DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Relatório Técnico

RT-MAC-2009-02

INEXACT GRAPH MATCHING FOR SEGMENTATION AND RECOGNITION OF OBJECT PARTS

Alexandre Noma, Ana B. V. Graciano,
Roberto M. César Jr., Luis A. Consularo and
Isabelle Bloch

Março de 2009

Inexact Graph Matching for Segmentation and Recognition of Object Parts

Alexandre Noma, Ana B. V. Graciano, Roberto M. Cesar Jr.,
Luis A. Consularo *, Isabelle Bloch †

Technical Report
Dept. of Computer Science, Institute of Mathematics and Statistics,
University of São Paulo, São Paulo, Brazil
e-mails: {noma,cesar,abvg}@ime.usp.br

February 13, 2009

Abstract

This article presents a novel graph matching technique for segmenting and recognizing object parts in images according to a model. Both model and input images are represented by means of attributed relational graphs which convey appearance features through object attributes and structural properties through relational attributes. Since the model and input graphs might present topological differences, a new structure called the *deformation* graph is introduced. The segmentation and recognition process is achieved through a new graph matching algorithm based on the deformation graph, which corresponds to labelling the input graph in order to minimize the deformations introduced in the model when it is updated with input information. This method is applied to the problem of multiple label segmentation and has shown to be fast with successful experimental results on both natural color images and video. Experiments have also allowed the evaluation of model robustness and reuse when segmenting different images using the same generated graph model to recognize similar objects in each of them, thus minimizing user intervention. The proposed technique has been implemented as an open-source application available at `http://structuralsegm.sourceforge.net/`.

1 Introduction

Segmentation and recognition of objects in digital images are essential steps in many important imaging applications. Solutions to such problems cover a wide range of techniques, among which, those based on graphs. A graph is a

*L.A. Consularo was with UNIMEP and is now with the TSE - Tribunal Superior Eleitoral, Brasília, Brazil, Brazil (e-mail: consularo@gmail.com).

†I. Bloch is with the Signal and Image Processing Department, TELECOM ParisTech (ENST), CNRS UMR 5141 LTCI, Paris, France (e-mail: isabelle.bloch@enst.fr).

powerful data structure for object representation, since its vertices and edges can be related, respectively, to pictorial elements and their relations and structure. Graphs may easily describe objects in terms of their component parts, being especially suitable when dealing with non-rigid objects such as facial features or the human body [1–3], or any kind of structured object in general [4]. When attributes are assigned to each vertex and to each edge of a graph, representing appearance and relational features, the graph is usually called an *attributed relational graph* (ARG).

The literature on graph-based image segmentation is extensive and introduces important methods, such as the Image Foresting Transform (IFT) [5], graph-cuts and Markov-Random-Fields-based (MRF) methods [6–9], amongst others. According to the resulting image partition, segmentation methods might be classified as binary, when only a foreground-background image is obtained, or as multi-label, when more than two meaningful image regions are produced. Graph-cut methods [6, 7, 9] are an important example of the binary segmentation case. Although not linear in the number of pixels of the input image, the graph-cut based approaches are reported to have very reasonable running times [10]. However, an extension of such methods to the multi-label problem is reduced to the multiway-cut problem [11], which is NP-Hard, requiring heuristics to obtain a solution and additional computational effort [12]. Grady [13] has also proposed a multi-label interactive image segmentation algorithm, in which the input image is viewed as a graph and the task is to determine the probability that a random walker starting at each unlabeled pixel will first reach one of the pre-labeled pixels.

Another class of graph-based segmentation methods has also been proposed in the literature [1, 14–18], usually under an object recognition perspective. In most cases, some supervised information about the target object(s) is expressed by means of a prototype model of the objects of interest, and segmentation is thus seen as a graph-matching problem between two graphs representing the input and the model. The matching step implies a mapping from the input vertices to their corresponding model vertices and, according to the restrictions imposed over such mappings, the algorithms may be classified either as exact or inexact (also known as error-tolerant).

The first category usually searches for strict correspondences between the sets of vertices and edges, accepting isomorphisms [19], sub-graph isomorphisms [20], or at most homomorphisms. All these guarantee that at least all edge relations shall be preserved. On the other hand, the inexact category [19, 21–25] allows approximate solutions when an exact mapping is not available, which is usually done through an optimization procedure which assigns a cost value to each solution and heuristics which search for minimizing or maximizing such values. These procedures might evaluate various aspects of the graphs, such as operations of vertex or edge inclusion or removal, vertex or edge (dis)similarities, among others.

Although the model and input images are related, their ARG representations often present different topologies due to the distinct natures of input and model partitioning, which produces different number of vertices and edges (Fig. 1). Such differences imply difficulties to determine a suitable mapping between the input and model graphs, as well as high computational

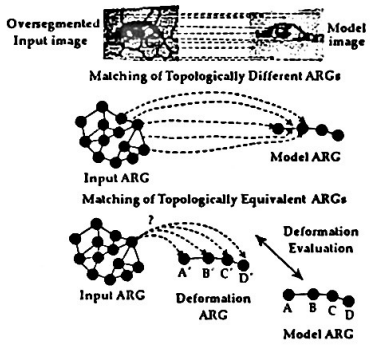


Figure 1: Matching of two topologically different attributed relational graphs versus two topologically equivalent ones under a deformation point of view.

cost. There exists a combinatorial number of possible graph matchings and the optimization procedure not only has to consider vertex similarities, but it also has to evaluate various structural matching configurations in order to avoid fragmented regions and preserve the borders of the objects. The structural information is used to rule out those "configurations" which are not plausible for a final segmentation. However, this might be misleading when both topologies are distinct and cause the method to look down on potential solutions, as shown in Fig. 1. For instance, consider an input edge connecting two input vertices which represent adjacent oversegmented regions related to two distinct objects (e.g. a subregion of the iris and an adjacent subregion of the sclera in Fig. 1). Consider also a model edge connecting model vertices which represent the same previous objects (e.g. the iris and the sclera). Although such a match is suitable in theory, the edges are quite different and may thus be evaluated as a bad solution.

In order to take structure into account and yet avoid the problem of directly matching two topologically different graphs, this paper introduces a novel inexact graph matching algorithm which deals with this matter by viewing each possible matching from an input ARG vertex to a model ARG vertex as a local deformation of the model graph. Such a deformation is expressed by means of the concept of the *deformation* ARG, which represents a distorted version of the model ARG that preserves its topological properties (i.e., its structure) while merging the attributes of a model vertex with the attributes of an input vertex.

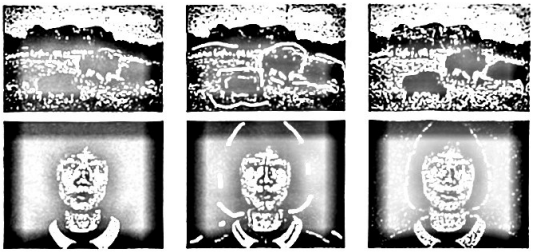
These ideas are a natural evolution of the methods proposed in [3, 26]. A first attempt to segment static images through inexact graph matching was taken in [3]. The method aimed at segmenting facial features according to a manually-created face model image. Both model and input were represented by complete attributed relational graphs, i.e., ARGs with a complete topology, which were then matched in the recognition step. Although these

graphs were complete, they were often considerably distinct since the input one was derived from an oversegmentation and the model one, from a simplified partition of the target regions. However, these differences were not explicitly taken into account. Three inexact approaches considering structure have been evaluated: tree-search, genetic and estimation of distribution algorithms. These optimization techniques have been evaluated in terms of computational efficiency and result accuracy, revealing the best performance for the tree-search approach. Although this work presented successful results, the algorithms were time-consuming and the usage of complete graphs did not necessarily convey a compact representation of object structure.

Thus, the method in [26] proposed a new graph matching algorithm applied to the interactive segmentation problem. The technique aggregated user interaction in order to create a model of the objects to be segmented based upon the user scribbles, similarly to what is done in the present work. Then, segmentation was performed through the proposed graph matching algorithm, which was based on the Sequential Forward Search (SFS) strategy for feature selection [27], and entailed structure in its cost function. The algorithm dealt with differences in graph topology when searching for a mapping: edges from the input were compared to those from the model according to a dissimilarity measure that considers four special topology-match cases in order to avoid the aforementioned structural problems. Each case consisted of one possible combination of presence or absence of an edge with respect to the model graph. However, this approach to the topological discrepancy problem did not cover all possible cases and segmentation could be impaired under certain circumstances.

Regarding the segmentation and recognition issues, the present paper proposes a new algorithm for multi-label object segmentation according to an interactively created part-based model using ARGs as the underlying representation. An object (Fig. 2 (a)) is considered to be a set of parts (subsets of pixels of an image) and their relations. An object model image presenting the relevant object(s) and composing parts of interest is defined by the user according to scribbles drawn over this image (Fig. 2 (b)), as described in [26]. However, the model image could as well be obtained manually as in [3], or automatically inferred as in [28]. Then, the input image is oversegmented, giving rise to two graphs: a model graph, obtained from the oversegmentation of the input image, which includes only the regions that intercept the user traces; and an input graph, which includes all the regions from the oversegmentation. Thus, user-defined traces are used to select a subset of regions of the oversegmented image. Finally, the recognition step is solved through a novel graph matching procedure proposed herein. The resulting image is a partition of the input, labelled according to the model (Fig. 2 (c)). Segmentation and recognition output have been assessed through tests performed over sample images from public libraries and sample frames from two video sequences. Results illustrating the reusability of a single model applied to similar objects in order to justify the model robustness are also presented. All these elements are the main original contribution of the present paper.

This paper is organized as follows. Section 2 presents the formal defini-



(a) (b) (c)

Figure 2: Sample images and the definition of objects and parts in this paper: (a) an object might be a whole scene subdivided into parts (buffalos, background), or a single entity (a person) subdivided into meaningful parts (face, clothing, hair); (b) all part-models are defined by the user through traces drawn over the regions of interest, and each trace color identifies a single object/part; (c) these colors also identify the recognized objects of interest in the segmented images.

tion of attributed relational graphs for image representation adopted in this paper, as well as an overview of the segmentation method proposed herein. Section 3 explains the proposed graph matching algorithm for the purpose of segmentation and recognition, which is based on an optimization technique, described in Section 4. Experimental results are described in Section 5, in which the benefits of the proposed method are highlighted. Finally, a few conclusions, as well as suggestions for future work, are given in Section 6.

2 Methodology overview

2.1 Graph-based representation

In this paper, an *attributed relational graph* (ARG) is a directed graph formally expressed as a tuple $G = (V, E, \mu, \nu)$, where V stands for the set of vertices of G and $E \subseteq V \times V$, its set of edges. Typically, a vertex represents a single image region (subset of image pixels) and an edge is created between vertices representing two image regions. $\mu : V \rightarrow L_V$ assigns an *object* attribute vector to each vertex of V, whereas $\nu : E \rightarrow L_E$ assigns a *relational* attribute vector to each edge of E. $|V|$ denotes the number of vertices in V, while $|E|$ denotes the number of edges in E. Similarly, $|L_V|$ and $|L_E|$ correspond to the number of attributes which compose the vectors μ and ν, respectively. $E(v)$ denotes the set of all directed edges with an endpoint at $v \in V$, i.e. $E(v) = \{e \in E : e = (v, w) \text{ or } e = (w, v), w \in V\}$.

For the purpose of the segmentation method, three instances of such ARGs are considered: an *input* ARG $G_i = (V_i, E_i, \mu_i, \nu_i)$, derived from the input image, a *model* ARG $G_m = (V_m, E_m, \mu_m, \nu_m)$, representing the objects of interest selected by the user, and a *deformation* ARG $G_d = (V_d, E_d, \mu_d, \nu_d)$, used as an auxiliary data structure for measuring deformations implied in the model when matching a vertex $v \in V_i$ to another $w \in V_m$. Subscripts shall be used to denote the corresponding graph, e.g.

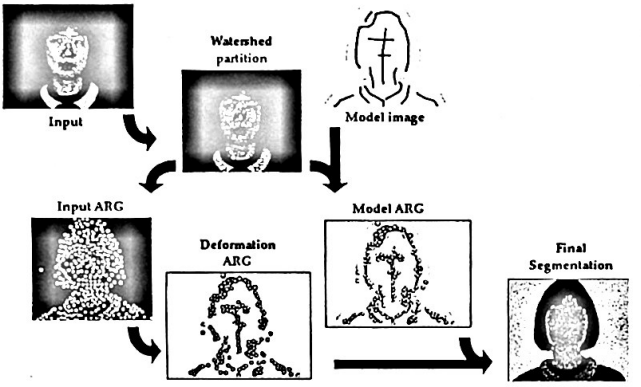


Figure 3: Overview of the methodology steps. The ARGs are depicted only with their vertices for better visualization.

$v_i \in V_i$ denotes a vertex of G_i, whereas $e_i = (v_i, w_i) \in E_i$ denotes an edge of G_i. Similar notations are used for G_m and G_d. Note that, when referring to a graph G_d, an association between a given pair of vertices v_i and v_m is always implicit.

- $\mu(v)[j]$ $(j = 1, \ldots, |L_V|)$ and $\nu(e)[k]$ $(k = 1, \ldots, |L_E|)$ refer to the "j-th" and "k-th" attributes of the respective vectors $\mu(v)$ and $\nu(e)$ associated to a vertex v and an edge e, respectively;
- $\mu'_d(v_d)[j]$ and $\nu'_d(e_d)[k]$ refer to the updated attribute vectors of a vertex $v_d \in G_d$;
- σ_μ^j and σ_ν^k are *update* functions used to compute attributes of a vertex belonging to G_d;
- δ_V^j and δ_E^k are *dissimilarity* functions used to compare the respective object and relational attributes of two given graphs.

2.2 Proposed framework

The segmentation process is depicted step-by-step in Fig. 3. First, a model image must be created and it should indicate a labelling of all the objects to be segmented and their parts of interest. In this paper, the model image is created interactively as described in [26]: given an input image to be segmented, the user first points out the target objects by placing traces over the input, thus creating a model image in which each color identifies an object (or object part) of interest. Next, an oversegmentation is performed using the watershed algorithm [29] in order to obtain a partition image in which the expected contours of each object are present.

This oversegmented image is used to create both an input ARG G_i and a model ARG G_m. The first is obtained in the following way: each watershed

region gives rise to a vertex and its attributes, whereas adjacent regions lead to an edge and its respective attributes. G_m is obtained similarly, but only those watershed regions which intercept the user-defined traces result in a model vertex. The model edges, however, are created according to the desired graph topology (in the present work, complete and triangulated graphs are used). Since it is desirable that the user inputs little effort when segmenting an image, only a few scribbles are expected over the image, thus implying a considerable difference in the number of vertices present in the input and model graphs (Fig. 3). The final segmentation should be a mapping of all $v_i \in V_i$ to vertices in V_m such that input vertices related to image regions corresponding to the same model object should be assigned to the same model vertex. This is equivalent to merging regions of an oversegmented object into a single region. The mapping of vertices from G_i to those in G_m characterizes an inexact graph matching problem [14, 15]. Although many mappings exist, a desirable solution should correspond to an image partition as similar to that defined by the model image as possible.

In order to guarantee that the final mapping follows the model ARG topology, the *deformation* ARG G_d is introduced. This graph is initialized as a copy of G_m and used to evaluate the local deformation effect produced by a given association between a vertex $v_i \in V_i$ and a vertex $v_m \in V_m$ (Fig. 4). This effect is computed through the update of the object attributes of vertex v_d, a clone of v_m, with the object attributes of v_i, as if the regions represented by such vertices were the same. Also, the relational attributes of all edges directly connected to v_d are updated. Note that this update preserves the topology of G_d, solely affecting the values of the object and relational attributes. Finally, this deformed version of the model graph is compared to the original one through a cost function. A desirable solution should minimize this cost and the final classification of a given v_i will be the vertex v_m that corresponds to the minimum cost association. In the next section, we discuss how these deformations are computed and used by the graph matching procedure.

3 The graph matching algorithm for model-based image segmentation

A segmentation of the input image according to the model under the graph-based representation is a solution for the graph matching problem between G_i and G_m, characterized as a mapping $f : V_i \rightarrow V_m$. This implies assigning a corresponding model vertex to each input vertex. Clearly, there are $|V_m|$ possible assignments for each input vertex and the decision of which to choose depends on an optimization procedure.

Let G_d be an ARG initially equal to G_m, i.e., $V_d = V_m$ and $E_d = E_m$. Let also v_d and v_m be two corresponding vertices in G_d and G_m respectively, with $\mu_d(v_d) = \mu_m(v_m)$.The quality of such an assignment may be assessed by computing the deformation which occurs in the model when this input vertex is associated with v_m, which is expressed by an updated version of v_d, whose attributes are thus merged with those from v_i. Thus, after the

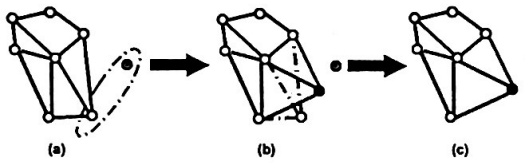


Figure 4: (a) A deformation graph and an input (green) vertex. Initially, each vertex $v_d \in V_d$ is a copy of its corresponding model vertex v_m, and the highlighted pair represents an input vertex v_i and a deformation graph vertex $v_d = v_m$ to be associated. This association represents the fusion of the image regions which are represented by these input and model vertices. (b) The red vertex represents the vertex from the deformation graph whose object attributes were updated with those from the input and model vertices. The blue edges depict the relational attributes which were also updated due to the changes in the attributes of v_d. In order to illustrate the deformation graph computation, let $\mu(v_i) = (10, 20, 30)$ and $\mu(v_m) = (10, 10, 10)$. In this case, $\mu'_d(v_d) = (10, 15, 20)$. Analogously, let (160, 100) and (100, 200) be the centroids of the regions represented by v_i and v_m, respectively. The resulting centroid is represented by the deformation vertex v_d and is the average point between v_i and v_m, at (130, 150). This fact leads to the update of attributes of all edges with an endpoint at v_d. For instance, let v be a neighbor of v_d with centroid at (90, 80), $e_d = (v_d, v) \in E_d$. Then, $\nu'_d(e_d) = (\frac{-40}{d_{max}}, \frac{-70}{d_{max}})$. (c) The resulting deformation graph.

update, $G_d(v_i, v_m)$ becomes a distorted version of the model in which each object attribute (appearance information) $j, j = 1, \ldots, |L_V|$, of the deformed vertex is computed as:

$$\mu'_d(v_d)[j] = \sigma^j_\mu(\mu_d(v_d)[j], \mu_i(v_i)[j]) \tag{1}$$

and each relational attribute $k, k = 1, \ldots, |L_E|$, of the edges connected to the deformed vertex is calculated as:

$$\nu'_d(e_d)[k] = \sigma^k_\nu(\nu_d(e_d)[k]) \tag{2}$$

$\forall e_d \in E_d(v_d) = \{e \in E_d : e = (v_d, w_d)$ or $e = (w_d, v_d), w_d \in V_d\}$, i.e. for all edges e_d linked to v_d.

The functions σ^j_μ and σ^k_ν are called *update* functions and they shall be covered in practice in Section 5.1. Thus, the new value of the j-th object attribute vector of v_d after its fusion with v_i is obtained through a function σ^j_μ which expresses the mixture of the "j-th" input object attribute with the "j-th" original model object attribute. For example, if the $j-$th object attribute is the average color of the region represented by a vertex, then $\sigma^j_\mu(\mu_d(v_d)[j], \mu_i(v_i)[j])$ is the average color of the regions represented by v_d and v_i as if they were a single region. Similarly, because v_d was updated with object attributes of v_i, the edges linked to v_d must have their associated relational attributes updated according to functions $\sigma^k_\nu, k = 1, \ldots, |L_E|$, of choice.

The impact of each local deformation due to the association of a pair of vertices $v_i \in V_i$ and $v_m \in V_m$ is measured according to the following *cost*

function:

$$f(G_d, G_m) = \alpha.c_V(v_d, v_m) + \frac{(1-\alpha)}{|E_d(v_d)|}. \sum_{e_d \in E_d(v_d)} c_E(e_d, e_m) \quad (3)$$

This cost function measures how the association of a vertex v_i with a copy v_d of a model vertex affects the local structure of the graph (second term), as well as the appearance attributes it holds (first term). The parameter α, $0 \leq \alpha \leq 1$, controls the importance of the appearance versus the structural attributes.

The term $c_V(v_d, v_m)$ is a measure of the deformation occurred in the object attributes of v_d in comparison with v_m, being defined as:

$$c_V(v_d, v_m) = \sum_{j=1}^{|L_V|} \beta_V^j . \delta_V^j(\mu_d'(v_d)[j], \mu_m(v_m)[j]) \quad (4)$$

whereas $c_E(e_d, e_m)$ is a measure of the deformation implied in the relational attributes of $e_d \in E_d(v_d)$ with respect to its corresponding original model edge $e_m \in E_m$, defined as:

$$c_E(e_d, e_m) = \sum_{k=1}^{|L_E|} \beta_E^k . \delta_E^k(\nu_d'(e_d)[k], \nu_m(e_m)[k]) \quad (5)$$

Both δ_V^j and δ_E^k are *dissimilarity* functions which compare each deformed attribute associated to a vertex or an edge with those from the original model. Thus, the higher the value of the dissimilarity function between attribute vectors is, the higher is the resulting value of $f(G_d, G_m)$. Constants β_V^j and β_E^K balance the importance among the vertex and edge attributes, respectively.

4 The optimization algorithm

The following graph matching algorithm was devised in order to deal with the topological issue using the deformation graph concept. It takes G_i and G_m as input and maps each vertex from G_i to a single vertex in G_m, as detailed in the pseudo-code below. The output of the algorithm is a mapping of all vertices of V_i to vertices of V_m.

1: define α, β_V^j, for $j = 1, \ldots, |L_V|$, and β_E^k, for $k = 1, \ldots, |L_E|$
2: $G_d \leftarrow G_m$
3: for each vertex $v_i \in V_i$ do
4: $\quad f_{min} \leftarrow \infty$
5: $\quad minlbl \leftarrow -1$
6: $\quad$ for each vertex $v_m \in V_m$ do
7: $\quad\quad$ let $v_d \in V_d$ be the vertex corresponding to v_m
8: $\quad\quad$ let $E_d(v_d)$ be the set of all edges linked to v_d
9: $\quad\quad$ save the original attribute vectors $\mu_d(v_d)$ and $\nu_d(e_d)$, $\forall e_d \in E_d(v_d)$
10: $\quad\quad$ for each attribute $j = 1, \ldots, |L_V|$ of vertex $v_d \in V_d$ do

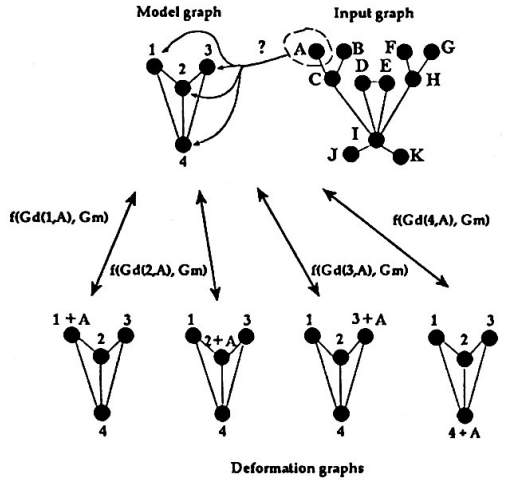


Figure 5: A graphical illustration for the matching of an input vertex to a model vertex according to the implied deformations. Input vertex A must be matched to one of the model vertices $(1 \ldots 4)$. At each iteration, the deformation graph starts as a copy of the model graph and it is then updated for each possible match of A with a model vertex, while the corresponding values of the cost function are computed. The deformations implied by each assignment are highlighted in the picture as red arcs and red vertices, and these distorted vertex and edges are compared to the original ones present in the model. Finally, the match leading to the lowest cost is chosen as the solution for A.

$\mu'_d(v_d)[j] \leftarrow \sigma_\mu(\mu_d(v_d)[j], \mu_i(v_i)[j])$
end for
for each $e_d \in E_d(v_d)$ **do**
for each attribute $k = 1, \ldots, |L_E|$ **of** e_d **do**
$\nu'_d(e_d)[k] \leftarrow \sigma^k_\nu(\nu_d(e_d)[k])$
end for
end for
compute the value f of the cost function $f(G_d(v_i, v_m), G_m)$ (Eq. 3)
if $f < f_{min}$ **then**
$f_{min} \leftarrow f$
$minlbl \leftarrow v_m$
end if
restore the original attribute vectors of $\mu_d(v_d)$ and $\nu_d(e_d)$, $\forall e_d \in E_d(v_d)$
end for
label$(v_i) \leftarrow minlbl$
end for

The algorithm works as follows. Firstly, it is necessary to set the parameters α, β_V^j, $j = 1, \ldots, |L_V|$, and β_E^k, $k = 1, \ldots, |L_E|$. These parameters weigh the influence of the appearance against the structural information (parameter α, Eq. 3), as well as the importance of each attribute.

Line 2 initializes the *deformation* graph G_d with a copy of the original model G_m. Line 3 loops through all the vertices $v_i \in V_i$ from the input ARG G_i, so that all of them are mapped to a vertex in G_m. Line 6 loops through all the vertices $v_m \in V_m$ from the model ARG G_m, in order to analyze each candidate pair (v_i, v_m). First, the attributes of the vertex v_d corresponding to v_m in the *deformation* graph are updated, as well as those of the edges linked to v_d (lines 10–17). Then (line 18), the algorithm computes the impact of the deformation expressed by $G_d(v_i, v_m)$ by calculating the corresponding value of the cost function (Eq. 3), while keeping track of the model vertex which led to the cheapeset solution found so far for that v_i (lines 19–22). Finally, v_i is mapped (labelled) to the model vertex which is associated to the cheapest solution (line 25). This process is repeated until all input vertices are evaluated and it results in each input vertex labelled with a single model vertex.

We shall briefly analyze the algorithm complexity. For each association of vertices from the input and model graphs, the deformation graph is recomputed. This implies constant-time computations for the update of an object attribute vector and, at most $|E_m|$ constant-time updates of relational attributes. The same holds true for the evaluation of the cost function value (Eq. 3). Thus, the algorithm is $\Theta(|V_i|.|E_m|)$, which also means that the running time varies according to the adopted model graph topology. The worst-case happens when the model graph is actually complete. Nevertheless, simpler topologies, such as that of a planar graph, require significantly less time and produce successful segmentations, as Section 5 shows.

Fig. 5 illustrates how the algorithm works using the *deformation* graph. In this example, an input vertex A shall be matched to one of the four model vertices (1 through 4). Thus, all possible assignments are evaluated and the cost function is computed for each possible match. The final label of A corresponds to the model vertex which was least affected by the deformation introduced by the attributes of v_i.

5 Experiments

In order to test the present technique, a Java application was designed and implemented [1]. Its interface allows the user to load images to be segmented, create a model image according to traces drawn over different regions of interest of the input image, and define the parameters of the cost and dissimilarity functions (Eqs. 3, 8 and 9).

Tests have been performed using sample natural static images from the Berkeley Image Segmentation Database [2] and frames of two sample video

[1] Visit `http://structuralsegm.sourceforge.net/` for more results and demo videos.

[2] `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/`

sequences. Model images have been created with the aid of the interface in order to point out the objects of interest for the purpose of segmentation.

In all experiments, the input ARGs derived for each input image present the same topology: they include a vertex for each region produced by the oversegmentation and there are edges connecting two vertices representing adjacent regions. On the other hand, the model ARGs followed either a triangulated topology [3] (Fig. 7 right) or a complete graph structure. These different levels of structural information embedded in the model graphs allow the assessment of how segmentation quality and computer performance vary in such cases.

A set of object and relational attributes, as well as update and dissimilarity functions, have been systematically used in all performed tests. However, the weight parameters of the cost and dissimilarity functions are not necessarily the same, since they have been adopted according to the characteristics of each image under analysis. All these shall be described in the remainder of this section.

5.1 Attributes

For color images, the object attribute vector was composed of a single attribute ($|L_V| = 1$): the average $L^*a^*b^*$ tuple characterizing the corresponding image region, i.e. $\mu(v) = (L^*_v, a^*_v, b^*_v)$. The CIELAB [31] color space was chosen because it is a closer representation of the way humans percept color and its metrics were suitable and straightforward for the idea of mixing attributes from two image regions when building the deformation graph, as explained in the next subsection. When dealing with gray-scale images, $\mu(v) = (g(v))$, where $g(v)$ denotes the average gray-level of the image region associated to vertex $v \in V$. Each $L^*a^*b^*$ component or gray-scale value is normalized between 0 and 1 with respect to the minimum and maximum possible range values.

Similarly, a single relational attribute ($|L_E| = 1$) is associated to each edge $e = (v, w) \in E$, $v, w \in V$, being defined as $\nu(e) = \frac{(p_w - p_v)}{d_{max}}$, where p_v and p_w are the centroids of their corresponding image regions. The constant d_{max} represents the largest distance between any two points of the input image region and it is used to keep the modulus of each vector between 0 and 1. Thus, the relational attribute is a geometric vector normalized by d_{max}, whose origin is the centroid of the source vertex and whose tip is the centroid of the sink vertex. Other attributes may easily be employed, since the methodology presented herein does not impose any restriction on the nature of μ and ν.

5.2 Update functions

When searching for the best correspondence between an input and a model vertex, different deformation graphs are obtained. As explained in Section 3,

[3] In the segmentation problem, each vertex stores the centroid of its respective region, and triangulated edges were obtained by a Delaunay Triangulation [30] based on these centroids, since it induces a planar graph and maintains the connectivity between adjacent regions.

each G_d expresses the fusion of a given input vertex v_i and a vertex v_d. Thus, the object and relational attribute vectors of v_d must be updated after each fusion, in order to reflect the deformation under analysis.

The following updating functions have been devised to update the aforementioned attributes:

$$\mu'_d(v_d) = \sigma_\mu(\mu_d(v_d), \mu_i(v_i)) = (\frac{L^*_{v_d} + L^*_{v_i}}{2}, \frac{a^*_{v_d} + a^*_{v_i}}{2}, \frac{b^*_{v_d} + b^*_{v_i}}{2}) \quad (6)$$

$$\nu'_d(e_d) = \sigma_\nu(\nu_d(e_d)) = \frac{(p_{w_d} - p'_{v_d})}{d_{max}} \quad (7)$$

$\forall e_d = (v_d, w_d) \in E_d(v_d)$, $p'_{v_d} = \frac{p_{v_d} + p_{v_i}}{2}$, $p_{v_d} = p_{v_m}$ before the update.

Thus, the new value of the object attribute $\mu'_d(v_d)$ after merging the attributes of v_i and v_d (or equivalently, v_m) expresses the average color of all pixels of the regions represented by v_i and v_m, as if they were a single region. The structural impact after associating v_i and v_m is represented in G_d by the updated relational attribute vector ν'_d of each directed edge linked to v_d. This update expresses the distorted geometric vectors which connect the points corresponding to the centroid of v_d, which is the average of the centroids of v_i and v_m, and the centroids of its neighbor vertices.

5.3 Dissimilarity functions

Once a deformation graph G_d is computed to express the distortion associated to a given matching candidate pair, the respective value of the cost function f (Eq. 3) must be computed in order to measure the level of dissimilarity between G_d and the original model G_m. Since f is defined in terms of vertex and edge dissimilarity costs, it is necessary to define distance metrics for object and relational attributes (appearance and structure, respectively).

To compute the deformation between the object attributes of v_d and v_m, the term $c_V(v_d, v_m)$ is defined as:

$$c_V(v_d, v_m) = \beta_V . \delta_V(\mu'_d(v_d), \mu_m(v_m)) = \sqrt{\sum_{C=L^*,a^*,b^*} (C_{v_d} - C_{v_m})^2} \quad (8)$$

Similarly, if $e_d \in E_d(v_d)$ and $e_m \in E_m$ is its corresponding model edge, then $c_E(e_d, e_m)$ is a measure of the deformation between both edges defined as:

$$c_E(e_d, e_m) = \beta_E . \delta_E(\nu'_d(e_d), \nu_m(e_m)) = \gamma_E . \frac{|\cos(\theta) - 1|}{2} + (1 - \gamma_E) . |\|\nu(e_d)\| - \|\nu(e_m)\|| \quad (9)$$

The value θ is the angle between the vectors $\nu(e_d)$ and $\nu(e_m)$, whereas the parameter γ_E, $0 \leq \gamma_E \leq 1$, controls the weights of the angular and modular dissimilarities between vectors. The angular cost (Fig. 6) is high for opposite vectors and low for vectors with similar orientation. Thus, the structural distance (the second term in Eq. 3) is computed as the average over all distorted edges e_d (directly connected to v_d) compared to their respective model edges $v_m \in V_m$. Note that, because a single object or relational attribute is being considered in each dissimilarity function, the weights β_V and β_E are both set to 1.

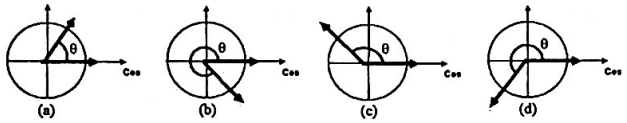


Figure 6: The angular cost in Eq. 9 results in lower values when the vectors have similar orientations (a, b) and higher values for opposite orientations (c, d).

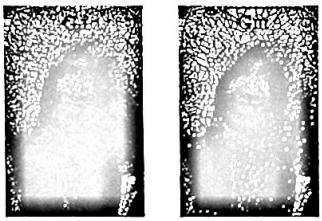


Figure 7: Input and model graphs: based upon the watershed regions, both input and model graphs are created. Edges are depicted by blue lines. Input edges (E_i) represent the adjacency between watershed regions and model edges (E_m) represent the adjacency of regions of a Delaunay triangulation obtained from the centroids of the regions which intercept the user scribbles, as highlighted in the image on the right.

5.4 Results on static images

The experiments presented herein depict several segmentations of sample natural gray-scale and color images obtained from the Berkeley Image Segmentation Database. The test environment comprised a machine with a 1.4MHz CPU and 1.5Gb RAM.

Each picture was segmented according to the following protocol. First, the user scribbles over the input image, using a different color to identify each object or part to be segmented. Next, the input image is oversegmented through the watershed algorithm, giving rise to the input and model graphs (Fig. 7) as described in Section 2.2. The topology used in these examples was that of a triangulated graph, in which the edges are obtained by a Delaunay triangulation [30] from the centroids of the regions represented by the vertices. Then, the α and γ parameters are set and the matching algorithm proposed herein is applied, leading to the object segmentation.

Sample resulting images (Figs. 8 and 9) show the final regions labelled according to the color of the strokes defined by the user. Although certain image regions, such as the geisha, the mountains, and the eskimo, present high variability due to textures or varying appearance, the final segmentation remains accurate and robust thanks to the structural constraints embedded in the model.

Another set of experiments aimed at evaluating the performance of the algorithm in terms of result quality and speed, when distinct topologies have been carried out. For this purpose, images were segmented using either a

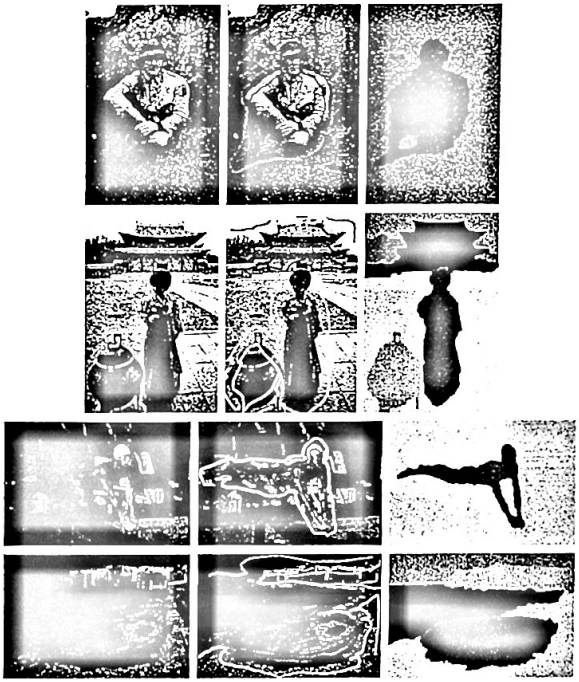


Figure 8: Sample color images from the Berkeley dataset used to test the proposed method: 'soldier', 'geisha', 'swimmer' and 'boat'. From left to right: input image, scribbles drawn by the user over the input image and the final segmentation.

triangulated model graph or a complete graph, in which there is an edge connecting every pair of vertices. Simultaneously, these experiments studied the influence of appearance versus structural information (Eq. 3), since segmentations were obtained using $\alpha = 0.0, \ldots, 1.0$, keeping $\gamma_E = 0.5$ in order to assign the same importance to the angular and modular costs (Eq. 9). Figs. 10 through 12 present sample results of this experiment.

Based upon the obtained results, it can be seen that, as α increases, the importance of the appearance information increases, resulting in more fragmented regions. Thus, the variation of α reflected the influence of the appearance versus the structural information over the segmentation, producing a better result for α between 0.1 and 0.4, for complete graphs, and between 0.1 and 0.6, for triangulated models. In terms of speed, because the algorithm runs in time proportional to the number of model edges, the running times for the segmentations using complete graphs were, as expected, much higher than those obtained for triangulated models, as shown in Tables 1 and 2.

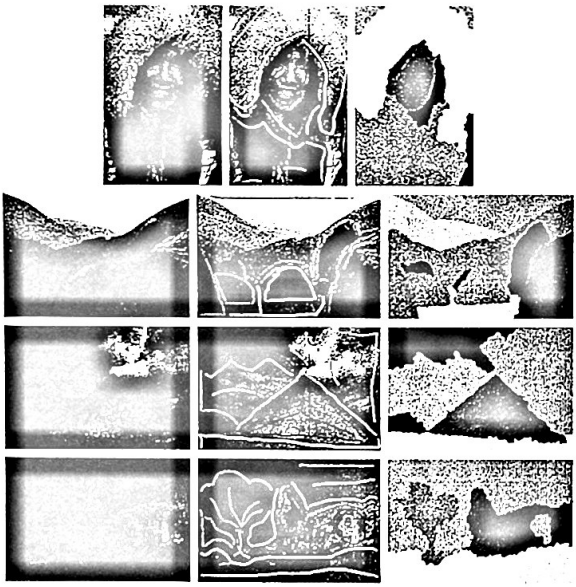


Figure 9: Sample gray-scale images from the Berkeley dataset used to test the proposed method: 'eskimo', 'rocks', 'pyramid' and 'landscape'. From left to right: input image, scribbles drawn by the user over the input image and the final segmentation.

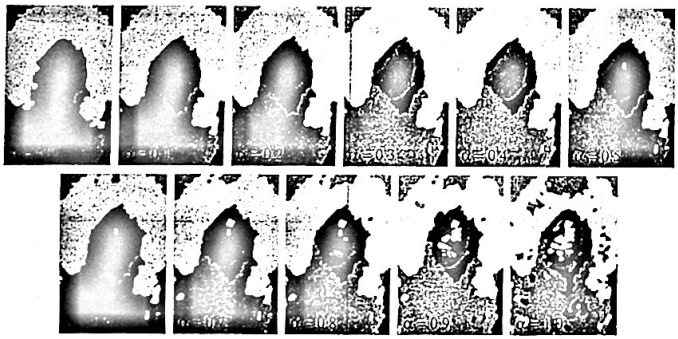


Figure 10: Results from the proposed algorithm using a complete model graph, for $\alpha = 0.0, \ldots, 1.0$.

Moreover, the deformation graph using triangulated models better preserved the uniform regions in the segmentation (as α increases) than using complete model graphs. This suggests that the triangulated models are

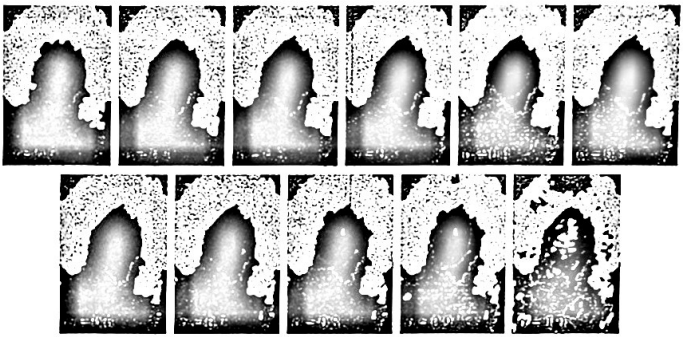


Figure 11: Results using a triangulated model graph, for $\alpha = 0.0, \ldots, 1.0$. Note that these results present more uniform regions than the ones obtained using a complete model (Fig. 10).

Segmented Image	$\lvert V_i \rvert$	$\lvert E_i \rvert$	$\lvert V_m \rvert$	$\lvert {E_m}^C \rvert$	$\lvert {E_m}^T \rvert$
eskimo	546	3050	224	49952	1308
rocks	841	4802	313	97656	1844
pyramids	822	4642	327	106602	1928
landscape	998	5658	367	134322	2170

Table 1: The sizes of the input and model graphs used in the experiments of performance evaluation and comparison between the proposed algorithm DG and the SFS method, in which $\lvert {E_m}^C \rvert$ and $\lvert {E_m}^T \rvert$ denote the number of edges in the complete and in the triangulated model graphs, respectively.

more adequate to represent structural information, since each region considers only the structural relations from its immediate neighbors, providing more relevant context information than farther regions.

Finally, the present algorithm was compared to the SFS - sequential forward search - matching algorithm described in [26]. In summary, this algorithm takes as input two graphs (the input and the model) and finds a mapping between their set of vertices. First, the algorithm computes, for each input graph vertex, its corresponding *supervertex*, an auxiliary structure that represents an ordered list of all possible mappings between that given input vertex and all model ones. This order is defined according to the increasing Euclidean distances between the input and model vertex centroid attributes and the closest pair defines the cost of its respective supervertex. Then, the set of supervertices is also sorted in increasing order of costs, defining an order in which input vertices should be visited to determine the total mapping. Each input vertex v_i is analysed in this particular order and the algorithm computes the cost of the whole solution obtained so far. However, the cost of the solution only gradually takes structure into account. Also, the algorithm copes with differences in the topologies of both graphs by evaluating four distinct possible configurations and by computing edge dissimilarities accordingly. Nevertheless, the considered configurations

Figure 12: Results using complete or triangulated model graphs, for $\alpha = 0.0, \ldots, 1.0$. Each line presents the results for a given α for all three images, in increasing order from top to bottom, using complete and triangulated model graphs.

do not cover the whole range of structural possibilities and may impair the final segmentation.

The comparison between both algorithms was achieved through tests performed over the same set of gray-scale images and using the same user scribbles for each image. Both algorithms were implemented under the same Java framework and they have been presented with the same pair of input and model graphs for each segmentation. As in the previous experiment, two types of graphs were used - triangulated and complete - and the α parameter

Segmented Image	Complete model		Triangulated model	
	DG	SFS	DG	SFS
eskimo	8.8	10.8	1.5	3.9
rocks	25.5	35.6	3.2	13.4
pyramids	27.7	35.9	3.2	17.6
landscape	40.5	59.2	4.3	27.7

Table 2: The average times (in seconds) for segmenting the given gray-scale images using either the matching procedure proposed in this paper (DG) or the SFS algorithm from [26]. Complete and triangulated models were used and both algorithms were implemented in Java and ran under the same conditions for the purpose of comparison.

was varied. Using complete model graphs, both algorithms produced similar results and responded alike to the variation of α. On the other hand, the use of triangulated model graphs shows that the present method outperforms the SFS algorithm in terms of segmentation accuracy.

Note that the bad segmentation produced by the SFS when $\alpha = 0.0$, i.e. only structural information is considered, is due to the fact that the small number of edges in the triangulated model impairs the accurate comparison between input and model edges used to compute the edge dissimilarity. Thus, due to the order given by the supervertices, the first input vertices classified by the SFS correspond to the input regions in the model, and they are not guaranteed to be associated to the correct model region, compromising the whole structure information embedded in the solution. On the other hand, the present matching algorithm does not depend on the order in which vertices from the input are labelled, since each vertex is treated separately as a deformation of the model.

The new algorithm proposed for the graph matching step, using triangulated model graphs, presents time complexity $O(|V_i|.|E_m|)$, with $|E_m| = O(|V_m|)$, for triangulated model graphs, and $|E_m| = O(|V_m|^2)$, for complete model graphs. The SFS algorithm, however, is bounded by a function $\Omega(|V_i||V_m|^2)$. Thus, even though they share the same upper bound for complete model graphs, the running times of the methods were not alike, with the SFS algorithm taking longer for all inputs. On the other hand, the use of triangulated model graphs decreases the time complexity by a factor of $|V_m|$, which drastically drops the time consumed by the present algorithm. Tables 1 and 2 highlight these differences in speed for the set of tests performed, presenting their approximate average running times.

5.5 Model robustness and reuse

To reduce user interaction, experiments have been performed in order to evaluate the robustness and reusability of the model when applied to different images presenting similar objects. The set of input images to be segmented comprised sample frames of a moving head video from the XM2VTS Database [4] and a set of random motorcycle images (Fig. 14). For these tests, the adopted model graph topology was that of triangulated graphs,

[4] http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/

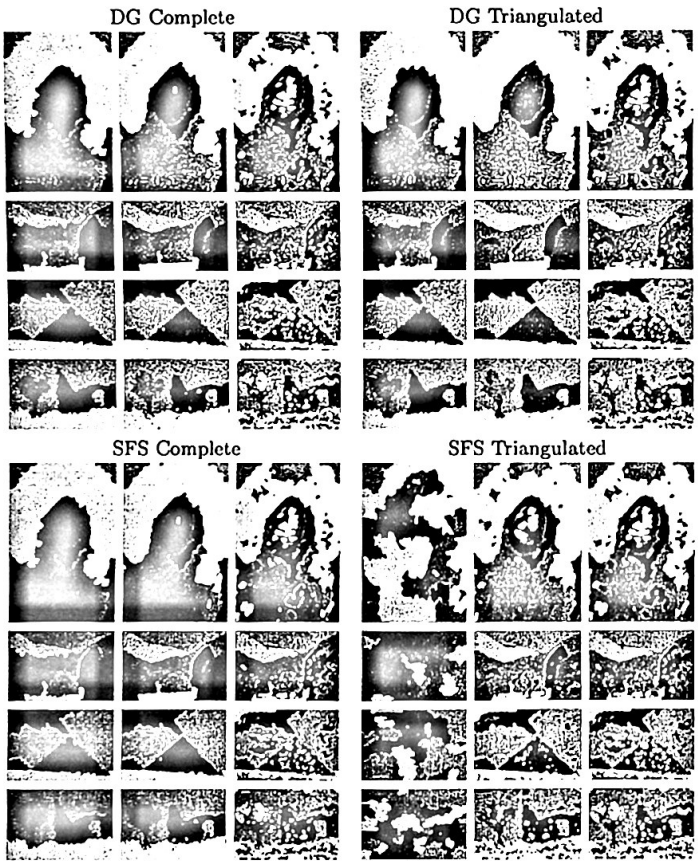


Figure 13: Sample segmentation results of the 'eskimo', 'rocks', 'mountains', and 'landscape' images obtained from both algorithms (the present method DG and the SFS) using complete and triangulated model graphs, for $\alpha = 0.0, 0.5$, and 1.0.

since they have proven to produce good results in less time, as previously discussed.

The first test is depicted in Fig. 15 and it works as follows. The first object, i.e. the woman's head subdivided into parts, is interactively segmented in the same manner as a single static image (Section 5.4). Thus, a triangulated model graph is obtained from the intersection of the user scribbles with the watershed regions that fall inside the subimage defined by the minimum rectangle that encloses the scribbles. The input graph is

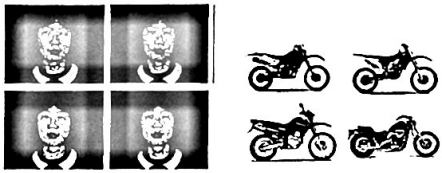


Figure 14: Input images with multiple similar objects in order to test the reusability of the triangulated models.

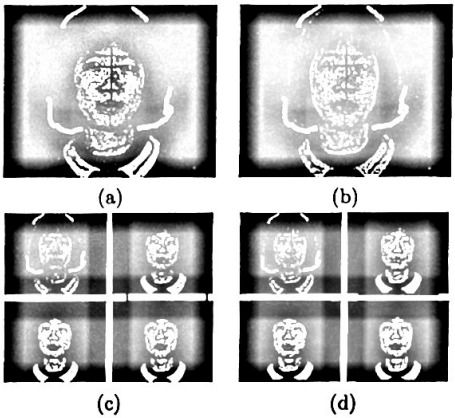
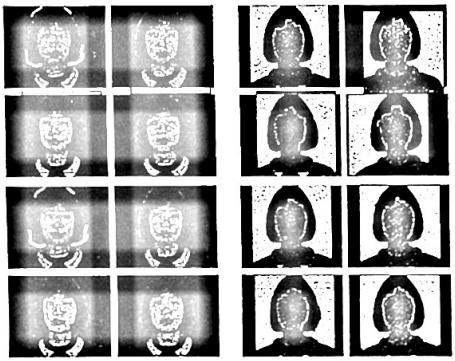


Figure 15: Reusing one single triangulated model in order to segment similar objects. First, the user performs a regular segmentation in one object by defining the scribbles (a) to segment it (b). Then the user places the red rectangle (c) over each desired object to be segmented, as indicated in (d). The same model created in the first segmentation is used to segment all other similar objects, thus reducing the level of user interaction.

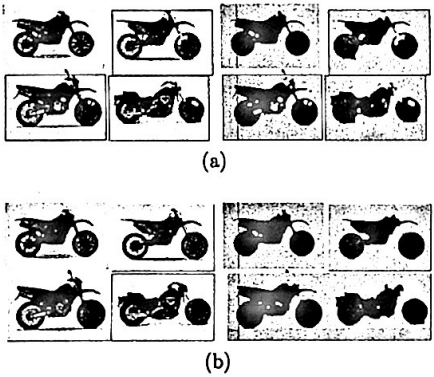
obtained similarly, but from all the watershed regions inside the rectangle. Then, the other segmentations are obtained by placing the rectangle over the other objects, obtaining new input graphs, and computing the segmentations through the matching between each input graph and the previously obtained model. Despite the fact that the deformation graph depends on the centroids of the regions in order to calculate the graph matching, the method is robust to small displacements when the appearance information compensates the structural 'noise' (Fig. 16 (a), bottom left result).

A similar test was applied to the motorcycles in Fig. 17. As the figure shows, the results are sensitive to the rectangle position since the appearance changes from one motorcycle to another. However, the structures are similar and, thus, for better results, greater influence was given to the structure ($\alpha = 0.1$). Because of this, the worst segmentation was that of the last motorcycle (on the bottom right), since its structure differs from the other



(a) (b)

Figure 16: (a) A distorted placement of the rectangles. (b) The 'optimum' placement of the rectangles. In both cases, it was used the proposed method with $\alpha = 0.4$ in order to obtain the matching.



(a)

(b)

Figure 17: (a) A distorted placement of the rectangles ($\alpha = 0.4$). (b) The 'optimum' placement of the rectangles ($\alpha = 0.1$).

motorcycles (e.g. larger distance between wheels).

Finally, the reusability of the model was also tested on the digital video sequence "Person walking straight" [5] presented in [32]. To obtain the triangulated model, user scribbles were drawn over one particular frame selected from the sequence (Fig. 18 (a)) and the proposed algorithm was applied using this single model in order to segment the rest of the sequence. The additional information of the X axis position of the person was also used to update the centroid coordinates of each part represented by a model vertex.

[5] http://www.csc.kth.se/~hedvig/data.html

(a) (b) (c)

Figure 18: (a) Selected frame and the user traces for the triangulated model. (b) Position calculated through background subtraction: first, the object is detected by background subtraction with threshold; then, the position of the object in the X axis is given by the red line dividing in the middle the minimum (green) rectangle that encloses the thresholded components. (c) The median pixels of the first five frames used as the background model.

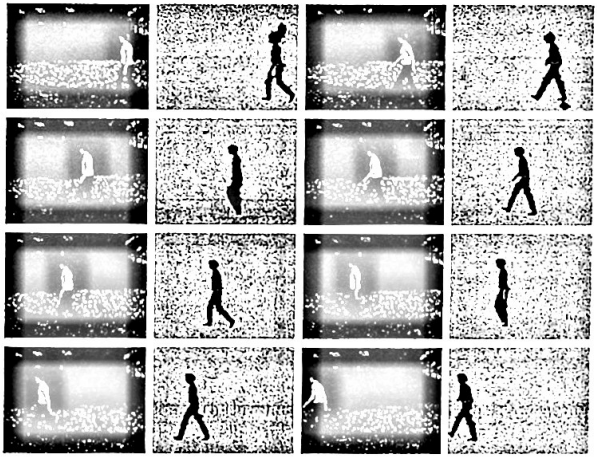


Figure 19: The results of the proposed method for a digital video sequence using the triangulated model in Fig. 18 (a).

This information was obtained by a simple background subtraction (Fig. 18 (b)), which used the median pixels of the first five frames as the background model (Fig. 18 (c)). The input graphs were created from the watersheds of each frame. Results are illustrated in Fig. 19.

6 Conclusion

This paper proposed a novel algorithm for performing model-based segmentation and recognition of object parts in images using attributed relational graphs to represent both input and model. This approach takes into account both appearance and structural (the spatial relations between the vertices) information in order to obtain a matching between input and model graphs.

Topological differences between both graphs are dealt with by means of a deformation ARG, a structure which evaluates the local impacts (or deformations) caused by the association of a given input vertex with a model vertex. Since structure is at stake, two graph topologies have been evaluated: triangulated model graphs and complete model graphs. The new algorithm using triangulated models produced successful segmentation results with fast performance. Also, the method has presented model robustness when recognizing object parts based on appearance and structural information, as well as model reusability when segmenting similar objects in several images.

Our ongoing work is devoted to improving model robustness. For instance, an algorithm robust to translation is desirable in order to allow the segmentation of multiple objects without placing a rectangle over each object and to recognize the object parts in a video sequence without providing the information about the position of the object. This shall be accomplished through an investigation of a MAP-MRF model within this framework.

Acknowledgements. This work was supported by FAPESP, CAPES, CNPq, COFECUB and FINEP.

References

[1] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *Int. J. Comput. Vision*, vol. 61, no. 1, pp. 55–79, 2005.

[2] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-based object detection in images by components," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 4, pp. 349–361, 2001.

[3] R. M. Cesar-Jr., E. Bengoetxea, I. Bloch, and P. Larrañaga, "Inexact graph matching for model-based recognition: Evaluation and comparison of optimization algorithms," *Pattern Recognition*, vol. 38, no. 11, pp. 2099–2113, 2005.

[4] D. Crandall, P. Felzenszwalb, and D. Huttenlocher, "Spatial priors for part-based recognition using statistical models," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition*, vol. 1, 2005, pp. 10–17.

[5] A. X. Falcão, J. Stolfi, and R. A. Lotufo, "The image foresting transform: Theory, algorithms, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 19–29, Jan 2004.

[6] A. Blake, C. Rother, M. Brown, P. Pérez, and P. H. S. Torr, "Interactive image segmentation using an adaptive GMMRF model," in *Proc. European Conf. Comput. Vision*, 2004, pp. 428–441.

[7] C. Rother, V. Kolmogorov, and A. Blake, ""GrabCut": interactive foreground extraction using iterated graph cuts," in *ACM SIGGRAPH*, 2004, pp. 309–314.

[8] A. K. Sinop and L. Grady, "A seeded image segmentation framework unifying graph cuts and random walker which yields a new algorithm," in *Proc. Int. Conf. Comput. Vision*, 2007, pp. 1–8.

[9] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient n-d image segmentation," *Int. J. Comput. Vision*, vol. 70, no. 2, pp. 109–131, 2006.

[10] Y. Boykov, O. Veksler, and R. Zabih, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1124–1137, 2004.

[11] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis, "The complexity of multiway cuts," *ACM Symp. Theory of Computing*, pp. 241–251, 1992.

[12] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, 2001.

[13] L. Grady, "Random walks for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, 2006.

[14] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition." *Int. J. Pattern Recognition and Artificial Intell.*, vol. 18, no. 3, pp. 265–298, 2004.

[15] H. Bunke, "Recent developments in graph matching." in *Int. Conf. Pattern Recognition*, 2000, pp. 2117–2124.

[16] H. Bunke and B. T. Messmer, "Recent advances in graph matching," *Int. J. Pattern Recognition and Artificial Intell.*, vol. 11, no. 1, pp. 169–203, 1997.

[17] H. Bunke, "Error correcting graph matching: On the influence of the underlying cost function," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 9, pp. 917–922, 1999.

[18] R. C. Wilson and E. R. Hancock, "Structural matching by discrete relaxation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 6, pp. 634–648, 1997.

[19] M. Gori, M. Maggini, and L. Sarti, "Exact and approximate graph matching using random walks," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1100–1111, 2005.

[20] B. T. Messmer and H. Bunke, "Efficient subgraph isomorphism detection: A decomposition approach," *IEEE Trans. Knowledge and Data Eng.*, vol. 12, no. 2, pp. 307–323, 2000.

[21] A. Cross, R. Wilson, and E. Hancock, "Inexact graph matching using genetic search," *Pattern Recognition*, vol. 30, no. 6, pp. 953–970, 1997.

[22] A. Cross and E. Hancock, "Graph matching with a dual step EM algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 11, pp. 1236–1253, 1998.

[23] L. Shapiro and R. Haralick, "Structural descriptions and inexact matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 3, no. 5, pp. 504–519, 1981.

[24] S. Medasani and R. Choi, "Graph matching by relaxation of fuzzy assignments," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 1, pp. 173–182, 2001.

[25] T. Caelli and S. Kosinov, "Inexact graph matching using eigen-subspace projection clustering," *Int. J. Pattern Recognition and Artificial Intell.*, vol. 18, no. 3, pp. 329–354, 2004.

[26] L. A. Consularo, R. M. Cesar-Jr, and I. Bloch, "Structural image segmentation with interactive model generation," in *Proc. IEEE Int. Conf. Image Processing*, vol. 6, San Antonio, Texas, USA, September 2007, pp. 45–48.

[27] A. Jain and D. Zongker, "Feature selection: Evaluation, application, and small sample performance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 2, pp. 153–158, 1997.

[28] D. Kim, I. Yun, and S. Lee, "A new shape decomposition scheme for graph-based representation," *Pattern Recognition*, vol. 38, no. 5, pp. 673–689, May 2005.

[29] L. Vincent and P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 6, pp. 583–598, 1991.

[30] L. Guibas and J. Stolfi, "Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams," in *Proc. 15th Annual ACM Symp. Theory of Computing*, 1983, pp. 221–234.

[31] G. Hoffmann, "Cielab colorspace," http://www.fho-emden.de/~hoffmann/cielab03022003.pdf.

[32] H. Sidenbladh, M. J. Black, and D. J. Fleet, "Stochastic tracking of 3d human figures using 2d image motion," in *Proc. European Conf. Comput. Vision*, vol. 2, 2000, pp. 702–718.

RELATÓRIOS TÉCNICOS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
Instituto de Matemática e Estatística da USP

A listagem contendo os relatórios técnicos anteriores a 2003 poderá ser consultada ou solicitada à Secretaria do Departamento, pessoalmente, por carta ou e-mail (mac@ime.usp.br).

Carlos H. Cardonha, Marcel K. de Carli Silva e Cristina G. Fernandes
COMPUTAÇÃO QUÂNTICA: COMPLEXIDADE E ALGORITMOS
RT- MAC 2005-01 – janeiro 2005

C.E.R. Alves, E. N. Cáceres and S. W. Song
A BSP/CGM ALGORITHM FOR FINDING ALL MAXIMAL CONTIGUOS SUBSEQUENCES OF A SEQUENCE OF NUMBERS
RT- MAC- 2005-02 – janeiro 2005

Flávio S. Corrêa da Silva
WHERE AM I? WHERE ARE YOU?
RT- MAC- 2005-03 – março 2005, 15pp.

Christian Paz-Trillo, Renata Wassermann and Fabio Kon
A PATTERN-BASED TOOL FOR LEARNING DESIGN PATTERNS
RT- MAC – 2005-04 – abril 2005, 17pp.

Wagner Borges and Julio Michael Stern
ON THE TRUTH VALUE OF COMPLEX HYPOTHESIS
RT- MAC – 2005-05 – maio 2005, 15 pp.

José de Ribamar Braga Pinheiro Jr., Alexandre César Tavares Vida and Fabio Konl
IMPLEMENTAÇÃO DE UM REPOSITÓRIO SEGURO DE APLICAÇÕES BASEADO EM GSS – PROJETO TAQUARA
RT- MAC – 2005-06 – agosto 2005, 21 pp.

Helves Domingues and Marco A. S. Netto
THE DYNAMIC DEPENDENCE MANAGER PATTERN
RT – MAC 2005-07 – dezembro 2005, 12 pp.

Marco A. S. Netto, Alfredo Goldman and Pierre-François Dutot
A FLEXIBLE ARCHITECTURE FOR SCHEDULING PARALLEL APLICATIONS ON OPPORTUNISTIC COMPUTER NETWORKS
RT- MAC 2006-01 – Janeiro 2006, 18 pp.

Julio M. Stern
COGNITIVE CONSTRUCTIVISM AND LANGUAGE
RT – MAC 2006-02 – Maio 2006, 67 pp.

Arlindo Flávio da Conceição and Fabio Kon
EXPERIMENTS AND ANALYSIS OF VOICE OVER IEEE 802.11 INFRASTRUCTURED NETWORKS
RT – MAC 2006-03 – Junho 2006,

Giuliano Mega and Fabio Kon
DISTRIBUTED SYMBOLIC DEBUGGING FOR THE COMMON PROGRAMMER
RT – MAC 2006-04 – Junho 2006

Pedro J. Fernandez, Julio M. Stern, Carlos Alberto de Bragança Pereira and Marcelo S. Lauretto
A NEW MEDIA OPTMIZER BASED ON THE MEAN-VARIANCE MODEL
RT – MAC 2006-05 – Junho 2006, 24 pp.

P. Feofiloff, C.G. Fernandes, C.E. Ferreira and J.C. Pina,
"A NOTE ON JOHNSON, MINKOFF AND PHILLIPS' ALGORITHM FOR THE PRIZE-COLLECTING STEINER TREE PROBLEM"
RT-MAC2006-06 – Setembro 2006, 11 pp.

Julio Michael Stern
DECOUPLING, SPARSITY, RANDOMIZATION, AND OBJECTIVE BAYESIAN INFERENCE
RT-MAC2006-07 – Novembro 2006, 36 pp.

Cristiane Maria Sato, Yoshiharu Kohayakawa
ENTROPIA DE GRAFOS
RT – MAC2006-08 – Dezembro 2006 , 44 pp.

Julio M. Stern
LANGUAGE, METAPHOR AND METAPHYSICS: THE SUBJECTIVE SIDE OF SCIENCE
RT-MAC-2006-09 – Dezembro 2006, 35 pp.

Thiago A. de André and Paulo J. S. Silva
EXACT PENALTIES FOR KKT SYSTEMS ASSOCIATED TO VARIATIONAL INEQUALITIES
RT-MAC-2007-01– Março 2007, 21 pp.

Flávio Soares Correa da Silva, Rogério Panigassi and Carlos Hulot
LEARNING MANAGEMENT SYSTEMS DESIDERATA FOR COMPETITIVE UNIVERSITIES
RT-MAC-2007-02 – Maio 2007, 12 pp.

Alexandre Freire da Silva, Fabio Kon, Alfredo Goldman
THREE ANTI-PRACTICES WHILE TEACHING AGILE METHODS
RT-MAC-2007-03 – Maio 2007, 20pp.

Silvio do Lago Pereira e Leliane Nunes de Barros
PLANEJAMENTO BASEADO EM PROCESSOS DE DECISÃO MARKOVIANOS
RT-MAC-2007-04 – Maio 2007, 17pp.

Silvio do Lago Pereira e Leliane Nunes de Barros
DIAGRAMAS DE DECISÃO BINÁRIA
RT-MAC-2007-05 – Maio 2007, 16pp.

Carlos Alberto de Bragança Pereira and Julio Michael Stern
AN ESSAY ON THE ROLE OF BERNOULLI AND POISSON PROCESSES IN BAYESIAN STATISTICS
RT-MAC-2007-06 – Junho 2007, 39pp.

Flávio Soares Corrêa da Silva
ARGUMENTS IN FAVOR OF A CONTROLLED PLURALITY OF OFFICE FORMATTING STANDARDS
RT-MAC-2007-07 – Junho 2007, 9pp.

Silvio do Lago Pereira, Leliane Nunes de Barros and Fábio Gagliardi Cozman
STRONG PROBABILISTIC PLANNING
RT-MAC-2007-08 – Junho 2007, 25pp.

Silvio do Lago Pereira and Leliane Nunes de Barros
FORMALIZING PLANNING ALGORITHMS FOR TEMPORALLY EXTENDED GOALS
RT-MAC-2007-09 – Junho 2007, 22pp.

Flávio S. Corrêa da Silva
THEPROLOGPLAY: AN INTERACTIVE VIRTUAL ENVIRONMENT FOR ARTIFICIAL INTELLIGENCE AND RELATED FIELDS
RT-MAC-2007-10 – Setembro 2007, 12pp

Vanessa Sabino and Fabio Kon
*Licenças de Software Livre História e Características**
RT-MAC-2009-01 - Março 2009, 40pp.

Alexandre Noma, Ana B. V. Graciano, Roberto M. César Jr., Luis A. Consularo and Isabelle Bloch
Inexact Graph Matching for Segmentation and Recognition of Object Parts
RT-MAC-2009-02 – Março 2009, 31pp.