}_i) = \frac{S_i(\mathbb{O}, \mathbb{B}) \cap \hat{S}_i}{S_i(\mathbb{O}, \mathbb{B}) \cup \hat{S}_i} \tag{10}$$

In the second metric, we evaluate the relative size between the proposed markers and the object of interest. This is computed with the size regularization metric over the binarized markers. For a given marker $\mathbb{O}$ and a segmentation ground-truth $\hat{S_O}$, the *relative size metric* is given by $R_{size}(\mathbb{O}, \hat{S_O})$.

The third metric measures the number of connected components on each proposed marker. We denote the number of connected components of a marker $\mathbb{O}$ by $CC(\mathbb{O})$.

Finally, we also propose a metric to measure the thinness of the proposed markers. Ideally, if the proposed markers look like scribbles or points located over the image, they should not be too thick. To measure the thinness of a marker we measure the relative size between a marker and a skeleton produced over the marker. Let $\mathbb{O}$ be a crisp marker and $sk(\mathbb{O})$ be the skeleton produced from a skeletonization process applied over $\mathbb{O}$. The skeleton $sk(\mathbb{O})$ is a binary map with the same size of $\mathbb{O}$, with values of 1 for the elements that belong to the skeleton, and 0 for the remaining elements. The thinness metric is defined by:

$$Thinness(\mathbb{O}) = \frac{\sum_{x \in V} \mathbb{O}(x)}{\sum_{x \in V} sk(\mathbb{O})(x)}. \tag{11}$$

TABLE I: Evaluation metrics for all datasets

	IoU	*CC*	R_{size}	*Thinness*
MSRA10K	0.767	13.164	0.048	4.058
Weizmann One Object	0.733	14.02	0.053	3.835
Weizmann Horses	0.745	19.92	0.041	4.458
GSCSEQ	0.652	14.245	0.042	4.275
Pascal VOC 2012	0.590	12.322	0.720	4.056

In other words, this metric measures how many times a maker is larger then its skeleton.

C. Results

We first evaluate our method over the MSRA10K dataset, the one used to train the network. Qualitative results are shown in Fig. 4. We can see in the figure some examples of the MSRA10K dataset, with the generated object markers and the produced segmentations. It is important to reiterate that for the experiments in this article, we learn only the object marker. We can observe that, for most cases, it is possible to recover good object segmentations by using the automatically generated markers. For the last case, on the bottom row, the produced segmentation is not good despite the markers being well-placed in the object of interest. This happens since there are strong boundaries inside the object in the constructed hierarchy, which makes the marker-based segmentation more difficult. It is also possible to see that the generated markers follow the desired shape characteristics, which are enforced by the regularization functions. Although some of the generated markers are placed near the borders of the object, they are relatively small, smooth and contain a small number of connected components.

In table I, the evaluation metrics are shown for all datasets, with the results for the MSRA10K dataset (the one used for training) highlighted in grey. The metrics shown are an average over the validation sets of the MSRA10K and Pascal VOC datasets, and over the complete dataset for Weizmann single object, Weizmann Horses and GSCSEQ.

We can observe from the *IoU* metric that the network is able to propose markers that recover a good segmentation for the MSRA10K validation set, as well as for the unseen datasets. The lowest IoU score was achieved on the Pascal VOC dataset. This is expected, as the images from the Pascal VOC dataset are more complex, containing multiple objects with varied sizes, including very small objects in some cases.

The *CC*, R_{size}, and *Thinness* metrics confirm that the generated markers follow the desirable characteristics. The average number of connected components over all datasets is 14.73, showing that the proposed markers are not formed by a very large number of components. Furthermore, the markers are approximately 4 times thicker than their skeletons, which have a thickness of 1 pixel, showing that they are also thin. According to the R_{size} metric, the proposed markers are also small when compared to the objects of interest on all datasets, with the exception of Pascal VOC dataset. Many images on the Pascal VOC dataset contain very small objects, and this explains why thin markers (4.056 times larger than

their skeletons) are relatively large with respect to the size of the objects of interest.

VI. Conclusion and future work

In this work, we have proposed a novel segmentation layer that can be used together with neural networks to create an end-to-end trainable marker learning architecture. In the limit case where the fuzzy markers tend to be binary, the results of the new proposed segmentation layer tend towards those of the widely used marker based segmentation described in [10], so that the proposed learned makers are adapted to (reduce the effort of) the users of this popular method. We showed that by using the proposed layer and a loss function composed by a segmentation loss and a set of regularization functions that control the shape of the markers, we can automatically generate markers that recover a good segmentation and have desirable shape characteristics. This behaviour is observed on the dataset used for training, as well as on four unseen datasets.

In future works, we plan to continue the studies on the marker learning problem, modifying the input for weaker object location cues and exploring new regularization functions that favour desirable features on markers. We would also like to insert users in the training loop, to learn from their feedback and corrections on the proposed markers in order to improve the quality of those generated in the future.

Acknowledgment

The authors thank the Pontifícia Universidade Católica de Minas Gerais (PUC-Minas), the Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq – (PQ 310075/2019-0), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES – (Grant COFECUB 88887.191730/2018-00) and Fundação de Amparo a Pesquisa do Estado de Minas Gerais – FAPEMIG – (Grants PPM-00006-18). Furthermore, this study was also financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[1] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *TPAMI*, vol. 35, no. 8, pp. 1915–1929, 2012.

[2] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *NNLS*, vol. 30, no. 11, pp. 3212–3232, 2019.

[3] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free? weakly-supervised learning with convolutional neural networks," in *CVPR*. IEEE, 2015, pp. 685–694.

[4] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification," in *CVPR*. IEEE, 2016, pp. 2285–2294.

[5] X. Bai and G. Sapiro, "Geodesic matting: A framework for fast interactive image and video segmentation and matting," *IJCV*, vol. 82, no. 2, pp. 113–132, 2009.

[6] J. Cousty, G. Bertrand, L. Najman, and M. Couprie, "Watershed cuts: Minimum spanning forests and the drop of water principle," *PAMI*, vol. 31, no. 8, pp. 1362–1374, 2008.

[7] L. Grady, "Random walks for image segmentation," *PAMI*, vol. 28, no. 11, pp. 1768–1783, 2006.

[8] C. Rother, V. Kolmogorov, and A. Blake, ""Grabcut": interactive foreground extraction using iterated graph cuts," *TOG*, vol. 23, no. 3, pp. 309–314, 2004.

[9] W.-D. Jang and C.-S. Kim, "Interactive image segmentation via backpropagating refinement scheme," in *CVPR*. IEEE, 2019, pp. 5297–5306.

[10] P. Salembier and L. Garrido, "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval," *TIP*, vol. 9, no. 4, pp. 561–576, 2000.

[11] F. Malmberg, R. Nordenskjöld, R. Strand, and J. Kullberg, "Smartpaint: a tool for interactive segmentation of medical volume images," *CMBBE*, vol. 5, no. 1, pp. 36–44, 2017.

[12] C. Vachier and F. Meyer, "The viscous watershed transform," *JMIV*, vol. 22, no. 2-3, pp. 251–267, 2005.

[13] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *ICCV*, vol. 1. IEEE, 2001, pp. 105–112.

[14] A. X. Falcão, J. Stolfi, and R. de Alencar Lotufo, "The image foresting transform: Theory, algorithms, and applications," *PAMI*, vol. 26, no. 1, pp. 19–29, 2004.

[15] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang, "Deep interactive object selection," in *CVPR*. IEEE, 2016, pp. 373–381.

[16] Z. Li, Q. Chen, and V. Koltun, "Interactive image segmentation with latent diversity," in *CVPR*. IEEE, 2018, pp. 577–585.

[17] A. X. Falcão and F. P. Bergo, "Interactive volume segmentation with differential image foresting transforms," *T-MI*, vol. 23, no. 9, pp. 1100–1108, 2004.

[18] B. Perret, J. Cousty, J. C. R. Ura, and S. J. F. Guimarães, "Evaluation of morphological hierarchies for supervised segmentation," in *ISMM*. Springer, 2015, pp. 39–50.

[19] J. K. Udupa and S. Samarasekera, "Fuzzy connectedness and object definition: Theory, algorithms, and applications in image segmentation," *Graphical models and image processing*, vol. 58, no. 3, pp. 246–261, 1996.

[20] G. B. da Fonseca, B. Perret, R. Negrel, J. Cousty, and S. J. F. Guimarães, "Fuzzy-marker-based segmentation using hierarchies," in *DGMM*. Springer, 2021, pp. 391–403.

[21] L. Cerrone, A. Zeilmann, and F. A. Hamprecht, "End-to-end learned random walker for seeded image segmentation," in *CVPR*. IEEE, 2019, pp. 12 559–12 568.

[22] J. Clough, N. Byrne, I. Oksuz, V. A. Zimmer, J. A. Schnabel, and A. King, "A topological loss function for deep-learning based image segmentation using persistent homology," *TPAMI*, 2020.

[23] B. Perret and J. Cousty, "Component tree loss function: Definition and optimization," *CoRR*, vol. abs/2101.08063, 2021. [Online]. Available: https://arxiv.org/abs/2101.08063

[24] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu, "Global contrast based salient region detection," *TPAMI*, vol. 37, no. 3, pp. 569–582, 2014.

[25] S. Alpert, M. Galun, A. Brandt, and R. Basri, "Image segmentation by probabilistic bottom-up aggregation and cue integration," *TPAMI*, vol. 34, no. 2, pp. 315–327, 2011.

[26] E. Borenstein and S. Ullman, "Learning to segment," in *ECCV*. Springer, 2004, pp. 315–328.

[27] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman, "Geodesic star convexity for interactive image segmentation," in *CVPR*. IEEE, 2010, pp. 3129–3136.

[28] M. Everingham and J. Winn, "The pascal visual object classes challenge 2012 (voc2012) development kit," *PASCAL, Tech. Rep*, vol. 8, 2011.

[29] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, "A perceptually motivated online benchmark for image matting," in *CVPR*. IEEE, 2009, pp. 1826–1833.

[30] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*. Springer, 2015, pp. 234–241.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*. IEEE, 2016, pp. 770–778.

[32] P. Yakubovskiy, "Segmentation models pytorch," https://github.com/qubvel/segmentation_models.pytorch, 2020.

[33] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 565–571.

[34] J. Cousty and L. Najman, "Incremental algorithm for hierarchical minimum spanning forests and saliency of watershed cuts," in *ISMM*. Springer, 2011, pp. 272–283.

[35] P. Dollár and C. L. Zitnick, "Fast edge detection using structured forests," *PAMI*, vol. 37, no. 8, pp. 1558–1570, 2014.

[36] N. Otsu, "A threshold selection method from gray-level histograms," *SMC*, vol. 9, no. 1, pp. 62–66, 1979.