Pattern Recognition 45 (2012) 1159–1179

Contents lists available at SciVerse ScienceDirect

Pattern Recognition

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

Interactive image segmentation by matching attributed relational graphs

Alexandre Noma [a,*], Ana B.V. Graciano [a], Roberto M. Cesar Jr [a], Luis A. Consularo [b], Isabelle Bloch [c]

[a] Institute of Mathematics and Statistics, University of São Paulo Rua do Matão, 1010, CEP 05508-090 São Paulo, Brazil
[b] Tribunal Superior Eleitoral, Praça dos Tribunais Superiores—Bloco C—SAS—CEP 70096-900, Brasília-DF, Brazil
[c] Telecom ParisTech, CNRS LTCI, 46 rue Barrault, 75013 Paris, France

ARTICLE INFO

Article history:
Received 27 October 2010
Received in revised form
10 August 2011
Accepted 13 August 2011
Available online 1 September 2011

Keywords:
Interactive image segmentation
Matching attributed relational graphs
Deformed graph
Spatial configuration

ABSTRACT

A model-based graph matching approach is proposed for interactive image segmentation. It starts from an over-segmentation of the input image, exploiting color and spatial information among regions to propagate the labels from the regions marked by the user-provided seeds to the entire image. The region merging procedure is performed by matching two graphs: the input graph, representing the entire image; and the model graph, representing only the marked regions. The optimization is based on discrete search using deformed graphs to efficiently evaluate the spatial information. Note that by using a model-based approach, different interactive segmentation problems can be tackled: binary and multi-label segmentation of single images as well as of multiple similar images. Successful results for all these cases are presented, in addition to a comparison between our binary segmentation results and those obtained with state-of-the-art approaches. An implementation is available at http://structuralsegm.sourceforge.net/.

1. Introduction

The goal of the image segmentation problem is to divide the input image into different meaningful regions, e.g. to extract the object of interest from the original background. User intervention is often required, encoding prior information into the process through seeds, such as scribbles that the user draws over an image to identify the regions of interest. Ideally, after roughly placing the scribbles, the remainder of the image is automatically segmented. In practice, the user should be allowed to add/remove scribbles to make corrections on the segmentation process and achieve the desired result. The system must be easy to use and the segmentation algorithm should be fast enough to interact with the user, producing accurate results with minimal effort.

A non-interactive model-based image segmentation method has been introduced in [10], where models were obtained from pre-selected images with their respective manual segmentations. The goal was to propagate these manual segmentations to other similar images, by matching model regions with those from the input images (see Fig. 1), for segmentation and recognition of image structures.

The present paper focuses on interactive image segmentation, where the user provides scribbles which are used as input to segment the image (either the same, or a set of similar ones). Therefore, it is natural to think about how to represent the information provided by the user through the scribbles. Most traditional approaches represent them as markers (watershed), seeds (region growing methods) or graph nodes (graph-cut-like methods). However, we are interested in explicitly representing both the image information (grey-level, color, texture) and the structural information (spatial relations) provided by the scribbles. This is useful in a number of situations where the structure of the scene is important to recognize individual objects. For instance, for disambiguating objects having similar appearance in an image, their spatial arrangement is very useful. In this sense, graphs are a natural choice, since we may represent both types of information in a single structure. Inspired by the model-based approach described in [10], the proposed framework is based on over-segmentation, e.g. produced by the watershed transform [34], to build both input and model graphs. Note that, differently from [10], the proposed method follows an interactive model generation approach. In order to be more user-friendly, instead of using manual segmentations for the models, the new method takes advantage of the scribbles provided by the user in order to propagate the segmentations, in which the model graphs can be interactively updated according to the changes on the scribbles. Moreover, the introduced approach allows segmenting both cases, single and multiple similar images, by using model and input graphs encoding information from the same image or from two different images.

Following a preliminary version of this work, described in [11], we focus on the usual version of the interactive image

* Corresponding author. Tel.: +55 11 5016 1873; fax: +55 11 3091 6134.
E-mail addresses: alex.noma@gmail.com (A. Noma), abvg@vision.ime.usp.br (A.B.V. Graciano), roberto.cesar@vision.ime.usp.br (R.M. Cesar Jr), luis.consularo@tse.jus.br (L.A. Consularo), isabelle.bloch@telecom-paristech.fr (I. Bloch).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.08.017

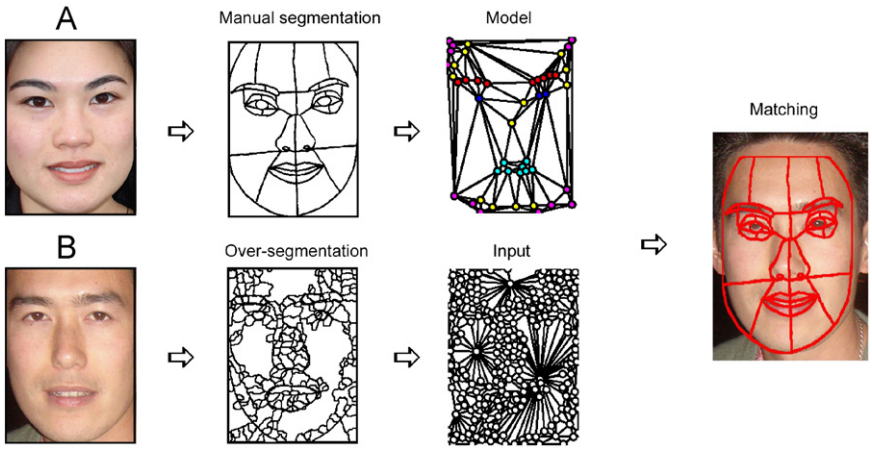

Fig. 1. Our method was inspired by the graph matching approach described in [10]. In that paper, a model graph represented a manual segmentation, while an input graph represented an over-segmentation of another image. The goal was to propagate manual information to other input images for segmentation and recognition of image structures.

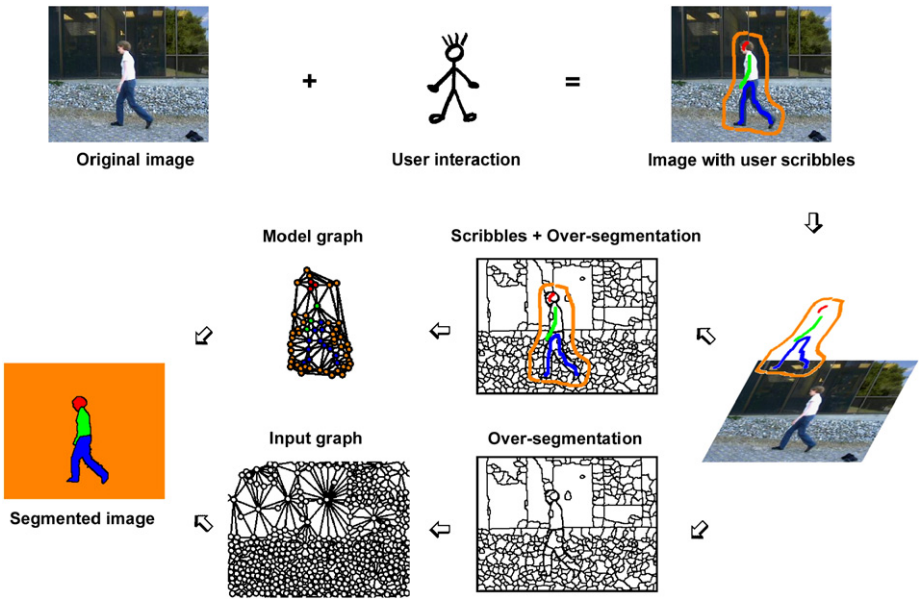

Fig. 2. Method overview. Initially, the user scribbles over the image to indicate what the objects of interest are. Next, an over-segmentation is computed from the input image. This over-segmentation originates two graphs: an input graph and a model one. In the model graph, the label of each vertex corresponds to the color of the scribble intercepting its respective region. Finally, a segmentation may be found by matching these graphs to propagate the labels from the marked regions to all non-marked ones, producing a labeling of the whole input image.

segmentation problem, involving single images, by formulating it as a matching task between (attributed relational) graphs. In this case, the input image is over-segmented to produce two attributed relational graphs (ARG), i.e., graphs in which vertices and/or edges are associated to feature vectors: the input ARG, representing all regions[1] (corresponding to the entire image), and the model ARG, representing only the regions intercepted by the scribbles (Fig. 2).

The goal is to map each input vertex to a model vertex, resulting in a many-to-one correspondence, in which the segmentation is achieved by propagating the labels given by the model vertices, as shown in Fig. 3.

To compute a match between these graphs, a cost function must be optimized in order to evaluate and choose a proper mapping among the exponential number of possible solutions. This cost function may consider appearance (e.g. average intensities/color of each region) and structural features of the image regions (e.g. relative positions among region centroids), encoded as vertex and edge attributes respectively.

In general, the evaluation of the similarity between input and model graphs can be too costly due to the presence of noise or distortions between input and model patterns, which directly affects vertex and edge attributes. Note that the interdependences represented by the graph edges turn the general graph matching problem into a challenging task, especially when we need to locate a 'small' graph within a 'large' one.

[1] Regions offer many advantages over pixel features: regions are more noise-tolerant and make constraints, such as contiguity/smoothness and adjacency, easier to formulate, as observed in [31,32].

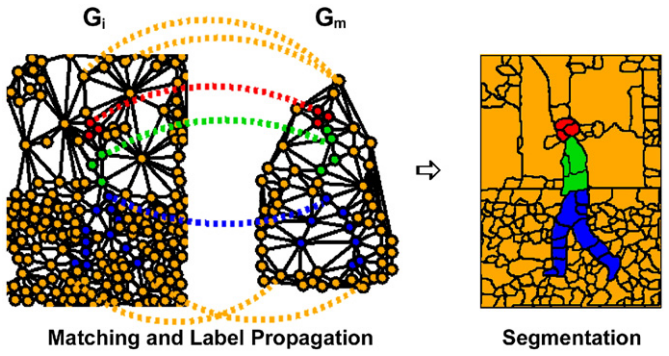

Fig. 3. Label propagation. Each scribble is characterized by a color. Each model vertex is associated to a label corresponding to the color of the scribble intercepting its respective region. During the matching step, these labels are propagated from the model G_m to the input graph G_i. This defines a complete segmentation for the image represented by G_i, by painting each pixel with the respective color assigned to the input region. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Nevertheless, when we have a 'good' initialization to align both graphs, these interdependences can be ignored and we can efficiently evaluate the spatial configurations of vertices by using deformed graphs (DG) [25], where each candidate pair of vertices (input, model) can be examined independently of the already mapped vertices. This is the key idea in the graph matching step adopted in this paper.

In our experiments, by replacing the previous tree search-based optimization algorithm [11] by the DG technique [25], we observed good matching results which were comparable to the tree search algorithm using complete model graphs, speeding up the segmentation process and allowing more interactivity with the user to produce more accurate segmentations. Moreover, we combined the DG algorithm with two post-processing steps. Firstly, by imposing a simple connectivity constraint between the segmented regions and the scribbles, we noticed that the user effort can be significantly reduced in the sense that few scribbles can produce the desired segmentation. For instance, the connectivity constraint is also present on segmentation approaches based on geodesic distance [5,26,27], random walks [12,14,17], and in the maximal similarity based region merging technique [24] by merging adjacent regions. In order to improve the boundaries of the extracted objects, the second step consisted in a pixel-based refinement, producing results which are comparable to the current state-of-the-art algorithms.

The remainder of this paper follows in Section 2 with a brief description about related works. Section 3 describes the proposed framework to match attributed relational graphs, and the optimization algorithm based on deformed graphs [25]. Section 4 presents successful results using simple scribbles, and quantitative comparison with some of the main current graph-based methods. Besides successful results for binary and multi-label segmentation for the usual case, examples illustrating the additional feature, involving multiple similar images segmentation with a single set of user seeds, are also provided. Finally, some conclusions are drawn in Section 5, besides a brief discussion about future work.

2. Related work

This section describes the main classes of methods designed for interactive segmentation relying on graphs (e.g. for applications such as photo-editing), taking into account the user intention, typically through interactions at the beginning of the segmentation process. This overview does not aim to be exhaustive, being provided to exhibit the main trends in current research.

A pioneering work involving seeds for image segmentation was due to Beucher and Meyer [6], in which the authors combined watershed with markers. Instead of using local minima to produce the regions/basins in the watershed, seeds may be provided (or found automatically) to produce the segmentation, dividing the image into the desired regions. Differently from the watersheds with markers [6], in the present work, the provided seeds are used to collect color information from the regions produced by the local minima, and structural information represented by the spatial relations, which are used to disambiguate regions having similar colors and belonging to different objects.

Currently, the main techniques using graphs for interactive segmentation are based on watershed, graph cuts, shortest paths (geodesic) and random walker.

The method based on interactive graph cuts (IGC) [7] was the first work involving graph cuts (GC) for interactive image segmentation, proposed by Boykov and Jolly. The segmentation is performed by the min-cut/max-flow algorithm. The user scribbles extract color information and are used as hard constraints. It has become very popular due to its strong mathematical foundation provided by the MAP-MRF framework. The GrabCut [28] algorithm extended the IGC method by simplifying user interaction.

A very useful segmentation benchmark, with a platform implementing important algorithms, has recently been proposed by McGuinness and Connor [23]. The authors compared important algorithms such as IGC [7], seeded region growing (SRG) [2], simple interactive object extraction (SIOX) [15] and binary partition tree (BPT) [1,29], in order to provide a good coverage of the various techniques currently available for foreground extraction, as stated in [23].

The SRG [2] method, proposed by Adams and Bischof, is very popular due to its simplicity and speed, assuming that regions of interest are characterized by connected pixels with similar colors. The SIOX [15] algorithm is also based on color information and has recently been integrated into the popular imaging program GIMP as the "Foreground Selection Tool". The BPT [1,29] algorithm is based on hierarchical region segmentation, exploiting the user interaction to split and merge regions in the tree.

Bai and Sapiro [5] proposed a method based on fast kernel density estimation [37] for the color statistics, improving the geodesic distance-based approach described in [27]. Combining the improved color statistics and the connectivity constraint imposed by the geodesic distance (between the segmented regions and the scribbles), the authors of [5] illustrate accurate segmentations using simple scribbles. Here, we also exploit the connectivity restriction in a separate post-processing step.

Grady [17] proposed random walks (RW), where each pixel is labeled based on the probability that a random walker reaches the pixel, starting from a scribble.

Variants of IGC, geodesic segmentation and RW have recently been proposed in the literature. An example is [14], where the authors formulated the interactive image segmentation problem as a statistical transductive inference, combining GC and RW. Another example is the geodesic GC [26], which uses geodesic distance information combined with edge information in a GC optimization framework.

An important work unifying GC, RW and shortest paths (geodesic) optimization techniques has recently been proposed by Couprie et al. [12], in which the authors introduced the power watershed technique, representing a new family of segmentation algorithms.

Excluding the BPT [1,29] method, all other approaches described above are based on pixels. Another important class of algorithms is represented by region merging techniques. Li et al. [21] presented a region merging method combining GC and watershed regions. Ning et al. [24] have recently proposed a novel maximal similarity based region merging (MSRM) mechanism for interactive image segmentation. The key idea of MSRM is to perform region merging between adjacent regions by exploiting an effective representation for the color statistics based on (quantized) color histograms computed from the regions. Compared to pixel-based approaches, region merging strategies aggregate robustness to noise and pixel variation in general.

Regarding the related literature, the relevance of the proposed technique has the following advantages. (1) It has been implemented as an open-source software. (2) It is trivially extensible from binary to multi-label, and from single to multiple images segmentation. (3) Good results were achieved when segmenting complex images, in which both foreground and background regions have similar colors, being comparable to the main current graph-based algorithms. (4) The graph matching step can be easily combined with the connectivity constraint between the segmented regions and the scribbles in order to reduce user effort. (5) By including a pixel-based refinement to improve the boundaries of the extracted objects, the proposed method produced accurate segmentations for a wide range of natural images.

3. Proposed framework

3.1. Segmentation by matching attributed relational graphs

Interactive segmentation relies on user hints (markers or scribbles) to segment objects in an image. In the case of binary segmentation, there are two types of scribbles, one for the foreground and the other for the background. In multi-label segmentation, the user defines as many scribbles as there are objects of interest in the image. In both cases, we consider each scribble to have a distinct color, which will also identify the referenced object during object segmentation. This prior information/labeling introduced by the user must be propagated to the non-marked regions in order to achieve a complete segmentation of the input.

In order to propagate the labels, we represent the input image data and the information encoded by the scribbles by means of attributed relational graphs (ARG) [33]. This data structure is a directed graph, in which we associate attribute vectors to vertices and edges. In the remainder of the text, all references to graphs imply a reference to an ARG.

We denote an ARG by $G=(V,E,\mu,\nu)$, in which V is a set of vertices, E is a set of (directed) edges, μ represents the vertex attributes, and ν represents the edge attributes. Cardinalities of the vertex and edge sets are denoted by $|V|$ and $|E|$ respectively.

In this paper, vertices represent image regions and edges represent spatial configurations among these regions. Also, region appearance information is encoded as vertex attributes, and the structural constraints as edge attributes. To obtain these graphs from an image, we first obtain an over-segmentation of the input image in which the contours of each object are expected to be present. In this paper, this is achieved by the watershed algorithm [34]. The resulting over-segmented image can then be used to create an input ARG and/or a model ARG, as explained next.

Input graph: We shall denote this graph by $G_i=(V_i,E_i,\mu_i,\nu_i)$ and similar subscripts will be adopted for its individual vertices and edges. It can be obtained from the input image as follows. The input vertex set V_i is defined by the watershed regions, in which there is a vertex representing each region. For the edge set E_i, there are different possibilities to represent the relations between vertices. For instance, an edge can be created to link vertices corresponding to adjacent regions. Then, vertices and edges can be attributed, as described in the next section.

Model graph: The model graph is denoted by $G_m=(V_m,E_m,\mu,\nu)$, and similar subscripts will be adopted for its individual vertices and edges. Differently from the input graph, this graph represents only specific regions of an over-segmented image: those which are intercepted by a scribble. For each of such regions, a vertex is created. If more than one scribble fall upon a single region issued from the watershed, then the dominant one (i.e. the one which intercepts more pixels from the region) can be chosen as the region label. Note that, in general, the marked regions may not induce a connected graph if edges are created based on region adjacency. By computing a Delaunay triangulation based on the considered region centroids to build the set E_m, close regions can be linked by an edge while keeping G_m connected. Finally, the attributes are computed.

3.2. ARG attributes

The choice of ARG attributes depends on the application. In our interactive segmentation experiments, we compute the same types of vertex and edge attributes for both G_i and G_m as follows:

Appearance information: The appearance information of a vertex v, denoted by $\mu(v)$, corresponds to the following attributes:

- the average intensity $\mu(v)(intensity)$ of its respective image region. Here, we focus on color images, in which we use three such values, one for each channel in the CIELAB color space, which is appropriate for computing distances between colors. This approach is significantly simpler than the color histograms used by the maximal similarity based region merging technique described in [24], each one having $16\times16\times16=4096$ bins and corresponding to a different region in the over-segmentation;
- a label $\mu(v)(label)$ identifying the class/scribble of the vertex. For vertices in G_m, this is precisely the color of the respective scribble, whereas for vertices in V_i, the label is initially undefined, being assigned later during the matching process.

Structural information: Besides the appearance information, structure is used to minimize the ambiguities caused by regions with similar intensities belonging to both foreground and background components (or distinct classes, in the case of multi-label segmentation). The structural information is represented by the spatial relations among the centroids of the regions issued from the over-segmentation. These relations are encoded by vectors in the 2D space, as shown in Fig. 4. The basic idea is that each directed edge holds a corresponding vector as its attribute. Inspired by the work described in [10], the relative positions are evaluated by the following equation, which compares two given vectors, $\vec{v_1}$ and $\vec{v_2}$, in terms of the angle between them and their lengths:

$$c_{\mathrm{vec}}(\vec{v_1},\vec{v_2})=\lambda_2\frac{|\cos\theta-1|}{2}+(1-\lambda_2)\frac{\left\|\vec{v_1}\right|-\left|\vec{v_2}\right\|}{C_S}, \tag{1}$$

where θ is the angle between $\vec{v_1}$ and $\vec{v_2}$, $|\vec{v_1}|$ and $|\vec{v_2}|$ denote the vector modulus/lengths.

The first term in Eq. (1) represents the angular cost, which assigns higher values to opposite vectors. The second term represents the modular cost, which assigns a value proportional to the difference of the vector lengths, normalized by a constant

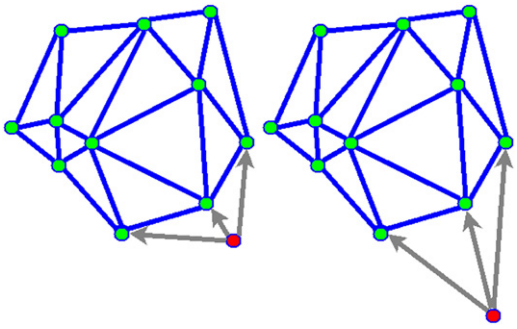

Fig. 4. For the edge attributes, we used vectors to encode spatial relations among vertices. The highlighted vertices (in red) have different coordinates, producing different vectors in terms of size and orientation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

C_S, representing the maximum distance, to keep the computed values between 0 and 1.[2] The parameter λ_2 ranges from 0 to 1.[3]

3.3. Segmentation by matching attributed relational graphs

With these two representations at hand, the goal is to find a mapping from input to model vertices to label each input image region with a scribble color. However, by construction, the model and the input graphs present distinct topologies. Also, since it is desirable that the user inputs little effort when segmenting an image, only a few scribbles are expected over the image, which causes $|V_m|$ to be much smaller than $|V_i|$.

This mapping qualifies as an inexact homomorphism between input and model graphs (Fig. 3), a problem which has been dealt within the literature of inexact graph matching [8,10,36]. One way to solve this many-to-one mapping is by optimizing a cost function which evaluates the similarities between attributes (of vertices and edges). For instance, one can use the following general form of the cost function:

$$E = \lambda_1 \sum_{vertices} d_A + (1-\lambda_1) \sum_{edges} d_S, \tag{2}$$

where the term d_A evaluates the dissimilarities between appearance attributes of pairs of vertices (v_i, v_m), the term d_S evaluates the structural dissimilarities between pairs of edges (e_i, e_m), and the parameter λ_1 ranges between 0 and 1, weighing the influence of each term over the result.

In the general case, by considering structural information to map input to model vertices, the direct comparison between input and model edge attributes can be too costly due to a combinatorial number of all possible edge interconnections. In our method, we try to overcome this problem by using deformed graphs [25].

Matching by deformed graphs: Instead of evaluating all the interdependences represented by the edges in G_i, we propose an alternative approach which evaluates individual input vertex mappings against the model graph.

In our particular problem, because an initial correspondence is available to align both graphs (regions intercepted by the scribbles), we can use it to evaluate local deformations among attributes, without compromising the existing interdependences. This simplification is achieved by successively matching deformed graphs with the model graph, and obtaining a label for each $v_i \in V_i$ (Fig. 5). Note that, differently from the tree search technique described in [10], the classification of each input vertex v_i is done independently of the already mapped input vertices v_i', $v_i' \neq v_i$.

A deformed graph (DG) [25] represents a deformation of the model graph by a vertex issued from G_i, as if its corresponding region had fallen under a scribble during user interaction. The (local) deformation occurs in G_m with respect to the attributes of a single vertex v_m and its neighboring edge attributes, caused when simulating a change in the coordinates of v_m. Given a pair (v_i, v_m), $v_i \in V_i$, $v_m \in V_m$, the induced deformed graph, denoted by $G_d(v_i, v_m)$, is computed as follows: $G_d(v_i, v_m)$ starts as a copy of the model G_m, with the same number of vertices and edges, and the same attributes for the vertices and edges, except for the copy of model vertex v_m and its adjacent edges. The centroid coordinates and intensity attributes of the copy of v_m are replaced by that of v_i, leading to the deformed vertex v_d. Similarly, a deformed edge e_d corresponds to the edges with an endpoint at v_d, as shown in Fig. 6(c) and their attributes are recomputed to take into account the new coordinates in v_d.

By ignoring the adjacency information in E_i, the matching technique based on DG always performs comparisons among similar graphs, in the sense that both graphs have the same topology, with the same number of vertices and edges, differing only on the attributes. Moreover, only the attributes from the deformed vertex and its neighboring edges in $G_d(v_i, v_m)$ may differ from the (original) model graph G_m. Therefore, when evaluating the structural dissimilarities between $G_d(v_i, v_m)$ and the model G_m, only the deformed edges (and their corresponding model edges) have to be examined. Thus, for the evaluation of this isomorphism, Eq. (2) simplifies to

$$E(v_i, v_m) = \lambda_1 d_A + (1-\lambda_1) \sum_{deformed\_edges} d_S, \tag{3}$$

where the term d_A evaluates the dissimilarities between the attributes of the deformed vertex v_d and the original model vertex v_m (Eq. (4)), the term d_S evaluates the structural dissimilarities between the deformed edges and their respective originals (Eq. (5)), and the other parameters remain the same as before. The appearance term of Eq. (3) compares vertex attributes from a pair (v_d, v_m), with v_d corresponding to an input vertex represented in G_d, as follows:

$$d_A(v_d, v_m) = \frac{\text{Euclidean distance } (\mu(v_d)(intensity), \mu(v_m)(intensity))}{C_A}, \tag{4}$$

where the Euclidean distance, between the intensity attributes of v_d and v_m, is normalized by a constant C_A, representing the maximum distance among intensity values, to keep the computed results between 0 and 1.[4] The structural term of Eq. (3) is evaluated by:

$$d_S(G_d(v_i, v_m), G_m) = \frac{1}{|E(v_d)|} \sum_{e_d \in E(v_d)} c_{\text{vec}}(v(e_d), v(e_m)), \tag{5}$$

in which $c_{\text{vec}}(.)$ is given by Eq. (1), $E(v_d)$ denotes the set of deformed edges originating from v_d, $|E(v_d)|$ denotes the cardinality, and e_m is the model edge corresponding to the deformed edge e_d. Eq. (5) is the average cost between deformed edges and their corresponding model edges.[5]

[2] In our experiments, we used $C_S = \sqrt{w^2 + h^2}$, where w is the image width and h is the image height.

[3] As in the original algorithm described in [25], we used $\lambda_2 = 0.5$ in all experiments to give the same importance to both angular and modular terms of Eq. (1).

[4] In our experiments, we used $C_A = 100$.

[5] Unless stated otherwise, we used $\lambda_1 = 0.5$ in all experiments to give the same importance to both appearance and structural terms in Eq. (3).

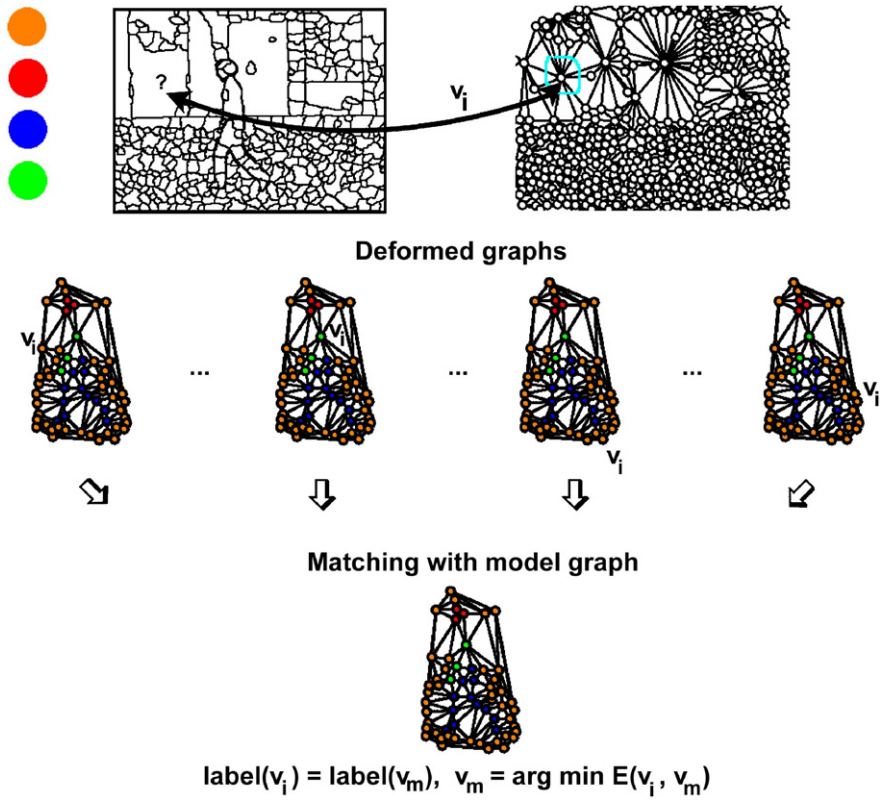

Fig. 5. Matching using deformed graphs. Because the input and model graphs can have very distinct topologies, deformed graphs are used to reduce the many-to-one mapping problem to a set of simpler inexact isomorphism problems between deformed and model graphs. The final image labeling is then given by evaluating these isomorphisms according to a greedy strategy, encoding color information and spatial configuration. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

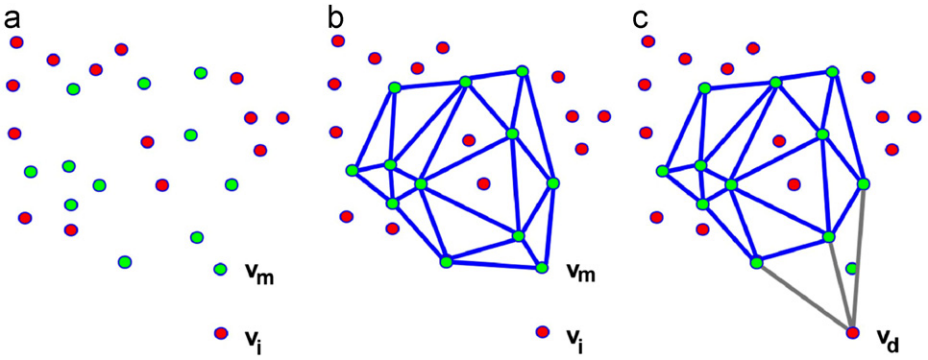

Fig. 6. (a) The model vertices (in green) are superposed to the input vertices (in red). For instance, v_m is a model vertex and v_i is an input vertex. (b) The edges denote a triangulation using the model vertices. (c) v_d is a deformed vertex, with the same coordinates as v_i, and the resulting deformed edges correspond to all (highlighted) edges with an endpoint at v_d. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.4. Optimization

Since the aim is to map all input vertices, a set of deformed graphs must be evaluated for each $v_i \in V_i$, as shown in the pseudo-code given in algorithm *MatchingByDG*.[6]

```
MatchingByDG (G_i,G_m)
1   P ← ∅
2   for each vertex v_i ∈ V_i
3       do
4           c_min ← ∞;
5           v_min ← NULL
6           for each vertex v_m ∈ V_m
7               do
8                   create G_d(v_i,v_m)
9                   c ← E(v_i,v_m)   ▷ compares G_d(v_i,v_m)
                    with G_m
10                  if c < c_min
11                      do
12                          c_min ← c;
13                          v_min ← v_m
14          μ(v_i)(label) ← μ(v_min)(label)
15          P ← P ∪ {(v_i,v_min)}
16  return P
```

[6] The proposed algorithm does not depend on the number of scribbles to map each input vertex to a model vertex. The same pseudo-code can be used to treat the binary (foreground/background) or the multi-label segmentation problem.

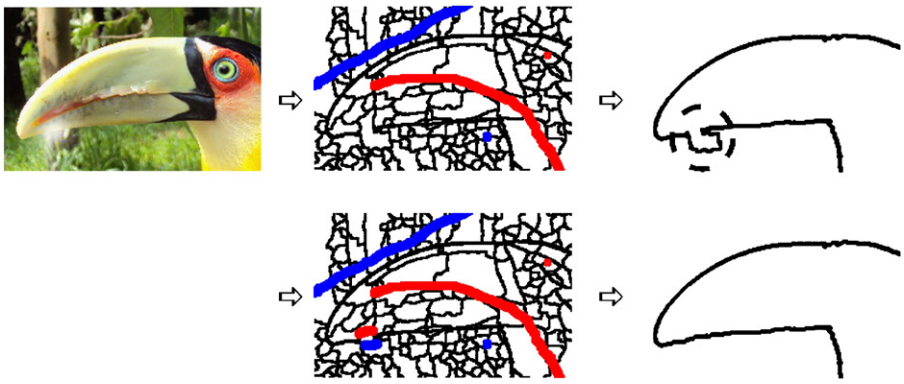

Fig. 7. Example of a coarse over-segmentation which produced an incorrect segmentation of the nose of the toucan. A correction can be achieved by manually dividing the highlighted coarse region. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Let P be a set of pairs representing a match from input to model vertices. Then the total cost of solution P is given by

$$E(P) = \sum_{(v_i,v_m) \in P} E(v_i,v_m). \quad (6)$$

In order to minimize Eq. (6), the algorithm uses a greedy strategy that relies on the individual matches between input and model vertices. The label to be assigned to each v_i should be that of the model vertex which corresponds to the DG $G_d(v_i,v_m)$ that minimizes Eq. (3), i.e., which minimizes the deformations induced in the model. Hence, we must evaluate $|V_m|$ isomorphisms between an induced DG $G_d(v_i,v_m)$ and the model graph G_m (lines 6–13).

In the end, the algorithm produces the set of pairs P and $\mu(v_i)(label) = \mu(v_m)(label)$, $\forall (v_i,v_m) \in P$, corresponding to a complete segmentation of the input image by coloring all the pixels from each input region with the respective vertex label, corresponding to a scribble color.

3.5. Computational complexity

The matching algorithm presented in the previous section has complexity in $O(|V_i||E_m|)$, which is basically determined by the number of deformed edges. Note that, for an efficient implementation, it is not necessary to rebuild $G_d(v_i,v_m)$ for each iteration. Instead, for each input vertex, the structural evaluation can be performed by traversing each (directed) model edge and each corresponding deformed edge only once. For planar model graphs, such as those generated by Delaunay triangulation, the number of traversed deformed edges is in $O(|E_m|) = O(|V_m|)$ during the classification of each input vertex, resulting in $O(|V_i||V_m|)$ edge comparisons to classify all the input vertices.

In the general tree search strategy, described in [10], an incremental solution is computed by expanding the nodes in the search tree. Each expanded node represents a pair which was included in the mapping, and its children represent all the possibilities of mapping the next input vertex. The key idea is to progressively consider more structural information, where the number of direct comparisons between input and model edges depends on the depth of the expanded node. At depth k, in order to select the cheapest model vertex for the current input vertex, $|V_m|(k-1)$ edge comparisons are performed.

The previous optimization technique [11] maintained the computational complexity in $O(|V_i||V_m|)$ by limiting the number of edge comparisons in $O(|V_m|)$ at each depth/level. During our experiments, we noticed that this modified version of the tree search [11] presented difficulties to compute the segmentations, especially when considering simple scribbles.[7] For the tested images, the original tree search technique presented a better performance when compared to the modified version [11], in which there is a total of $|V_m|(0+1+2+\cdots+|V_i|-1)$ edge comparisons, resulting in an algorithm with computational complexity in $O(|V_i|^2|V_m|)$. Therefore, in practice, the deformed graph-based algorithm decreased the computational complexity from $O(|V_i|^2|V_m|)$ to $O(|V_i||V_m|)$.

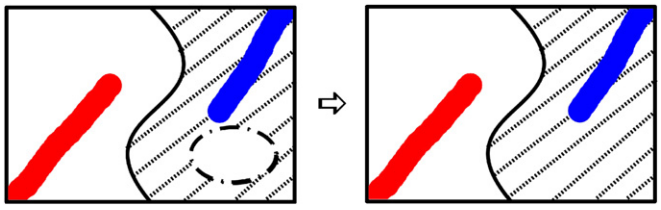

Fig. 8. (P1) Merging of regions to keep the connectivity constraint between the segmented regions and the scribbles provided by the user. In the image on the left, there is a small (solid) region, which is not connected with its corresponding (red) scribble. In the image on the right, this disconnected region is merged into the dashed region. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.6. Post-processing

For single images, the scribbles can be imposed as hard constraints, hence each scribbled pixel keeps its label in the final segmentation. We assume that the scribbles are correctly placed, i.e. each scribbled pixel corresponds to its correct region of interest. In our experiments, this constraint was useful to improve the results, especially for coarse over-segmentation, providing a way to manually divide a region (Fig. 7).

In order to provide ease of scribbling with minimal effort by the user and to compute accurate segmentation for object extraction, the proposed DG approach can include two post-processing steps:

(P1) Removal of spurious regions by keeping the connectivity constraint between the segmented regions and the scribbles.

[7] The examples presented in the previous work [11] required complex scribbles, which were placed near the object boundaries.

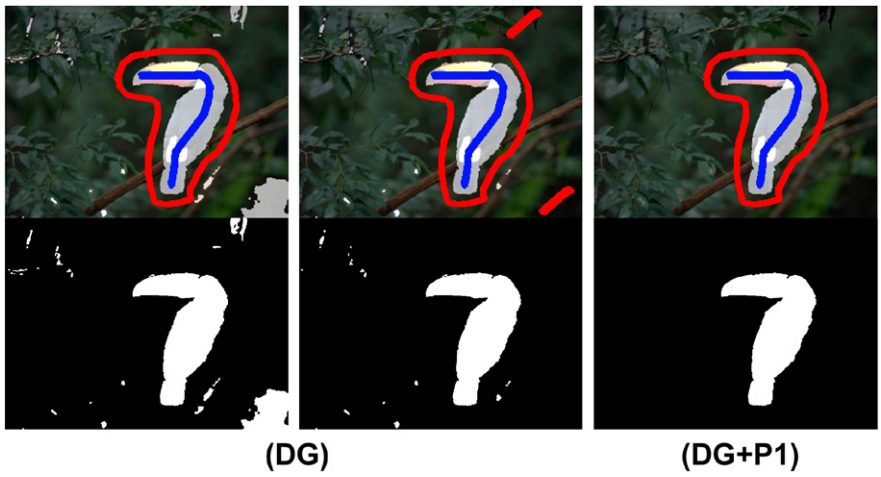

Fig. 9. Segmentation produced by the proposed DG-based approach, without and with the connectivity constraint (P1), respectively. From left to right, the two first columns represent the first case: (P1) is not imposed and the DG method requires more scribbles to remove all the spurious components from the solution (especially when considering $\lambda_1 = 0.5$).

(P2) Pixel-level refinement to improve the boundaries of the extracted objects.

In step (P1), after performing the matching algorithm (Section 3.4), the initial segmentation can be post-processed in order to guarantee that each segmented region is connected to a scribble. For instance, foreground components in the initial segmentation that are not connected to a foreground scribble are merged into the background components. Similarly, the background components that are not connected to a background scribble are merged into the foreground (Fig. 8). This simple step can significantly reduce the user effort, in the sense that less scribbles are necessary for the desired segmentation, as illustrated in Figs. 9(DG+P1) and 11(DG+P1). By considering scribbles as hard constraints, step (P1) can be used to divide coarse regions. As shown in Fig. 7, by adding scribbles, part of the (highlighted) coarse region was separated from the foreground (red) scribble to produce the correct result.

Methods combining both region merging and pixel-based approaches have been proposed in the literature (e.g. [21]), aggregating robustness (to noise and pixel variation) and accuracy (especially on the object boundaries). Following this idea, for the experiments involving binary segmentation for object extraction, we also applied the post-processing step (P2) for re-classification of pixels on a narrow band on the object boundaries, e.g. to circumvent possible imprecision in the initial over-segmentation, by performing the following heuristic. First, all pixels in a narrow band are marked as unknown pixels to indicate that they need to be re-classified (Fig. 10). In our case, the narrow band consisted of the internal and external borders of the foreground components. Then, each pixel p from the band is examined and labeled with the most similar pixel (by evaluating their colors using Eq. (4)) from all pixels at distance d from p. This step is repeated K times.[8] The improvements are illustrated in Fig. 11(DG+P1+P2).

Note that both steps, (P1) and (P2), can be executed in linear time, without affecting the computational complexity of the proposed method.

3.7. Model update

In order to achieve the desired segmentation, the user can add/remove scribbles to make corrections on the segmentation process, resulting in updates in the model graph. Fig. 12 is an example of binary segmentation, illustrating the results computed from the initial scribbles and after an additional scribble to eliminate the initial misclassified regions. After adding/removing a scribble, the model graph is rebuilt and the matching algorithm is executed using the new updated model. Then, the post-processing steps described in the previous section can be performed to improve the solution.

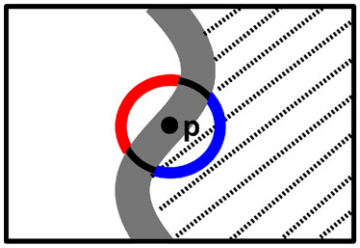

Fig. 10. (P2) Re-classification of pixels on a narrow band on the object boundaries. For each pixel p in the narrow band, p is re-classified according to the most similar pixel (in terms of color information), considering only the classified pixels that fall in a circumference with a fixed size.

4. Experiments

In order to show the benefits of the proposed approach, we tested it with different databases and real natural images, including the Berkeley Segmentation Dataset and Benchmark [22], the Microsoft GrabCut database [28], images from Ning et al. [24], Bai and Sapiro [3–5] and Levin et al. [20]. Some examples using simple scribbles are illustrated in Fig. 13.

In general, region merging approaches do not require thick scribbles to mark a large number of pixels, since regions from the initial over-segmentation can be marked by using thin scribbles. Thus, compared to pixel-based methods, more color statistics can be collected by region merging techniques when considering thin scribbles. In our implementation, we used scribbles which are 2 pixels wide to provide a clean and elegant fashion of scribbling. The results are presented with dilated scribbles for better visualization. Also, the colors of the scribbles were modified to facilitate the reading when printing in grey scale.

[8] In our case, we set $d=10$ pixels and repeated this refinement $K=10$ times.

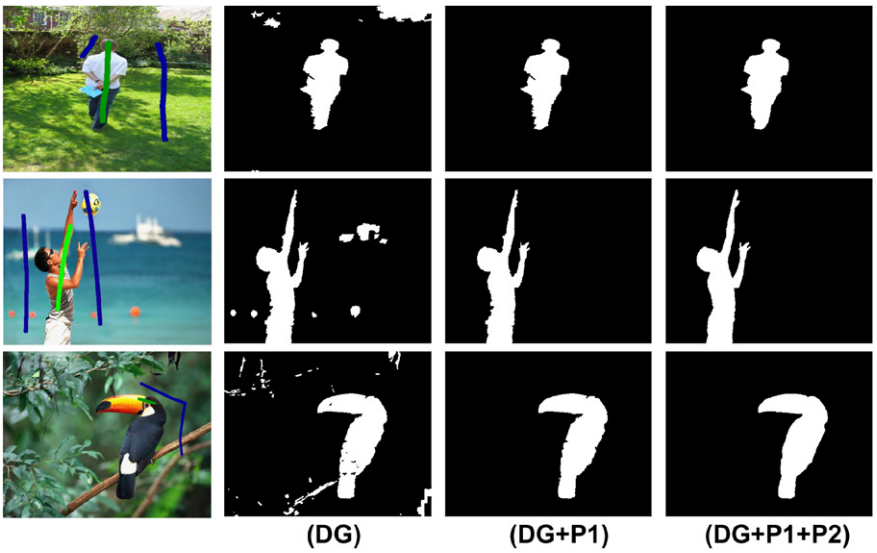

Fig. 11. Our results using images from the Grabcut database [28], Bai and Sapiro [3], and the toucan image, similar to the one used in [24], but with higher resolution (1024×768), respectively. The first column on the left shows the original images with scribbles. (DG) Computed labels by the matching algorithm described in Section 3.4. (DG+P1) Improved results after applying the post-processing step (P1) by imposing the connectivity constraint. (DG+P1+P2) Results after applying the post-processing step (P2) by improving the object boundaries.

Fig. 12. Flowers from the Berkeley database [22]. The misclassified regions (in the middle column) are corrected by adding a new scribble (in the right column). The extracted objects are highlighted.

Fig. 14 shows examples of image compositions with new backgrounds, computed by the proposed method. Fig. 15 illustrates similar segmentations using different fashions of scribbling, indicating the robustness of our method, especially when the color information is discriminative enough to separate the different components in the image.

4.1. Analysis of parameter λ_1

The parameter λ_1 weighs the influence between the appearance and the structural terms in Eq. (2). For instance, for $\lambda_1 = 0$, only structural information is used for the matching step (Section 3.4). Similarly, for $\lambda_1 = 1$, only appearance information is used to compute the segmentation. Intermediate values of λ_1 are used to combine both appearance and structural information for classification.

Additional natural images are illustrated in Fig. 16, presenting a significant variability of colors, resulting in several regions with similar colors on both foreground and background components. For these examples, the segmentation task is ambiguous, especially when using only the average colors from each region, as illustrated in Fig. 16(c). When considering only color information, these examples require more elaborated representations, such as the quantized color histograms used by the maximal similarity region merging (MSRM) technique [24]. Here, we exploit the structural information to decrease the ambiguities among similar foreground/background regions, while keeping the simplicity and the efficiency of the proposed method.

The behavior of the parameter λ_1 is illustrated in Fig. 17, in which the Jaccard index [16] was computed for different values of λ_1. As pointed out in [23], the Jaccard index is an important measure for object extraction accuracy, which can be computed by the following expression:

$$\frac{|\mathcal{G}_{\mathcal{F}} \cap \mathcal{R}_{\mathcal{F}}|}{|\mathcal{G}_{\mathcal{F}} \cup \mathcal{R}_{\mathcal{F}}|}, \tag{7}$$

where $\mathcal{G}_{\mathcal{F}}$ is the set of foreground pixels belonging to the ground-truth, while $\mathcal{R}_{\mathcal{F}}$ is the set of foreground pixels in the segmentation result. $|\mathcal{G}_{\mathcal{F}} \cap \mathcal{R}_{\mathcal{F}}|$ and $|\mathcal{G}_{\mathcal{F}} \cup \mathcal{R}_{\mathcal{F}}|$ represent the cardinalities of the intersection and the union, respectively. In Fig. 17, by testing three examples with ambiguous foreground/background regions, the best results were robust in a neighborhood around 0.5, indicating the importance of the structural information to improve the segmentations.

For a deeper evaluation, we tested the Microsoft GrabCut database [28], which is commonly used for quantitative comparisons involving different methods. This database provides the labeling-Lasso, where the unknown pixels corresponds to a roughly symmetric narrow band along the boundaries of the foreground object, as shown in Fig. 18 (middle column). For this particular type of seeds, algorithms such as those based on adaptive thresholding [14,18] can exploit the same effect of the skeleton of the unknown regions to improve the segmentations. In order to avoid any bias, instead of using the labeling-Lasso, Couprie et al. [12] proposed the use of asymmetrically eroded seeds, such as the one shown in Fig. 18 (right column).

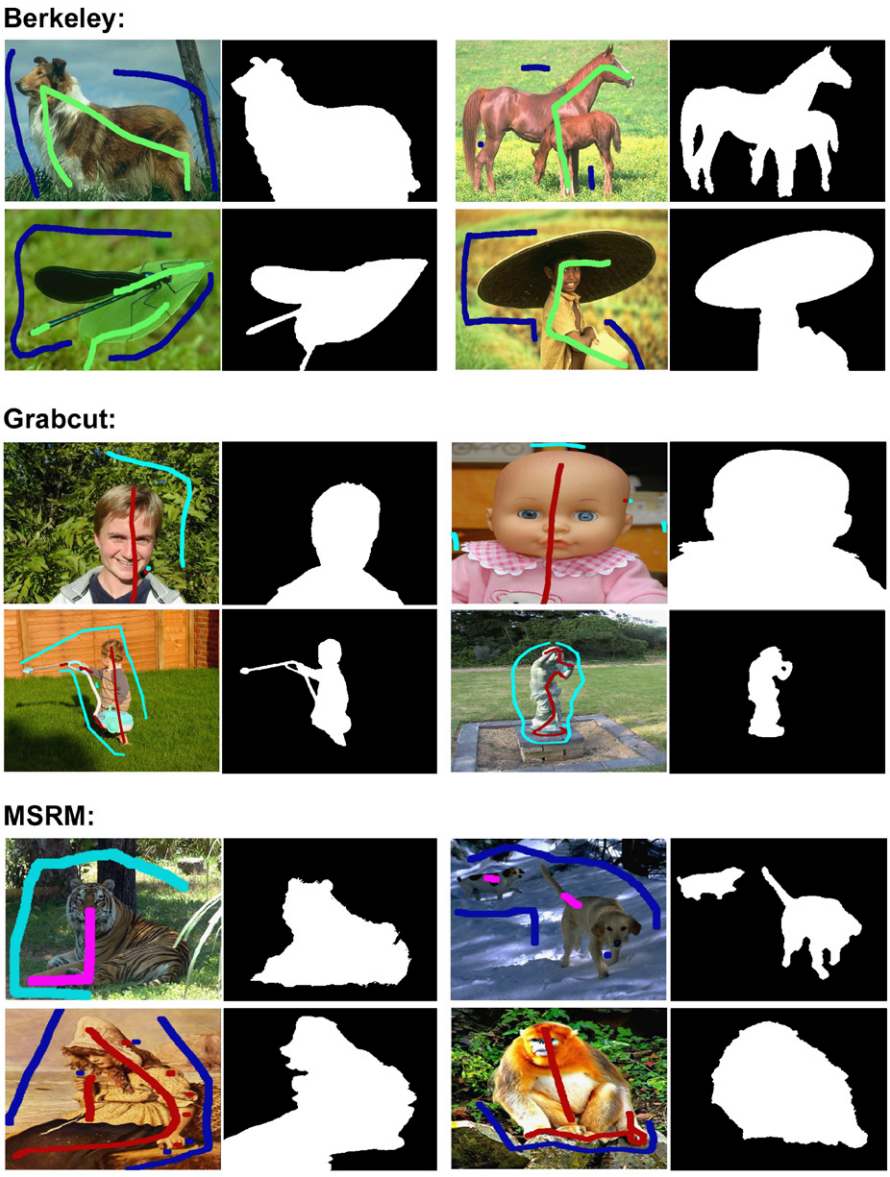

Fig. 13. Our results using images from Berkeley Image Segmentation and Benchmark [22], GrabCut [28], and MSRM database [24]. For each example, the image on the left shows the original image with scribbles, and the image on the right shows the computed labels by DG+P1+P2.

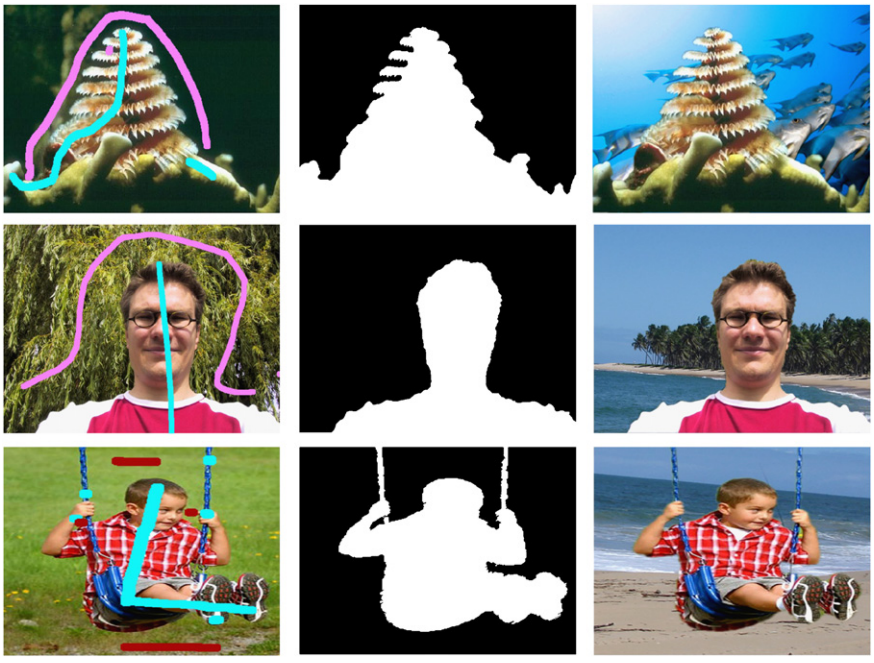

Fig. 14. Examples of background replacement using images from different databases: Berkeley [22], GrabCut [28], and Bai and Sapiro [3,5], respectively. Left column: original images with scribbles. Middle column: labels computed by the proposed approach (DG+P1+P2). Right column: compositions using new backgrounds.

Fig. 15. Flowers from the Berkeley database [22] and three different ways of placing the scribbles. The respective segmented regions (outlined in white) were computed by DG+P1+P2.

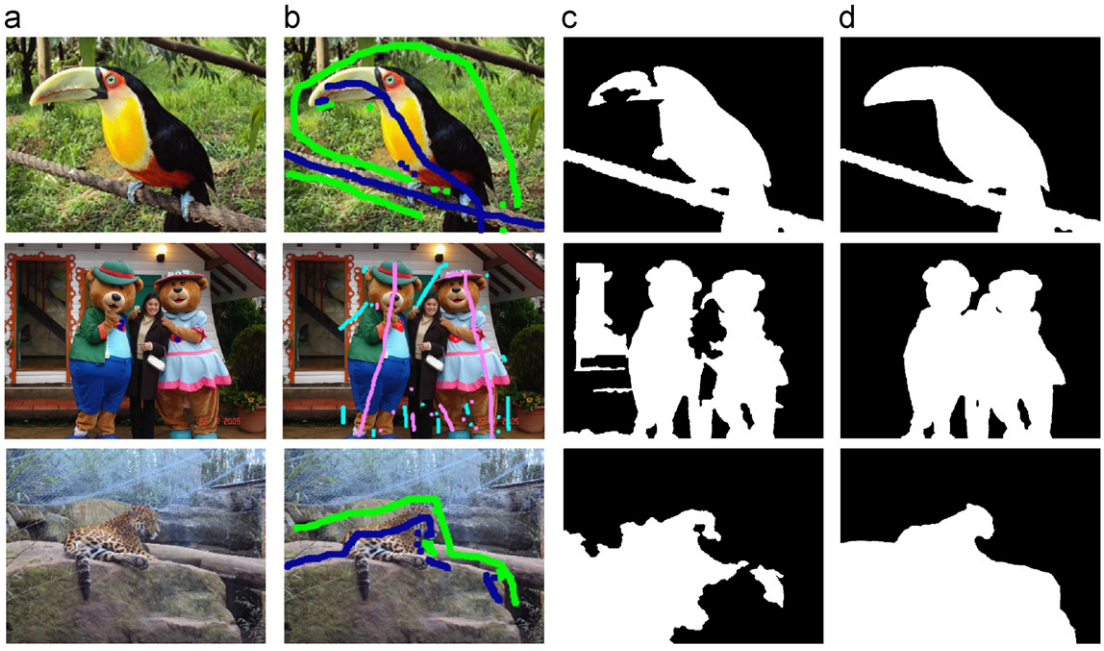

Fig. 16. Additional segmentation examples. (a) Input images: bird, bears and jaguar. (b) Scribbles over the input images. (c) Computed labels using only the appearance information ($\lambda_1 = 1.0$). (d) Computed labels using both the appearance and the structural information ($\lambda_1 = 0.5$). The above results illustrate the importance of the structural component in Eq. (2). The segmentations were computed by DG+P1+P2.

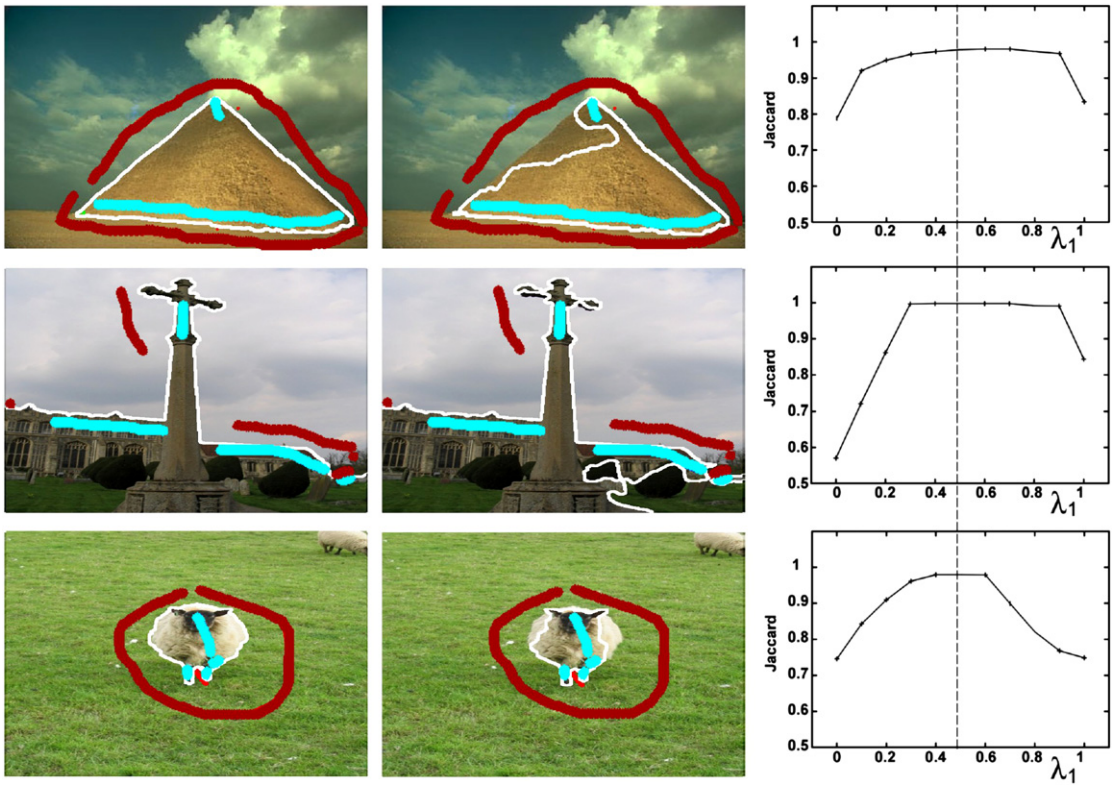

Fig. 17. Pyramid from the Berkeley database [22]; cross and sheep from the Grabcut database [28]. For each example, we tested the same scribbles for different values of λ_1, by using DG+P1+P2. The segmentations are outlined in white. (a) Segmentations by using both the appearance and the structural information ($\lambda_1 = 0.5$). (b) Using only the appearance information ($\lambda_1 = 1.0$). (c) Jaccard indices for different values of λ_1.

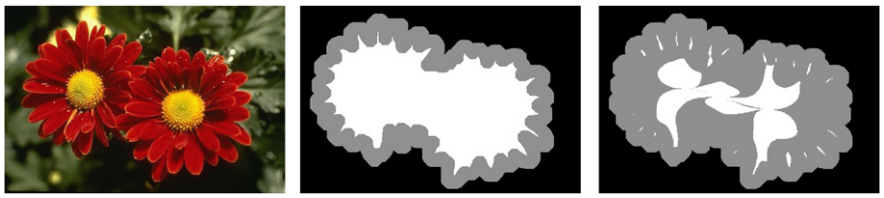

Fig. 18. Left column: flowers from the Berkeley database [22]. Middle column: respective labeling-Lasso provided by the GrabCut database [28]; the unknown pixels correspond to a symmetric narrow band (in grey) along the boundaries of the flowers. Right column: the corresponding asymmetrically eroded seeds from Couprie et al. [12].

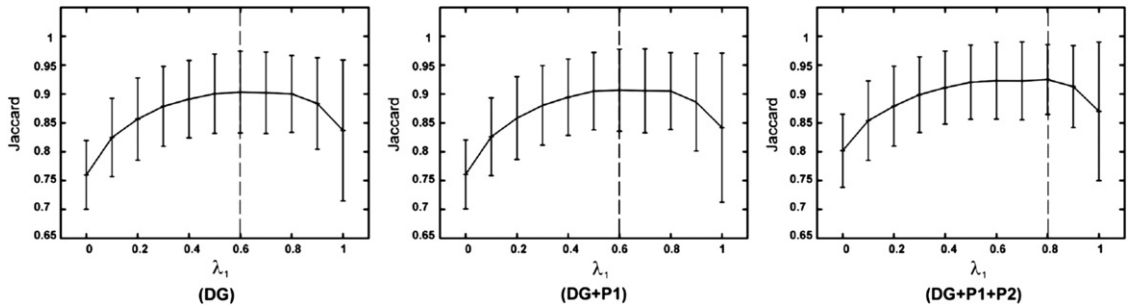

Fig. 19. The average Jaccard indices and the respective standard deviations for the fifty images from the GrabCut database [28], using the eroded seeds provided by Couprie et al. [12].

Fig. 19 shows the average Jaccard indices, for $\lambda_1 = 0.0, 0.1, \ldots, 1.0$, using the asymmetrically eroded seeds provided by Couprie et al. [12], and considering the three cases: (DG) the matching algorithm (Section 3.4) without any post-processing step; (DG+P1) combined only with post-processing step (P1); and (DG+P1+P2) combined with both (P1) and (P2). The largest averages were: (DG) 0.9033 for $\lambda_1 = 0.6$; (DG+P1) 0.9066 for $\lambda_1 = 0.6$; and (DG+P1+P2) 0.9250 for $\lambda_1 = 0.8$. Nevertheless, all three cases resulted in good average object accuracy when using $\lambda_1 = 0.5$ (0.9005, 0.9049 and 0.9204, respectively). Thus, for the comparison with the current state-of-the-art techniques, we test our approach with the fixed parameter $\lambda_1 = 0.5$.

4.2. Multi-label segmentation and image matting

The proposed formulation can also be applied to the multi-label segmentation problem in a straightforward manner. As shown in Fig. 20, the proposed approach has a good performance, even in the presence of multiple objects with similar colors, since the initial over-segmentation included the important boundaries in these examples.

In order to extract objects with complex boundaries, such as hair strands and animal fur, image matting [20,35] is suitable to reconstruct the foreground/background components and the alpha value (transparency) of each pixel. In the image matting problem, the interaction can be performed by using trimaps which divide the image into three regions: foreground, background and unknown regions. Our approach can also be used to automatically generate such trimaps, in which a narrow band is automatically spanned across the current foreground/background boundaries to define the unknown regions. Following the same idea of the fixed width band described in [5], Fig. 21 shows examples in which our approach can minimize the effort by avoiding the need to track all the object boundaries when building the complete trimaps manually.

As observed in [20], for complex cases, especially when the objects have too many holes, image matting can also be performed by using scribbles, which can require a significant effort, as suggested by the examples presented in [20]. For the cases in which the trimaps are preferable, such as Fig. 22, our approach can simplify the scribbles used for image matting.

In all matting experiments, we used the Robust Matting [35] binaries provided by the authors, which produced accurate matting results.

4.3. Comparison with current state-of-the-art

For a quantitative comparison with other methods, we used the 50 images from the Microsoft GrabCut database [28]. We considered different techniques, including pixel-based and region-based methods, for a good coverage of the literature. More specifically, we used the source codes provided by the authors of the power watershed (PW) [12], the maximal similarity based region merging technique (MSRM) [24], and random walker (RW) [17]. For the interactive graph cuts (IGC) [7], seeded region growing (SRG) [2], simple interactive object extraction (SIOX) [15] and binary partition tree (BPT) [1,29], we used the segmentation tool due to McGuinness and Connor [23], combined with a context creator utility provided by the authors of [23] to load the seeds from pre-existing image files. All these implementations are freely available, corresponding to recent important works [12,17,23,24].

In order to better exploit the capabilities of the well-known IGC method [7], we tested it with fixed parameter $\sigma = 3.5$,[9] and with the best σ for each individual image (considering $\sigma = 0.5, 1.0, 3.5, 7.0, 10.0, 15.0$ in the segmentation tool provided by McGuinness and Connor [23]), corresponding to the largest object accuracy (Jaccard index). The parameter σ is used in the boundary penalty function described in the IGC paper [7], corresponding to the distribution of noise among neighboring pixels. For the remaining competing approaches, we did not set any parameter.

Table 1 shows the results for the symmetrically and asymmetrically eroded seeds: the Lasso form [28] and the ones provided

[9] When considering IGC [7] with fixed parameter in the segmentation tool provided by [23], $\sigma = 3.5$ produced the largest average Jaccard indices in our experiments with the GrabCut database [28].

Fig. 20. Examples from the Berkeley database [22] for multi-label segmentation, including multiple objects with similar colors. The results were computed by DG+P1, using only the connectivity constraint (P1). Left column: original images. Middle column: scribbles and segmented regions outlined in black. Right column: computed labels. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

by Couprie et al. [12], respectively. When compared to the Lasso form, the asymmetrically eroded seeds from Couprie et al. [12] correspond to smaller markers for the foreground, as shown previously in Fig. 18.

The quantitative results produced by the tree search algorithm [10], considering complete (TSC) and triangulated (TST) model graphs, were also included in Table 1. For both TSC and TST, we considered a fixed parameter α (corresponding to the largest average Jaccard index) and the best α (corresponding to the largest Jaccard index for each image) for Eq. (1) of the previous work described in [11].[10]

In Table 1, our approach DG+P1+P2 with $\lambda_1 = 0.5$ (Eq. (2)) produced the largest average Jaccard index among all the considered methods for both types of seeds. Moreover, for the smallest set of seeds [12], the performance of DG (using triangulated model graphs, without any post-processing step) was comparable to TSC (using complete model graphs), but within a lower complexity time, indicating the robustness of the proposed method according to seed quantity. Differently from the DG technique, the TST method presented a poor performance (comparable to SRG), indicating insufficient structural information by the triangulated model graphs when considering the tree search technique [10] and 'small' seeds. For TST, the largest average Jaccard index was achieved when using $\alpha = 0.1$, indicating that increasing the influence of the structural term was not enough to produce good results. Moreover, by choosing the best α, the

[10] The parameter α in Eq. (1) of [11] has the same role as our parameter λ_1, where lower values for α correspond to larger weights to the structural term.

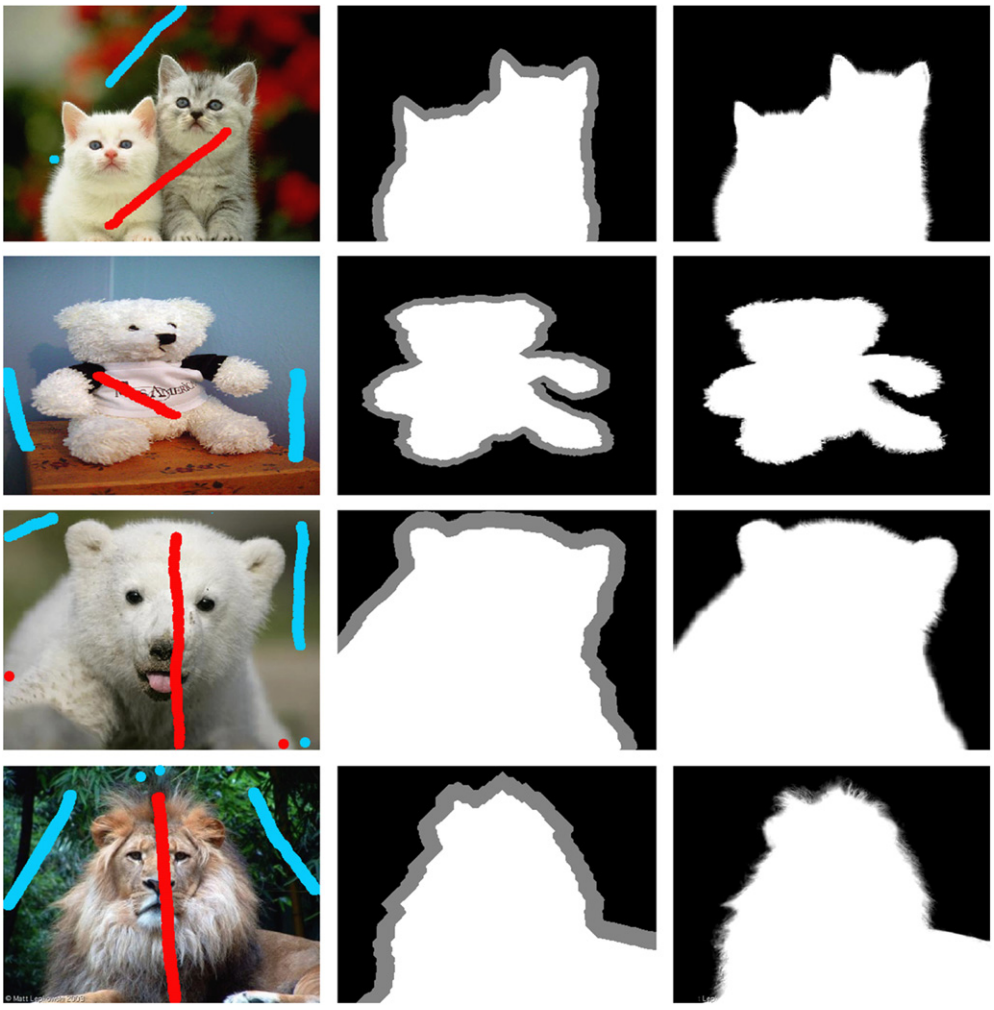

Fig. 21. Input images from Bai and Sapiro [4,5]. Left column: original images with scribbles. Middle column: automatically computed trimaps by DG+P1+P2. Right column: alpha mattes computed by Robust Matting [35] using the trimaps from the middle column.

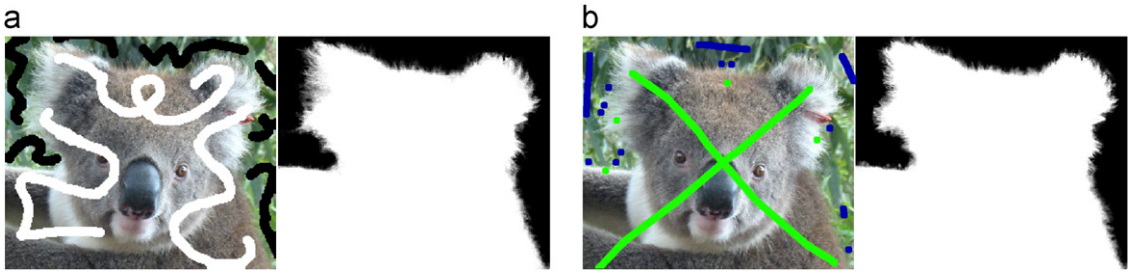

Fig. 22. (a) Original image with scribbles and the corresponding alpha matte, both images were extracted from the paper due to Levin et al. [20]. (b) Original image with scribbles and the respective alpha matte computed by the Robust Matting [35] using the automatically computed trimap by DG+P1+P2.

stability of TST was not improved, which presented large standard deviations for the object accuracy.

In the ground-truth provided by the Grabcut database [28], grey level 255 corresponds to the foreground objects, 0 to the background components, and 128 to the mixed foreground-background pixels. In all experiments, we discarded the mixed pixels from the (binary) Jaccard index computations for the object accuracy analysis, since this binary index considers only full foreground/background pixels. Moreover, based on the binary Jaccard index used for object accuracy, the authors of [23] proposed the fuzzy boundary accuracy to evaluate the segmentation results. One way to use it on the Grabcut database is to interpret the mixed/unknown pixels in the ground-truth as part of the boundary, which may increase the degree of 'uncertainty' in the boundary. Following this idea, we also present the respective fuzzy boundary accuracies in Table 1, in which we considered (besides the internal boundaries of the ground truth) the mixed regions for the corresponding fuzzification.[11] Note that, by considering this boundary accuracy

[11] As described in [23], we used $\sigma = 4$ for the fuzzification.

Table 1
Quantitative comparison using the Microsoft GrabCut database [28], with the symmetrically (the Lasso form [28]) and the asymmetrically eroded seeds (provided by Couprie et al. [12]). For the object and boundary accuracy measures, we used the Jaccard index [16] and the fuzzy boundary accuracy [23], respectively.

Method	Object accuracy (Jaccard)		(Fuzzy) Boundary accuracy	
	Average	Standard deviation	Average	Standard deviation
Symmetric seeds (Lasso)				
SIOX [15]	0.8852	0.1718	0.6338	0.1935
SRG [2]	0.8881	0.0860	0.5409	0.1797
MSRM [24]	0.9017	0.0856	0.6082	0.1467
IGC [7], $\sigma=3.5$	0.9275	0.0651	0.6376	0.1454
RW [17]	0.9324	0.0659	0.6547	0.1432
PW [12]	0.9341	0.0636	0.6653	0.1384
IGC [7],best σ	0.9366	0.0621	0.6661	0.1452
BPT [1,29]	0.9533	0.0531	**0.7266**	0.1230
DG, $\lambda_1=0.5$	0.9102	0.0835	0.6366	0.1264
DG+P1, $\lambda_1=0.5$	0.9429	0.0508	0.6898	0.1075
DG+P1+P2, $\lambda_1=0.5$	**0.9536**	0.0501	0.7239	0.1137
TSC, $\alpha=0.5$	0.9444	0.0471	0.6999	0.1012
TSC, best α	0.9525	0.0430	0.7305	0.0781
TST, $\alpha=0.1$	0.9415	0.0440	0.6888	0.0949
TST, best α	0.9438	0.0429	0.6975	0.0966
Asymmetric seeds (Couprie et al. [12])				
SRG [2]	0.8378	0.1272	0.4948	0.2010
RW [17]	0.8752	0.0986	0.5471	0.1936
IGC [7], $\sigma=3.5$	0.8784	0.0946	0.5613	0.1834
SIOX [15]	0.8830	0.1232	0.5863	0.1956
PW [12]	0.8891	0.0881	0.5842	0.1705
IGC [7], best σ	0.8955	0.0889	0.5948	0.1831
MSRM [24]	0.9017	0.0856	0.6082	0.1467
BPT [1,29]	0.9188	0.0688	**0.6368**	0.1717
DG, $\lambda_1=0.5$	0.9005	0.0687	0.5747	0.1422
DG+P1, $\lambda_1=0.5$	0.9049	0.0672	0.5846	0.1371
DG+P1+P2, $\lambda_1=0.5$	**0.9204**	0.0644	0.6255	0.1494
TSC, $\alpha=0.5$	0.9080	0.0635	0.5927	0.1274
TSC, best α	0.9195	0.0593	0.6142	0.1318
TST, $\alpha=0.1$	0.8485	0.1040	0.4879	0.1516
TST, best α	0.8546	0.1056	0.4985	0.1526

measure, the BPT technique [1,29] presented the highest fuzzy boundary accuracy. Table 2 presents additional measures, which are the same indices used in [12], in which our method presented the best performance for the smallest set of seeds from Couprie et al. [12].

For the tree search algorithms (TST and TSC), the input edges were created between adjacent regions in the over-segmentation. For all the experiments involving the MSRM approach, we used watershed for the initial segmentation for a fair comparison, which significantly increased its computational time (see Table 5 in next section).

The region-merging techniques, MSRM [24], BPT [1,29] and DG+P1+P2 with $\lambda_1=0.5$, produced the largest average accuracy indices for the eroded seeds from Couprie et al. [12]. By using 'small' sets of seeds, besides the robustness to noise and color variability, a larger number of pixels can be marked by the region merging methods, when compared to the pixel-based approaches, providing more information for the classification.

Fig. 23 illustrates a qualitative comparison, in which SRG and SIOX failed for all the three additional examples. Although RW produced a good result for the jaguar example, BPT, TSC+P1 and DG+P1 provided more consistent results for the other two examples with the given scribbles, by producing less leaking effects and less missing regions. Excluding SRG and SIOX, the remaining competing methods basically required more scribbles to achieve good results, such as those shown in Fig. 16(d), which were produced by DG+P1+P2 with $\lambda_1=0.5$.

Table 2
Quantitative comparison using the Microsoft GrabCut database [28], testing both symmetrically and asymmetrically eroded seeds, with the same accuracy indices used in [12]: Rand Index (RI), Global Consistency Error (GCE), Variation of Information (VoI), and Boundary Error (BE). Good segmentations correspond to high RI, low GCE, low VoI and low BE.

	RI	GCE	VoI	BE
Symmetric seeds (Lasso)				
SIOX [15]	0.9451	0.0378	0.2928	5.1512
SRG [2]	0.9523	0.0404	0.3022	4.0483
IGC [7], $\sigma=3.5$	0.9680	0.0265	0.2235	3.3025
RW [17]	0.9704	0.0241	0.2090	3.1617
PW [12]	0.9706	0.0247	0.2102	2.8888
MSRM [24]	0.9719	0.0235	0.2099	2.4678
BPT [1,29]	0.9781	0.0176	0.1695	**1.7780**
DG+P1+P2, $\lambda_1=0.5$	**0.9784**	**0.0175**	**0.1692**	1.9124
Asymmetric seeds (Couprie et al. [12])				
SRG [2]	0.9342	0.0527	0.3692	5.9340
RW [17]	0.9457	0.0430	0.3113	5.8707
IGC [7], $\sigma=3.5$	0.9483	0.0420	0.3059	5.3369
PW [12]	0.9518	0.0400	0.2930	4.8480
SIOX [15]	0.9543	0.0369	0.2852	4.7063
MSRM [24]	0.9594	0.0338	0.2666	3.6166
BPT [1,29]	0.9634	0.0299	0.2395	3.2795
DG+P1+P2, $\lambda_1=0.5$	**0.9653**	**0.0282**	**0.2331**	**3.1799**

4.4. Running time

The experiments were carried out in a computer with an Intel Core i3 2.13 GHz processor and 4 GB of RAM. Table 3 illustrates the running times for object extraction. For each entry, we present the image dimensions (width and height), the number of input and model vertices ($|V_i|$ and $|V_m|$, respectively), and the computational times for: building the input and model graphs (G_i and G_m, respectively), the graph matching algorithm (Section 3.4) and the post-processing steps (Section 3.6). Note that the largest computational times are required for building G_i, mainly due to a pre-processing step which was performed to reduce the number of watershed regions, by merging small[12] regions with its most similar adjacent region (in terms of appearance/color). This may slow down the launching of the interactive segmentation program, but it was very important to reduce the computational times for the proposed approach, which considerably reduced the sizes of the input and model graphs, as shown in Table 4, without affecting the interactivity with the user during the segmentation process. In Table 3, although our code was implemented in Java, which is often slower than a C++ implementation, note that the model graph update and the graph matching algorithm presented fast performances, justifying the fact that our method can be used for the interactive image segmentation problem.

Table 5 compares the running times of different approaches. SRG [2], IGC [7] and PW [12] were the fastest implementations (in C++), while the tree search methods [11] TST and TSC were the slowest ones. Note that only the BPT running time included the pre-processing time, required to compute the binary trees. In our experiments, the user interaction of BPT was very fast, being comparable to the other C++ implementations like IGC [7] and

[12] In our implementation, we merged small regions with area <25 pixels.

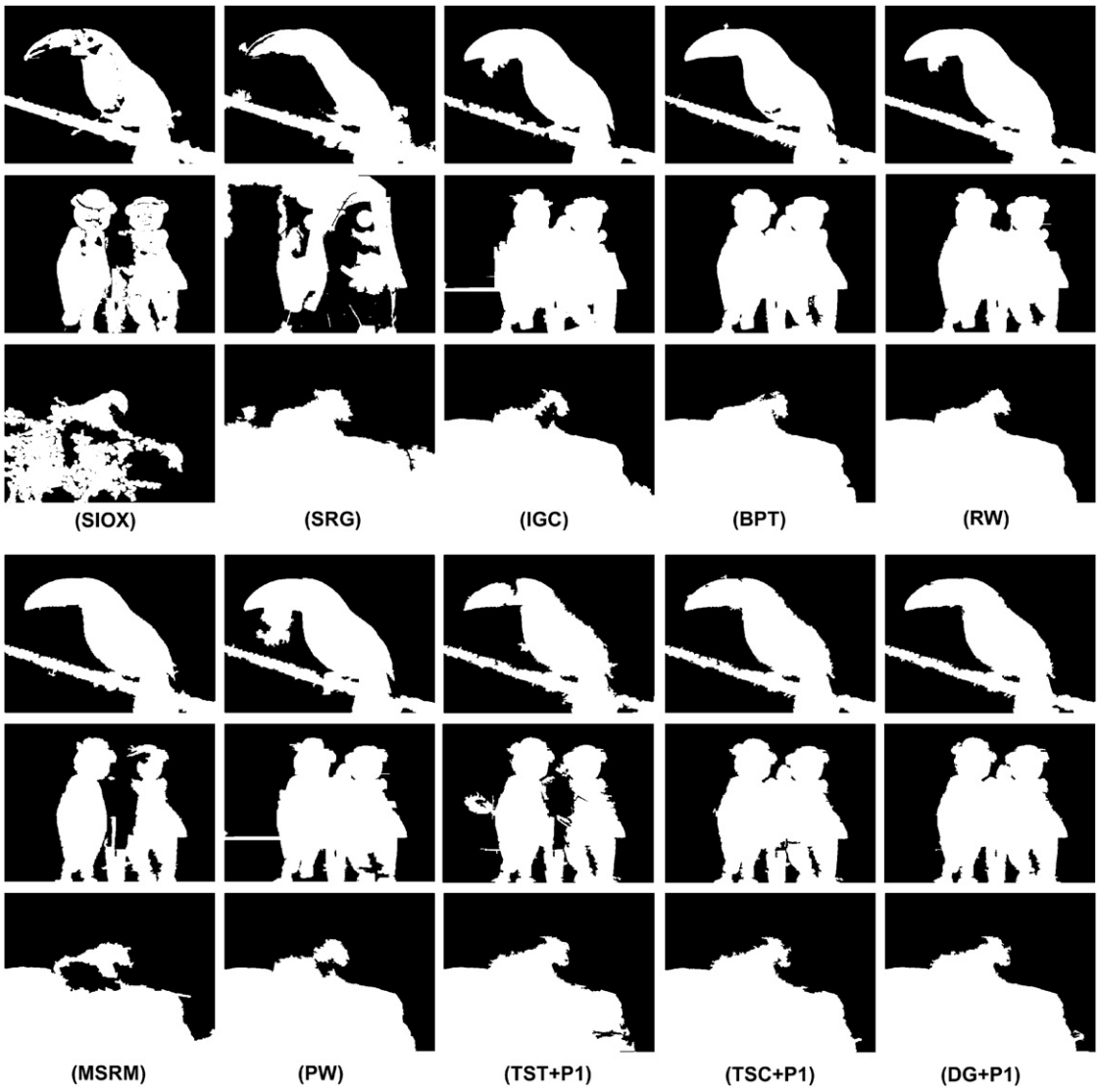

Fig. 23. Qualitative comparison using the additional images and the respective scribbles from Fig. 16. The matching techniques TST, TSC and DG were combined only with the post-processing step (P1).

PW [12]. DG, TS [11] and MSRM [24] did not include the over-segmentation computation time.

4.5. Segmentation of multiple similar images

The proposed approach can also segment multiple similar images. In this case, an important question arises: "Would it be possible to take advantage of the same user scribbles applied to one specific image in order to segment another similar image?".[13] Specifically, given two similar images A and B (e.g. corresponding to different frames from a video sequence), and seeds marking the regions of interest in A, the goal is to segment B by using the same prior information represented by the scribbles drawn over A. In general, direct use of scribbles from A on B does not result in good segmentation due to possible deformations between A and B, especially when dealing with articulated objects (Fig. 24). Thus, a flexible representation for the scribbles is necessary in order to produce a successful segmentation for B.

By using attributed relational graphs as described in Section 3, the proposed approach represents scribbles from A and image B as a model graph and an input graph, respectively (Fig. 25). Then, for instance, by performing the algorithm *MatchingByDG* (Section 3.4) to obtain a solution for the graph matching problem, encouraging segmentation results were achieved for roughly aligned graphs.

In Fig. 26, we considered five different frames (63–67) from the video sequence of a person walking straight from [30], assuming frame 65 as the model graph to segment all five given frames. In this example, the results from our approach outperformed the ones achieved by directly applying the scribbles with usual methods to segment each frame. For the third row in Fig. 26, we tested DG+P1+P2, $\lambda_1 = 0.5$ and PW [12] for the usual single image segmentation by directly applying the scribbles on each frame. Both methods produced similar leaking effects, especially on the legs.

Fig. 27 illustrates our results on the Tsukuba images, which are very popular for stereo benchmarking. Differently from most

[13] Note that, differently from [13], which relied on manual segmentations, our goal is to take advantage of the scribbles to propagate the segmentation to other similar images.

Table 3
Running times for initial segmentation computed by graph matching and for post-processing using Figs. 13–16, 20–22.

Image	Size	$\|V_i\|$	$\|V_m\|$	Computational times (in ms)				
				G_i	G_m	Matching	P1+P2	Total
Binary segmentation								
berk-anemone	480 × 320	1752	170	3588	78	1863	2599	8128
berk-flowers	480 × 320	1780	30	3341	42	365	2604	6352
berk-dog	480 × 320	1431	185	3768	72	1638	2071	7549
berk-horses	480 × 320	1015	63	3835	46	485	3503	7869
berk-insect	480 × 320	2144	244	3541	87	3135	2463	9226
berk-boy	480 × 320	1976	186	3599	77	2200	2175	8051
grab-person1	600 × 450	1963	181	6585	80	2208	2851	11724
grab-person2	600 × 450	2350	117	6355	139	1663	3702	11859
grab-doll	462 × 549	2836	128	5612	129	2175	3234	11150
grab-child	1024 × 768	6214	314	19 340	252	11 032	8224	38 848
grab-statue	768 × 1024	5175	315	20 322	236	9892	13 412	43 862
msrm-tiger	264 × 192	314	72	1203	69	211	1709	3192
msrm-dogs	335 × 295	1135	102	2092	79	757	1936	4864
msrm-girl	303 × 397	982	196	2922	107	1208	2053	6290
msrm-monkey	360 × 414	1136	145	3317	105	1013	3724	8159
add-bird	640 × 480	2659	389	7471	142	5956	6526	20 095
add-bears	800 × 600	5557	386	10 810	186	11 823	9838	32 657
add-jaguar	640 × 480	2019	182	7583	83	2300	7394	17 360
bai-boy	255 × 308	743	78	1840	31	421	2075	4367
bai-cat3	535 × 412	2465	67	4384	156	967	3026	8533
bai-toy	338 × 450	1344	73	3650	62	671	2325	6708
bai-bear	450 × 344	1863	107	3573	47	1263	2278	7161
bai-lion	400 × 300	1013	92	2636	47	593	2090	5366
levin-teddy	486 × 416	1716	191	4527	109	2028	4508	11 172
Multi-label segmentation								
berk-anemone	480 × 320	1752	249	3597	193	2606	2988	9384
berk-astronauts	480 × 320	1256	269	3410	196	1988	3607	9201
berk-horses	480 × 320	1015	106	3835	161	697	3682	8375
berk-flowers	480 × 320	1780	63	3355	148	581	4113	8197
berk-boy	480 × 320	1976	241	3589	193	2755	3222	9759
berk-bears	480 × 320	1109	161	3804	171	1116	3306	8397

Table 4
Number of original watershed regions and the corresponding reduced amount after the pre-processing step.

Image	Number of watershed regions	
	Original	Pre-proc
berk-dog	14,420	1431
berk-horse	17,471	1015
berk-insect	9620	2144
berk-boy	11,205	1976
berk-flowers	11,324	1780
grab-person1	27,796	1963
grab-person2	27,705	2350
grab-doll	15,446	2836
grab-child	84,025	6214
grab-statue	90,294	5175
msrm-tiger	6052	314
msrm-dogs	7603	1135
msrm-girl	12,296	982
msrm-monkey	16,258	1136
add-bird	29,117	2659
add-bears	34,591	5557
add-jaguar	34,603	2019
bai-cat3	14,580	2465
bai-toy	14,767	1344
bai-bear	9938	1863
bai-lion	10,993	1013
levin-teddy	20,396	1716

Table 5
Average running times by using the grabcut database [28] with the asymmetric seeds by Couprie et al. [12].

Method	Mean time (s)
IGC [7], $\sigma = 3.5$	0.4259
SRG [2]	0.4593
PW [12]	1.2759
SIOX [15]	1.4072
RW [17]	1.8866
BPT [1,29]	2.9126
DG, $\lambda_1 = 0.5$	11.4762
DG+P1, $\lambda_1 = 0.5$	12.7314
DG+P1+P2, $\lambda_1 = 0.5$	16.5837
MSRM [24]	153.7778
TST [11], $\alpha = 0.1$	187.2444
TSC [11], $\alpha = 0.5$	386.1613

of the usual interactive segmentation approaches, by using a model graph, we produced encouraging results by avoiding most of the segmentation errors due to incorrect placement of the seeds. More challenging results are shown in Figs. 28 and 29, illustrating large object displacements, camera position and color variability. Despite the differences among the images, the model created for the first one is successfully used to segment the others.

5. Conclusions

In this paper, we have proposed an interactive algorithm for image segmentation, which produced accurate results for a wide range of natural images. The core of the technique corresponds to

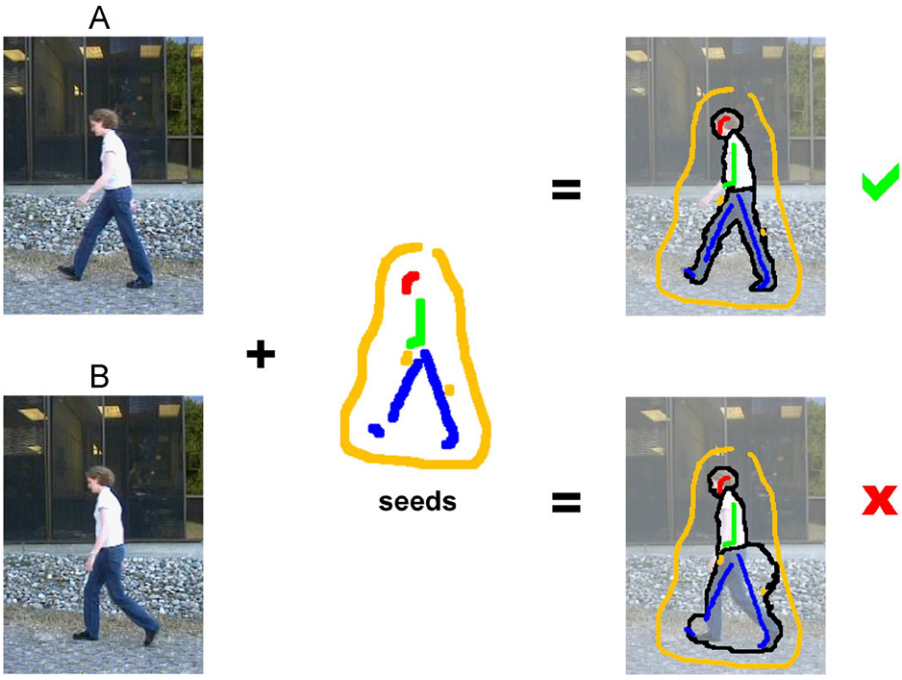

Fig. 24. Motivation: "Could we use the scribbles drawn over *A* to segment a similar image *B*?". In general, given two similar images *A* and *B* and scribbles marking the regions of interest in *A*, their direct application to the segmentation of *B* does not produce good results. This is because markers may fall over wrong regions in *B*. Results shown were both produced by our DG+P1+P2 algorithm for segmenting single images.

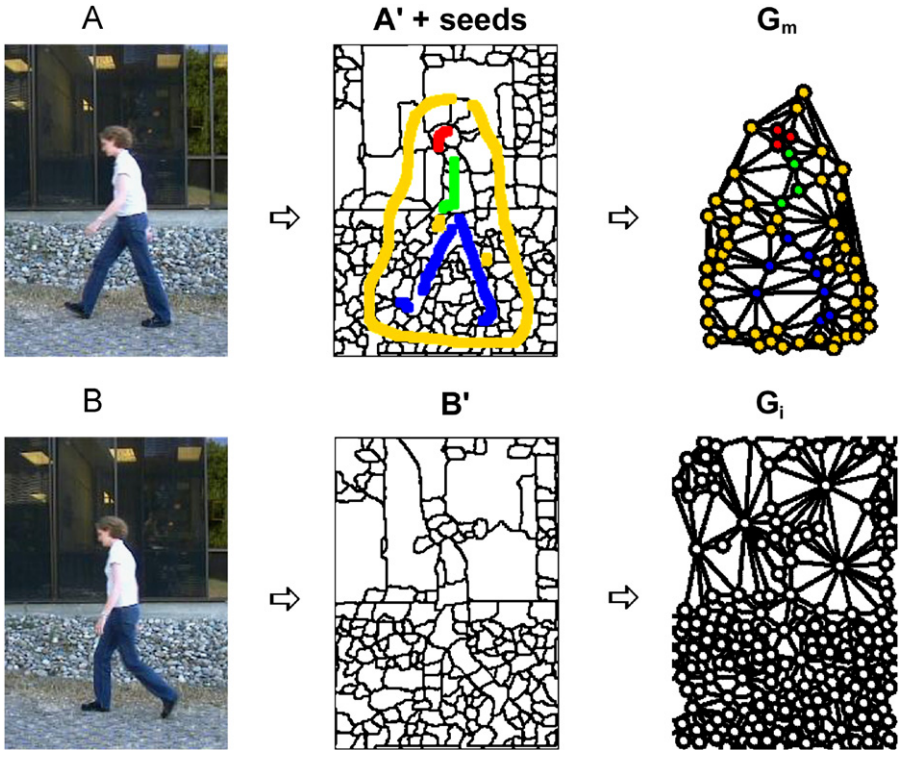

Fig. 25. Model and input graphs representing similar images, *A* and *B*, respectively. The corresponding over-segmentations are A' and B'. The scribbles are placed over A', indicating the regions of interest, resulting in the model graph G_m. The input graph G_i represents all the regions in B'.

a framework to match attributed relational graphs by exploiting the spatial relations among vertices.

For the optimization, we exploited deformed graphs [25], which can be applied to multi-label segmentation (representing multiple objects or multiple object parts) without any change in the core algorithm, in contrast to the GC-based algorithms (e.g. [7,28]), which are not trivially extensible from binary to multi-label segmentation.

In our experiments, besides quantitative analysis involving current state-of-the-art techniques, we have also illu-

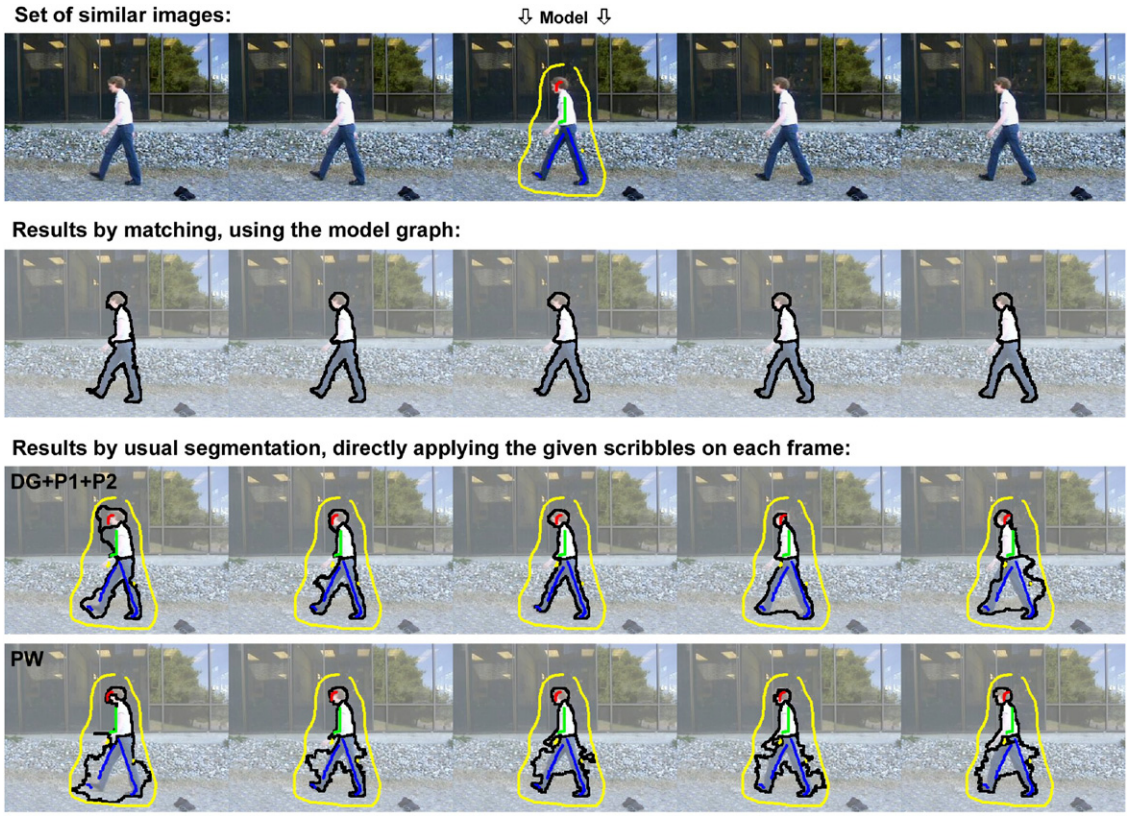

Fig. 26. First row: input images from the video sequence of a person walking straight [30], frames 63–67. Second row: segmentations by matching the corresponding attributed relational graphs by using frame 65 and its scribbles as the model graph. Third row: segmentations by using the usual interactive image segmentation and the same scribbles specifically designed for frame 65.

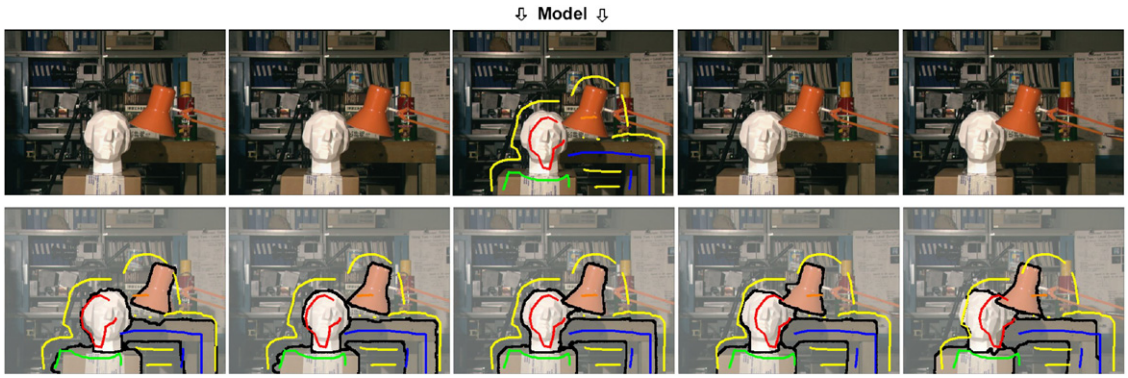

Fig. 27. First row: images provided by Dr. Y. Ohta and Dr. Y. Nakamura from University of Tsukuba. A single set of scribbles was specifically designed for the middle image, chosen as the model. No scribbles were defined on the other images of the sequence. Second row: for each image, the segmentation was obtained by matching the input attributed relational graph, built from this image, and the model graph. The single set of scribbles was placed over each segmentation result just to emphasize the object displacements.

strated the importance of the structural information by showing how λ_1 affects the performance when segmenting single images. The parameter λ_1 was used to balance the influence between color and spatial information on the segmentation result.

Moreover, we explored the re-usability of model graphs to segment multiple similar images, which can be applied to, for example, multi-label video segmentation based on models from key frames.

As a future step, further studies involving other graph matching techniques are necessary for more challenging instances involving segmentation of multiple similar images. In particular, the proposed algorithm is not suitable for very large displacements. For instance, in our experiments, good results were achieved when the objects were not very far from the single set of scribbles. More specifically, when each object was at least intersected by its correct scribble, limiting the displacements of the objects.

Another step would be to verify the possibility of automatically tuning λ_1 by using training data.

Acknowledgements

We are grateful to FAPESP, CAPES, CNPq, COFECUB and FINEP for their financial support. We thank Camille Couprie for providing the eroded seeds and an implementation for the accuracy

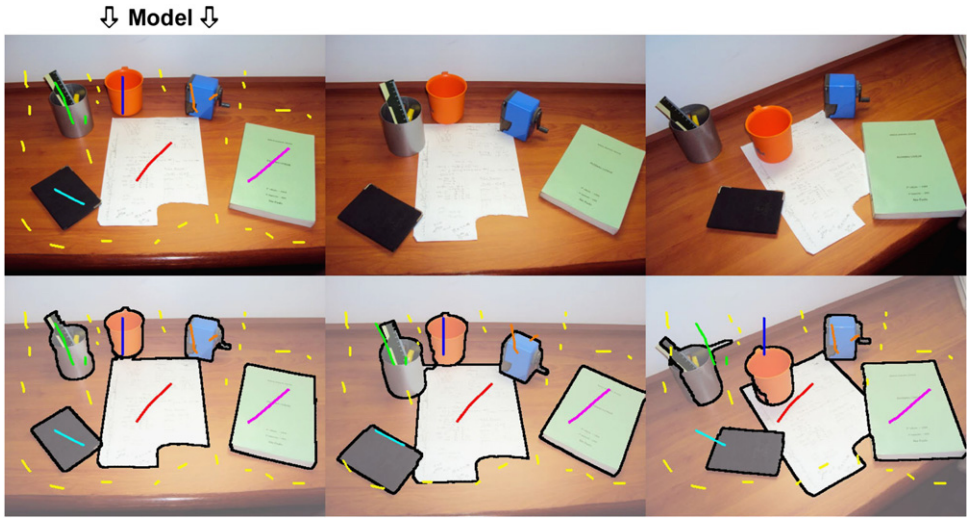

Fig. 28. Example with large changes on the object positions. A single set of scribbles was specifically designed for the image on the left side, chosen as the model. Each segmentation corresponds to the match between the model graph and each input graph representing each image. As in the previous figure, the single set of scribbles was placed over each segmentation result just to emphasize the object displacements.

Fig. 29. Example with large variability of the color information. Similarly to the former figure, the scribbles were drawn over the left image, chosen as the model. Each segmentation was obtained by matching the model graph with each input graph corresponding to each image.

indices used in [12], Kevin McGuinness for the context creator utility to load markers from image files and the implementation of the fuzzy boundary accuracy measure [23], Xue Bai for the input images from [3–5], Jue Wang for the executables for the Robust Matting [35], and all who made the source codes and databases freely available for our experiments. Finally, we would like to thank all the reviewers, which significantly contributed to improve the manuscript.

References

[1] T. Adamek, Using Contour Information and Segmentation for Object Registration, Modeling and Retrieval, Ph.D. Thesis, Dublin City University, 2006.
[2] R. Adams, L. Bischof, Seeded region growing, IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (6) (1994) 641A–647.
[3] X. Bai, G. Sapiro, Distancecut: interactive segmentation and matting of images and videos, in: Proceedings of the International Conference on Image Processing, 2007.
[4] X. Bai, G. Sapiro, A geodesic framework for fast interactive image and video segmentation and matting, in: Proceedings of the IEEE International Conference on Computer Vision, 2007.
[5] X. Bai, G. Sapiro, Geodesic matting: a framework for fast interactive image and video segmentation and matting, International Journal of Computer Vision 82 (2) (2009) 113–132.
[6] S. Beucher, F. Meyer, The morphological approach to segmentation: the watershed transformation, in: E.R. Dougherty (Ed.), Mathematical Morphology in Image Processing, 1993, pp. 433–481.
[7] Y. Boykov, M. Jolly, Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images, Proceedings of the IEEE International Conference on Computer Vision, vol. 1, 2001, pp. 105–112.
[8] H. Bunke, Recent developments in graph matching, in: Proceedings of the International Conference on Pattern Recognition, 2000, pp. 2117–2124.
[10] R.M. Cesar Jr, E. Bengoetxea, I. Bloch, P. Larrañaga, Inexact graph matching for model-based recognition: evaluation and comparison of optimization algorithms, Pattern Recognition 38 (11) (2005) 2099–2113.
[11] L.A. Consularo, R.M. Cesar-Jr., I. Bloch, Structural image segmentation with interactive model generation, in: Proceedings of the IEEE International Conference on Image Processing, Piscataway, NJ, 2007.
[12] C. Couprie, L. Grady, L. Najman, H. Talbot, Power watershed: a unifying graph-based optimization framework, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (7) (2011) 1384–1399.

[13] J. Cui, Q. Yang, F. Wen, Q. Wu, C. Zhang, L.V. Gool, X. Tang, Transductive object cutout, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[14] O. Duchenne, J.-Y. Audibert, R. Keriven, J. Ponce, F. Segonne, Segmentation by transduction, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[15] G. Friedland, K. Jantz, R. Rojas, SIOX: simple interactive object extraction in still images, in: Proceedings of IEEE International Symposium on Multimedia, 2005, pp. 253–259.
[16] F. Ge, S. Wang, T. Liu, New benchmark for image segmentation evaluation, Journal of Electronic Imaging 16 (3) (2007).
[17] L. Grady, Random walks for image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (11) (2006).
[18] J. Guan, G. Qiu, Interactive image segmentation using optimization with statistical priors, in: ECCV International Workshop on the Representation and Use of Prior Knowledge in Vision, 2006.
[20] A. Levin, D. Lischinski, Y. Weiss, A closed-form solution to natural image matting, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2) (2008) 228–242.
[21] Y. Li, J. Sun, C.-K. Tang, H.-Y. Shum, Lazy snapping, ACM Transactions on Graphics 23 (3) (2004).
[22] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, Proceedings of the IEEE International Conference on Computer Vision, vol. 2, 2001, pp. 416–423.
[23] K. McGuinness, N.E. O'Connor, A comparative evaluation of interactive segmentation algorithms, Pattern Recognition 43 (2) (2010) 434–444.
[24] J. Ning, L. Zhang, D. Zhang, C. Wu, Interactive image segmentation by maximal similarity based region merging, Pattern Recognition 43 (2) (2010) 445–456.
[25] A. Noma, A. Pardo, R.M. Cesar-Jr, Structural matching of 2D electrophoresis gels using deformed graphs, Pattern Recognition Letters 32 (1) (2011) 3–11.
[26] B.L. Price, B. Morse, S. Cohen, Geodesic graph cut for interactive image segmentation, in: IEEE Conference on Comput Vision and Pattern Recognition, 2010.
[27] A. Protiere, G. Sapiro, Interactive image segmentation via adaptive weighted distances, IEEE Transactions on Image Processing 16 (4) (2007) 1046–1057.
[28] C. Rother, V. Kolmogorov, A. Blake, "GrabCut": interactive foreground extraction using iterated graph cuts, ACM Transactions on Graphics 3 (2004) 309–314.
[29] P. Salembier, L. Garrido, Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval, IEEE Transactions on Image Processing 9 (2000) 561–576.
[30] H. Sidenbladh, M.J. Black, D.J. Fleet, Stochastic tracking of 3D human figures using 2D image motion, in: Proceedings of the European Conference on Computer Vision, 2000, pp. 702–718.
[31] S. Todorovic, N. Ahuja, Region-based hierarchical image matching, International Journal of Computer Vision 78 (1) (2008) 47–66.
[32] S. Todorovic, N. Ahuja, Unsupervised category modeling, recognition, and segmentation in images, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (12) (2008) 2158–2174.
[33] W.H. Tsai, K.S. Fu, Error-correcting isomorphisms of attributed relational graphs for pattern analysis, IEEE Transactions on Systems, Man, and Cybernetics 9 (12) (1979) 757–768.
[34] L. Vincent, P. Soille, Watersheds in digital spaces: an efficient algorithm based on immersion simulations, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (6) (1991).
[35] J. Wang, M.F. Cohen, Optimized color sampling for robust matting, in: IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.
[36] R.C. Wilson, E.R. Hancock, Structural matching by discrete relaxation, IEEE Transactions on Pattern Analysis and Machine Intelligence 19 (6) (1997) 634–648.
[37] C. Yang, R. Duraiswami, N.A. Gumerov, L. Davis, Improved fast gauss transform and efficient kernel density estimation, in: Proceedings of the IEEE International Conference on Computer Vision, 2003.

Alexandre Noma received his M.Sc. (2003) and Ph.D. (2010) degrees in computer science from University of São Paulo, USP. He is currently pursuing a post-doctoral position at National Institute for Space Research at São José dos Campos, Brazil. His main research interests include graph matching, image segmentation and pattern recognition.

Ana B.V. Graciano received her M.Sc. (2007) in computer science from the University of São Paulo, USP. She is currently a Ph.D. candidate at USP, in collaboration with Télécom ParisTech, France. Her research interests are model-based object segmentation/classification, fusion of statistical and relational cues, graph-based pattern recognition, and medical imaging.

Roberto M. Cesar-Jr received his M.Sc. (1993, Electrical Engineering) from Unicamp and his Ph.D. (1997, Physics) from USP. He is currently a Full-Professor in the Department of Computer Science, IME-USP. He has experience in computer science, with emphasis on computer vision, pattern recognition, image processing and bioinformatics.

Luis Augusto Consularo is Systems Analyst at Informatics Department of Brazilian Superior Electoral Court, Brasília. He received his M.Sc. (1995) in computer science from Federal University of São Carlos, and his Ph.D. (2000) in computational physics from University of São Paulo. His research interests include computer vision and pattern recognition.

Isabelle Bloch is professor at Signal and Image Processing Department, Telecom ParisTech, CNRS LTCI, and coordinates the Image Processing and Understanding group. Her interests include 3-D image/object processing, computer vision, fuzzy and logics mathematical morphology, information fusion, fuzzy set theory, structural, graph-based and knowledge-based recognition, spatial reasoning, and medical imaging.