Graph Edit Distance for Active Graph Matching in Content Based Retrieval Applications

Stefano Berretti*, Alberto Del Bimbo and Pietro Pala

Dipartimento di Sistemi e Informatica, Universita' degli Studi di Firenze, Italy

Abstract: Application of multimedia technologies to visual data, like still images and videos, is receiving an increasing attention especially for the large number of potential innovative solutions which are expected to emerge in the next years. In this context, techniques for retrieval by visual similarity are expected to boost the interest of users through the definition of novel paradigms to access digital repositories of visual data. In this paper, we define a novel model for active graph matching and describe its application to content based retrieval of images. The proposed solution fits with the class of edit distance based techniques and supports active node merging during the graph matching process. A theoretical analysis of the computational complexity of the proposed solution is presented and a prototype system is experimented on the images of two sample image collections.

Keywords: Content based image retrieval, active graph matching, edit operations.

INTRODUCTION

In recent years, digital technologies have opened the way to novel and promising solutions for supporting representation and access to visual data, like still images or videos. In particular, the advent of multimedia technologies and the ever increasing diffusion of wired and wireless telecommunication opens the way to new paradigms to access digital repositories of visual data.

Thanks to the availability of devices, tools and formats for acquisition and representation in digital form of visual material, large repositories of digitalized photos, videos or 3D models are rapidly growing. However, in order to exploit the valuable assets contained in these ever growing collections, some tool should be available to support users in the process of finding information out of these data.

So far, several modeling approaches have been defined to support content based access to image repositories. In general, these include several components: processing each database image so as to extract a descriptor of its content; organizing image content descriptors into an efficient index structure; defining a distance function for measuring the (dis)similarity between user queries and image content descriptors; and embodying an effective query metaphor into a visual interface [1-3].

In particular, description of visual patterns and computation of their similarity on a perceptually motivated basis is a key issue for content based image retrieval applications. Among all data structures that can be used to represent features of visual patterns and their relationships, graphs are one of the most versatile and powerful.

Modeling visual data using graphs, typically requires segmentation of visual data into parts and use of graph nodes to represent features of parts and graph edges to represent their relationships. Nodes and edges are associated (labeled) with additional information (descriptors) capturing salient visual features of parts and of their relationships, respectively. Descriptors can be either symbolic or numeric. In the former case, descriptors are symbolic labels identifying one or more predefined classes to which the part belongs to. In the latter case, descriptors are in the form of feature vectors capturing through numeric values, prominent features of each part (such as, color, texture, shape).

Determining the similarity between two graphs is usually referred to as *graph matching* and ultimately corresponds to the problem of identifying a correspondence between nodes and edges of the two graphs. For a general review of methods and techniques related to graph matching, the interested reader can refer to [4,5].

Broadly speaking, graph matching techniques can be grouped into three distinct classes: (*i*) graph isomorphism; (*ii*) subgraph isomorphism; (*iii*) error tolerant subgraph isomorphism. Each technique can be further classified based on the method adopted to find a solution to the isomorphism problem. Under this perspective, two different approaches can be distinguished: *stochastic* and *deterministic*. In the former, matching is achieved by means of a stochastic relaxation process that guarantees identification of the optimal solution only as the limit of an increasing number of iterations. Generally, if the number of computation steps is polynomially bounded, only a suboptimal solution is guaranteed. In the latter, matching is accomplished through exploration of the solution space, that is the space defined by all possible combinations of graph node associations until the optimal solution is found. This is a NP-complete problem, though techniques for avoiding exhaustive exploration of the solution space have been proposed. For instance, the most common approach is based on tree search using the A* algorithm [6].

Early approaches to graph matching addressed the problem of comparison of two graphs by means of finding a

*Address correspondence to this author at the Dipartimento di Sistemi e Informatica, Universita' degli Studi di Firenze, via S. Marta 3, 50139, Firenze, Italy; Tel: +39 055 4796540/415; Fax: +39 055 4796363; E-mail: berretti@dsi.unifi.it; s.berretti@gmail.com

1874-0618/07

graph isomorphism, that is a bijective function that associates with every node/edge in the first graph one node/edge in the second one. It is assumed that the two graphs have the same number of nodes/edges, the same labels and the same edge structure. However, this approach to graph matching poses several constraints that are seldom satisfied in the practice. This is mainly related to the fact that the process of eliciting graphs from visual data is usually affected by noise and errors of various types. This is particularly true in the case visual data represents generic images where the assumption of smoothness of color changes is rarely verified. As a result, two graphs built on the same data set using two different, though equivalent, processes may be different and thus do not match through an exact graph isomorphism.

As an example, the segmentation of an image can be considered. This is a process by which an image is partitioned into disjoint regions according to some homogeneity criteria (usually homogeneity of color and/or texture). Results of the segmentation process find a natural representation through a graph structure: nodes correspond to regions and are labeled with a description of region features; edges encode information about relationships between regions (e.g., region adjacency). A multitude of approaches to image segmentation have been proposed (for example, region growing, split and merge, clustering in the feature space) and even within the same class of approaches, different settings of parameters almost always result in different segmentations. These differences exacerbate if the input images are different to begin with. For instance, this can be the case of images representing the same scene from slightly different viewpoints or under different illumination conditions. In all these situations, graph matching by exact graph isomorphism has proved to be inadequate.

A first solution for matching graphs that are not identical is to address *subgraph* rather than graph isomorphism: a subgraph isomorphism between two graphs being an isomorphism between one of the two graphs and a subgraph of the second one. In classical subgraph isomorphism methods [7,8], the best match is found using the A* search method [6,9], or its improvements aiming to anticipate the discarding of unfeasible solutions through look-ahead estimations [10]. In [8], a system is presented that exploits subgraph matching to support retrieval of graphic logos from a database of color images taken from advertisements and magazines. The system is invariant with respect to translation and scale of images and can accurately locate a query logo in a target image. However, its ability to retrieve similar objects in addition to identical ones is very limited.

In order to cope effectively with comparison of similar, but not identical objects, the concept of *inexact* matching has been considered [7,11-14]. Only the adoption of inexact graph matching techniques can enable accurate and effective measure of similarity between objects that are visually similar but not identical. It should be considered that the measure of similarity is not only intended to model differences between two objects in terms of translation and scale. Rather, the ability of a system to capture the similarity between two objects is useful for comparison of objects that are different to begin with, or that become different due to noise, distortions or image processing operations (e.g., segmentation).

In [15], image content is represented through region adjacency graphs (RAG) and the similarity between two images is evaluated using a recursive neural network. However, the recursive neural network is not able to manage region adjacency graphs directly. Rather, each RAG has to be converted into a directed order acyclic graph in order to be processed by the network. This conversion is typically associated with a loss of information that penalizes the effectiveness of the similarity measure.

In [16], Stochastic Petri Net graphs are used to represent shape and color content of images. Graph matching is accomplished through a deterministic approach, by exploring all possible node associations in the solution space.

In [17], several graph-based techniques are experimented for object recognition tasks. Exact as well as approximate algorithms are used to compute the similarity between two graph representations.

In [18] and [19], color adjacency graphs are used to capture prominent chromatic features of imaged objects. A deterministic subgraph matching technique is used to locate a template colored object within a generic image.

Though embodying the concept of relational inexactness, all these techniques feature a common trait, in that they exploit a static representation of the elements under match (nodes and edges). Differently, the matching task should be considered as an active process by which two objects are allowed to change in order to find the best correspondence between their constituting elements.

A few approaches have been presented in the past to model graph matching as an active process. In [20,21], a model is proposed to consider raw data, that originated the graphs, during the matching process. Raw data and its relational abstraction (the graphs) are never decoupled. Rather, interpretation of raw data can change during the graph matching process so as to accommodate for a better matching of the two graphs. However, the applicability of this approach for content based retrieval of information from large repositories is highly unpractical since it requires a continuous interaction between raw data and their relational abstractions during the matching process.

In [22], a method is presented to deal with subgraph matching by operating at a structural level: a relational distance metric is defined to accommodate for the effects of noise or segmentation errors by inserting dummy nodes into the graphs. A similar approach develops on the notion of *graph edit distance* [23]. This is defined with respect to a set of edit operations, namely, *delete*, *insert*, *substitute*, that can be applied to alter the first graph until a subgraph isomorphism to the second graph exists. Each edit operation is associated with a cost. In this way, the overall effect of all the edit operations that are applied to one graph can be quantified through an overall cost that sums up the costs associated with individual operations. The higher the cost of the operations that are applied the more dissimilar the two graphs.

In this paper, a novel solution is proposed for error tolerant graph matching. The solution belongs to the class of edit distance based techniques. In particular, the original edit distance based framework is extended so as to account for a new operator to support *node merging* during the matching process. It should be considered that, in the context of edit distance based techniques, node merging is not equivalent to a sequence of node deletion and insertion. Indeed, the graph that results from the application of a node merging operation can also be obtained through the application of appropriate deletion and insertion operations. However, the cost that is associated to the two transformations is not the same: in the general case, the cost of one operation (merging) is less that the cost of two operations (deletion and insertion). Furthermore, the merging operation should be associated with a much lower cost than deletion and insertion, as the former condenses in one node the information scattered in two or more nodes, while the latter two either remove or add new information to the graph.

Techniques for graph matching based on node splitting and merging have been previously used for object tracking [24], and for image content description [25-27]. However, in the proposed solution, instead of applying edit operations only to one of the two graphs, graph matching is achieved by editing both graphs. In this way, the application of edit operations is equivalent to a process by which the two graphs evolve toward a common graph structure.

The paper is organized as follows: in the next Section the graph matching problem is formally stated with reference to the new operator of node merging. The algorithmic implementation and an estimate of its computational complexity are discussed in the implementation Section. Experimental results are reported in the results Section. Finally, conclusions and current and future research directions are drawn in the last Section of the paper.

GRAPH MATCHING BY NODE MERGING

In the following, a graph is represented, according to the same formalism used in [25]. In particular, a graph g is a tuple, $g = (V, E, \alpha, \beta)$, being V a set of nodes, $E \subseteq V \times V$ a set of edges, $\alpha: V \rightarrow L_V$ a node labeling function, and $\beta: E \rightarrow L_E$ an edge labeling function. L_V and L_E are the set of nodes and edge *labels*. The term *label* refers here to a generic descriptor representing the information (either in symbolic or numeric form) associated with the node/edge. In this sense, a label may be a symbolic descriptor as well as a feature vector retaining prominent characteristics of the part of an object associated with the node/edge.

Given two graphs, $g_1 = (V_1, E_1, \alpha_1, \beta_1)$, and $g_2 = (V_2, E_2, \alpha_2, \beta_2)$, a graph isomorphism is a *bijective function* $f: V_1 \rightarrow V_2$ that preserves all edges and labels, that is:

$$D_v(\alpha_1(x), \alpha_2(f(x))) = 0$$

$$D_e(\beta_1(E(x,y)), \beta_2(E(f(x),f(y)))) = 0$$

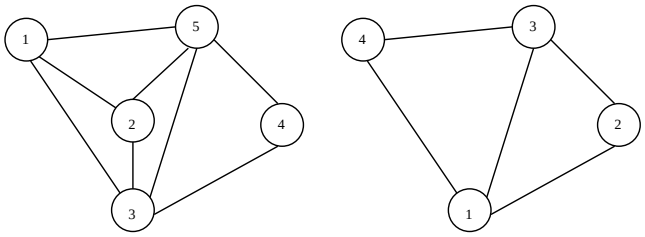
being D_v and D_e two distance measures defined in L_V and L_E, respectively.

Given a generic graph $g = (V, E, \alpha, \beta)$, a subset $W \subseteq V$ of its nodes is *connected* if for every pair of nodes $i, j \in W$ there is a path in W leading from i to j.

In traditional graph matching based on the edit distance, a set of edit operations is defined to transform one graph into another one. The set of edit operations is composed of *deletion*, *insertion* and *substitution*, the effect of this latter operation being the change of the value of a node or edge label.

More precisely, the node deletion operator removes one graph node; the node insertion operator inserts a new node into the graph; the node substitution operator changes the label value associated with one graph node. Similarly, the edge deletion operator removes one graph edge; the edge insertion operator inserts a new edge between two graph nodes; the edge substitution operator changes the label value associated with one graph edge.

Each edit operator δ_i is associated with an edit cost $EC(\delta_i)$, so that the overall change due to the application of the sequence of edit operators $\Delta = (\delta_1, \delta_2, ..., \delta_k)$ can be quantified.



G_1 G_2

	α_1
1	A
2	B
3	C
4	D
5	E

	β_1
E(1,2)	A
E(1,3)	A
E(1,5)	A
E(2,3)	B
E(2,5)	B
E(3,4)	C
E(3,5)	C
E(4,5)	D

	α_2
1	C
2	D
3	E
4	A

	β_2
E(1,2)	c
E(1,3)	c
E(1,4)	a
E(2,3)	d
E(3,4)	a

Edit sequence
δ_1: delete node 2 in G_1
δ_2: delete edge E(1,2) in G_1
δ_3: delete edge E(2,3) in G_1
δ_4: delete edge E(2,5) in G_1

f values
f(1) = 4
f(3) = 1
f(4) = 2
f(5) = 3

Fig. (1). Comparison of two similar graphs using basic editing operators.

As an example, in Fig. (**1**) two graphs are shown, together with the sequence of edit operations that can be applied to the first graph to match the second one. The dissimilarity between the two graphs is evaluated as the cumulative cost of all the edit operations that are applied, that is, $EC(\Delta) = \sum_{i=1}^{k} EC(\delta_i)$.

In the proposed solution, this set of basic edit operators is augmented by the *node merging* operator. This replaces a set of connected nodes with one node. The label value that is assigned to the new node is computed based on the values of the labels of the original nodes. In this way, information about the original nodes is not lost, rather it is recombined in the new node.

In Fig. (**2**), an example of graph matching by node merging is shown. The two graphs in Fig. (**2**), are derived by graphs in Fig. (**1**), by replacing symbolic descriptors with numeric descriptors (given by α_i and β_i values). In particular, the two graphs in Fig. (**2**) represent the output of the segmentation of two images, node and edge descriptors representing intra-region (normalized region area and region mean color in the RGB space) and inter-region (region adjacency) features, respectively. In this example, the edge labels are all equal to 1, in that they only capture the fact that two regions in the image are adjecent each other (and the corresponding nodes in the graph are also connected each other).

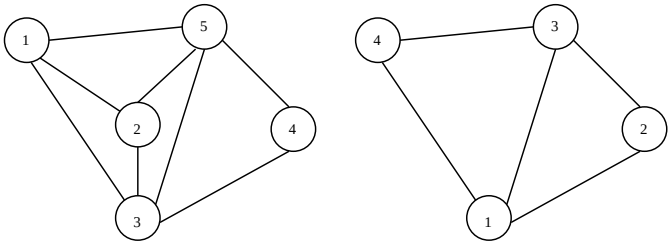
The cost associated to each operator reflects the amount of alteration it introduces on one graph. With reference to the example shown in Fig. (**2**), we assume that region descriptors (i.e., node labels) are in the form $\alpha(i) = (s_i, r_i, g_i, b_i)$, being s_i, r_i, g_i, b_i, respectively, the area of the region normalized to the area of the image, and the normalized color components in the RGB color space of the average color of the region. According to this, the edit costs $EC(\delta_i)$ of edit operators δ_i, can be evaluated as follows:

- *node delete*: distance $d_d^{(n)}$ between descriptors of the deleted region and the *null* region, that is, $EC(\delta) = d_d^{(n)}((s_i,r_i,g_i,b_i),(\varnothing,\varnothing,\varnothing,\varnothing)) = s_i$;
- *node insert*: distance $d_i^{(n)}$ between descriptors of the newly created region and the *null* region, that is, $EC(\delta) = d_i^{(n)}((s_i,r_i,g_i,b_i),(\varnothing,\varnothing,\varnothing,\varnothing)) = s_i$;
- *node substitute*: distance $d_s^{(n)}$ between descriptors of the original region and the substituted region, that is, $EC(\delta) = d_s^{(n)}((s_i,r_i,g_i,b_i),(s_j,r_j,g_j,b_j)) = ((s_i-s_j)^2+1/3((r_i-r_j)^2+(g_i-g_j)^2+(b_i-b_j)^2))^{1/2}$.

The contribution to the distance given by the difference in the three color components is multiplied by the constant *1/3*. This aims to balance the contribution to the distance given by the difference in the area, and by the differences in the color of the regions;

- *node merge*: distance $d_m^{(n)}$ between descriptors of the two regions to be merged, that is, $EC(\delta) = d_m^{(n)}((s_i,r_i,g_i,b_i),(s_j,r_j,g_j,b_j)) = (1/3((r_i-r_j)^2+(g_i-g_j)^2+(b_i-b_j)^2))^{1/2}$.

The cost of merging two nodes does not depend on the area of the two merging regions in that we assume there is no cost to pay in merging regions of different area. The only cost is related to the difference in the color of the regions. The constant 1/3 guarantees that the cost of merging is normalized in the [0,1] interval;



G_1 G_2

	α_1
1	(0.2, 1.0, 0.8, 0.4)
2	(0.2, 0.9, 0.6, 0.3)
3	(0.3, 0.9, 0.5, 0.3)
4	(0.1, 0.6, 0.3, 0.2)
5	(0.2, 0.1, 0.0, 0.0)

	β_1
E(1,2)	1
E(1,3)	1
E(1,5)	1
E(2,3)	1
E(2,5)	1
E(3,4)	1
E(3,5)	1
E(4,5)	1

	α_2
1	(0.5, 0.9, 0.5, 0.3)
2	(0.1, 0.6, 0.3, 0.2)
3	(0.2, 0.1, 0.0, 0.0)
4	(0.2, 1.0, 0.8, 0.4)

	β_2
E(1,2)	1
E(1,3)	1
E(1,4)	1
E(2,3)	1
E(3,4)	1

Edit sequence
δ_1: *Merge nodes 2 and 3 in* G_1
δ_2: *Merge edges E(1,2) and E(1,3) in* G_1
δ_3: *Merge edges E(2,5) and E(3,5) in* G_1
δ_4: *substitute description of node 3 in* G_1

f values
f(1) = 4
f(3) = 1
f(4) = 2
f(5) = 3

Fig. (2). Comparison of two similar graphs by using extended editing operators.

- *edge delete*: distance $d_d^{(e)}$ between descriptors of the deleted edge and the *null* edge, that is, $EC(\delta) = \gamma * d_d^{(e)}(\beta(i),\varnothing) = \gamma * \beta_i$;
- *edge insert*: distance $d_i^{(e)}$ between descriptors of the newly created edge and the *null* edge, that is, $EC(\delta) = \gamma * d_i^{(e)}(\beta(i),\varnothing) = \gamma * \beta_i$;
- *edge substitute*: distance $d_s^{(e)}$ between descriptors of the original edge and the substituted edge, that is, $EC(\delta) = \gamma * d_s^{(e)}(\beta(i),\beta_j) = \gamma * ||\beta_i - \beta_j||$;

- *edge merge*: distance $d_m^{(e)}$ between descriptors of the two edges to be merged, that is, $EC(\delta) = \gamma * d_m^{(e)}(\beta(i), \beta_j) = \gamma * ||\beta_i - \beta_j||$;

being γ a weight coefficient used to balance the relative relevance of operators acting on nodes and edges.

It should be noticed that values of distances $d_d^{(n)}$, $d_i^{(n)}$, $d_s^{(n)}$, $d_m^{(n)}$, are not necessarily the same. For instance, deleting one node is associated with a cost that amounts to the relevance of the region corresponding to that node. This relevance is estimated with the area of the region normalized to the area of the image. The same consideration holds for inserting a new node. Differently, the cost of substituting one node descriptor depends not only on its area, but also on its average color. Finally, the cost of merging two nodes depends only on how much different the average colors of the two regions are, regardless of their size. If merging occurs between two regions with very similar colors, there is almost no penalty in terms of editing costs.

Table **1** summarizes the cost of matching the two graphs in Fig. (**2**) using the merging operator (see Table **1a**), and not using the merging operator (see Table **1b**).

Table 1. Cost of Matching the Two Graphs in Fig. (2): (a) Using the Merging Operator; (b) Not Using the Merging Operator

(**a**)

Edit Operation	Operation Cost	Cumulative Cost
δ_1: *Merge node 2 and 3 in G1 resulting in (0.5,0.9,0.55,0.3)*	0.058	0.058
δ_2: *Merge edges E(1,2) and E(1,3) in* G_1	0.0	0.058
δ_3: *Merge edges E(2,5) and E(3,5) in* G_1	0.0	0.058
δ_4: *Substitute description of node 3 in* G_1 *with (0.5,0.9,0.55,0.3)*	0.029	0.087

(**b**)

Edit Operation	Operation Cost	Cumulative Cost
δ_1: *Delete node 2 in G1*	0.2	0.2
δ_2: *delete edge E(1,2) in* G_1	γ	$0.2 + \gamma$
δ_3: *delete edge E(2,3) in* G_1	γ	$0.2 + 2\gamma$
δ_4: *delete edge E(2,5) in* G_1	γ	$0.2 + 3\gamma$
δ_5: *substitute description of node 3 in* G_1 *with (0.5,0.9,0.55,0.3)*	0.2	$0.4 + 3\gamma$

IMAGE MATCHING

In order to account for node merging, image matching is organized as an iterative process that performs the following actions at each iteration step:

- For each graph node, the set of *compatible nodes* is computed. This set is defined as the set of nodes that can be merged with the current node (for instance, because they have very similar colors and/or textures);
- For each graph node, the set of *virtual nodes* is computed. This is defined as the set of nodes originated by merging the current node with one or more adjacent nodes;
- Nodes of the two graphs are compared so as to decide which combinations of nodes should be actually fused.

Detailed description of each iteration step is provided in the following Section.

COMPATIBLE NODES

Each node is associated with a label that represents information about node features. We assume that a dissimilarity metric is defined that enables comparison of node labels so as to derive the dissimilarity between two nodes. In our case, this dissimilarity metric is in the form of a *weighted Euclidean* distance.

Let $g = (V, E, \alpha, \beta)$ be a graph, $i, j \in V$ two nodes and $\alpha(i), \alpha(j) \in L_V \subseteq R^n$ their labels. The dissimilarity between nodes i and j is measured as:

$$D_\omega(i,j) = [\alpha(i)-\alpha(j)]' diag(\omega_1, \omega_2, \ldots, \omega_v)\,[\alpha(i)-\alpha(j)] \quad (1)$$

being $(\omega_1, \omega_2, \ldots, \omega_v)$ a set of weights used to balance the relative relevance of node features.

It should be noticed that this definition of node dissimilarity is not restricted to nodes of the same graph. In fact, the dissimilarity between nodes of two distinct graphs can be computed provided that they adopt homogeneous labels (feature vectors).

Given a generic graph $g = (V, E, \alpha, \beta)$, two nodes $i, j \in V$ are *compatible nodes* if both the following conditions hold: i and j are adjacent nodes; $D_\omega(i, j) < \tau_c$; being τ_c a fixed node compatibility threshold (in the experimental results, this threshold was set to $\tau_c = 0.3$).

For a node i of a graph, the set of compatible nodes can be defined. This set is indicated as $C(i)$ and is composed of the current graph node and all its compatible nodes. For node i, the set of compatible nodes $C(i)$ is used to derive the set of *virtual nodes*. In general, a virtual node results from the combination (merging) of one node with one or more compatible nodes. Given a node i, let $N+1$ be the cardinality of the set $C(i)$ (i.e., the number N of node compatible with the node i, plus the node i itself).

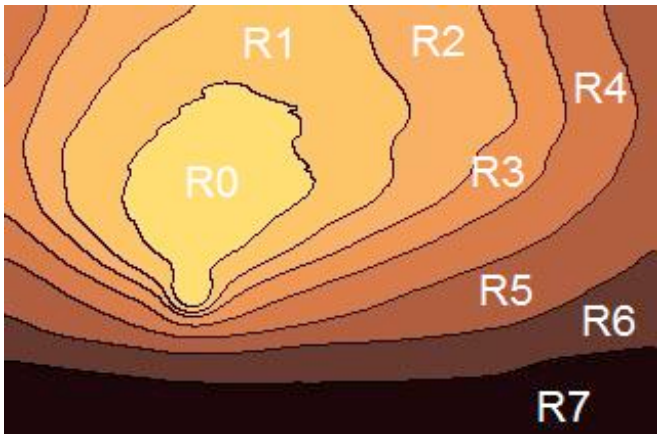
Let $C^N_k(i)$ be the set of all k-combinations of the elements of $C(i)$ that include node i. The set of virtual node combinations $VN(i)$, for node i is defined as:

$$VN(i) = \{ C^N_k(i) \}^N_{k=0}$$

being $C^N_0(i) = i$. Since the cardinality of $C^N_k(i)$ is the binomial coefficient $\binom{N}{k}$, the cardinality of $VN(i)$ is the sum $\sum^N_{k=0} \binom{N}{k} = 2^N$.

(**a**)

(**b**)



(**c**)

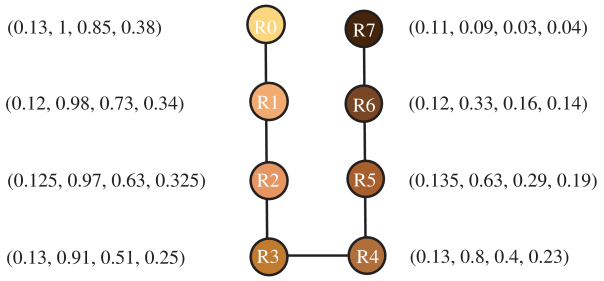


Fig. (3). (**a**) A sample image. (**b**) The segmented image obtained applying a color based segmentation algorithm to the image in (**a**). (**c**) The segmented image in (**b**) is represented through a graph, where: nodes correspond to the segmented regions and are labeled with the region area (in percentage with respect to the overall area of the image), and the average color of the region (in the RGB color space); edges account for the adjacency between regions in the image plane.

Indeed, each element of $VN(i)$ is a node. In particular, it can be the node i, or any node obtained by merging node i with one or more compatible nodes. Given a virtual node ψ, the *node originating* function $\Omega(\psi)$ returns the set of nodes that were merged to originate it. When two or more nodes are merged to create a new node, some criteria must be followed in order to compute the label to assign to the new node. New nodes should inherit information from the nodes from which they originate. This is accomplished through the definition of a *feature propagation* function F_{fp}: $R^n \times R^n \rightarrow R^n$. Given a pair of nodes and their labels, this function outputs the label that should be associated with the node originating from the merge of the first two.

The way in which the feature propagation function acts on the pair of feature vectors associated with the nodes to be merged depends on what is represented in the elements of the feature vector. In general, the feature propagation function may entail ad-hoc knowledge about rules to be applied for each element of the feature vector.

A sample case is shown in Fig. (**3**). The feature vector of one node combines information about area and color. In particular, the feature vector of node i is in the form $f_i = (f_1^i, f_2^i, f_3^i, f_4^i) \in R^4$, being f_1^i the area of the region represented by node i, and f_2^i, f_3^i, f_4^i the three components of its color. In order to combine the feature vectors of N nodes, the feature propagation function applies a *summation rule* for the first element of the feature vector, and a *mean rule* for the last three elements. That is:

$$F_{fp}\left(\bigcup_{j=1}^{N} f_j\right) = \left(\sum_{j=1}^{N} f_1^j, 1/N \sum_{j=1}^{N} f_2^j, 1/N \sum_{j=1}^{N} f_3^j, 1/N \sum_{j=1}^{N} f_4^j\right)$$

In the above equation, normalized values of the area and of the color components of the regions are used.

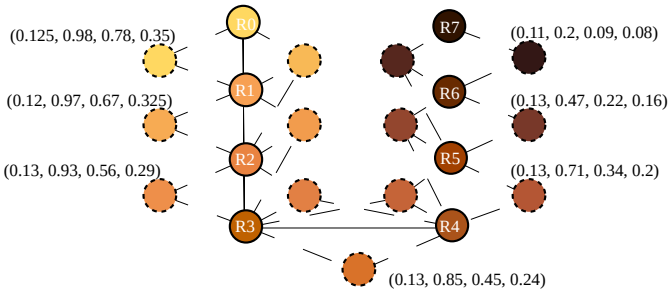


Fig. (4). Virtual nodes originated from the graph shown in Fig. (**3c**). For the readability of the graph, feature vectors associated with the original graph nodes, and with the virtual nodes obtained as combination of three nodes are not displayed. Each virtual node is evidenced through a dotted contour and is connected to its originating nodes through dotted edges.

Virtual nodes that are originated from nodes of the graph in Fig. (**3**) are shown in Fig. (**4**). Each virtual node is evidenced through a dotted contour and is connected to its originating nodes through dotted edges.

VIRTUAL NODES COMPARISON

When two graphs have to be compared, for each graph node the virtual node combination set is computed. Virtual

node combination sets of the two graphs are compared so as to determine the best node correspondences. In order to find the best node correspondences both actual-to-actual, actual-to-virtual and virtual-to-virtual node comparisons are explored.

In this way, the purpose of selecting the best node correspondences is twofold: on the one hand, it favorites aggregation of nodes that find a counterpart in both graphs; on the other, it favorites aggregation of nodes that are surrounded by similar nodes (aggregations of nodes).

Nodes are not compared using the dissimilarity function defined in Eq.(1). Rather, a *context dissimilarity* function $D_{cd}(.,.)$ is defined for this purpose. Given two nodes i and j, the value of $D_{cd}(i,j)$ accounts not only for the dissimilarity of the feature vectors associated with nodes i and j, but also for the dissimilarity of their adjacent nodes:

$$D_{cd}(i,j) = \frac{\alpha \cdot \left|A_i - A_j\right| + \beta \cdot d_{col} + \gamma \cdot \left[1 - \frac{\#Adj_{comp_{i,j}}}{\max Adj}\right]}{\alpha + \beta + \gamma}$$

where: A_i and A_j are the area of the two regions; $d_{col}=(((R_i-R_j)^2+(G_i-G_j)^2+(B_i-B_j)^2)/3)^{1/2}$ is the distance between the average colors of regions i and j; $max\ Adj = max\ (Adj^i, Adj^j)$ is the maximum between the number of adiacent regions for regions i and j; and α, β, γ are weights of the individual distance components.

According to this equation, the difference in area and color between the regions is accounted as well as the number of adjacent compatible regions.

Comparison of two virtual node combination sets results in the identification of two nodes (belonging to the first and second graphs, respectively) that correspond to each other. Each one of these two nodes can be either an actual node or a virtual node. In the case the node is a virtual node, it becomes an actual node and replaces, in the original graph, all the nodes that originated it.

ALGORITHM IMPLEMENTATION

The proposed solution to graph matching, requires comparison not only of the nodes in two graphs, but also of the virtual nodes that originate from possible merging occurring between adjacent nodes in each graph involved in the comparison. Assuming the computational complexity of a traditional subgraph matching problem to be $O(m_1^{m2})$ [28] (being m_1 and m_2 the number of nodes of the two graphs), the complexity of the proposed solution scales to $O((m_1*\xi)^{(m2*\xi)})$ being ξ the average number of virtual nodes originated from each actual node.

In order to be effectively used for graph comparison, the complexity of the proposed solution needs to be reduced. This is accomplished by adopting a greedy approach for node comparison. According to this strategy, graph comparison is accomplished through an iterative matching process. At each iteration step, the following actions are performed:

- One node i in the first graph is selected and the most similar node j in the second one is found.
- *VN(i)* and *VN(j)*, that is, the virtual nodes originated by nodes i and j are computed.
- The elements of *VN(i)* and *VN(j)* are compared to find the best match
- If the best match involves some virtual nodes (e.g., node ψ), replace all nodes $\Omega(\psi)$ with ψ. In the next iteration steps, ψ is regarded as an actual node (not a virtual one).

Adoption of this greedy approach for node comparison reduces the computational complexity of the matching process. In fact, node correspondences are found through an iterative exploration of the best possible node mappings and selection of the best mapping at each iteration. However, this approach does not guarantee to find the optimal solution to the matching problem.

However, at the cost of an increased computational complexity, the proposed approach could be also included in a matching approach that attains exact match between the graphs under comparison in the style of the A* algorithm.

COMPLEXITY ANALYSIS

The computational complexity of the proposed graph matching technique is evaluated as follows. Let us consider two graphs, G_1 and G_2 with m_1 and m_2 nodes, respectively. In addition, without loss of generality, we assume that $m_1 \leq m_2$. The matching algorithm is constructed around a main loop which iteratively considers all the nodes in graph G_1 in order to subsequently assign them to nodes in G_2.

According to this, the worst computational complexity can be estimated for the case in which, for each node i in G_1 the following operations are performed:

(a) Find the node j in G_2 that is the most similar to node i.

(b) Build *VN(i)* and *VN(j)*, that is the sets of virtual nodes for nodes i and j.

(c) Compare *VN(i)* and *VN(j)* to find the best match between their elements.

Complexity of step (a) is $O(m_2)$. Assuming that each node has an average number of N adjacent compatible nodes, the average cardinality of the virtual combination sets is:

$$E[\#VN(i)] = E[\#VN(j)] = \sum^{N}_{k=0} \binom{N}{k} = 2^N$$

Therefore, the computational complexity of step (c) is $O(2^{2N})$. Instead, the computational complexity of step (b) is negligible with respect to $O(2^{2N})$.

Steps (a), (b) and (c) have to be performed for each node i in G_1, so that the overall computational complexity is given by $O(m_1*(m_2+2^{2N}))$.

As a consequence, the management of node merging penalizes the overall complexity of the matching process only in the case in which $m_2 << 2^{2N}$. This condition is rarely verified in graphs originated from segmented images, in that typically the number N of compatible node is small.

EXPERIMENTAL RESULTS

The proposed approach for graph matching by node merging has been experimented in the application context of image retrieval by visual similarity. In particular, two test set have been considered: the *WebMuseum* collection of painting images [29], and the ALOI object image database [30].

The first image dataset comprises about 1000 images, representing paintings by different authors, styles and artistic period collected from the *WebMuseum* [29]. Images were initially described by segmenting them into regions according to chromatic content using the approach proposed in [31]. Color regions identified during this phase are approximately homogeneous, but there are several cases in which the segmentation process may produce over-segmented or under-segmented images. This can hinder an effective retrieval due to the difficulty to map regions of similar, but not identical images. For each image, a graph model is constructed, where each node represents a region and is labeled with a feature vector capturing region area and color. Edges between nodes are used to encode region adjacency.

The example reported, aimed at testing the improvement of retrieval effectiveness determined by the use of the merging strategy applied during graphs comparison. To this end, we compared retrieval results obtained by running the matching algorithm with two different settings of parameter τ_c which thresholds the nodes compatibility: in the first case, we used $\tau_c = 0.3$, thus allowing the combination of adjacent nodes (*node merging enabled*); in the second case we used $\tau_c = 0$, thus preventing all nodes to merge with adjacent nodes (*node merging disabled*). This latter choice reduces the matching problem to the assignment of best fitting nodes in the two graphs.

Retrival results are presented using the standard measures of *precision* and *recall*. Precision is defined for each query image as the number of correctly retrieved images with respect to the overall number of images retrieved from the database (fixed a maximum similarity threshold). Recall is the number of correctly retrieved images with respect to the overall number of images in the database which are relevant to a given query image. The ideal result is to get precision equal to one for every value of recall.

In Fig. (**5**) a manually authored color sketch is used in order to retrieve paintings representing faces. Actually, this is the application scenario which least exploits the potentiality of the proposed approach, since the probability that nodes merging takes place in the query is quite low. This is mainly due to the fact that a user will probably draw a small number of patches identified by very different colors. As a consequence, it is highly probable that node merging will be performed only in the database graphs. So, testing the method in this case should provide a lower bound in the improvement that can be expected in the application of this approach.

The query and the first six retrieved images are shown in Fig. (**5**) (from left to right and from top to bottom). It can be noticed that the top three retrieved images represent portrait paintings. The fourth image does not represent any face, but is retrieved since its segmented regions have colors similar to the colors used in the query sketch.

Fig. (**6a**) shows the precision curves obtained by running the matching algorithm with dynamic merging of graph nodes enabled and disabled, respectively. The horizontal axis represents the size of the retrieval set, while the vertical axis is the precision of retrieval. Values of precision are reported for different sizes of the retrieval set (from 1 to 30), showing that node merging can significantly improve the effectiveness of the retrieval process, expecially for the top ranked, and most relevant, images.

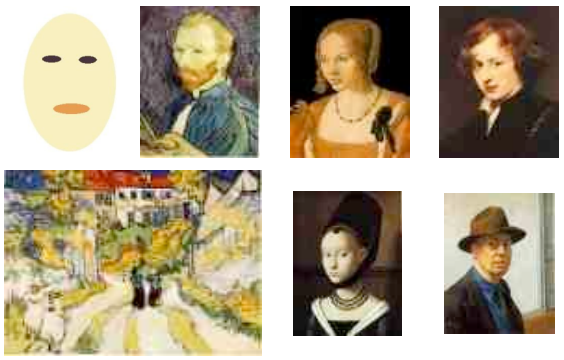
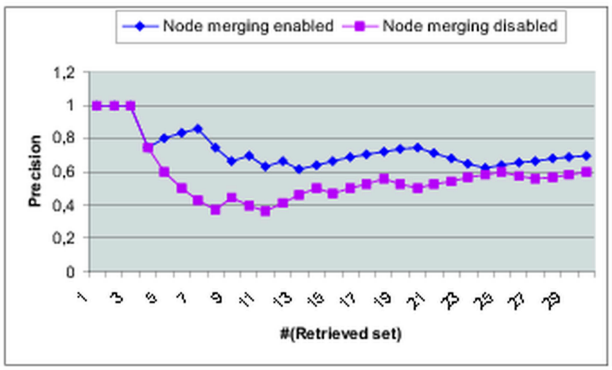


Fig. (5). Retrieval example based on the matching algorithm. The query image is on the upper-left, followed by the six top ranked results (listed from left to right and from top to bottom).

(**a**)



(**b**)

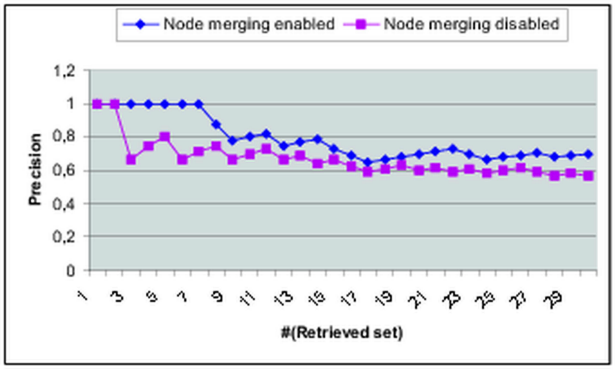


Fig. (6). Precision curves obtained by running the matching algorithm with dynamic node merging enabled and disabled for the examples of Fig. (**5** (plot **a**)) and Fig. (**7** (plot **b**)), respectively.

A different example is shown in Fig. (**7**). In this case, the query is an image selected from the database. Such a query could be useful, for example, to search images with the same theme painted from the same author. Retrieval results are shown for the seven best ranked images (including the query), according to the proposed matching algorithm. It is interesting to note that the query and some of the retrieved images are from the same painter (*Cezanne*).

Fig. (7). Retrieval example based on the matching algorithm. The query image is on the upper-left, followed by the six top ranked results (listed from left to right and from top to bottom).

In Fig. (**6b**), precision curves are shown for the cases in which the matching algorithm runs with dynamic merging of nodes enabled and disabled, respectively. Also in this case, it can be observed the improving of the results obtained by the matching strategy with node merging enabled.

In the experiments on the WebMuseum, the ground truth for the retrieval has been established by randomly selecting 50 images out of the 1000 images of the dataset to be used as query. For these images, 10 users have been required to score the 8 most similar images to the query. Based on these user based evaluation the set of relevant images for each query is defined. Using this experimental set up, Table **2** shows the average values of precision at different recall.

Experiments on the *Amsterdam Library of Object Images* (ALOI database) have been also performed. This database has been used elsewhere to perform retrieval experiments for content based retrieval applications [32]. This allows the comparison of the results reported in this work with those obtained using different retrieval approaches.

Table 2. Value of Precision at Different Recall for a Set of 50 Query Images Randomly Selected from the WebMuseum Dataset

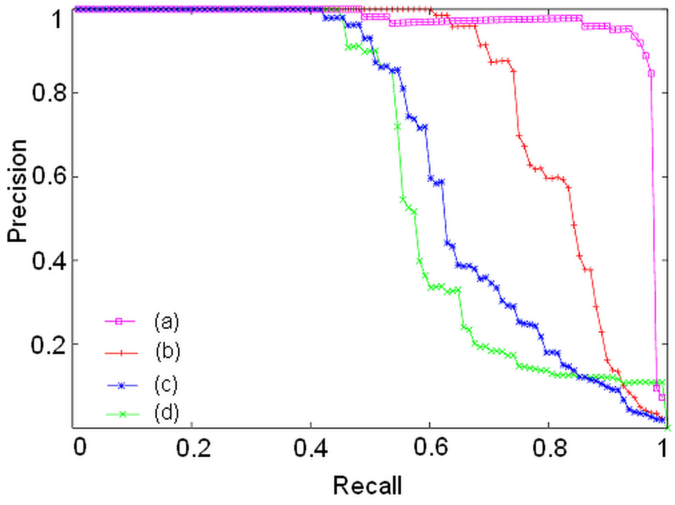
Precision	0.82	0.75	0.69	0.6	0.47	0.35
Recall	0.5	0.6	0.7	0.8	0.9	1.0

The ALOI database comprises 1000 objects recorded under various imaging conditions [30]. In order to capture the sensory variation in objects recordings, the viewing angle, the illumination angle, and the illumination color of every object are systematically varied. Additionally, wide-baseline stereo images are captured. In thiw way, 12 images are captured for each object, yielding a total of 12000 images for the overall collection.

As an example, Fig. (**8e**) shows the *precision-recall* curves computed for four different query images which belong to four different objects categories of the ALOI database (the four query images are reported in Fig. (**8a-d**).

From Fig. (**8**), it can be observed that the proposed solution is capable to provide quite effective retrieval perform-

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



(**e**)

Fig. (8). (**a-d**) Images of four different objects from the ALOI database. These images are used as queries against the entire ALOI database. (**e**) Precision-recall curves obtained using the four images in **a-d** as queries.

ance for objects belonging to different categories. In fact, due to the active aggregation of image regions during the match, the approach can overcome problems originated during the feature extraction process. In particular, one common difficulty occurring in image segmentation is the possible fragmentation of image objects, that can result in the split of individual objects into different parts. Differently from traditional approaches that only perform match between static graph structures, the proposed active graph matching solution is capable to dynamically recombine image regions, thus recovering erroneous segmentations during the match.

Avrage values of precision at different recall are reported in Table **3** for the ALOI database. These experiments have been conducted by using, for every object of the ALOI dataset, the image that has been acquired with illumination at 0°. These 1000 query images have been compared against the entire ALOI dataset.

Table 3. Value of Precision at Different Recall for the ALOI Dataset

Precision	0.91	0.83	0.77	0.72	0.61	0.43
Recall	0.5	0.6	0.7	0.8	0.9	1.0

In these experiments, the ground truth is directly obtained by considering the 12 images of each object as the correct retrieval set for the object. This implies that, for every query, it is expected that the 12 images acquired under different conditions for the same object are ranked in the first 12 positions of the retrieval set. In so doing, we did not consider any particular ordering between these images, assuming that all the 11 images (apart the query image that is obviously ranked in the first position) are equivalent.

CONCLUSIONS

In this paper a novel solution has been proposed for error tolerant graph matching. The solution fits with the class of edit distance based techniques. In particular, the traditional set of edit operations is extended so as to allow node merging during the matching process.

An analysis of the computational complexity of the proposed approach has been presented to evidence that the larger the size of the two graphs being compared, the smaller is the increase of complexity associated with the management of node merging. Results are reported to demonstrate the potential and effectiveness of the proposed solution.

Future work will address a more extensive experimentation and testing as well as a comparison with alternative techniques using edit distance for graph matching, both in terms of computational complexity and matching accuracy.

ACKNOWLEDGEMENTS

This work is partially supported by the Information Society Technologies (IST) Program of the European Commission as part of the DELOS Network of Excellence on Digital Libraries (Contract G038-507618).

A preliminary version of this work appeared in [33]. This work advances [33] by providing insight in the theory and implementation details of the proposed approach, and in the experimental results.

REFERENCES

[1] A. Del Bimbo, *Visual Information Retrieval*. San Francisco: Morgan Kaufmann Publishers, 1999.

[2] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain "Content-Based Image Retrieval at the End of the Early Years", *IEEE Trans Pattern Anal Mach Intell* vol. 22, no. 12, pp. 1349-1380, December 2000.

[3] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain, "Content-Based Multimedia Information Retrieval: State of the Art and Challenges", *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 2, no. 1, pp. 1-19, February 2006.

[4] H. Bunke, "Recent Developments in Graph Matching", in Proc. 15th International Conference on Pattern Recognition, Barcelona, Spain, 2000, vol. 2, pp. 117-124.

[5] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of Graph Matching in Pattern Recognition", *Intern J Pattern Recognit Artif Intell*, vol. 18, no. 3, pp. 265-298, May 2004.

[6] J. Ullman, "An Algorithm for Subgraph Isomorphism", *Journal of the ACM*, vol. 23, no. 1, pp. 31-42, January 1976.

[7] W. H. Tsai, and K. S. Fu, "Error-Correcting Isomorphism of Attributed Relational Graphs for Pattern Analysis", *IEEE Trans Syst Man Cybern,* vol. 9, no. 12, pp. 757-768, December 1979.

[8] M. Das, E. M. Riseman, and B. A. Draper, "FOCUS: Searching for multi-colored objects in a diverse image database", in Proc. International Conference on Computer Vision and Pattern Recognition, 1997, pp. 756-761.

[9] N. J. Nilsson, *Principles of Artificial Intelligence*. Berlin: Springer Verlag, 1980.

[10] S. Berretti, A. Del Bimbo, and E. Vicario, "Efficient Matching and Indexing of Graph Models in Content-Based Retrieval", *IEEE Trans Pattern Anal Mach Intell*, vol. 23, no. 10, pp. 1089-1105, October 2001.

[11] H. Bunke, "Error-correcting Graph Isomorphism Using Decision Tree", *Intern J Pattern Recognit Artif Intell*, vol. 12, no. 6, pp. 721-742, September 1998.

[12] A. Massaro, and M. Pelillo, "Matching graphs by pivoting", *Pattern Recog Lett*, vol. 24, no. 8, pp. 1099-1106, May 2003.

[13] A. Sanfeliu, and K. S. Fu, "A Distance Measure Between Attributed Relational Graphs for Pattern Recognition", *IEEE Trans Syst Man Cybern*, vol. 13, no. 3, pp. 353-362, May/June 1983.

[14] A. Hlaoui, and S. Wang, "A New Algorithm for Inexact Graph Matching", in Proc. 16th International Conference on Pattern Recognition, Quebec City, Canada 2002, vol. 2, pp. 465-468.

[15] C. De Mauro, M. Diligenti, M. Gori, and M. Maggini, "Similarity learning for graph-based image representations", *Pattern Recognit Lett*, vol. 24, no. 8, pp. 1115-1122, May 2003.

[16] N. G. Bourbakis, "Emulating Human Visual Perception for Measuring Difference in Images Using an SPN Graph Approach", *IEEE Trans Syst Man Cybern*, vol. 32, no. 2, pp. 191-201, April 2002.

[17] A. Sanfeliu, R. Alquezar, J. Andrade, J. Climent, F. Serratosa, and J. Verges, "Graph-based Representation and techniques for image processing and image analysis", *Pattern Recognit,* vol. 35, no. 3, pp. 639-650, March 2002.

[18] J. Matas, R. Marik, and J. Kittler, "On Representation and Matching of Multi-Coloured Objects", in Proc. of International Conference on Computer Vision, 1995, pp. 726-732.

[19] I. K. Park, I. D. Yun, and S. U. Lee, "Color Image Retrieval using Hybrid Graph Representation", *Image and Vision Computing,* vol. 17, no. 7, pp. 465-474, May 1999.

[20] R. C. Wilson, and E. R. Hancock, "Relational Matching with Dynamic Graph Structures", in Proc. of the Fifth International Conference on Computer Vision, 1995, pp. 450-456.

[21] 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, June 1997.

[22] L. Shapiro, and R. M. Haralick, "A metric for Comparing Relational Descriptions", *IEEE Trans Pattern Anal Mach Intell*, vol. 7, no. 1, pp. 90-94, January 1985.

[23] B. T. Messmer, and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection", *IEEE Trans Pattern Anal Mach Intell*, vol. 20, no. 5, pp. 493-504, May 1998.

[24] C. Gomila, and F. Meyer, "Tracking Objects by Graph Matching of Image Partition Sequences", in Proc. of International Workshop of Graph based Representation for Pattern Recognition, 2001, pp. 1-11.

[25] R. Ambauen, S. Fischer, and H. Bunke, "Graph Edit Distance with Node Splitting and Merging and its Application to Diatom Identification", in Proc. of International Workshop on Graph based Representations in Pattern Recognition, York, UK, 2003, pp. 95-106.

[26] R. Cesar, E. Bengoetxea, and I. Bloch, "Inexact Graph Matching Using Stochastic Optimization Techniques for Facial Feature Recognition", in Proc. 16th International Conference on Pattern Recognition, Quebec City, Canada, 2002, vol. 2, pp. 465-468.

[27] L. Gregory, and J. Kittler, "Using Graph Search Techniques for Contextual Colour Retrieval," in Proc. of Structural, Syntactic and Statistical Pattern Recognition, 2002, vol. 186-194.

[28] M. R. Garey, and D. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: Freeman, 1979.

[29] WebMuseum of Art: http://www.ibiblio.org/wm/, January 2007.

[30] J. Geusebroeck, G. Burghouts, and A. Smeulders, "The Amsterdam library of object images," *International Journal of Computer Vision,* vol. 66, no. 1, pp. 103-112, January 2006.

[31] A. Del Bimbo, M. Mugnaini, P. Pala, and F. Turco, "Visual Querying by Color Perceptive Regions", *Pattern Recognit,* vol. 31, no. 9, pp. 1241-1253, September 1998.

[32] G. P. Nguyen, and M. Worring, "Optimization of interactive visual similarity based search", *ACM Transactions on Multimedia Computing, Communications, and Applications,* vol. 4, no. 1, February 2008.

[33] S. Berretti, A. Del Bimbo, and P. Pala, "A Graph Edit Distance Based on Node Merging," in Proc. International Conference on Image and Video Retrieval, Dublin, Ireland, Lecture Notes in Computer Science, 2004, vol. 3115, pp. 464-472.

Received: August 1, 2007 Revised: October 9, 2007 Accepted: October 11, 2007