Quadtree-based Inexact Graph Matching for Image Analysis

Luís Augusto Consularo
UNIMEP - Methodist University of Piracicaba
Piracicaba-SP, Brazil - laconsul@unimep.br
Roberto M. Cesar-Jr.
Department of Computer Science - IME - University of São Paulo
São Paulo, Brazil - cesar@ime.usp.br

Abstract

This paper presents a new method for segmentation and recognition of image objects based on structural pattern recognition. The input image is decomposed into regions through a quadtree algorithm. The decomposed image is represented by an attributed relational graph (ARG) named input graph. *The objects to be recognized are also stored in an ARG named* model graph. *Object segmentation and recognition are accomplished by matching the input graph to the model graph. The possible inexact matches between the two graphs are cliques of the association graph between them. An objective function, to be optimized, is defined for each clique in order to measure how suitable is the match between the graphs. Therefore, recognition is modeled as an optimization procedure. A beam-search algorithm is used to optimize the objective function. Experimental results corroborating the proposed approach are presented.*

1. Introduction

Among the different methods for pattern recognition, the structural approach plays an important role because pattern structure is made explicit in order to accomplish the recognition task. Two different ways are commonly explored to represent the patterns in this approach: (a) primitives and strings (syntactical pattern recognition) [7]; (b) graphs [9]. This paper presents a new method that belongs to the graph-based structural pattern recognition approach. The pattern recognition task is carried out by matching an input (unknown) graph to a model graph. An attributed relational graph (**ARG**) is used to represent the input and the model information. In the present paper, both graphs are automatically extracted from digital images to perform image segmentation and recognition of image structures.

Graph matching is one of the most important problems in structural pattern recognition. Graph isomorphisms are often used and many works deal with the problem of searching for the best isomorphism between two graphs or subgraphs [9]. Nevertheless, in many applications the isomorphism condition is too strong, and the problem is expressed rather as an inexact graph matching problem where a one-to-one mapping between the graphs is not required [12]. Most works on inexact graph matching rely on the optimization of some objective function. This function usually measures the similarity between vertices and between edges of both graphs, as well as between the structure of both graphs. Examples of optimization techniques include combinatorial optimization [4], estimation of distribution algorithms (EDA's) [8], tree search [5], and graph editing [1], to name but a few.

A framework for segmentation and recognition of image structures based on graph morphisms was introduced by Bloch and Perchant some years ago [10]. The research that followed that work lead to the introduction of new algorithms in [3, 2]. In this approach, it is supposed that the image is composed by regions of interest which are to be segmented and recognized. For instance, in face recognition, such image regions are facial features such as eyebrows, pupils, nostrils and lips. The image is represented by an ARG where each vertex is associated to an image region, whereas the edges represent structural information between the regions. The model graph is created from a manual segmentation of a prototype image. On the other hand, the input graph is created from an over-segmentation of the input image using watershed. The segmentation and recognition of the regions of interest in the input image is obtained by matching the input and model graphs. Although this approach has lead to very good results, it presents some drawbacks [2].

The present paper explores an alternative approach that presents some advantages with respect to the aforementioned method [3, 2]. Instead of edge detection-based techniques (i.e. watershed) and manual segmentation to generate the ARG's, quadtrees [6] are used to extract both input and model graphs. The quadtree approach allows a very efficient way for partitioning the images to extract the graphs. Besides, region granularity may be controlled directly from the quadtree structure. Another contribution of the present paper is that, instead of considering complete graphs as in [3, 2], connectivity graphs are adopted. A new dissimilarity measure between graph edges is defined in order to deal with this new graph structure.

This paper is organized as follows. Section 2 presents our method, introducing the necessary notations and definitions, graph attributes, dissimilarity measures and optimization algorithms. Experimental results are shown in Section 3. This paper is concluded with some comments on our ongoing work in Section 4.

2. Inexact Matching: The Clique Search Approach

2.1. Problem Statement

In this work, $\tilde{G} = (N, E)$ denotes a directed graph where N represents the set of vertices of $\tilde{G}$ and $E \subseteq N \times N$ the set of edges. Two vertices a, b of N are adjacent if $(a, b) \in E$. If each vertex of $\tilde{G}$ is adjacent to all others, then $\tilde{G}$ is said to be complete. We define an attributed relational graph as $G = (N, E, \mu, \nu)$, where $\mu : N \rightarrow L_N$ assigns an attribute vector to each vertex of N. Similarly, $\nu : E \rightarrow L_E$ assigns an attribute vector to each edge of E. We typically have $L_N = \mathbb{R}^m$ and $L_E = \mathbb{R}^n$, where m and n are the numbers of vertex and edge attributes, respectively. The vertices and the edges attributes are called object and relational attributes, respectively. Two ARGs $G_1 = (N_1, E_1, \mu_1, \nu_1)$ and $G_2 = (N_2, E_2, \mu_2, \nu_2)$ are used, which will be henceforth referred to as the *input* (i.e. derived from the image) and the *model* graphs, respectively. $|N_1|$ denotes the number of vertices in N_1, while $|E_1|$ denotes the number of edges in E_1. We use a subscript to denote the corresponding graph, e.g. $a_1 \in N_1$ denotes a vertex of G_1, while $(a_1, b_1) \in E_1$ denotes an edge of G_1. We use a superscript to enumerate the nodes of a graph, i.e. $a_1^1, a_1^2, \ldots, a_1^{|N_1|} \in N_1$. Similar notations are used for G_2.

An inexact match between G_1 and G_2 is searched on the association graph between G_1 and G_2. The *association graph* $\tilde{G_A}$ between G_1 and G_2 is defined as the complete graph $\tilde{G_A} = (N_A, E_A)$, with $N_A = N_1 \times N_2$. An inexact match between G_1 and G_2 can be expressed as a sub-graph $\tilde{G_S} = (N_S, E_S)$ of the association graph $\tilde{G_A}$ between G_1 and G_2 with $N_S = \{(a_1, a_2), a_1 \in N_1, a_2 \in N_2\}$ such that $\forall a_1 \in N_1, \exists a_2 \in N_2, (a_1, a_2) \in N_S$ and $\forall (a_1, a_2) \in N_S, \forall (a_1', a_2') \in N_S,\ a_1 = a_1' \Rightarrow a_2 = a_2'$ which guarantees that each vertex of the image graph has exactly one label (vertex of the model graph) and $|N_S| = |N_1|$. $\tilde{G_S}$ is built as a clique of $\tilde{G_A}$.

Figure 1 illustrates these concepts. Starting from the input and model images, the corresponding graphs are generated. In Figure 1, this step is indicated as *quadtree segmentation* because quadtrees are used in this paper, as explained in Section 2.5. The expected inexact match between the input and model graphs are shown as correspondences between nodes of these two graphs (dashed lines in the middle of the figure). The association graph, and the corresponding clique that represents the expected inexact match are shown in the bottom part of Figure 1. Although the association graph is a complete graph, only the edges that belong to the indicated clique are shown for visualization's sake.

Clearly, there are many possible cliques that represent an inexact match between G_1 and G_2, and we need to define an objective function to assess the quality of a given clique and its suitability with respect to each specific application. This criterion should include, additionally to the structural aspects, information on the attributes. In particular, the clique should minimize the dissimilarity between the object attributes of the mapped vertices from G_1 to G_2 and the respective relational attributes associated to the matched edges. The evaluation of the quality of a solution expressed by $\tilde{G_S}$ is performed through an objective function:

$$f_1(\tilde{G_S}) = \frac{\alpha}{|N_S|} \sum_{(a_1,a_2) \in N_S} c_N(a_1, a_2) + \frac{(1-\alpha)}{|E_S|} \sum_{e \in E_S} c_E(e) \tag{1}$$

where $c_N(a_1, a_2)$ is a measure of dissimilarity between the attributes of a_1 and a_2. Similarly, if $e = ((a_1, a_2), (b_1, b_2))$, $c_E(e)$ is a measure of the dissimilarity between edge (a_1, b_1) of the image and edge (a_2, b_2) of the model. Typically, $c_N(a_1, a_2)$ will be defined as a decreasing function of the similarity between vertex attributes. If two vertices a_1 and a_2 have the same attributes (high similarity), then c_N will be very low and the association of a_1 and a_2 will be favored when minimizing f_1. On the other hand, associations between nodes having different attribute values will be penalized. The term depending on edge comparison can be interpreted in a similar way.

2.2. Object and Relational Attributes

The function to be optimized (Equation 1) depends on dissimilarity measures between the graph attributes. The vertices attributes are calculated from quadtree regions (Section 2.5) in the image while relational attributes are based on the spatial disposition of these regions. The adopted attributes for the experiments presented in this paper are:

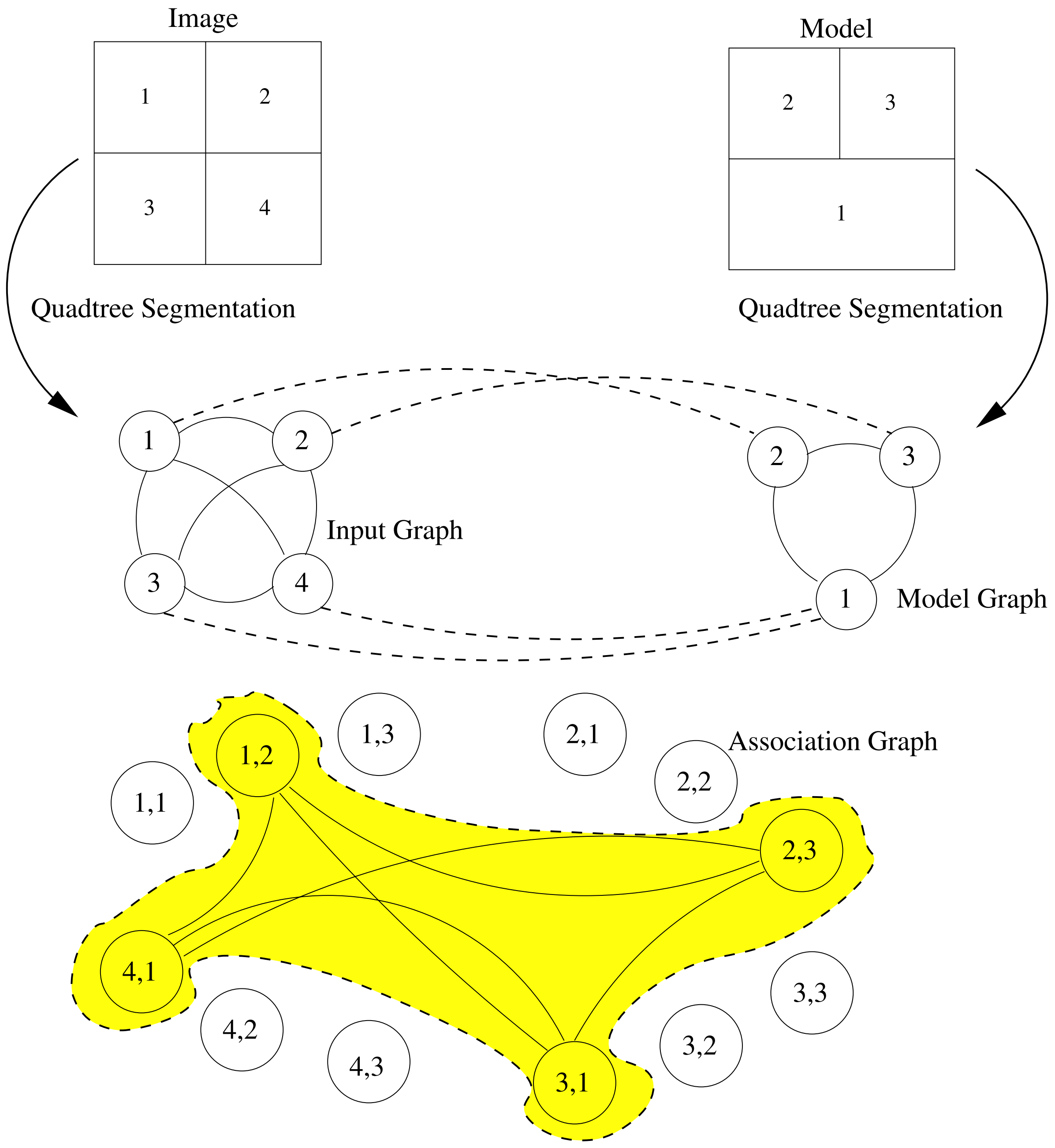

Figure 1. General scheme of the proposed approach: the input and model images are segmented by a quadtree procedure and represented as graphs. The inexact match between these graphs is obtained by searching for a suitable clique in the association graph. The inexact match of this example is indicated as the shaded gray clique of the association graph.

- *Object attributes*: Let $G = (N, E, \mu, \nu)$ be an ARG and let $a \in N$. The set of object attributes $\mu(a)$ is defined as $\mu(a) = (g(a))$, where $g(a)$ denotes the average grey-level of the image region associated to vertex a. $g(a)$ is normalized between 0 and 1 with respect to the minimum and maximum possible grey-levels,
- *Relational attribute*: Let $a, b \in N$ be any two vertices of G, and p_a and p_b be the centroids of the re-

spective corresponding image regions. The relational attribute $\nu(a,b)$ of $(a,b) \in E$ is defined as the vector $\nu(a,b) = (p_b - p_a)/(2d_{max})$, where d_{max} is the largest distance between any two points of the considered image region.

2.3. Dissimilarity Measures

There are two dissimilarity measures c_N and c_E used by the objective function f_1 (Equation 1), associated to vertices and edges of the association graph, respectively. The measure c_N is related to object attributes, while c_E is related to relational attributes. Let (a_1, a_2) denote a vertex of $\tilde{G}_A$, with $a_1 \in N_1$ and $a_2 \in N_2$. The dissimilarity measure $c_N(a_1, a_2)$ is defined as:

$$c_N(a_1, a_2) = |g_1(a_1) - g_2(a_2)|$$

where $g_i(a_i)$ is the object attribute of a vertex $a_i \in G_i$, $i = 1, 2$.

Let e denote an edge of $\tilde{G_A}$, with end-points $(a_1, a_2) \in N_A$, $a_1 \in N_1$ and $a_2 \in N_2$ and $(b_1, b_2) \in N_A$, $b_1 \in N_1$ and $b_2 \in N_2$. We compute the modulus and angular differences between $\nu(a_1, b_1)$ and $\nu(a_2, b_2)$ as

$$\phi_m(e) = |\|\nu(a_1, b_1)\| - \|\nu(a_2, b_2)\||$$

and

$$\phi_a(e) = \frac{|\cos(\theta) - 1|}{2}$$

where θ is the angle between $\nu(a_1, b_1)$ and $\nu(a_2, b_2)$, i.e. $\cos(\theta)$ is calculated as

$$\cos(\theta) = \frac{\nu(a_1, b_1) \cdot \nu(a_2, b_2)}{\|\nu(a_1, b_1)\|\|\nu(a_2, b_2)\|}$$

In order to define the dissimilarity measure $c_E(e)$, we need an auxiliary function:

$$\hat{c}_E(e) = \gamma_E \phi_a(e) + (1 - \gamma_E)\phi_m(e) \qquad (2)$$

where $\nu(a_i, b_i)$ is the relational attribute associated to edge $(a_i, b_i) \in E_i$. The parameter γ_E $(0 \leq \gamma_E \leq 1)$ controls the weights of ϕ_m and ϕ_a. It is important to note that $\nu(a, a) = \vec{0}$. This fact means that, when two vertices in G_1 are mapped onto a single vertex of G_2 by the homomorphism, we have $c_E(e) = \|\nu(a_1, b_1) - \vec{0}\| = \|\nu(a_1, b_1)\|$, which is proportional to the distance between the centroids of the corresponding regions in the over-segmented image (in such cases, we define $\cos(\theta) = 1$). Therefore, $\hat{c}_E$ would give large dissimilarity measures when assigning the same label (i.e. the target vertex in G_2) to distant regions and lower measures when assigning the same label to near regions, which is intuitively desirable in the present application.

If G_1 and G_2 were complete graphs, then $\hat{c}_E$ could be used as the dissimilarity graph to help evaluating the objective function. This was the case in the original paper that introduced the inexact matching approach explored here [2]. Nevertheless, this paper does not assume that G_1 and G_2 complete graphs. Therefore, it is possible that edges in G_1 do not have corresponding edges in G_2 and vice-versa. An illustrative example is shown in Figure 2 where the model is an image composed of two parts labeled 1 and 2. The model graph has two vertices, a_2^1 and a_2^2, and an edge linking them, as shown in the figure. Figure 2 also shows an input image with a possible quadtree segmentation. Four regions are identified as 1, 2, 3 and 4 and the corresponding nodes (denoted a_1^1, a_1^2, a_1^3 and a_1^4) are also shown in the figure, as well as their expected match onto the model graph. Because of the match between the graph vertices, some edge comparisons are expected but not possible because the edges do not actually belong to the graphs. For example, because a_1^1 is matched to a_2^1 and a_1^4 is matched to a_2^4, the dissimilarity measure should be evaluated to compare (a_1^1, a_1^4) and (a_2^1, a_2^2). It is important to note that this comparison is not possible in a straightforward manner since $(a_1^1, a_1^4) \notin E_1$ because region 1 in not connected to region 4.

Let $a_1, b_1 \in N_1$ and $a_2, b_2 \in N_2$ be nodes of G_1 and G_2, respectively. Suppose that a_1 and b_1 is matched to a_2 and b_2, respectively, i.e. $(a_1, a_2), (b_1, b_2) \in N_S$. In this case, the edge (a_1, b_1) would be matched to (a_2, b_2) and the dissimilarity measure between them should be evaluated. However, because the graphs are not complete, it is possible that one or both edges do not actually exist and the dissimilarity measure should properly deal with such situations. The edge dissimilarity measure is then defined as:

$$c_E(e) = \begin{cases} \hat{c}_E(e), & (a_1, b_1) \in E_1, (a_2, b_2) \in E_2 \\ \hat{c}_E(e'), & (a_1, b_1) \notin E_1, (a_2, b_2) \in E_2 \\ \infty, & (a_1, b_1) \in E_1, (a_2, b_2) \notin E_2 \\ 0, & (a_1, b_1) \notin E_1, (a_2, b_2) \notin E_2 \end{cases} \qquad (3)$$

The above dissimilarity measure c_E deals with all possibilities of missing edges. The case where $(a_1, b_1) \notin E_1$ and $(a_2, b_2) \in E_2$ is of particular interest, and arises because of the over-segmentation imposed on the input image and the fact that a connectivity graph is used to generate the ARGs. This is the case discussed above on the example of Figure 2. In such situations, (a_1, b_1) is expected to be compared to (a_2, b_2) and the vertex attributes is created on-the-fly, i.e. by the dissimilarity measure procedure itself. This is indicated in Equation 3 by edge e' instead of e.

2.4. The Optimization Algorithm

It is necessary to optimize the objective function 1 in order to find a suitable inexact match between G_1 and G_2.

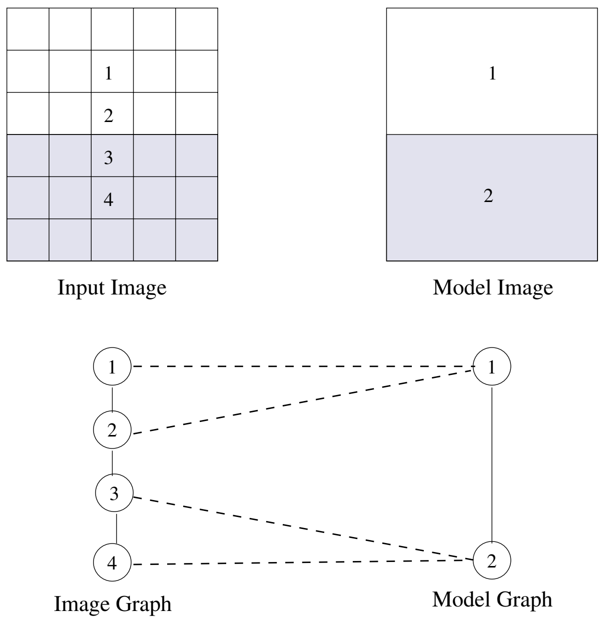

Figure 2. Example of the over-segmentation effect and the problem due to the connectivity graph approach: edges like that between vertices 1 and 4 does not exist because the corresponding regions are not adjacent. Such situation should be properly dealt on-the-fly by the edges dissimilarity measure procedure.

There are many different optimization algorithms that may be used and the reader is referred to [2] for a comparative review. In the experiments reported in this paper, beam-search algorithm has been used. The algorithm starts with empty cliques and incrementally increases a number of different cliques in parallel, evaluating function 1 for each one. The most cheaper clique is chosen and a new vertex is added to it at each iteration. The algorithm stops when a clique that represents a complete solution, as defined above, is found. The different cliques are represented in a search tree, as explained below. This algorithm starts by creating a search tree with each vertex representing a pair (k, l), where k represents the k-th vertex of the input graph (i.e. a_1^k) and l represents the l-th vertex of the model graph (i.e. a_2^l). The initial empty clique is represented by initializing the tree with a dummy root vertex $(0, 0)$ that is expanded in $|N_2|$ sons $(1, 1), (1, 2), \ldots, (1, |N_2|)$. The objective function f_1 is calculated for each son. The cheapest leaf in the tree is taken to be expanded in the next loop. In this first expansion of the tree, the only leaves are the nodes that have just been expanded, but this will not be the case after the second node expansion. It is hence necessary to calculate $c_N(a_1^1, a_2^j)$, $j = 1, \ldots, |N_2|$, i.e. the cost of matching the input graph node a_1^1 to each model graph node a_2^j. This first step does not involve calculating the edge costs c_E since only one node of each graph is being considered so far. The cheapest node is hence expanded in $|N_2|$ sons $(2, 1), (2, 2), \ldots, (2, |N_2|)$, the objective function for each newly born son is calculated and the cheapest tree leaf among all leaves (including nodes left unexpanded in previous steps) is taken to be expanded. It is necessary to calculate $c_N(a_1^2, a_2^j)$, as well as the edge costs $c_E((a_1^1, a_1^2), (a_2^3, a_2^j))$, $j = 1, \ldots, |N_2|$. As new nodes are expanded more terms c_N and c_E are taken into account by the objective function. All matchings between edges must be taken into account once a tree leaf is expanded. The normalization terms $|N_A|$ and $|E_A|$ (Equation 1) must be set properly when calculating the objective function for each node since the number of considered vertices and arcs depends on the depth of the exploded nodes. It is worth noting that all leaves are considered at each step, i.e. tree nodes previously left unexpanded are also candidates to be expanded. The process is repeated until a tree vertex $(|N_1|, l)$ is reached, meaning that all $|N_1|$ vertices in G_1 have been assigned to a vertex in G_2, thus defining a suitable homomorphism between the two graphs. Because taking all possible solutions in parallel is impossible because of the combinatorial explosion, only a fixed maximum number of solutions are considered. This is implemented by defining a priority queue that returns a pointer to the cheapest vertex each time a tree leaf should be chosen to be expanded. Because of the limited the maximum size allowed for the priority queue, once this limit is reached, the more expensive vertices are discarded from the queue. This solution is similar to the beam search algorithm, which saves time and space complexity at the cost of not considering many paths in the tree (and therefore possibly loosing a better solution). Therefore, we have implemented a post-processing step in order to get the solution subsequently improved. Once a solution is reached by the above procedure, the algorithm tracks its path in the tree, from leaf to root, and verifies the price of the solution obtained by changing the node label by the other possible labels for that node. If the obtained solution is cheaper than the previous one, then it is updated with the new label for that node (otherwise, nothing is done). The algorithm then proceeds for the next node in the tree. If the solution has been improved at least once after traversing a leaf-to-root path, this procedure is repeated again from the leaf. Convergence is reached after traversing the leaf-to-root path with no improvements on the solution. More details on this optimization algorithm may be found at [2].

2.5. Graph Formation Through Quadtrees

Quadtree is a recursive data decomposition method mapping the input image onto a DAG (Directed and Acyclic Graph) [11]. Each tree node corresponds to an image region and has at most four children associated to four image subregions. The quadtree root node corresponds to the whole image, which is taken as the initial region. Some *a priori* defined criterion is evaluated to this region and, if not verified, the region is subdivided in four subregions, each one corresponding to the children of the root. This procedure is recursively applied to each subregion, thus leading to the formation of the quadtree (and the corresponding partitioned image). There are different subdivision criteria such as the number of pixels inside each region, the variance and the entropy of the pixel gray-levels within the region. Examples of resulting partitioned images using these criteria are shown in Figure 3.

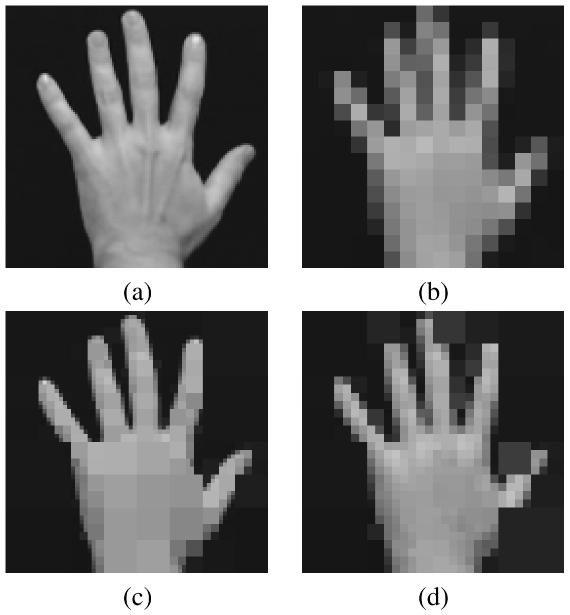

Figure 3. (a) Hand image and some quadtree partitioning results with division criterion based on: (b) the number of pixels (64 pixels); (c) the standard deviation ($\pm$30 gray levels); and (d) the entropy (4 bits).

After quadtree partitioning, each leaf becomes an ARG node of the input or model graphs. The ARG edges are then created to build the ARG as an adjacency graph, as previously explained. Figure 4 illustrates the respective ARGs for the partitioning results shown in Figure 3(b) and (c). Because region adjacency information represents overall image structure in a limited way (for only local structure is represented, i.e. direct neighborhood), we also apply transitive closure of different orders to the original ARG.

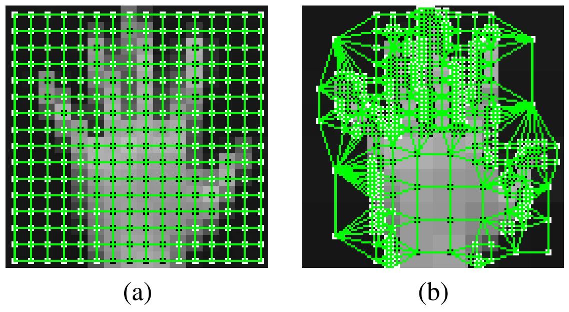

Figure 4. The graphs for respective partitioning results illustrated in Figure 3b and c .

3. Experimental Results

In order to illustrate the introduced methods, results are shown for one synthetic and one real image. Figure 5 shows the result for a synthetic image. The original image is shown in Figure 5(a) while the model graph is shown in Figure 5(b). The quadtree partitioned image and corresponding input graph are shown in Figures 5(c) and (d), respectively. The final segmentation is shown in Figure 5(e) together with the final segmentation image whose quadtree regions hold their respective average gray level pixels (Figure 5(f)). It is worth noting the important role played by the structural information: although many regions present very similar gray level, they have been correctly identified because of the structure information stored by the graph edges.

The importance of structural information is more emphasized in the case of the results shown in Figures 7, 8, and 9. The hand image has been divided into subregions (Figure 6(a)) so that each finger has a different label (Figure 6(b)). Segmenting and recognizing each hand structure could not be carried out based only on gray-level information because this is relatively homogeneous throughout the hand. Nevertheless, as shown in the final segmentation (Figure 7 (a) and (b)), the method is capable of correctly identifying these hand structures, though some errors are present. Two examples based on the graph model in Figure 6b illustrates the structural recognition. In the first example, the Figure 8a shows an open flat hand with closed fingers. The fingers distinction would be a hard task if only the gray level feature would be taken into account. In fact, the gray level got a low weight (α = 0.2). The edge modulus was the structural feature here, since the γ_E parameter was set in 0.04. Although the parts distinction was not completely correct, it is perceptible that the structural pattern provided in the model was preserved. In the second example, the Figure 9a shows a hand with spread fingers. Although it could be supposed an easier task than that for closed fingers, the final segmentation presented some finger distinction mistakes, as pointed in Figure 9d. These mistakes may have been due to poor choices of parameters α (0.1) and γ_E (0.04) or poor models. An improvement for next developments is a parameter searcher which could find low cost homomorphisms. Both examples have been obtained with transitive closure of order 2 applied to the input and model graphs.

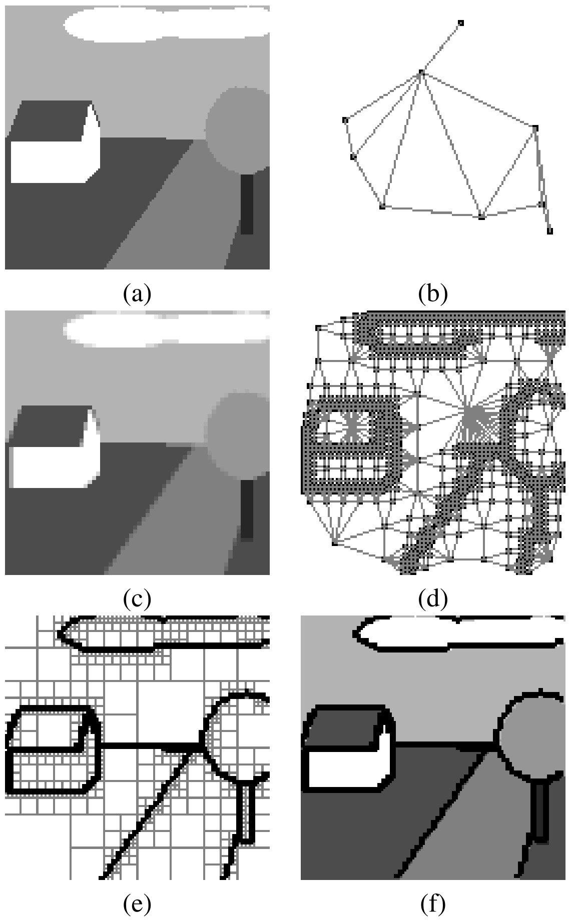

Figure 5. Landscape illustration (a); the graph given as model (b); the gray level quadtree partitioning result (c); the input graph from quadtree (d); the quadtree superposed to the final segmentation result (e); the final segmentation result superposed to the original image (f).

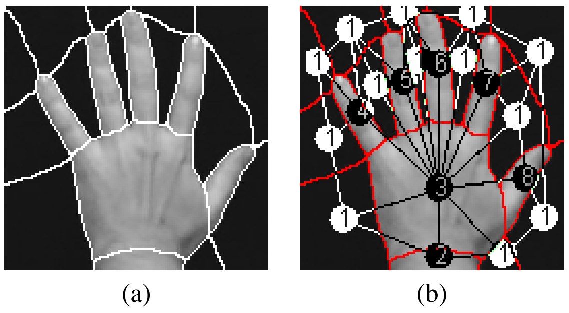

Figure 6. Hand image model (a); the partitioned hand image taken as model (b); the hand parts and their respective labels. Each part corresponds to a node in the model graph whose edges are also illustrated in this picture.

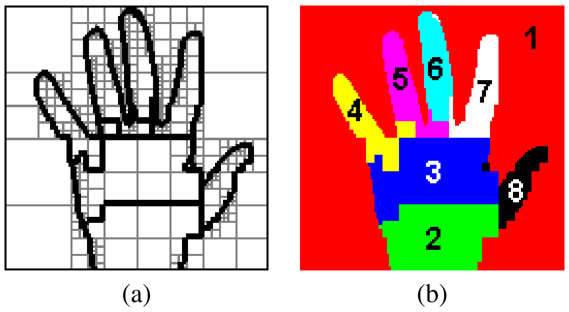

Figure 7. Quadtree regions (Figure 3c) superposed to the final segmentation (a); final segmentation result where each label identifies each hand part (b).

4. Concluding Remarks and Future Work

This paper presented a new method for segmentation and recognition of image structures based on graph matching. It is a structural pattern recognition approach for image segmentation that is solved by an optimization procedure that searches for suitable cliques in the association graph between image and model graphs. The ARGs are generated from a quadtree decomposition of the images in an efficient and elegant way. Experimental results corroborating the introduced approach were presented. Our ongoing work aims at speeding the search algorithm to allow real-time applications, as well as improving the quality of the final solution. We are also working on the extension of the proposed approach to deal with video sequences. We would like to explore the solution found in previous frames to guide and speed the segmentation in the current frame.

IEEE COMPUTER SOCIETY

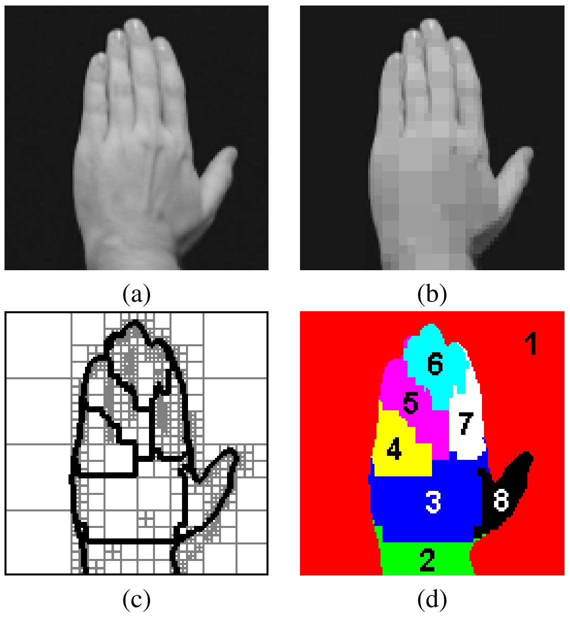

Figure 8. Open flat hand with closed fingers original image (a); gray level partitioning result (st. dev. ±10 gray levels) (b); quadtree regions superposed to the final segmentation (c); final segmentation result where each label identifies each hand part (d).

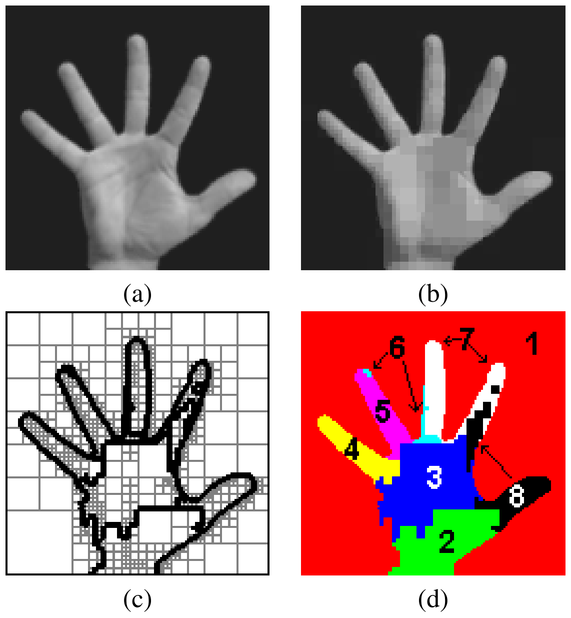

Figure 9. Spread fingers hand original image (a); gray level partitioning result (st. dev. ±12.0 gray levels) (b); quadtree regions superposed to the final segmentation (c); final segmentation result where each label identifies each hand part (d).

Acknowledgements

R. Cesar-Jr. is grateful to FAPESP (99/12765-2) and to CNPq (300722/98-2 and 474596/2004-4).

References

[1] H. Bunke. Error Correcting Graph Matching: On the Influence of the Underlying Cost Function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):917–922, 1999.

[2] 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*, 2005 (accepted).

[3] O. Colliot, A. Tuzikov, R. Cesar-Jr., and I. Bloch. Approximate reflectional symmetries of fuzzy objects with an application in model-based object recognition. *Fuzzy Sets and Systems*, 147(1):141–163, 2004.

[4] A. D. J. Cross and E. R. Hancock. Convergence of a Hill Climbing Genetic Algorithm for Graph Matching. In *Lecture notes in Computer Science*, volume 1654, 1999.

[5] A. Deruyver and Y. Hodé. Constraint Satisfaction Problem with Bilevel Constraint: Application to Interpretation of Over-segmented Images. *Artificial intelligence*, 93:321–335, 1997.

[6] D.H.Ballard and C.M.Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[7] K. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[8] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.

[9] B. T. Messmer and H. Bunke. Efficient subgraph isomorphism detection: A decomposition approach. *IEEE Transactions on Knowledge and Data Engineering*, 12(2):307–323, 2000.

[10] A. Perchant and I. Bloch. A New Definition for Fuzzy Attributed Graph Homomorphism with Application to Structural Shape Recognition in Brain Imaging. In *IMTC'99, 16th IEEE Instrumentation and Measurement Technology Conference*, pages 1801–1806, Venice, Italy, 1999.

[11] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):2049–2051, 1984.

[12] R. Wilson and E. Hancock. A Bayesian compatibility model for graph matching. *Pattern Recognition Letters*, 17(3):263–276, 1996.

IEEE COMPUTER SOCIETY