

---

# How and Why Pattern Recognition and Computer Vision Applications Use Graphs

Donatello Conte, Pasquale Foggia, Carlo Sansone and Mario Vento

**Summary.** In this chapter, we present a review of graph-based methodologies for pattern recognition and computer vision, by considering three different points of view: the algorithms, the applications, and the performance evaluation. Preliminarily, a survey of graph-matching approaches, including a synthetic description of a plenty of algorithms and their inspiring rationale, is discussed. Afterward, a detailed taxonomy of pattern recognition applications using graphs is organized, motivating, for each of them, why graph-based approaches can be profitably used and how a specific technique can be exploited. Finally, a section reporting the state-of-the-art of benchmarking activities is present, together with a discussion of the performance issues of well-known graph-based algorithms.

## 1 Introduction

Starting from the late 1970s, graph-based techniques have been proposed as a powerful tool for pattern representation and classification. After the initial enthusiasm, graphs have been practically left unused for a long period of time and only recently are obtaining a growing attention from the scientific community of pattern recognition (PR) and computer vision.

Due to their expressive power, graphs are conquering a primary role as a smart data structure for representing complex visual patterns, especially in structural methods, whose rationale is a vision of the objects as made of parts suitably connected to each other. Under this assumption, nodes of the graphs, enriched with properly defined attributes, can be thought as descriptors of the component parts of the objects, while the edges of the graphs represent the relationships between the parts. The reason why the literature on graph-based approaches is so wide depends on the fact that description schemes generally lead to a variety of graph representations differing from each other for the graph topology, the nature of the nodes and edges (deterministic or stochastic), the type of the attributes (numeric values, symbols, probabilities), etc. Of course, for each representation scheme, some methods for comparing the obtained graph representations must be defined, so obtaining also a variety of algorithms able to calculate exact or somewhat inexact correspondence between graphs, or a sort of distance between them.

The huge material available on graphs often discourages a researcher who intends to use these smart and promising approaches; in other cases, the complexity of this material is the main cause of unsuccessful attempts. The under-evaluation of the complexity of the literature may suggest to the researcher a quick and superficial choice of an algorithm, because of the wrong conviction that almost all the algorithms differ from each other for minor performance issues.

In the recent past some surveys of graph-based techniques have been published (see for example [1]). Since they are mainly focused to presenting almost all the existing algorithms (even if sometimes they are organized in a taxonomy), they often result really useful only to experienced researchers of the field. The idea of the present paper is the attempt of filling the “knowledge gap” existing between the graph-based techniques and their use in PR applications. This is done by considering both the above-mentioned issues in successive sections of the paper, carefully bridging techniques and applications, retracing the history of almost all the PR applications using graphs in the last decades.

In Sect. 2, a survey of graph-matching approaches, including a synthetic description of a plenty of algorithms with their inspiring rationale, is discussed. The survey groups similar approaches and algorithms in a few categories, each described in terms of the underlying technique, purposely neglecting inessential algorithmic details. This is done for both the classes of exact and inexact graph-matching methods. In a following subsection a commented bibliography is given, presenting for each group of algorithms the most important papers, with the aim of explaining the main differences among them, and reconstructing their publication sequence.

In Sect. 3 a detailed taxonomy of PR applications using graphs is organized; this section highlights, for each application, why graph-based approaches can be profitably used and how a specific technique can be exploited. The taxonomy is organized so as to render more understandable by a practitioner of the field the relationship among the techniques and the applications, so as to help him to choose the more suitable structural method using graphs. To complete the review, a final section reports the state of the art of benchmarking activities, together with a discussion of the performance issues of well-known graph-based algorithms. It is mainly organized in order to give general criteria for selecting the most effective algorithm for dealing with the problem at hand, with respect to some common classes of graphs.

## 2 Graph-Matching Taxonomy

In this section we will present a review of the algorithms that have been proposed and used in the PR field for the graph-matching problem. We have divided the matching methods into two broad categories: the first contains exact matching methods that require a strict correspondence between the two object being matched or at least between subparts of them. The second category defines inexact matching methods, where a matching can occur even if the two graphs being compared are structurally different to some extent.

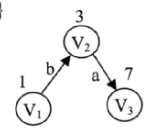
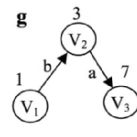
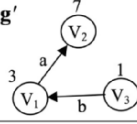
**2.1 An Introduction to Exact Graph-Matching Problems**

Exact graph matching is characterized by the fact that the mapping between the nodes of the two graphs must be *edge-preserving* in the sense that if two nodes in the first graph are linked by an edge, they are mapped to two nodes in the second graph that are linked by an edge as well.

Conceptually, the simplest form of graph matching is *graph isomorphism* (see Fig. 1), where an exact structural correspondence is sought: there must be a bijective mapping between the nodes of the two graphs that preserves the edges of both graphs.

A slightly weaker form of matching is *subgraph isomorphism* (see Fig. 2), that requires the existence of an isomorphism between one of the graphs and a subgraph of the other. In other words, one of the graphs may have extra nodes and extra edges linking these new nodes to the rest. Subgraph isomorphism is often confused with *monomorphism*, which is a little more relaxed matching: in monomorphism extra edges in the larger graph are allowed also between nodes that do have a correspondent in the smaller graph. In subgraph isomorphism, instead, one of the ends of the extra edges must be an extra node. In other words, while isomorphism and subgraph isomorphism impose a two-way constraint on the edges of the graphs, monomorphism imposes a one-way constraint.

A more robust form of graph matching is based on the computation of the maximum common subgraph (MCS - see Fig. 3), that is the largest subgraph of one of the two graphs that is isomorphic to a subgraph of the other. This kind of matching allows both graphs to have extra nodes and edges, but is also significantly more expensive from a computational viewpoint.

	Definition	Example
Graph	<p>A <i>graph</i> is a 4-tuple <math>g = (V, E, \alpha, \beta)</math></p> <ul style="list-style-type: none"> <li>- <math>V</math> is the finite set of vertices</li> <li>- <math>E \subseteq V \times V</math> is the set of edges</li> <li>- <math>\alpha : V \rightarrow L</math> is a function assigning labels to the vertices</li> <li>- <math>\beta : E \rightarrow L</math> is a function assigning labels to the edges</li> </ul>	<p> <math>V = \{V_1, V_2, V_3\}</math>  <math>E = \{(V_1, V_2), (V_2, V_3)\}</math>  <math>\alpha: V_1 \rightarrow 1</math>  <math>V_2 \rightarrow 3</math>  <math>V_3 \rightarrow 7</math>  <math>\beta: (V_1, V_2) \rightarrow b</math>  <math>(V_2, V_3) \rightarrow a</math> </p> 
Graph Isomorphism	<p>Let <math>g</math> and <math>g'</math> be graphs. A <i>graph isomorphism</i> between <math>g</math> and <math>g'</math> is a bijective mapping <math>f: V \rightarrow V'</math> such that</p> <ul style="list-style-type: none"> <li>- <math>\alpha(v) = \alpha'(f(v)) \forall v \in V</math></li> <li>- for any edge <math>e = (u, v) \in E</math> there is an edge <math>e' = (f(u), f(v)) \in E'</math> such that <math>\beta(e) = \beta'(e')</math>, and for any edge <math>e' = (u', v') \in E'</math> there is an edge <math>e = (f^{-1}(u'), f^{-1}(v')) \in E</math> such that <math>\beta(e) = \beta'(e')</math>.</li> </ul>	<p> <math>g</math>  </p> <p> <math>g'</math>  </p> <p> <math>f:</math>  <math>(V_1, V_3)</math>  <math>(V_2, V_1)</math>  <math>(V_2, V_3)</math> </p>

**Fig. 1.** Definitions of graph and graph isomorphism

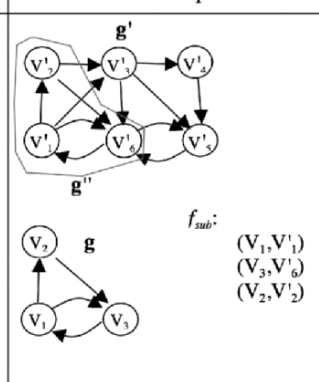
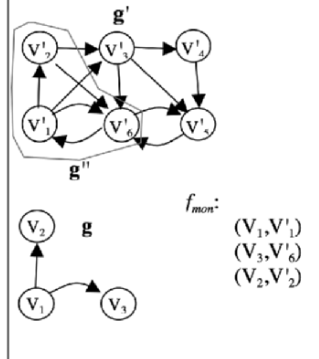
	Definition	Example
Sub-Graph Isomorphism	<p>Let <math>g</math> and <math>g'</math> be graphs. If <math>f_{sub} : V \rightarrow V''</math> is a graph isomorphism between graphs <math>g</math> and <math>g''</math>, and <math>g''</math> is a subgraph of graph <math>g'</math>, i.e. <math>g'' \subseteq g'</math>, then <math>f_{sub}</math> is called a <i>subgraph isomorphism</i> between <math>g</math> and <math>g'</math>.</p>	
Monomorphism	<p>Let <math>g</math> and <math>g'</math> be graphs. A monomorphism is an injective mapping <math>f_{mon} : V \rightarrow V'</math> such that:</p> <ul style="list-style-type: none"> <li>- <math>\alpha(v) = \alpha'(f_{mon}(v)) \forall v \in V</math></li> <li>- for any edge <math>e = (u, v) \in E</math> there is an edge <math>e' = (f_{mon}(u), f_{mon}(v)) \in E'</math> such that <math>\beta(e) = \beta'(e')</math>.</li> </ul>	

Fig. 2. Definitions of subgraph isomorphism and monomorphism

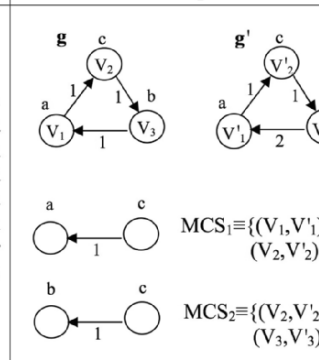
	Definition	Example
Maximum Common Subgraph	<p>Let <math>g</math> and <math>g'</math> be graphs. A <i>Common Subgraph</i> of <math>g</math> and <math>g'</math>, <math>CS(g, g')</math>, is a graph <math>g''</math> such that there are subgraph isomorphisms between <math>g''</math> and <math>g</math> and between <math>g''</math> and <math>g'</math>. We call <math>g''</math> a <i>Maximum Common Subgraph</i> of <math>g</math> and <math>g'</math>, <math>MCS(g, g')</math>, if there is no other common subgraph of <math>g</math> and <math>g'</math> that has more nodes than <math>g''</math>.</p>	

Fig. 3. Definition of maximum common subgraph

It has been demonstrated that the MCS problem is equivalent to the determination of the maximum *clique* (i.e., fully connected subgraph) in a so-called *association graph*, encoding the possible mappings between the nodes of the two graphs being matched; hence, many authors formulate the graph-matching algorithm in terms of clique detection. A generalization of MCS is weighted graph matching (WGM), where the edges of the graphs have a weight, and the goal is to find the common subgraph with the largest total weight.

Because of its strict requirements, isomorphism is not very used in PR applications, where it is customary that the graphs representing different instances of a same pattern have some structural differences due to noise or occlusions or to other causes. Subgraph isomorphism, monomorphism, and MCS are generally used in PR for finding an object, represented by a graph, as a part of a larger model graph (prototype), or for detecting the parts shared by two objects, structurally represented by graphs. Although, exact graph matching has exponential time complexity in the worst case. However, in many PR applications the actual computation time can be still acceptable, because of two factors: first, the kinds of graphs encountered in practice are usually different from the worst cases for the algorithms. Second, node and edge attributes can be used very often to reduce dramatically the search time.

## 2.2 A Commented Bibliography on Algorithms and Techniques for Exact Graph Matching

The first attempts for reducing the computational complexity of graph matching were aimed to define algorithms devised for special kinds of graphs. Among them, we find algorithms for some common graph topologies, as trees (special cases of graphs), proposed by Aho et al. in 1974 [2], planar graphs by Hopcroft and Wong in 1974 [3], and bounded valence graphs by Luks in 1982 [4]). Despite the historical relevance, this family of graph-matching algorithms can be used only in specific applicative areas, where the graphs being matched always have a same predefined structure.

Most of the algorithms for exact graph matching are based on some form of tree search with backtracking. The basic idea is that a partial match (initially empty) is iteratively expanded by adding to it new pairs of matched nodes; the pair is chosen using some necessary conditions that ensure its compatibility with the constraints imposed by the matching type with respect to the nodes mapped so far, and usually using also some heuristic condition to prune as early as possible unfruitful search paths. Eventually, either the algorithm finds a complete matching, or it reaches a point where the current partial mapping cannot be further expanded because of the matching constraints. In this latter case the algorithm backtracks, i.e., undoes the last additions until it finds a partial matching for which an alternative extension is possible. If all the possible mappings that satisfy the constraints have already been tried, the algorithm halts. Several different implementation strategies of this kind of algorithm have been employed, differing in the order the partial matches are visited. Probably the simplest is *depth-first search* that requires less memory than others and lends itself very well to a recursive formulation; it is also known as *branch and bound*.

A nice property of such algorithms is that they can be very easily adapted to take into account the attributes of nodes and edges in constraining the desired matching, with no limitations on the kind of attributes that can be used. This is very important for PR applications where often attributes play a key role in reducing the computational time of the matching. The first important algorithm of this family is due to Ullmann, in 1976 [5], still widely used today. Also the approach proposed by Schmidt and Druffel in 1976 [6] adopts the same strategy, with the addition of a preprocessing that creates an initial partition of the graph nodes on the basis of the distance matrix, to reduce the search space. Another interesting monomorphism algorithm based on backtracking has been proposed by Ghahraman et al. in 1980 [7]; it prunes the search space, using a so-called *netgraph* obtained from the Cartesian product of the nodes of two graphs being matched. Monomorphisms between these two graphs correspond to particular subgraphs of the netgraph. A major drawback of the algorithm is that the netgraph is represented using a matrix of size  $N^2 \times N^2$ , where  $N$  is the number of nodes of the largest graph. Consequently, only small graphs can be reasonably dealt with.

A more recent algorithm for both isomorphism and subgraph isomorphism is the VF algorithm [8, 9]. The authors define a heuristic that is based on the analysis of the sets of nodes adjacent to the ones already considered in the partial mapping. This heuristic is fast to compute leading in many cases to a significant improvement over Ullmann's and other algorithms, as shown in [10, 11]. Successively, the authors propose a modification of the algorithm [12, 13], called VF2, that reduces the memory requirement from  $O(N^2)$  (that compares favorably with other algorithms) to  $O(N)$  with respect to the number of nodes in the graphs, thus making the algorithm particularly interesting for working with large graphs. One of the most recent tree search methods for isomorphism has been proposed by Larrosa and Valiente in 2002 [14]; the authors reformulate graph isomorphism as a constraint satisfaction problem (CSP), a problem that has been studied very deeply in the framework of discrete optimization and operational research. Thus the authors apply to graph matching some heuristics derived from the CSP literature.

The backtracking approach has been applied also to problems different from graph isomorphism and subgraph isomorphism. For instance, Durand et al. [15] have used this approach to solve the maximal clique detection problem. Probably the most interesting matching algorithm that is not based on tree search is *Nauty*, developed by McKay in 1981 [16]. The algorithm deals only with the isomorphism problem, and is regarded by many authors as the fastest isomorphism algorithm available today. It uses some results coming from group theory for constructing the *automorphism* group of each of the input graphs. From them, a *canonical labeling* is derived, so that two graphs can be checked for isomorphism by simply verifying the equality of their canonical forms. The equality verification can be done in  $O(N^2)$  time, but the construction of the canonical labeling can require an exponential time in the worst case. In the average case this algorithm has quite impressive performance, although in [11, 17] it has been verified that under some conditions it can be outperformed by other algorithms like the above mentioned VF2. Furthermore, it does not lend

itself very well to exploit node and edge attributes of the graphs, that in many PR applications can provide an invaluable contribution to reduce the matching time.

Some matching algorithms are specifically aimed at reducing the cost of matching one input graph against a large library of graphs, suitably preprocessed. Messmer and Bunke proposed a very impressive algorithm in 1997 [18,19]. The algorithm, that deals with isomorphism and subgraph isomorphism, in a preprocessing phase builds a decision tree from the graph library. Using this decision tree, an input graph can be matched against the whole library in a time that is  $O(N^2)$  with respect to the input graph size. An extension to MCS is presented in a paper by Shearer et al. in 1997 [20], further improved in [21].

Other two recent papers, by Lazarescu et al. in 2000 [22] and by Irniger and Bunke in 2001 [23], proposed the use of decision trees for speeding up the matching against a large library of graphs. In these cases, the decision tree is not used to perform the matching process, but only for quickly filtering out as many library graphs as possible, applying then a complete matching algorithm only to the remaining ones.

### 2.3 An Introduction to Inexact Graph-Matching Problems

The stringent constraints imposed by exact matching are in some circumstances too rigid for the comparison of two graphs. In many applications, the observed graphs are subject to deformations due to several causes: intrinsic variability of the patterns, noise in the acquisition process, presence of nondeterministic elements in the processing steps leading to the graph representation, are among the possible reasons for having actual graphs that differ somewhat from their ideal models.

So the matching process must accommodate the differences by relaxing, to some extent, the constraints that define the matching type. Usually, in these algorithms the matching between two nodes that do not satisfy the edge-preservation requirements of the matching type is not forbidden. Instead, it is penalized by assigning to it a cost that may take into account other differences (e.g., among the corresponding node/edge attributes). So the algorithm must find a mapping that minimizes the matching cost.

*Optimal* inexact matching algorithms always find a solution that is the global minimum of the matching cost so implying that if an exact solution exists, it will be found. Hence they can be seen as a generalization of exact matching algorithms. Optimal algorithms face the problem of graph variability, and they do not necessarily provide an improvement of the computation time, usually resulting fairly more expensive than their exact counterparts.

*Approximate* or *suboptimal* matching algorithms, instead, only ensure to find a local minimum of the matching cost, generally not very far from the global one. Even if an exact solution exists, they may not be able to find it and for some applications this may not be acceptable, but the suboptimality of the solution is abundantly repaid by a shorter, usually polynomial, matching time.

A significant number of inexact graph-matching algorithms base the definition of the matching cost on an explicit model of the errors (deformations) that may occur (i.e., missing nodes, etc.), assigning a possibly different cost to each kind of error.

These algorithms are often denoted as *error correcting* or *error tolerant*. Another way of defining a matching cost is to introduce a set of *graph edit operations* (e.g., node insertion, node deletion, etc.), each assigned a cost; the cheapest sequence of operations needed to transform one of the two graphs into the other is computed, and called *graph edit cost*.

Some of the inexact matching methods also propose the use of the matching cost as a measure of dissimilarity of the graphs, e.g., for selecting the most similar in a set of graphs, or for clustering. In some cases, the cost formulation verifies the mathematical properties of a distance function (e.g., the triangular inequality); then we have a *graph distance* that can be used to extend to graphs some of the algorithms defined in metric spaces. Of particular interest is the *graph edit distance*, obtained if the graph edit costs satisfy some constraints (e.g., the cost of node insertion must be equal to the cost of node deletion).

Some papers demonstrate equivalences holding between the graph edit distance and relevant graph-matching problems, as the graph isomorphism and subgraph isomorphism and MCS [24–28].

## 2.4 A Commented Bibliography on Algorithms and Techniques for Inexact Graph Matching

Tree search with backtracking can also be used for inexact matching. In this case the search is usually directed by the cost of the partial matching obtained so far, and by a heuristic estimate of the matching cost for the remaining nodes. This information can be used either to prune unfruitful paths in a branch and bound algorithm, or also to determine the order in which the search tree must be traversed, as in the A\* algorithm. In this latter case, if the heuristic provides a close estimate of the future matching cost, the algorithm finds the solution quite rapidly; but if this is not the case, the memory requirement is considerably larger than for the branch and bound algorithm.

The first tree-based inexact algorithm is due to Tsai and Fu, in 1979 [29], and in an extended version in 1983 [30]. The paper introduces a formal definition of error-correcting graph matching of attributed relational graphs (ARG), based on the introduction of a graph edit cost, and defines a search method ensuring to find the optimal solution. A more recent paper by Wong et al. in 1990 [31] proposes an improvement of the heuristic of Tsai and Fu for error-correcting monomorphism, taking into account also the future cost of edge matching.

A similar approach is used in a paper by Sanfeliu and Fu in [32–34], where the definition of a true graph edit distance is attempted, and a suboptimal method, working in a polynomial time, for the distance computation is introduced. In a paper of 1980, Gharaman et al. [35], propose an optimal inexact graph monomorphism algorithm that is based on the use of branch and bound together with a heuristic derived from the netgraph.

Interesting early papers are due to Shapiro and Haralick in 1981 [36] and later in 1985 [37], with algorithms for finding the optimal error-correcting homomorphism and for evaluating the distance between two hypergraphs.



Among the more recent proposals based on tree search we can cite the algorithm using A\*, for different purposes: Dumay et al. in 1992 [38], for evaluating a graph distance and Berretti et al. in their 2000 and 2001 papers [39–41], for finding the largest matching between two sets of nodes forming a *bipartite graph*, with the constraint that each node must be used at most once. A\* search appears also in a recent paper by Gregory and Kittler in 2002 [42], where a fast, simple heuristic is used that takes into account only the future cost of unmatched nodes. The authors assume that at least for small graphs the less accurate estimate of the future cost is abundantly repaid by the time savings obtained in computing a less complicated heuristic.

Another recent inexact algorithm has been proposed by Cordella et al. in two papers of 1996 and 1997 [43,44]. This algorithm deals with deformations by defining a transformation model in which under appropriate conditions a subgraph can be collapsed into a single node. The transformation model is contextual, in the sense that a given transformation may be selectively allowed depending on the attributes of neighboring nodes and edges.

Along the same lines, Serratosa et al. in 1999 [45,46] present an inexact matching method that also exploits some form of contextual information. The authors define a distance between function described graphs (FDG) that are ARGs enriched with additional information relative to the joint probability of the nodes in order to model with one FDG a set of observed ARGs. As in the case of exact approach, efficiently inexact matching algorithms have been proposed for dealing with special, restricted classes of graphs, as planar graphs and region adjacency graphs (RAGs). For planar graphs, Rocha and Pavlidis [47] present an optimal algorithm for error-correcting homomorphism, while in a paper by Wang and Abe (1995) [48], a distance between RAGs is proposed, and is computed using a suboptimal algorithm. More recently, Lladós et al. in a 2001 paper [49] define a graph edit distance for RAGs using edit operations that are devised to model common distortions in image segmentation; the distance is computed using an optimal algorithm based on branch and bound.

The matching methods examined so far rely on a formulation of the matching problems directly in terms of graphs. A radically different approach is to cast graph matching, that is inherently a discrete optimization problem, so as to use one of the many continuous, nonlinear optimization algorithms. The found solution needs to be converted back from the continuous domain into the initial discrete problem by a process that may introduce an additional level of approximation. Nevertheless, in many application contexts this approach is very appealing because of its extremely reduced computational cost that is usually polynomially dependent (and with a low exponent) on the size of the graphs.

The first family of methods based on this approach uses *relaxation labeling*. One of the pioneering works is due to Fischler and Elschlager in 1973 [50]. The basic idea is that each node of one of the graphs can be assigned one label out of a discrete set of possible labels, that determines which node of the other graph it corresponds to. During the matching process, for each node there is a vector of the probabilities of each candidate label, dynamically re-evaluated until the process converges to a stable solution. At this point, for each node the label having the maximum probability is chosen. In 1989 Kittler and Hancock [51] provide a probabilistic framework for

relaxation labeling, in which the update rules previously used for the probabilities are given a theoretical motivation. In 1995, Christmas et al. [52] propose a method, based on the theoretical framework of Kittler and Hancock, that is able to take into account during the iteration process both node and edge attributes. Wilson and Hancock, in 1997 [53], extended the probabilistic framework by introducing a Bayesian consistency measure, that can be used as a graph distance. An extension of this method has been proposed by Huet and Hancock in 1999 [54]. This method also takes into account edge attributes in the evaluation of the consistency measure.

Myers et al. [55] in 2000 propose a new matching algorithm that introduces the definition of a Bayesian graph edit distance, approximated by considering independently the *supercliques* of the graphs, so as to perform the computation in polynomial time. Finally, in a recent paper (2001), Torsello and Hancock [56] propose the use of relaxation labeling also for computing an edit distance between trees.

A recent method by Luo and Hancock [57] is based on a probabilistic model of matching: the nodes of the input graph play the role of observed data while the nodes of the model graph act as hidden random variables; the matching is then found by using the expectation–maximization (EM) algorithm [58].

A different family of methods is based on a formulation of the problem as a WGM problem that permits the enforcement of two-way constraints on the correspondence. It consists in finding a matching, usually expressed by means of a matching matrix  $M$ , between a subset of the nodes of the first graph and a subset of the nodes of the second graph. The edges of the graphs are labeled with weights, that are real numbers, usually between 0 and 1. The desired matching must optimize a suitably defined goal function. Usually the problem is transformed into a continuous one by allowing  $M$  elements to have continuous values so making the WGM problem a quadratic optimization problem. An important limitation of this approach, from the perspective of PR applications, is that nodes cannot have attributes and edges cannot have other attributes than their weight. This restriction imposes a severe limit on the use of the semantic information often available in real applications.

Among the first papers based on this formulation is the work by Almohamad and Duffuaa in 1993 [59]. In this paper the quadratic problem is linearized and solved using the simplex algorithm [60]. The approximate, continuous solution found this way is then converted back into discrete form using the so-called Hungarian method [60] for the assignment problem. Rangarajan and Mjolsness [61], in 1996, proposed a method based on Lagrangian relaxation networks in which the constraints on the rows and on the columns of the matching matrix are satisfied separately and then equated through a Lagrange multiplier. Also in a 1996 paper, Gold and Rangarajan [62] present the graduated assignment graph-matching (GAGM) algorithm. In this algorithm a technique known as *graduated nonconvexity* is employed to avoid poor local optima. Another approach is based on a theorem by Motzkin and Straus that establishes a close relation between the clique problem and continuous optimization. Namely, they prove that all the maximum cliques of a graph correspond to maxima of a well-defined quadratic functional. In 1997, Bomze [63] proposed a modified functional for which the correspondence holds in both senses.

The papers by Pelillo and Jagota in 1995 [64, 65] propose a matching method based on the above cited theorem and an implementation where the quadratic problem is solved by means of relaxation networks [66]. In [67] a unified framework for relational matching based on the Bomze functional is presented. In 1999, Pelillo et al. [68] introduced a technique to reduce the MCS problem between trees to a clique problem and then solved it using replicator equations. Branca et al. [69] proposed in 1999 an extension of the framework defined by Pelillo [67] that is able to deal with a weighted version of the clique problem.

Several other inexact matching methods based on continuous optimization have been proposed in the recent years, as the fuzzy graph matching (FGM) by Medasani et al. [70, 71], that is a simplified version of WGM based on fuzzy logic. Another recent approach, proposed by van Wyk et al. in 2002 [72, 73] is based on the theory of the so-called reproducing Kernel Hilbert spaces (RKHS) for casting the matching problem into a system identification problem; this latter is then solved by constructing a *RKHS interpolator* to approximate the unknown mapping function.

Spectral methods are based on the following observation: the eigenvalues and the eigenvectors of the adjacency matrix of a graph are invariant with respect to node permutations. Hence, if two graphs are isomorphic, their adjacency matrices will have the same eigenvalues and eigenvectors. Unfortunately, the converse is not true: we cannot deduce from the equality of eigenvalues/eigenvectors that two graphs are isomorphic. However, since the computation of eigenvalues/eigenvectors is a well-studied problem, that can be solved in polynomial time, there is a great interest in their use for graph matching. An important limitation of these methods is that they are purely structural, in the sense that they are not able to exploit node or edge attributes, that often, in PR applications, convey information very relevant for the matching process. Further, some of the spectral methods are actually able to deal only with real weights assigned to edges by using an adjacency matrix with real-valued elements.

The pioneering work on spectral methods is the paper by Umeyama, in 1988 [74], proposing an algorithm for the weighted isomorphism between two graphs. It uses the eigendecomposition of adjacency matrices of the graphs to derive a simple expression of the orthogonal matrix that optimizes the objective function, under the assumption that the graphs are isomorphic. From this expression he derives a method for computing the optimal permutation matrix when the two graphs are isomorphic, and a suboptimal permutation matrix if the graphs are nearly isomorphic. In 2001, Xu and King [75], propose a solution to the weighted isomorphism problem, by approximating the permutation matrix with a generic orthogonal matrix. An objective function is defined using Principal Component Analysis and then gradient descent is used to find the optimum of this function.

In 2001 Carcassoni and Hancock [76] propose a spectral method that is based on the use of spectral features to define clusters of nodes that are likely to be matched together in the optimal correspondence; the method uses hierarchical matching by first finding a correspondence between clusters and then between the nodes in the clusters. Another method that combines a spectral approach with the idea of clustering has been presented by Kosinov and Caelli in 2002 [77]: a vector space, called the

*graph eigenspace*, is defined using the eigenvectors of the adjacency matrices, and the nodes are projected onto points in this space and a clustering algorithm is used to find nodes of the two graphs that are to be put in correspondence.

A method that is partly related to spectral techniques has been proposed in 2001 by Shokoufandeh and Dickinson [78]. The authors use the eigenvalues to associate to each node of a Directed Acyclic Graph a topological signature vector (TSV) that is related to the structure of the subgraph made of the descendants of the node. These TSV are used both for a quick indexing in a graph database and for the actual graph-matching algorithm. This latter is based on the combination of a greedy search procedure and of bipartite graph matching.

Finally, we must say that other heuristic approaches to inexact graph matching have been proposed: at least in principle, any of the heuristic techniques that have been used for combinatorial problems or for continuous global optimization problems can be adapted to some approximate form of graph matching. With no presumption of completeness, we can cite here, as examples, simulated annealing (Jagota et al. [79]) and tabu search (Gendreau et al. [80]; Williams et al. [81]).

### 3 Application Taxonomy

In the last decade several applications of graph matching in PR and machine vision have been reported in the literature. As regards the role of such applications within the global context of the containing work, we can recognize two situations. In a first type of works, that we could name *application-driven papers*, the main concern of the authors is solving an applicative problem. They present their graph-based techniques as a solution that is as effective or more effective than other, nongraph-based methods, for solving the problem at hand. Application-driven papers are of course the most interesting ones for an audience involved in deciding whether a graph-based technique is more or less suitable for a given problem, since they usually provide a comparison, either theoretical or experimental (or both), between their proposal and other approaches of different kind to the same problem. A second type of works, say *technique-driven papers*, instead is more centered around the presentation of a novel graph-based algorithm or technique that could be potentially applicable in several situations, and make use of an applicative problem to provide a performance benchmark of the proposed technique in comparison to other, often similar, methods. Since the application is not the main concern, in these papers the authors do not investigate thoroughly the advantages of a graph-based method over a different kind of approach. Nevertheless, these papers provide very useful insight to a slightly different audience: that is, the researchers that have already chosen to cast their problem in a graph framework and are now looking for the best performing technique or algorithm known to solve a problem of that kind. In this review, we will present both application-driven and technique-driven papers, but will focus mainly on the first type of works, providing only a shallow overview of the second type.

We have grouped the applications of graph matching according to the topic within the PR and machine vision fields. Namely, we have individuated five broad areas that cover the vast majority of the applications:

- 2D and 3D image analysis
- Document processing
- Biometric identification
- Image databases
- Video analysis

Among the applications to 2D and 3D image analysis, we have found both low-level problems such as *edge detection* and *stereomatching*, and middle/high-level problems such as *automatic navigation*, *robotic vision*, and *object recognition*. *Handwritten recognition*, *OCR*, and *symbol recognition* are the most relevant document processing applications addressed in the literature, while *face recognition and authentication*, *facial expression recognition*, *hand posture recognition*, *ear recognition*, and *fingerprint recognition* are examples of biometric identification applications. In the field of image databases, *indexing* and *retrieval* have been considered. *Retrieval* from video databases, *annotation* of video databases, *object tracking* and *motion estimation* are the typical applications in the context of video analysis.

While most graph-matching applications fall in the previously outlined categories, there are also a few, isolated works (mainly in the fields of biology and biomedicine) that do not fall neatly into one of the above-mentioned areas. We have presented some of these papers as *miscellaneous applications*.

In the following subsections we will provide details about each application areas, discussing the peculiarity of how graph-based methods have been applied in their context, highlighting (wherever it is possible) recurring patterns of usage and correspondence between the problem, the representation, and the matching technique.

### 3.1 2D and 3D Image Analysis

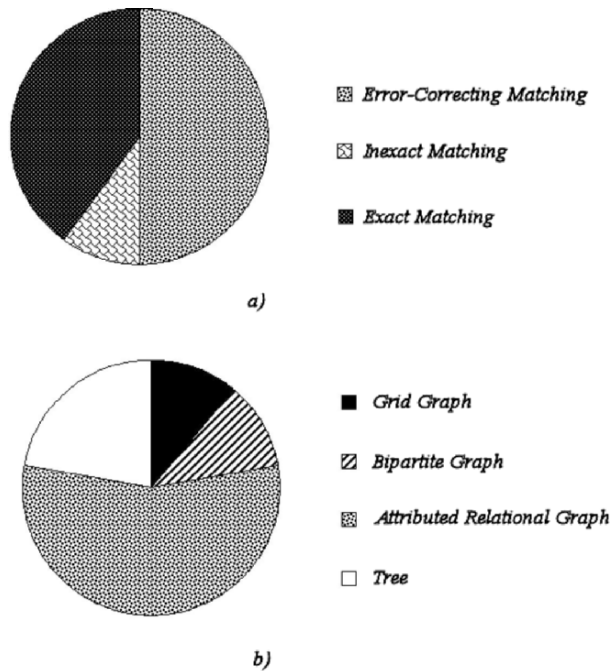
Among the papers that address 2D and 3D image analysis problems with a graph-matching technique, a significant number lies in the technique-driven category we have previously defined. For the 2D image analysis, this is the case of papers that report applications in the object recognition field [31, 73, 82–86], of papers that address the shape recognition [56, 68, 77, 87–89] or the scene recognition [90, 91] problem, and of papers that work on SAR images [53, 92]. As regards 3D image analysis, papers that report results on robotic vision [34], stereomatching [52, 93] object matching [57], object recognition [94–97], and object reconstruction [98] applications can be cited.

However, there are several interesting application-driven papers that we will now examine with more detail. In the field of 2D image analysis, the problems faced by the application-driven papers are three: *object recognition* (by Meth and Chellappa [99], Li and Lee [100], Belongie and Malik [101]), *shape recognition* (Sebastian et al. [102]) and *visual inspection* (Koo and Yoo [103]).

In object recognition the goal is to find all the occurrences, within an image, of a distinguished set of objects. Usually objects of interest belong to different classes (having different shapes) and neither their number nor their positions within the image are known; the image often contains also background elements (possibly complex) that should not be detected by the system. In some cases, the objects of interest may be partially occluded by other objects or by background elements. The basic idea of graph-based object recognition is to decompose the whole image into smaller parts, obtaining a graph representation describing those parts and their relations, and then to look for subgraphs of this large graph that correspond to the shapes of the objects of interest, by means of some kind of inexact matching algorithm. The above-mentioned papers differ in the adopted representations, ranging from low-level [99] to middle-level representations [100, 101], and also in matching techniques (error-correcting subgraph isomorphism with a similarity measure [99], inexact matching with a neural approach [100], weighted bipartite matching [101]); Shape recognition is very similar to object recognition, differing for the fact that only shape information is available (and not, say, color or texture information), and usually the image is not cluttered with background elements. If the shapes are simply connected (i.e., they not contain holes), they can be represented using a tree instead of a fully general graph, and the matching can be performed using error-correcting tree isomorphism [102]. Visual inspection is also similar to the object recognition problem, with the important difference that a model of which objects are expected to be in the image and which should be their positions is known; indeed, the purpose of visual inspection is actually to spot any difference with respect to the expected situation. For this reason, exact matching methods can be more appropriate for this problem [103].

The distribution of the matching algorithms and of the graph representations used within this applicative area is shown in Fig. 4.

Entering into details, in the field of 2D object recognition, Meth and Chellappa in [99] work on SAR images. They use a low-level representation: a node of the graph is associated to each pixel of the image. The node labels depend on the so-called topographical primal sketch (TPS). A TPS assigns to each pixel a label that is invariant under monotonic transformations of the grey levels. This is obtained by fitting a local two dimensional cubic surface on the image for estimating the intensity surface around each pixel. On the basis of the derivatives of this surface, one of the following six labels is given to the pixel: peak, pit, ravine, ridge, saddle, "no zero crossing." Two graph-matching techniques are proposed, the first one is based on a distance measure between node labels, while the second one is based on a similarity measure between features associated to node labels. In both cases, the test and the model image are first registered with respect to the node labels position. The first matching technique calculates a cost based on the relative distance between nodes with the same label in the test and in the model image. The second one associates a feature vector (by calculating the second derivative extrema, the directions of the second derivative extrema and the gradient) to nodes that have a certain label; on the basis of these feature vectors a similarity measure is computed. Results are reported on 81 images belonging to three different categories.



**Fig. 4.** Distribution of (a) the matching algorithms and (b) the graph representations used within applications in the 2D and 3D image analysis field

In [101] Belongie and Malik define a new middle-level shape descriptor, that they call *shape context*, for measuring shape similarity. Given an image, the edges are extracted and a certain number of uniformly spaced points, say  $N$ , on these edges is selected. A compact descriptor for each sample point is obtained by computing a coarse histogram of the relative coordinates of the remaining points, in a log-polar coordinate system. All the  $N$  histograms are flattened and concatenated so as to obtain the so-called shape context of the image. In addition to this representation, another one based on the local appearance, in particular on the tangent angle calculated for each of the  $N$  points, is also used. So, if a node of a graph is associated to each of these points, the cost of matching two nodes relative to points on two images can be expressed by taking into account two contributions, one relative to the difference between histograms and the other one relative to the tangent angles dissimilarity. The object recognition problem is then viewed as a weighted bipartite matching problem that can be solved with the Hungarian method. Results on the same database used in [104] are presented and also on other silhouette image databases. Furthermore, the authors suggest that their method can be also used for the retrieval from an image database, as it provides a similarity measure between 2D objects.

In the paper by Li and Lee [100], a graph represents a 2D scene that is described by using a polygonal approximation. The nodes of the graph are the vertices of the

polygon and the edges represent the sides. The angle subtended by each vertex is the node attribute, while the distance between two nodes is used as edge attribute. Given such a graph (called *scene graph*), in order to cope with distortions and occlusions, the authors propose to divide it into smaller pieces, called *subscene graphs*. Then, an inexact subscene graph matching is performed between each subscene graph and a model graph, by using an Hopfield neural net. The correct match for the complete scene graph can be obtained from the statistics of the matching results between each subscene graph and the model graph. In the paper, tests are made on images representing 2D hand tools.

Sebastian et al. [102] propose a system to recognize object shapes on the basis of their silhouette. They represent each object using a shock tree, which is derived from a thinning of the shape. For the matching, they propose the definition of a tree edit distance, in which the edit costs are not fixed arbitrarily but are derived analytically from a small set of hypotheses related to the cost of deforming a silhouette. This distance is computed by means of an error-correcting tree isomorphism algorithm based on dynamic programming.

Finally, Koo and Yoo [103] address the problem of visual inspection by using an high-level representation scheme. They consider Printed Circuit Board images, that are represented by means of a tree. Images are first binarized, then the binarized image is partitioned into nonoverlapping regions (*blobs*) each one made up of adjacent pixels having the same value. A node is associated to each blob; a tree is constructed by adding an edge between two nodes if the blobs they represent are spatially included into each other and have different pixel values. The root of the tree is the blob that contains the outer boundary pixels of the image. The tree obtained from a test image is compared with the tree derived from a defect-free image by means of the tree isomorphism algorithm proposed in [2]. If the two trees are not isomorphic a defect is detected and the inspection process stops. Otherwise, additional polygonal-boundary information are extracted and a second matching step is performed. In particular, a tolerance zone is defined and the proposed algorithm through a polygonal-boundary matching function checks if such tolerance is respected between pairs of matched nodes.

In the field of the 3D image analysis applications, Branca et al. [69] addressed the problem of automatic navigation, while Bauckhage et al. [105] and Olatunbosun et al. [106] the recognition of 3D objects, and Fuchs and Le Men [107, 108] the 3D object reconstruction.

The 3D object recognition problem is quite similar to the 2D version; the main differences are the need to take into account the changes in the object appearance due to a different point of view, and the increased importance of the occlusion phenomenon. These differences lead to the use of graph-matching algorithms more tolerant to structural changes.

The 3D object reconstruction is aimed at deriving the three-dimensional structure of a scene from 2D images. Under some constraints on the objects being reconstructed, it can be faced with an approach that reduces this problem to object recognition, by defining a set of 3D structural primitives whose occurrence can be recognized in the image. Automatic navigation consists in the detection of still or



moving objects (such as obstacles a vehicle has to avoid) in a 3D scene, usually represented by means of a pair of stereo images. The problem is different from object recognition since the shape of the objects is not known a priori. However, it turns out that this problem can be faced with techniques very similar to the ones used for 3D object recognition. In fact, by finding for each part of one of the two images the corresponding part in the other (which is similar to what an object recognizer does), the distance of each part from the camera can be estimated.

The 3D applications mainly use ARGs [69, 105, 107, 108] for representing objects, and two different matching techniques: either perform a MCS detection (by means of a maximal clique search on the association graph) [69, 106] or employ an error-correcting subgraph isomorphism algorithm [105, 107, 108]. With more detail, Branca et al. [69] presents an application of graph matching to autonomous navigation. In particular, the detection of ground floor obstacles and of moving objects are considered. Relational graphs are used for object representation, where the object features extracted by means of the Moravec interest operator are the nodes and the edge linking them are weighted by projective invariant values. Given two graphs obtained from two different images acquired by a TV camera mounted on a mobile vehicle, the goal is to determine, into the association graph, the maximal clique of nodes that are mutually compatible according to the similarity imposed by the invariant relations encoded into the edges. The nodes of this clique will belong to the same object and this permit to detect into a given image the features that belong to an obstacle, or to individuate the feature pertaining to a moving object. In the paper an algorithm for finding the maximum edge weighted clique in an high-order association graph is presented, based on an optimization procedure that use the Motzkin–Straus theorem.

As regards 3D object recognition, Bauchhage et al. present in [105] a system that uses graphs to recognize mechanical assemblies in a dynamic construction environment. In particular, the main objective of their project is to develop a robot that assembles parts from a wooden construction-kit for children, made up of bolts, rings, bars, and cubes. So, given an assembly described by a graph, they use a graph-matching technique for recognizing if that assembly is already present in the knowledge base of the robot. In the negative case it will be added to the database. They introduce the *mating feature* graph for representation. The nodes of this graph represent mating features or subparts of an assembly and are labeled with the type of the subpart. Nodes connected with a pair of edges represent subparts that belong to the same object, while single directed edges between nodes represent the fact that the corresponding subparts are attached to the same bolt. If an object is connected to several bolts, the nodes that correspond to these bolts are linked by a bidirectional edge; this edge is labeled with a value that indicates the angle between the bolts. For matching two mating feature graphs, they use the error-correcting matching procedure presented in [109]. They also propose an application of the mating feature graph to the 3D reconstruction of assembled objects.

Another approach is proposed in Olatunbosun et al. [106], where a special kind of RAGs, called color region adjacency graph (CRAG), is used for representing 3D objects. Graph nodes are the segmented regions, using the coordinates of their

centroids as node attributes, and edges represent connections between regions. By using the line length ratio, i.e., the ratio of the distance between a pair of nodes into the model image and a pair of nodes in a test image, and the line angles, i.e., angles between three nodes into the model and the test image, an association graph where the nodes are provisionally matched is built. Then the Bron–Kerbosh [110] algorithm is used to find the maximal clique on this association graph: an high clique value imply high similarity between the model image and a test image. The authors also propose to reduce the computational complexity of the maximal clique search method, by adopting a model-based approach. In order to recognize an object, a test images is first filtered, by eliminating from it all the color regions that do not belong to the model CRAG. So, the maximal clique search is performed on a smaller association graph.

Finally, Fuchs and Le Men in [107] and [108] use graph matching in the field of 3D building reconstruction from aerial stereopairs. In particular, in [108] the 3D object extraction problem is addressed, while in [107] the goal is the reconstruction of the structure of the roofs. They use a model driven strategy: the models used are ARGs, where each nodes represent a 3D feature (a 3D line segment, or a 3D planar region, or a facade of a building), while each edge encodes a geometric property (such as parallelism, orthogonality, and so on) between nodes. The building reconstruction is based on the computation of a subgraph isomorphism between a model and a graph built on a set of 3D features derived from the images. As regards the matching procedure, in [108] they use the error-correcting subgraph isomorphism detection presented in [109], with an estimation of the subgraph distance based on a stochastic heuristic, while in [107] propose a modification of the algorithm proposed in [109] in order to take benefit of an external information (e.g., an user input or a precomputed information). If the correspondence between some nodes of the model and some nodes of the input data is already known before the matching, the search space of the matching problem can be pruned by integrating the external information in the error-correcting subgraph isomorphism algorithm.

### 3.2 Document Processing

Among the various document processing applications, OCR, handwritten recognition, string recognition, symbol and graphic recognition have been addressed in the literature by using graph-matching techniques.

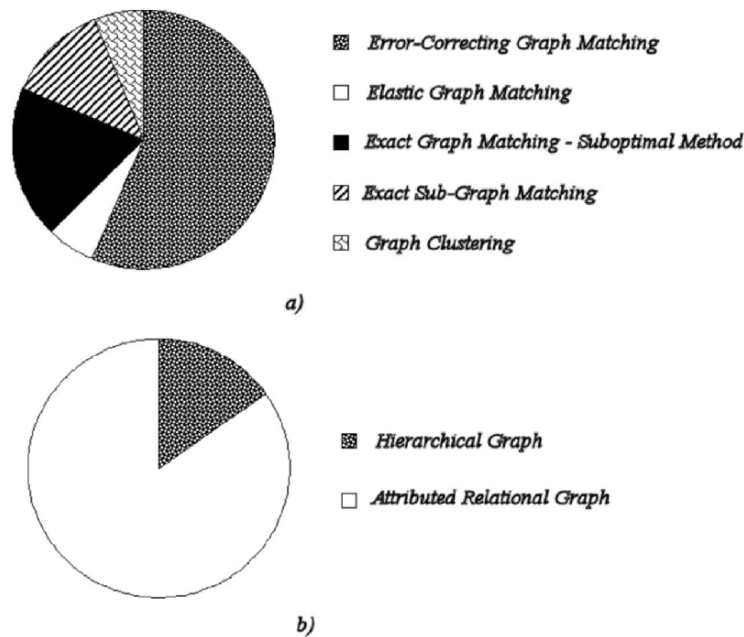
These problems are relatively similar to each other, entailing the recognition of small elements having a definite meaning within a printed or handwritten document. The number of different categories (classes) to be considered varies from ten to several hundreds, and also the shape variability of the elements belonging to a same class can range from reasonably small (e.g., for high-resolution printed characters) to very high (for handwritten characters or symbols). As regards the strategy adopted to face the problem, entities to be recognized (characters, symbols, or graphics) are usually decomposed into geometric primitives, which are in most cases approximated as thin lines (also called *strokes*), since in handwriting and in printed scripts, thickness does not convey useful information. This decomposition is then represented as

a graph, and the recognition process is performed as a graph matching with model graphs corresponding to the different classes of characters, symbols, or graphics to be recognized. The proposed approaches differ in complexity of the geometric primitives, in the way the decomposition is translated into a graph, and in the kind of graph matching performed. A problem that is strongly related is the construction of such model graphs from a set of examples, which is usually performed by means of algorithms that involve a graph matching as one of their step.

The distribution of the matching algorithms and of the graph representations used within the document processing field is shown in Fig. 5. Now we will examine with more detail each of the problems belonging to this field.

Since OCR and handwritten recognition are among the most classical PR problems, and many of large datasets are available, they are often used as test cases in technique-driven papers. This is the case of the papers by Sanfeliu and Fu [34], Foggia et al. [43, 111, 112], Chan [113] and Rangarajan and Mjølness [61]. But also several application-driven papers have been written on the handwritten recognition problem (both offline and online) or on the optical character recognition problem: this is the case of the papers by Lee and Liu [114] and Suganthan and Yan [115], and Liu et al. [116], Lu et al. [117], Chen and Lieu [118], and Rocha and Pavlidis [47].

Independently on the main focus of the considered papers, both printed and handwritten characters are typically described by ARGs [34, 43, 47, 111, 112, 115, 117, 118]. Two description schemes have been used (1) the nodes of the graph represent



**Fig. 5.** Distribution of (a) the matching algorithms and of (b) the graph representations used within applications in the document processing field

the structural primitives in which a character can be decomposed after a thinning process and the edges represent the relations between them these primitives or (2) the nodes are the junctions between strokes (singular points) and edges represent the primitives into which characters are decomposed. Authors dealing with Latin characters and Arabic digits [34, 43, 47, 111, 112] use circular arcs and segments as primitives, while straight line segments or strokes are typically used in case of Chinese characters [115–118]. As regards the representation, a quite different approach is proposed in [114], where the authors propose an architecture for the recognition of handwritten Chinese character that integrates the feature extraction, the segmentation, and the recognition phase. The feature extraction phase is performed by means of Gabor filters; such features are used to segment characters using an optimization module based on a genetic algorithm. Finally, elastic graph matching is used in the recognition phase. Besides this paper, other authors mainly use error-correcting graph matching for dealing with the high variability of handwritten characters. Another feature that is peculiar to this kind of applications is the graph size: graphs describing characters are typically made up of few nodes.

As regards technique-driven papers on handwritten recognition, in [34] and [111] the authors use respectively handwritten characters and handwritten digits to validate a distance measure between ARGs in the framework of error-correcting graph matching. In the same framework, a matching algorithm using subgraph transformations is applied to handwritten characters [43]. Chinese characters are used in [113] as test case for a learning algorithm that build templates starting from fuzzy-attribute graphs; while in [61] the authors present a suboptimal method for exact graph matching, based on a lagrangian relaxation network, using handwritten digits for testing. Finally, handprinted digits are used as application of a graph learning algorithm [112].

As regards the recognition of Chinese characters, both offline and online approaches are present in the literature. In [115] the matching between input graph and model graph for offline Chinese character recognition is performed by means of an Hopfield network (presented in [119]) that is specifically devised to allow the segments of a broken stroke of an input character to be matched to a stroke of the model graph.

The recognition of online handwritten Chinese character addressed by graph matching has the additional problem of a significant computational cost due to the large number of categories. Therefore for developing an online recognition system it is mandatory to find an adequate structural representation together with matching algorithms that can efficiently address this recognition problem. To this aim, some authors [117, 118] used a sort of hierarchical graphs to represent a character. Such graphs have two layers: nodes and edges in the first layer represent high-level components and relations between them; while in the second layer each component is described by a graph in which nodes and edges represent the strokes of that component and their relations. In [116] a Chinese character is described with a complete relational graph (CRG), where each node describe one of the segments in which a stroke obtained from a pen down–pen up movement on a digitizer can be decomposed. In order to reduce matching time, a suboptimal solution is proposed. The

problem of matching CRGs is transformed into a two-layer assignment problem and solved with the Hungarian method. Within the OCR field, in [47] an error-correcting subgraph-matching algorithm is used. It allows a multiple-to-one matching from a set of feature (a path) of an input graph to a feature of a model graph (prototype), on the basis of a set of predefined transformations. These graph transformations regard straightening of strokes, rewriting of strokes into arcs, insertion and deletion of features, and attribute transformations. Having associated a cost to each transformation, the matching procedure for each input graph selects the prototype that gives rise to the matching with the minimum cost. It is worth noting that prototypes are manually defined, without using a specific learning procedure.

A quite peculiar approach is proposed in [120], where the OCR problem is addressed with an ad hoc matching defined between the so-called graph embeddings. A graph embedding, used in this paper for representing characters, is a labeled graph where each node is labeled with its coordinates in the  $x$ - $y$  plane.

The handwritten digit string recognition problem has been addressed by [121]. Starting from an input image and after a thinning process, the authors construct a graph whose nodes are the branches or the ending points of the thinned image and whose edges represent lines of the thinned image. The input graph is then submitted to a segmentation process by using a set of heuristic rules. It gives rise to a number of separate symbols, called *blocks*. The recognition procedure consists in matching the input blocks with the prototype graphs of the digits, by applying a set of transformations to each input block. The matching is therefore an error-correcting graph-subgraph isomorphism. As transformations, the combination of two nodes into one, the transformation of a loop to an edge and the deletion of edges or nodes are considered.

Finally, in the field of symbol and graphics recognition fall the paper of Llads et al. [49, 122], Changhua et al. [123], Cordella et al. [8] and Jiang et al. [124]. While the last two papers are devoted to exploit the performance of an exact subgraph-matching algorithm in detecting component parts within technical drawings [8] and of a graph-clustering algorithm [124], respectively, the others have their main focus on the application domain.

As in case of character recognition, almost all the approaches use ARGs for representing symbols or graphical drawings; as already said, in [8] an exact subgraph-matching algorithm is used, while other authors employ different kinds of error-correcting subgraph-matching algorithms for recognition. The main difference with respect to the case of character recognition is in the number of the nodes of graphs representing maps, diagrams, or technical drawings, that can be up to some hundreds or even thousands.

In [49] the problem of finding a model graph, that represents a prototype symbol, as a subgraph of an input graph, that represents a drawing, is addressed. To do this, a two-level graph representation for graphical symbols is used. In the first level, a vectorized document is approximated by graphs whose nodes represent characteristic points (i.e., junctions, end or corner points, and so on) and whose edges approximate the segments between them. In the second level, data is organized in terms of RAGs. The RAG nodes represent the regions, i.e., minimal closed loops of the first level

graphs, and the edges are the neighboring relations between regions. Symbols are then recognized by means of an inexact subgraph-matching procedure that computes the minimum distance from a model RAG to an input RAG. This distance is considered to be the weighted sum of the costs of edit operations to transform one RAG into another one.

In [122] the authors try to identify building blocks in a hand-drawn floor plan. After a scanning and a vectorization process, drawings are described by means of ARGs. An inexact subgraph isomorphism algorithm based on discrete relaxation is used for matching the obtained ARG against model graphs representing the building elements. In order to speed up the process, a straight line Hough transform is also used. It allows the detections of regions filled with parallel straight lines, such as walls that are typically characterized by hatching patterns.

Finally in [123] graphical hand-sketched symbols are represented through ARGs and a similarity measure calculated using the A\* algorithm is used for recognition.

### 3.3 Biometric Identification

Graph-based techniques have been widely used within the context of biometric applications, mainly with reference to identification problems implemented by means of elastic graph-matching procedures. Rarely, in this application area, graph-learning algorithms have been used.

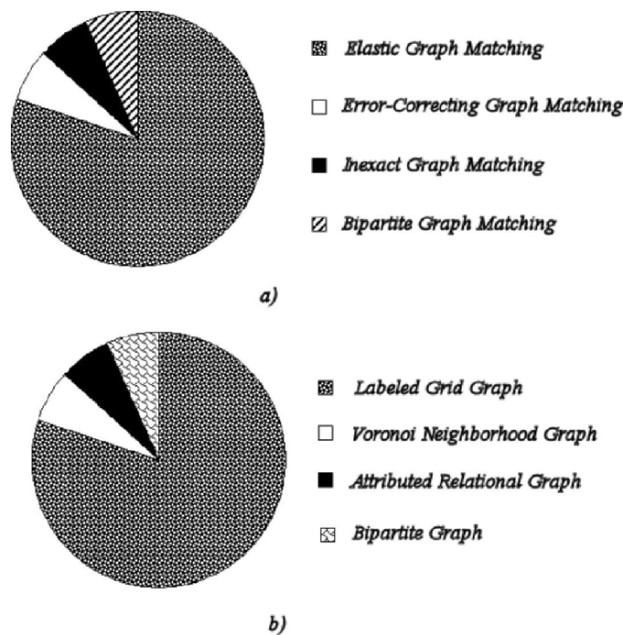
Among all the biometric identification problems, a key role is played by *face authentication*, *face recognition*, and *fingerprint recognition*. Moreover, there are other applications based on facial images, as *facial expression recognition* and *face pose estimation*, as well as other probably less-known applications, as *hand posture recognition* and *ear recognition*. In all these problems, the goal is to compare a graph representation obtained from a sample image of some biometric trait of an individual with a model graph. This comparison has to take into account the possibility of severe distortion of the sample graph with respect to the model, due to the extreme variability in the appearance of biometric traits. In the case of authentication, there is only one model graph, and the problem is to decide whether the model and the sample correspond to the same person. In biometric recognition problems, instead, there are several models (corresponding to different persons, but also possibly to different gestures of a single person) and the system has to identify the person (or the gesture) shown in the sample. A characteristic that is common to many applications of this category, is that the reliability of the identification is extremely important, since the cost of errors is significantly larger than, for example, that of document processing applications.

In almost all cases, papers falling in this area are mainly application-driven, so using rather standard graph-based techniques; sometimes minor adjustments to classical algorithms have been introduced so as to take into account peculiarities of the problem at hand. Generally, the graphs used for describing the patterns are made of tens of nodes and have a rather simple structure (sometimes regular ones, as the grids of nodes used for applications dealing with facial images). The attributes of the nodes of the graphs are rather complex, and frequently are given by feature vectors made

of many components, while the edge attributes are simpler and typically represent distances between given points in the original image. For these characteristics, the matching technique that is most commonly used is elastic graph matching.

The distribution of the matching algorithms and of the graph representations used in the Biometric Identification field is summarized in Fig. 6.

In the areas of face authentication and face recognition, graph matching has been used in the systems proposed by Van Der Malsburg, Wiskott et al. [125–127], by Lim and Reinders [128], by Kotropoulos, Pitas, and Tefas [129, 130], by Duc et al. [131] and by Lyons et al. [132]. All these approaches use a graph, in particular a labeled rectangular grid, as an intermediate representation level for representing a face. In this grid, each node of the graph is associated to a specific facial landmark, called *fiducial point*. The labels associated to the nodes are of two different types: those based on *Gabor coefficients* [125–128, 131, 132], the so-called *jets*, and those made up of a vector of features evaluated on small areas of interest in the input image by means of multiscale dilation–erosion techniques [129, 130]. The face identification process is carried out by standard elastic graph-matching algorithms. The grid representing the input face is compared with the ones representing face models. During the matching process the feature vectors associated to matched nodes are used to calculate a distance, so as to evaluate an overall distance between the two compared input graphs. The matching procedure is elastic in the sense that it copes with deformations, rotations, or scale variations in the areas of interest of the input image.



**Fig. 6.** Distribution of (a) the matching algorithms and of (b) the graph representations used within applications in the biometric identification field

In more details, one of the simplest description schemes is the one proposed in [128], where the authors describe a face image by a graph made of four nodes representing prefixed landmark points of the face as eyes, the nose, and the mouth. Each node is labeled with a jet, while an edge of the graph is associated an attribute representing the distance existing between the points of the images relative to the nodes it connects. In this paper the elastic graph-matching procedure is specifically tailored for dealing with affine transformations on the considered images in the neighbors of the landmark points, and the authors denote their matching algorithm as *affine graph matching*. The algorithm is used for localizing a face within an image, and this task is accomplished by maximizing the similarity measure proposed in [127]; they take into account only the magnitude value of the jets, and use a genetic algorithm for exploring the search space more efficiently.

In the papers by Van der Malsburg, Wiskott et al. [125–127] faces are described by a larger graph, in particular a rectangular graph (a *grid graph*) where each node label is associated to a vector of Gabor wavelet complex coefficients. In [125] only the magnitude of these coefficients is used in the recognition process; while in [127] the addition of the phase of the coefficients allowed to achieve a more accurate location of the landmark points within the considered image. Moreover, in the latter paper, a new data structure, called *bunch graph*, is introduced for dealing with generalized representations of faces. A face bunch graph (FBG) is a sort of prototype of a set of images. As the previous graphs, it has a grid structure, and each node is devoted to represent the homologous nodes (fiducial points) of the represented graphs. The term *bunch* is used to denote the set of jets referring to the same fiducial point, and associated to a node of a FBG. The FBGs used to represent the images are obtained by an elastic graph-matching procedure, described in more details in [126]. The latter paper also explores the possibility of determining facial attributes, as sex, presence or absence of glasses or beard by using FBGs.

The magnitude of Gabor coefficients as features associated to grid nodes have been also used by Duc et al. [131] and Lyons et al. [132] in combination with techniques based on discriminant analysis. In particular in [131], after the elastic graph-matching phase, the authors use a local discriminant analysis on the feature vectors associated to grid node to verify the correct identity of the input face. In [132], instead, the authors use discriminant analysis before the matching. In particular, they submit the feature vectors to a principal component analysis so as to reduce the dimensionality of the feature space. They also present results on sex, race, and expression recognition.

Instead of using Gabor coefficients, Pitas et al. in [129] associate to each node of the grid a feature vector obtained by applying a multiscale dilation–erosion operator to the input image; they also propose a variant of the elastic graph matching, called morphological elastic graph matching (MEGM) that uses in the elastic graph-matching procedure the feature vectors obtained by morphological operators. The use of such operators is justified by considering that the computation of Gabor coefficients is time consuming while dilation and erosions can be computed in a very fast way. Moreover, dilations and erosions deal with local minima or maxima in an image and revealed to provide an effective characterization of facial features.



In a more recent paper [130] the same authors describe a method to improve the recognition performance of MEGM. In particular they propose to estimate the best coefficients for weighting the similarity values associated to the grid nodes by means of discriminant analysis techniques and support vector machines.

Among the other applications dealing with face images, papers by Wang et al. [133] and Hong et al. [83] make use of graph-matching techniques in the context of *facial expression recognition* while Elagin et al. [134] use graph matching for *pose estimation*.

In particular, as usual in this application area, Hong et al. [83] use grid graphs, labeled with jets, for representing faces and rather standard elastic graph-matching algorithm for recognizing seven face expressions: neutrality (that means no expression), happiness, sadness, anger, disgust, fear, and surprise.

Only three expressions are instead considered by Wang et al. in [133]: happiness, surprise, and anger. Indeed, their main goal is rather different and is aimed to estimate the changes of face expression from sequences of facial images. To this concern, the correspondence between images relative to successive frames is viewed as an elastic matching, even if the authors call it “labeled graph-matching problem.” In detail, 19 nodes are used to represent a face image. Each node is labeled using a template matrix of the  $17 \times 17$  pixels (in gray levels) around each node, while to each edge is associated a measure of the distance between the nodes it links. The graph matching is carried out by minimizing a cost function that takes into account both the template similarity and the topological information.

In the framework of the pose estimation problem, Elagin et al. [134] use graphs with 16 nodes to represent a face. Each node is associated to a facial landmark, as the pupils, the tip of the nose, the mouth angles, and so on. Also in this case, the nodes are labeled with Gabor coefficients, while the labels of the edges represent the distances between the points of the image associated to the nodes. Five different orientations are considered for the pose. As in [127] a bunch graph is used to represent set of faces, and so a bunch graph-matching procedure is used in order to perform the estimation.

The use of graph matching in the context of hand posture recognition, is described in the paper by Triesch and von der Malsburg [135]. The authors employ a description and recognition scheme similar to those typically utilized in the field of face recognition. In fact, Gabor coefficient as graph labels and elastic graph matching for recognition are used. In addition to conventional Gabor jets, a *color-Gabor jet* is introduced. It measures the similarity of each pixel to the skin color and together with the Gabor jet constitutes the so-called *compound jet*. The elastic graph-matching procedure is also modified in order to cope with this compound jet. After describing each hand by graphs made up of 15 nodes manually placed at anatomically significant points, 12 different hand postures are recognized.

Another biometric system is the one proposed by Burge and Burger [136], that make use of features extracted by ear images for subject identification. They consider  $300 \times 500$  pixels images, acquired using a CCD camera. Also in this case a middle level representation is used; after the localization of the ear within the images, an edge extraction based on the Canny operator is performed, followed by a curve

extraction. On the basis of the regions delimited by the obtained curves, a Voronoi neighborhood graph is constructed. The identification process is accomplished by a subgraph error-correcting graph matching between the model graph and the input graph. To this aim, the authors propose a matching procedure that specifically takes into account the possibility of broken curves into the input graph. This procedure tries to merge neighboring curves if their Voronoi regions indicate that they are part of the same underlying feature.

Finally, fingerprint recognition by means of graph matching, has been addressed in the papers by Maio and Maltoni [137] by Fan et al. [138] and by Neuhaus and Bunke [93]. This latter is a technique driven paper, while the other two are application driven. They use different approaches both for representing fingerprint and for recognizing them.

The first paper [137] uses ARG for describing fingerprints. The original fingerprint image is first processed in order to calculate a directional image. Then the directional image is segmented into regions, and each region is represented by a node of the graph. Each node has an attribute that measures the area of the region it represents, while each edge has three attributes: the phase difference between the average directions, the distance between the centroids, the length of the boundary between the regions represented by the two nodes it links. For the recognition phase an inexact graph matching is proposed, based on a branch and bound search within the space state.

On the other hand, Fan et al. [138] use bipartite graphs for representing the sample fingerprint image and template fingerprints. A fingerprint image is preprocessed in order to extract clusters of feature points (minutiae). A set of 24 attributes is then calculated for each feature point cluster and is associated to a node of the graph. The feature point clusters of a test image are the set of the left nodes of a fuzzy bipartite weighted graph while the feature point clusters of the template fingerprint are the right nodes. Fingerprint verification is then treated as a fuzzy bipartite graph-matching problem.

### 3.4 Image Databases

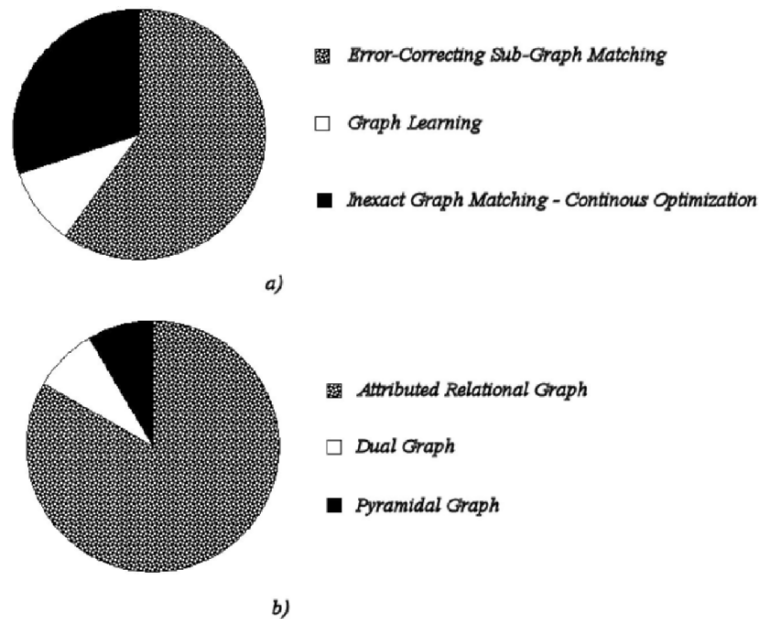
Another field in which graph-based techniques have been successfully employed is the one of image databases. Typical applications involving this kind of databases are indexing and retrieval: few papers addressed both the aspect [41, 139], while the most part [42, 54, 104, 140–147] investigated only the retrieval problem. Among all these papers, in [144] Hlaoui and Wang use a simple image database only for testing the performance of a new error-correcting matching algorithm with edit operations.

The indexing and retrieval problems are very similar from a conceptual point of view, but their different requirements in terms of performance and accuracy have brought to the use of different techniques and algorithms. In both cases, the goal is to find the images in the database that are similar to a given query image. While this can bear some resemblance to a recognition problem, there is an important conceptual difference: the images in the database are not partitioned in a set of fixed, nonoverlapping classes, to which the unknown class of the query image belongs.

Instead, the images have to be considered relevant to the query only on the basis of a vaguely defined perceptual similarity; there is no clear-cut, exact desired response for the system. Furthermore, the number of images in the database can be really large, imposing strong constraints on the performance of the algorithm. In the retrieval problem it is usually desired that the result images are provided in an order that reflects a similarity scoring, to allow the user to choose interactively the one that fits his needs. This mandates for a matching technique that yields some sort of cost or distance, such as error-correcting graph-matching techniques. As regards the indexing problem, the focus is to obtain a fast screening of the images before performing a retrieval operation, to reduce the search time. So it is not required to provide a distance measure, and it is acceptable if some images that are not relevant to the query are returned in the result set (the converse is not true, i.e., it is not acceptable if indexing excludes strongly relevant images). The main concerns for indexing are how fast it performs, and how many nonrelevant images it is able to filter out.

The distribution of the matching algorithms and of the graph representations used in the image databases field is shown in Fig. 7.

A peculiarity of this applicative context is that there is little agreement on the choice of the kind of graph representation to be used for the images. In most cases images are represented by ARGs [41, 139, 140, 143], but also RAGs [42], directed ordered acyclic graphs [141], shock graphs [104], dual graphs [147], pyramidal



**Fig. 7.** Distribution of (a) the matching algorithms and of (b) the graph representations used within applications in the image databases field

graphs [142], and  $n$  nearest-neighbor graphs [54, 145, 146] are used. As regards the matching phase, mainly error-correcting subgraph isomorphism algorithms are used. In some cases, however, learning techniques have also been employed [141].

Among the papers that address both the indexing and the retrieval problem, Berretti et al. in [41] propose the use of a metric indexing scheme for managing the organization of large archives of ARGs with a common size. In particular, the indexing is performed using  $m$ -trees. They also propose a new algorithm for retrieval, combining the  $A^*$  search with an original look ahead estimate. The estimate is derived as the optimal solution of a weighted assignment, which relaxes the optimal look-ahead problem so as to remove its basic factor of exponential complexity. This sort of minimal simplification results in an extremely well-informed estimate which can still be computed in polynomial time. The database used for testing the approach is composed of about 1,000 images, coming from paintings of the library of a web-museum. For each image of the database 10 further images are generated, synthetically changing color and color positions. In the system they proposed, for querying the database, an user can both select an example image or submit a query by sketch by drawing a set of colored regions and by arranging them in order to represent the expected appearance of the searched images. All the images are modeled with ARGs having a fixed number of nodes, namely eight. Each node come from the clustering of the color histogram in the  $L^*u^*v^*$  color space and the node attributes encode the triple of normalized coordinates of the average color of the cluster. For any two objects corresponding to different regions in the user sketch, the edge attribute encodes the relationship between the regions themselves.

In the paper by Petrakis and Faloutsos [139] ARGs that model medical images are reduced to a vectorial representation, so enabling R-tree indexing, under the assumption that all the graphs contain a set of anchor entities with predefined labels. Non-anchor entities are also allowed, but their number determines a linear degradation in the efficiency of the index. In addition to this indexing technique, the authors propose a subgraph isomorphism algorithm with a distance measure for retrieval. In particular, given an iconic query, all the images under a suitably chosen threshold are selected. As regards the representation, each image is segmented into regions, each one represented by a node. Size, roundness, and orientation of each region have been chosen as node attributes.

Among the papers that mainly address the retrieval problem, Cho and Yoo in [140] use graphs whose nodes represent objects of the image, while the edges encode spatial relations between objects. An object is characterized by its color, the ratio between its area and the whole image area, the ratios between the  $x$ -coordinate and the width and the ratio between the  $y$ -coordinate and the height of the image. The attribute edge can assume one of the eight possible spatial relations between two objects (N, NE, E, SE, S, SW, W, NW). They also define the *prime edge graph*, obtainable from a graph by deleting edges that are unnecessary for representing the structure of the image. The matching is realized with a subgraph isomorphism algorithm that makes use of a similarity measure.

In [143] Folkers et al. propose an exact subgraph isomorphism with a bottom-up strategy. They also define a similarity measure for pruning some isomorphism

checks. The proposed measure takes into account both the contextual and the spatial similarity between ARGs. In their description scheme, once again nodes represent the symbols of the image and the edge the relationships between them.

In the papers by Hancock and Huet [54, 145, 146], the aim is to retrieve 2D images from large databases. In their description scheme, a set of line-patterns are represented by means of a special type of ARG, i.e., a  $N$ -nearest-neighbor graph. In a  $N$ -nearest-neighbor graph, each node represent a line structure segmented from a 2D image. For each node  $n$  of the graph exactly  $N$  edges are created, the ones that link  $n$  to the nodes representing the  $N$  line segments having the closest distances from the line represented by  $n$  itself. Distances between lines are computed by considering distances between their centers.

In [145] the authors use six nearest-neighbor graphs. The line orientation and the line length constitute the attributes of the nodes, while the measure of the relative position and of the relative orientation of two lines whose representing nodes are linked by an edge are the attribute of that edge. The proposed matching is of inexact type; in particular a fuzzy variant of the Hausdorff distance that use only the values of the edge attributes is proposed for comparing graphs. For each image graphs of 3–400 nodes are considered. A first screening of a possible query result is made by considering only the histograms of the edge attributes, that are compared using the Bhattacharyya distance. Then, the fuzzy version of the Hausdorff distance is employed on the  $N$ -nearest images that are found, for refining the search. In [54] the node attributes are two normalized histograms, the one of the relative angles and the one of the relative lengths with respect to the remaining line segments in the pattern. The matching process is realized by means of a Bayesian graph-matching algorithm that utilize a two-step process. Firstly, a correspondence matches between the nodes in the a query pattern and each of the patterns in the database is established. This is made by maximizing an a posteriori measurement probability. In particular, the authors use an extension of the graph-matching technique reported by Wilson and Hancock in [53]; in order to minimize the computational overheads associated with establishing correspondence matches only edge information are used. Once the maximum a posteriori probability correspondence matches have been established for each pattern in the database, the pattern which has maximum matching probability is selected. This is made by using the Bhattacharyya distance for comparing the histogram attributes of the matched nodes.

In [146] Huet et al. present an application of the image retrieval for verifying similarities among different technical drawings representing patents. They use ARGs obtained as six-nearest-neighbor graph from the line drawings, using the same description model of [145]. The matching is of inexact type, and is realized by means of the fuzzy variant of the Hausdorff distance presented in [145].

Among the other representation schemes employed in the literature, in [42] Gregory and Kittler utilizes RAGs. Images are segmented so that a RAG can be built. Each pixel in the image is represented as a 5D vector, where the first three dimensions are the RGB color values for the pixel and the last two dimensions are the pixel coordinates. This feature space is then clustered and to every pixel a label corresponding to the cluster which it has been classified to is given. The region labels

correspond to homogeneous color regions within the image. A connected component analysis stage ensures that only to connected pixels may be assigned the same label. At this point each obtained region is represented by a node, whose attributes are the number of pixels and the average values of the red, green, and blue pixels within the region it represent. The segmentation is further improved by merging adjacent nodes which have a small number of pixels, or “similar” feature space representation. The database used for the testing phase is made up of flag images, that give rise to graphs of about 15 nodes. The matching is performed by using an error-correcting subgraph isomorphism with edit operations and the A\* procedure.

On the other hand, in [104] Sharvit et al. use shock graphs for representing images. The shock graph is directly extracted from the image on the basis of the symmetries exhibited by the image itself. As regards the matching procedure, they use a WGM that is a variant of the method presented in [62]. For testing, they employ a database consisting of binary shapes, and match grayscale images of isolated objects and user-drawn sketches against this database. The resulting shock graphs are made up of few nodes.

Finally, in [147] Park et al. propose the use of dual graphs for representing images. In particular, an ARG called modified color adjacency graph (MCAG) is used for indexing and a spatial variance graph (SVG) is used to disambiguate different images having equal MCAG representations. In a MCAG each node represents a bin of the quantized RGB color histogram. Node attributes are then the pixel count of each RGB chromatic component, while the edge attributes encodes spatial adjacency (based on 8-connectivity) between two color regions. The average number of nodes of a MCAG is about 100. On the other hand, each node of the SVG graph has as attribute the within-class variance relative to the pixels of the node it represents, while each edge attribute encodes the between-class variance. Graph matching is performed by defining a similarity measure directly obtainable by the adjacency matrices of the graphs.

Finally, in [141] a learning technique for facing the retrieval problem is proposed by De Mauro et al. Database images are described by means of RAG that are successively transformed into directed ordered acyclic graphs (DOAG). This transformation becomes necessary because it is more difficult to process undirected graphs than directed ones. The task of learning the search criteria for visual retrieval is accomplished by means of a Recursive Neural Network that map DOAGs into vectors. This net learn to map DOAG representing similar images into near vectors. Then, the retrieval problem is reformulated as the one of finding the  $N$ -nearest neighbors of the vector into which the net transform the DOAG of the query image.

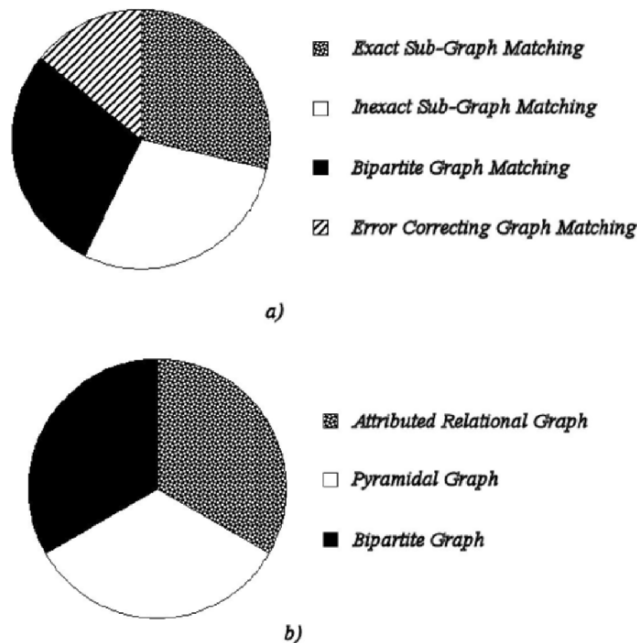
### 3.5 Video Analysis

Among the video analysis problems addressed by using graph-based techniques, retrieval from video databases [21, 142, 148], annotation of video databases [149], object tracking [150–153], and motion estimation [154] have been proposed.

Retrieval from video databases is similar to retrieval from static image databases from a conceptual point of view; the main differences are the considerably larger

size of the databases and the possibility to exploit information about the motion of parts of the scene to improve the retrieval performance. The other problems, instead, are rather peculiar of video analysis. In particular, their common aspect is that they are focused on extracting some kind of information from the sequence of the frames composing a video. This implies a comparison between successive frames, and the need to establish a correspondence between regions of two frames representing the same object or the same part of it. In motion estimation, the goal is to measure the velocity of moving elements of the scene. In object tracking, which can be considered as an evolution of motion estimation, where the application should be able to follow the motion of an object and compute its trajectory, distinguishing the different objects presents in the scene. A further evolution consists in the recognition, on the basis of the object trajectories, of events that bear a specific meaning within the context of the application: this gives rise to the possibility of automatic annotation of the video sequence, allowing a user to perform retrieval with classic, keyword-based search.

Since these problems are quite different each other, it should not be amazing the fact that very different kinds of graphs have been used. In particular, ARGs [21, 148], pyramidal graphs [142, 153], bipartite graphs [150, 152], multivalued neighborhood graphs [151], and medial graphs [154] are used. Obviously, also the matching techniques proposed in the various paper are quite different. The distribution of the matching algorithms and of the graph representations used in this applicative area is summarized in Fig. 8.



**Fig. 8.** Distribution of (a) the matching algorithms and of (b) the graph representations used within applications in the video analysis field

In the framework of the retrieval from video databases, Shearer et al. [21, 148] proposed two different approaches that do not make particular hypotheses on the nature of the video at hand, where Doulamis et al. [142] propose a system specifically tailored for the retrieval of people images in a video database. Furthermore, in the first two papers ARGs are used for representing video frames, while the last one propose the use of pyramidal graphs.

Entering into details, in [21] Shearer et al. describe a new algorithm to solve the largest common subgraph problem. Such algorithm significantly reduces the computational complexity of detection of the largest common subgraph between a known database of models, and a query given online. This approach can be fruitfully applied to video databases. In fact, when searching a video database, we are typically interested in the largest subpicture match that can be found. So, the largest common subgraph method will find the largest subpicture in common between a query image and a database of video frames. As regard the representation, ARGs are used. The authors consider each frame of the video and decompose it into objects. Then, graph nodes represent objects, while the edges are labeled with one of five categories (*Disjoint – Meets – Contains – Belongs to – Overlaps*) that represent the relationships between two objects. The proposed retrieval procedure is realized by using a decision tree algorithm based on a decision tree constructed using the adjacency matrix representation for the model graphs.

A different approach is presented in [148], where a modified version of an algorithm presented by Bunke and Messmer in [18] is proposed. It is able to cope with dynamically changing graphs. Such graphs can be employed for representing videos: the sequence of images that make a video can be represented by means of an initial graphs that represent the initial image and a sequence of graph edit operation that represent the successive images. As in the previous paper, for each image the nodes of the graph represent objects, while the edges encode the spatial relations between objects. An experimental evaluation of the algorithm is also presented, by using query graphs with 9 nodes against models having 4–10 nodes. In particular, the application of this algorithm consists in querying a video database with a sequence of frames. Each query frame is built starting from a number of object labels that can be spatially arranged by the user. The system transforms these query frames into a graph representing the initial frame and into a set of edit operations. Then, it uses the proposed matching algorithm in order to find the video sequence that match the sequence of selected frames.

In [142] Doulamis et al. propose a system for extracting people images from MPEG-coded videos. After a segmentation phase in which objects such as the face, the human body, and the background are extracted from each frame, graphs are used for representing these objects and their spatial relationships. As attribute of the nodes, the average color and the texture of an object, as well as its size and location within the scene are considered. The authors make use of two different types of graphs, one with edge attributes, that encode the direction and the orientation between two objects, and another one without edge attributes. Moreover, in order to enhance the querying flexibility of their system, they also propose a further decomposition of



each node into other graphs, so giving rise to a pyramidal graph representation of the visual content. As an example, the human face can be considered as an object containing the regions of eyes, mouth, and lips, each having their own properties. As regards the problem of retrieval from a video database, they do not clarify in the paper what type of graph-matching technique they use.

A quite peculiar approach to the problem of retrieval from databases is the one presented by Ozer et al. in [149]. The aim of this work is to annotate images and/or videos where a particular object of interest (OOI) is present. So, a simple textual query can be performed to extract images of OOI from a preprocessed database. As an example, they consider cars in video and image libraries, that they describe using ARGs. In case of video sequences, the feature points of an object are tracked and then grouped together according to their moving directions and distances. The object extraction is performed by means of a color image segmentation technique combined with an edge detector algorithm. Since an object usually contains several subobjects (in this case wheels, windows, lights, etc. of a car) a hierarchical segmentation scheme is also proposed. Three different views of a car are considered – front view, rear view, and side view. The three subgraphs relative to these views are joined together to form a unique graph representing all the possible views of the object. As attributes of the nodes, Hu moments and the compactness of the segmented regions are considered. Given two adjacent regions represented by two nodes, the ratio of the areas, the ratio of the perimeters, the relative position and orientation, and the overlapping area between two adjacent regions are the attribute of the edge that links those nodes. As regards the graph-matching procedure, they propose an inexact subgraph matching with a matching cost based on the attribute values, using a depth-first search with a brute force approach.

The papers by Chen et al. [150], Gomila and Mayer [151], and Conte et al. [152, 153] exploit the use of graph matching for object tracking in video sequences. They use different middle level representations and also different matching techniques.

In Conte et al. [152] the definition and the performance assessment of a tracking method devised for video-surveillance applications are presented. The tracking problem is factorized into two subproblems: the first is the definition of a suitable measure of similarity between regions in adjacent frames. Provided with this measure, the second subproblem is the search for an optimal matching between the regions appearing in the frames. As regards the first subproblem (the definition of a similarity measure), several different metrics are proposed, jointly used during the detection phase, according to a sort of signal fusion approach. The subproblem of the optimal matching has been instead formulated in a graph-theoretic framework, and then reduced to a weighted bipartite graph matching, for which a standard algorithm has been used.

Chen et al. [150] apply a shape contour extraction and a shadow deletion to each frame. Therefore they obtain the silhouette of each object within the scene. To each object a probability distribution is associated, that takes into account the intensity values of the area within the object contour. To model the multiobject tracking problem a bipartite graph is used. Each node represents an object and has as attributes its

position, its intensity distribution, and the dimension of its enclosing bounding box. The two classes of nodes in the bipartite graph are the so-called *profile nodes* and *object nodes* that correspond to the objects in the past and the present frame respectively. A bipartite matching algorithm is used to find the best match among nodes of the two successive frames in order to resolve the identities of the objects. If there are unmatched nodes, it implies that new objects have been detected and so new profiles will be created for tracking them within the successive frames.

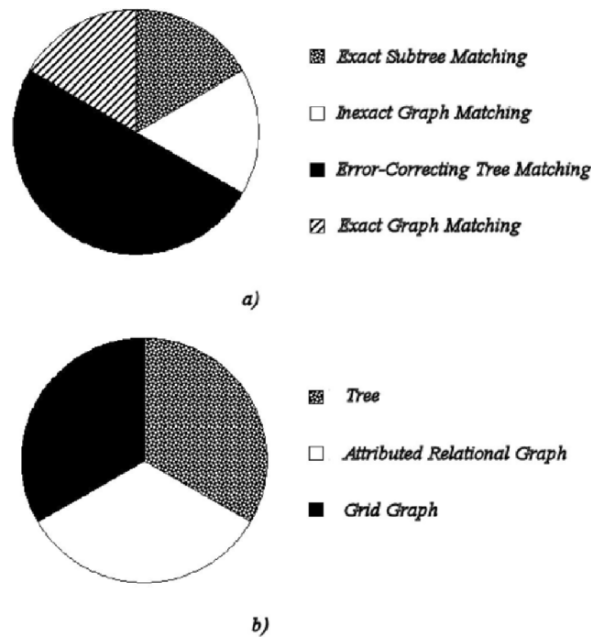
On the other hand, Gomila and Mayer [151] segment the image of each frame on the basis of the color information and represents the segmented image with a multivalued neighborhood graph. Node attributes measure the intrinsic features of the region they represents, while edge attributes represent relational constraints between nodes. Matching graphs relative to two successive frames permits to follow the objects along the video sequence. In order to cope with different segmentation of the same object in two successive frames, split and merge operation are performed on the images before the matching. The proposed matching algorithm is an error correcting one using the relaxation labeling.

Conte et al. [153] use a multiresolution graph pyramid for representing objects at different levels of detail. They use a hierarchical graph-matching procedure to deal with partial occlusions of the objects being tracked. The advantage of their approach is that it uses a fast, coarse grained, weighted bipartite graph matching as long as there are no occlusions in the scene. When two tracked objects come to overlap, a more refined subgraph isomorphism procedure is used to distinguish the parts of the occluding objects, possibly recurring to a finer level or detail until a reasonable solution is found.

Finally, Salotti and Laachfoubi in [154] present an application of motion estimation in aerial videos. Given an aerial video, their aim is to estimate the shift of the part of the image that represents the smoke, in order to collect information for preventing fires. They use *topographic graphs* (that are similar to medial graphs) for describing aerial images. Each frame of the video is segmented on the basis of the color information and the smoke area is described by means of a topographic graph. The shift estimation is performed by means of an inexact matching procedure that defines a cost function for matching nodes of two topographic graphs relative to successive images. These cost function examines only shifts in a small square window centered on each node, since it is reasonable that the move of the smoke is not too fast.

### 3.6 Miscellanea

Besides the application areas detailed in previous sections, there are other application-driven papers that are not strongly related to each other, neither on the basis of the problem they face, nor on the basis of the adopted approach (besides, obviously, the fact that they are based on graph matching). For the sake of completeness, we present here some of these works, although in these cases we cannot individuate any sort of common scheme in the exploitation of graph matching. The papers we have chosen deal with biomedical [38, 133, 155, 156] and the



**Fig. 9.** Distribution of (a) the matching algorithms and of (b) the graph representations used within applications in the miscellanea field

biological [157, 158] applications (see also Fig. 9 for the distribution of the matching algorithms and the graph representations used in this case).

Namely, the paper by Wang et al. [133] is technique driven, and presents an algorithm for finding the largest approximately common substructure of two trees. In the experimental results, the authors discuss its application for finding motifs in multiple RNA secondary structures. On the contrary, both the biomedical application described in [38, 155] have their main focus on the application context and address the problem of the correct identification of coronary arteries (artery labeling) starting from medical images, and are based on prior knowledge about the expected coronary arterial tree (CAT) structure and attributes. They use different graph-matching techniques to realize the labeling.

In the paper by Dumay et al. [38], the authors start from an arteriogram image and project a geometric model of the artery against the image. From this projected model, an ARG made up of about ten nodes is constructed: the nodes of the graph represents arterial segments and have as attributes the position, the mean diameter, and the orientation of the segments, while edges represent the parental relationship (parent-child and grandparent-child) between segments. Starting from the anatomy of a left coronary tree of normal functioning hearts, an inexact graph-matching procedure is used in order to assign anatomic labels to the node of an input image. Since missing branch and/or false structures can corrupt the input image, a cost function is defined in order to cope with transformations (substitution, insertion, and deletion of a node

and/or an edge) of the input graph. An A\* algorithm is used to perform the state space search.

Also the paper by Charnoz et al. [156] faces a somewhat analogous problem: the matching of several CAT-images of the intrahepatic vascular system of a same patient, acquired at different times. The authors propose an error-correcting tree matching algorithm that is robust with respect to topological modifications.

In the paper by Haris et al. [155], the authors use ARGs to represent the CAT. Starting from the input image, the CAT is detected by constructing an approximation of its centerline and borders. This results in a directed acyclic graph representing the CAT. The attribute of each node of this graph are the position of the artery element it represents, the direction of the artery and its approximate width. Given the input graph and a 3D CAT model which encapsulates the expected anatomic and geometric structure of a normal human CAT, a graph-matching algorithm assigns the appropriate labels to the input CAT using weighted maximal cliques on the association graph corresponding to the two given graphs. So the labeling problem is reformulated as one of finding the best maximal clique of the association graph.

A biological application is the identification of diatoms described by Fischer et. al. [157] and Ambauen et al. [158]. Diatoms are unicellular algae found in water and in other places where there is sufficient humidity and light for allowing photosynthesis. The technique used for describing diatom images is the same used in the face recognition field. A middle-level representation based on labeled grid graphs is used. On each image a rectangular  $16 \times 8$  grid is superimposed and each node of the graph is associated to a rectangle of the image. Each node is labeled with 13 features derived from the gray-level co-occurrence matrices and from the Gabor coefficients. In [157] the matching procedure can be seen as a simple form of error-correcting graph matching. A dissimilarity measure is evaluated between two grid graphs as the sum of the distance between the feature vectors associated to the nodes. Moreover, in order to cope with geometric distortions, also translations of the nodes are allowed and a specific cost is introduced into the dissimilarity measure in order to weigh such translations. In [158] a more complex matching algorithm is proposed, based on the addition of new edit operations to the classical set of deletion, insertion, and mutation.

## 4 Performance Comparison

By reviewing the wide literature in the field of graph matching, it appears evident that the habit of proposing more and more new algorithms is prevailing against the need of assessing the performance of the existing ones in an objective way. The characterization of the graph-matching algorithms proposed up to now would instead allow potential users to predict the performance of an algorithm – at least to some degree – and could thus lead to substantial savings in system development time.

Starting from these considerations, the IAPR-TC15 community in the second GbR workshop held in 1999 (see the TC15 website at the address: <http://>

[www.iapr-tc15.unisa.it](http://www.iapr-tc15.unisa.it)) declared the need for a serious benchmarking activity in the context of graph-matching algorithms. According to the proposal made in [159], such activity could start with exact graph-matching algorithms, by considering different kinds of morphism and by suitably defining the number of graphs to be matched, the size of the input graphs and the graph structure. Even if only exact graph-matching algorithms are considered, it is also worth noting that the matching problem may have, as pointed out in [160]:

- No solution (e.g., if the graphs are not isomorphic, or if there is no subgraph isomorphic to the given graph)
- One solution
- More than one solution (e.g., a square mesh matches its versions rotated by 90, 180, and 360°)

Generally, not only a fast solution for the second case is required, since a small disturbance caused by noise may transform it into the first case. Then, it is relevant the time needed by an algorithm for finding out that there is no perfect solution (*nonmatching* time). Nonmatching times are also crucial factors in the context of graph database filtering, as noted in [161]. Finally, the third case concerns algorithms that can find one or all the possible solutions to the given matching problem.

If we try to understand the reasons why up to now only a small number of serious attempts [11, 162–164] has been made for comparing graph-matching algorithms, we can easily recognize that one of the main difficulties is the fact that only in the last few years some standard databases of graphs specifically designed for this purpose have been made available. The creation of a graph database, in fact, is definitely not a simple task, since several issues have to be faced [11]. Generally speaking, two approaches can be followed for generating a database; a first way is to start from graphs obtained from real data, otherwise the database can be obtained synthetically. Although the first approach allows us to obtain rather realistic graphs, it is generally more expensive as it requires the collection of real data and the selection of the set of algorithms to be used for obtaining graphs from data. In this case the graphs are dependent on both the domain under consideration and the preprocessing algorithm used, reducing significantly the generality of the database and its reusability in other contexts. On the contrary, the artificial generation of graphs is not only simpler and faster than collecting graphs from real applications, but also allows us to control the variation of several critical parameters of the underlying graph population, such as the average number of nodes, average number of edges per node, number of different labels, and so on.

By following the latter approach, three proposals have been recently made in the scientific community, in order to provide standard graph databases. The first two [164, 165] gave rise to databases of synthetically generated graphs explicitly devised for benchmarking (sub)graph isomorphism algorithms and MCS algorithms, respectively, while the third proposal [166] is based on the generation of a database of artificial images – by using a set of attributed plex grammars – and of their corresponding graph representations.

#### 4.1 Graph Databases

The choice of the kind of graphs to be included in the first two cited database derived from an analysis of the graphs mainly used by members of the IAPR-TC15 community. Both databases are structured in pairs of graphs. In the first database, two categories of pairs of graphs have been introduced, namely pairs made of isomorphic graphs and pairs made of graphs in which the second graph is a subgraph of the first one. In the second database each pair of graphs has a MCS of at least two nodes.

In both cases, each category of pairs is made up of graphs that are different for structure and size. In particular, the following kinds of graphs have been considered (see [165] for a detailed discussion about their properties and the parameters characterizing them):

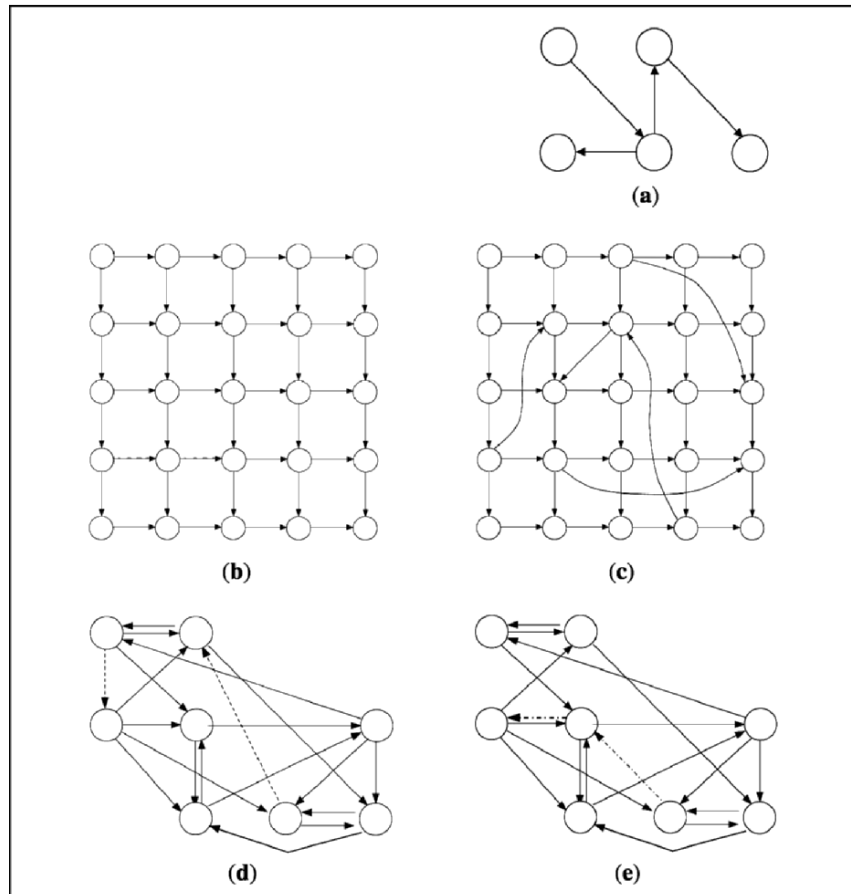
- *Randomly Connected Graphs*
- *Regular Meshes*, with different dimensionality: 2D, 3D and 4D
- *Irregular Meshes*
- *Bounded Valence Graphs*
- *Irregular Bounded Valence Graphs*

*Randomly connected graphs* are graphs in which the edges connect nodes without any structural regularity (see Fig. 10a). They can model applications in which objects (represented by nodes) can establish relations (represented by edges) with any other objects (not only the surrounding ones) independently of the relative positions. This hypothesis typically occurs in the middle and high processing levels of an image processing task.

*Regular meshes* (see Fig. 10b) have been introduced for simulating applications dealing with regular structures as those operating at the lower levels of an image processing task; while *Irregular mesh-connected* graphs (see Fig. 10c) can be used for simulating the behavior of graph-matching algorithms in presence of slightly distorted meshes. *Bounded valence graphs* (see Fig. 10d) model applications in which each object establish a fixed number of relations with other object, not necessarily with those belonging to its neighborhood. In order to introduce some irregularities in these kind of graphs, *Irregular bounded valence graphs* have been introduced too (see Fig. 10e).

So, in the first database (hereinafter denoted as *ISO-DB*) a total of 72,800 pairs of graphs have been generated: 18,200 pairs of isomorphic graphs and 54,600 pairs for which a subgraph isomorphism exists. Each kind of graphs has pairs of different size, ranging from few dozens to about 1,000 nodes (i.e., small and medium size graphs according to the classification presented in [159]). For each size and kind of graphs 100 different pairs have been generated. Moreover, in case of graph subgraph isomorphism, pairs in which the two graphs have three different size ratios have been generated.

The graphs composing the whole database have been distributed on a CD during the third IAPR-TC15 Workshop on Graph-Based Representations in Pattern Recognition and are also publicly available on the web at the URL: <http://amalfi.dis.unina.it/graph>.



**Fig. 10.** (a) An example of a randomly connected graph. (b) A 2D regular mesh with size  $5 \times 5$  and (c) an irregular mesh obtained by adding five further edges to the graph (b). For each added edge, the starting and the ending nodes are randomly determined according to a uniform distribution. (d) A bounded valence graph with a valence equal to 5; (e) an irregular bounded valence graph, obtained from the graph (d) by moving the two dashed edges

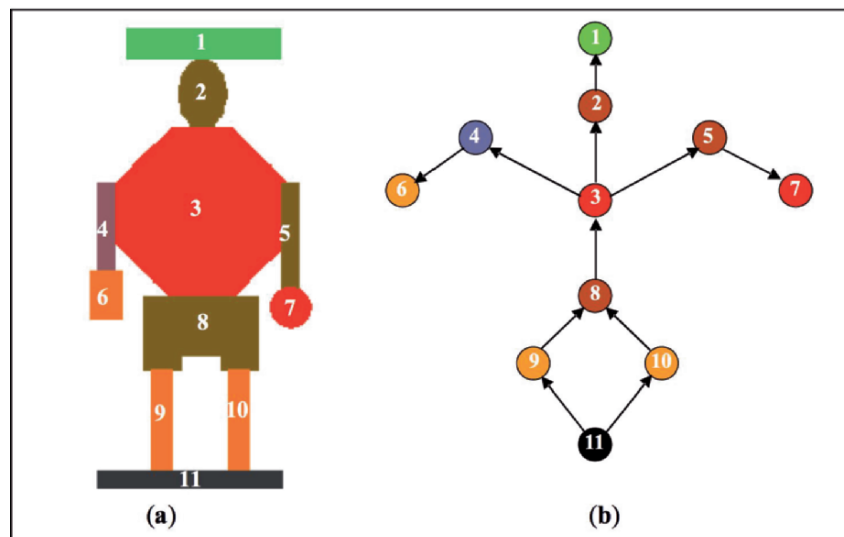
For the second database (from now on denoted as *MCS-DB*), a total of 81,400 pairs of labeled graphs have been generated. The choice of the labeling comes from the need of restricting the number of possible node or edge pairings because of the complexity of the MCS problem. For each of the above-mentioned kind of graphs, pairs of graphs having different sizes  $N$ , ranging from 10 to 100, have been included in the database. Moreover, for each value of  $N$ , five different sizes of the MCS have been taken into account and 100 pairs of graphs have been generated for each size. As regards the labeling, the authors proposal was to generate random values for the attributes, since any other choice would imply assumptions about an application

dependent model of the represented graphs. The whole database is publicly available on the web at the URL: <http://amalfi.dis.unina.it/graph>.

The third database comes from an image generation method (downloadable from the URL: <http://www.artificial-neural.net/>) based on a combination of attribute grammars and plex grammars [167]. A plex grammar is a generic mechanism allowing to specify a number of rules that describe how an image is built up from simpler constituents. Because the rules can be recursive, a potentially infinite set of images can be described by a finite number of rules.

According to the authors' proposal [166], an image generated according to the previous method can be simply converted into a graphical representation. An example referring to the image of a policeman is reported in Fig. 11. The nodes of the graph correspond to regions in the image, while the edges represent spatial relations between the regions. A number of different attributes can be computed for each node and each edge. Examples of node attributes are color, center of gravity, and size of a region; while edge attributes represent geometric relations between regions (e.g., angle and distance of centers of gravity).

Since the rules of the attributed plex grammar are defined by the user, there are no restrictions on the underlying domain and there is not a predefined graphical representation of an image. In fact, through a number of parameters the user can choose the representation that is most suitable for the problem at hand. It is then possible to create image databases, together with their graphical representations, for various kinds of PR problems involving graphs. Exact and inexact graph matching, supervised and unsupervised learning of graphical representations from examples and graph clustering are actual examples.



**Fig. 11.** (a) The image of a policeman generated according to the method proposed in [166] and (b) its representation as a graph. Labels associated with each node are not shown



Even if some authors [141, 163, 168] used the above-described image database for testing their graph-based approaches, no paper reporting a serious comparison of algorithms on graphs generated from it has been presented so far. On the other hand, there are some papers presenting benchmarking activities using graphs extracted from both the *ISO-DB* and the *MCS-DB*.

## 4.2 Benchmarking Activities

One of the first attempts to compare graph-matching algorithms has been made in [162]. Five inexact graph-matching algorithms have been considered and compared with respect to both the speed of algorithms, capacity for classification and suitability for different kind of graphs. Algorithms were tested in two real-world classification problems. Nevertheless, since the size of the graphs used in the tests ranges from three to nine nodes, the current usefulness of the obtained results is quite limited.

More recently, three works have tried to follow the indications of the IAPR TC 15 community for carried out a noteworthy activity in the context of graph-matching algorithms.

In particular, in [11], four exact graph-matching algorithms have been compared with respect to the times needed for finding a match on pairs of isomorphic graphs extracted from the *ISO-DB*. In particular the Ullmann's algorithm [5], the algorithm by Schmidt and Druffel [6], the VF2 algorithm [13] and Nauty [16] have been considered. As it could be expected, the authors conclude that an algorithm performing definitively better than all the others does not exist. In particular, for randomly connected graphs, the Nauty algorithm is the better if the graphs are quite dense and/or of quite large size. For smaller and quite sparse graphs, on the contrary, VF2 performs better. On more regular graphs, i.e., on 2D meshes, VF2 is definitely the best algorithm: in this case the Nauty algorithm is even not able to find a solution for graphs bigger than few dozens of nodes. In case of bounded valence graph, if the valence is small, VF2 is always the best algorithm, while for bigger values of the valence the Nauty algorithm is more convenient if the size of the graphs is small.

Two of the above-described algorithms, namely VF2 and Ullmann, have been also extensively compared in [161] with respect to their *nonmatching* times, i.e., the time needed for declaring that there is not a match between two given graphs. The comparison has been carried out on graphs extracted from the *ISO-DB*; both the cases of pairs of graphs with the same number of nodes and with a different number of nodes have been considered.

According to the tests reported in the paper, the *nonmatching* times obtained by the Ullmann's algorithm are almost always smaller than those achieved by the VF2 algorithm when pairs of graphs with the same number of nodes are taken into account. Only in case of very regular graphs, i.e., high-dimensional meshes, the *nonmatching* times of the two algorithms are practically identical. This behavior is substantially confirmed in case of graphs with different number of nodes, even if in this case the *nonmatching* times of VF2 are smaller for regular bounded valence graphs with at least 70 nodes and for high-dimensional meshes.

Finally, in [164] the matching times of three MCS algorithms, namely those proposed by McGregor [169], by Balas and Yu [170], and by Durand et al. [15], have been compared by using pairs of graphs extracted from the *MCS-DB*.

According to the authors, also in this case it does not exist an algorithm that is definitively better than the others. In particular, for randomly connected graphs, the McGregor algorithm is the best one if the graphs are quite sparse and/or of quite small size. For larger and quite dense graphs, on the contrary, the Durand et al. algorithm performs better. If the MCS has a more regular structure, i.e., on mesh-like MCS, the McGregor algorithm is in most cases the best algorithm; the Durand et al. algorithm performs better only for small and dense graphs. On the contrary, when the irregularity degree grows up, the Balas–Yu algorithm performs better for large and dense graphs. In case of bounded valence graphs, the McGregor algorithm is the best one if the valence is small, while for larger values of the valence the Durand et al. algorithm is more convenient when the graphs to be matched are dense. If the graphs are both dense and large, the Balas Yu algorithm is the best one. Finally, when an irregularity degree is added to the bounded valence graphs, the Durand–Pasari algorithm performs better in most cases, even if the Balas–Yu algorithm is still winning for large graphs.

## 5 Conclusions

In this paper we have presented a comprehensive review of graph-matching methods used in PR and computer vision applications, highlighting the relationships between the application domain and specific problem on one side, and the adopted graph representation and matching technique on the other side. An evaluation of the proposed methods and tools for assessing the performance of a graph-matching algorithm completed our work.

All together, these two parts provide useful information to applied researchers for deciding which graph-based technique best fits their needs.

## References

1. Conte D, Foggia P, Sansone C, Vento M (2004) Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence* 18(3):265–298
2. Aho AV, Hopcroft JE, Ullman JD (1974) *The Design and Analysis of Computer Algorithms*. Addison Wesley, Reading, MA
3. Hopcroft JE, Wong J (1974) Linear time algorithm for isomorphism of planar graphs. *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, pp. 172–184
4. Luks EM (1982) Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and Systems Sciences* 25:42–65
5. Ullman JR (1976) An algorithm for subgraph isomorphism. *Journal of the Association for Computer Machinery* 23:31–42

6. Schmidt DC, Druffel LE (1976) A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. *Journal of the ACM* 23:433–445
7. Ghahraman DE, Wong AKC, Au T (1980) Graph monomorphism algorithms. *IEEE Transactions on Systems, Man and Cybernetics* 10(4):189–197
8. Cordella LP, Foggia P, Sansone C, Vento M (2000) Fast graph matching for detecting CAD image components. *Proceedings of 15th International Conference on Pattern Recognition*, pp. 1034–1037
9. Cordella LP, Foggia P, Sansone C, Tortorella F, Vento M (1998) Graph matching: a fast algorithm and its evaluation. *Proceedings of 14th International Conference on Pattern Recognition*, pp. 1582–1584
10. Cordella LP, Foggia P, Sansone C, Vento M (1999) Performance evaluation of the VF graph matching algorithm. *Proceedings of the International Conference on Image Analysis and Processing*, pp. 1172–1177
11. De Santo M, Foggia P, Sansone C, Vento M (2003) A large database of graphs and its use for benchmarking graph isomorphism algorithms. *Pattern Recognition Letters* 24(8):1067–1079
12. Cordella LP, Foggia P, Sansone C, Vento M (2001) An improved algorithm for matching large graphs. *Proceedings of the Third IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 149–159
13. Cordella LP, Foggia P, Sansone C, Vento M (2004) A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(10):1367–1372
14. Larrosa J, Valiente G (2002) Constraint satisfaction algorithms for graph pattern matching. *Mathematical Structural in Computer Science* 12:403–422
15. Durand PJ, Pasari R, Baker JW, and Tsai CC (1999) An efficient algorithm for similarity analysis of molecules. *Internet Journal of Chemistry* 2
16. McKay BD (1981) Practical graph isomorphism. *Congressus Numerantium* 30:45–87
17. Foggia P, Sansone C, Vento M (2001) A performance comparison of five algorithms for graph isomorphism. *Proceedings of the Third IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 188–199
18. Bunke H, Messmer BT (1997) Recent advances in graph matching. *International Journal of Pattern Recognition and Artificial Intelligence* 11(1):169–203
19. Messmer BT, Bunke H (1999) A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition* 32(12):1979–1998
20. Shearer K, Bunke H, Venkatesh S, Kieronska S (1997) Efficient graph matching for video indexing. *Computing. Supplement* 12:53–62
21. Shearer K, Bunke H, Venkatesh S (2001) Video indexing and similarity retrieval by largest common subgraph detection using decision trees. *Pattern Recognition* 34(5):1075–1091
22. Lazarescu M, Bunke H, Venkatesh S (2000) Graph matching: fast candidate elimination using machine learning techniques. *Proceeding of the Joint IAPR International Workshops SSPR and SPR*, pp. 236–245
23. Irniger C, Bunke H (2001) Graph matching: filtering large databases of graphs using decision trees. *Proceedings of the Third IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 239–249
24. Bunke H (1997) On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters* 18(8):689–694
25. Bunke H, Shearer K (1998) A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters* 19(3):255–259

26. Bunke H (1999) Error correcting graph matching: On the influence of the underlying cost function. *IEEE Transactions on PAMI* 21(9):917–922
27. Fernndez ML, Valiente G (2001) A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters* 22(6):753–758
28. Wallis WD, Shoubridge P, Kraetz M, Ray D (2001) Graph distances using graph union. *Pattern Recognition Letters* 22(6):701–704
29. Tsai WH, Fu KS (1979) Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics* 9(12):757–768
30. Tsai WH, Fu KS (1983) Subgraph error-correcting isomorphisms for syntactic pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics* 13(1):48–61
31. Wong AKC, You M, Chan SC (1990) An algorithm for graph optimal monomorphism. *IEEE Transactions on Systems, Man and Cybernetics* 20(3):628–638
32. Eshera MA, Fu KS (1984) A similarity measure between attributed relational graphs for image analysis. *Proceedings of the Seventh International Conference on Pattern Recognition*, pp. 75–77
33. Eshera MA, Fu KS (1984) A graph distance measure for image analysis. *IEEE Transactions on Systems, Man and Cybernetics* 14(3):398–408
34. Sanfeliu A, Fu KS (1983) A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics* 13(3):353–363
35. Ghahraman DE, Wong AKC, Au T (1980) Graph optimal monomorphism algorithms. *IEEE Transactions on Systems, Man and Cybernetics* 10(4):181–188
36. Shapiro LG, Haralick RM (1981) Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 3(5):504–519
37. Shapiro LG, Haralick RM (1985) A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 7(1):90–94
38. Dumay ACM, van der Geest RJ, Gerbrands JJ, Jansen E, Reiber JHC (1992) Consistent inexact graph matching applied to labelling coronary segments in arteriograms. *Proceedings of the International Conference on Pattern Recognition, Conference C*, pp. 439–442
39. Berretti S, Del Bimbo A, Vicario E (2000) A look-ahead strategy for graph matching in retrieval by spatial arrangement. *International Conference on Multimedia and Expo*, pp. 1721–1724
40. Berretti S, Del Bimbo A, Vicario E (2000) The computational aspect of retrieval by spatial arrangement. *Proceedings of 15th International Conference on Pattern Recognition*, pp. 1047–1051
41. Berretti S, Del Bimbo A, Vicario E (2001) Efficient matching and indexing of graph models in content-based retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(10):1089–1105
42. Gregory L, Kittler J (2002) Using graph search techniques for contextual colour retrieval. *Proceeding of the Joint IAPR International Workshops SSPR and SPR*, pp. 186–194
43. Cordella LP, Foggia P, Sansone C, Vento M (1996) An efficient algorithm for the inexact matching of ARG graphs using a contextual transformational model. *Proceedings of the 13th International Conference on Pattern Recognition*, pp. 180–184
44. Cordella LP, Foggia P, Sansone C, Vento M (1997) Subgraph transformations for inexact matching of attributed relational graphs. *Computing. Supplement* 12:43–52

45. Serratos F, Alquezar R, Sanfeliu A (1999) Function-described graphs: A fast algorithm to compute a sub-optimal matching measure. Proceedings of the Second IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, pp. 71–77
46. Serratos F, Alquezar R, Sanfeliu A (2000) Efficient algorithms for matching attributed graphs and function-described graphs. Proceedings of 15th International Conference on Pattern Recognition, pp. 867–872
47. Rocha J, Pavlidis T (1994) A shape analysis model with applications to a character recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(4):393–404
48. Wang C, Abe K (1995) Region correspondence by inexact attributed planar graph matching. Proceedings of the Fifth International Conference on Computer Vision, pp. 440–447
49. Llados J, Marti E, Villanueva JJ (2001) Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(10):1137–1143
50. Fischler M, Elschlager R (1973) The representation and matching of pictorial structures. *IEEE Transactions on Computers* (22):67–92
51. Kittler J, Hancock ER (1989) Combining evidence in probabilistic relaxation. *International Journal of Pattern Recognition and Artificial Intelligence* 3:29–51
52. Christmas WJ, Kittler J, Petrou M (1995) Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17(8):749–764
53. Wilson RC, Hancock ER (1997) Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(6):634–648
54. Huet B, Hancock ER (1999) Shape recognition from large image libraries by inexact graph matching. *Pattern Recognition Letters* 20:1259–1269
55. Myers R, Wilson RC, Hancock ER (2000) Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(6):628–635
56. Torsello A, Hancock ER (2001) Computing approximate tree edit-distance using relaxation labeling. Proceedings of the Third IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, pp. 125–136
57. Luo B, Hancock ER (2001) Structural graph matching using the EM algorithm and singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(10):1120–1136
58. Dempster AP, Laird AP, and Rubin DB (1977) Maximum-likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B* 39:1–38
59. Almohamad HA, Duffuaa SO (1993) A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15(5):522–525
60. Lawler ES (2001) *Combinatorial Optimization: Networks and Matroids*. Dover, New York
61. Rangarajan A, Mjolsness ED (1996) A Lagrangian relaxation network for graph matching. *IEEE Transactions on Neural Networks* 7(6):1365–1381
62. Gold S, Rangarajan A (1996) A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18(4):377–388
63. Bomze IM (1997) Evolution towards the maximum clique. *Journal of Global Optimization* 10:143–164
64. Pelillo M (1995) Relaxation labeling networks for the maximum clique problem. *Journal of Artificial Neural Networks* 2:313–328

65. Pelillo M, A Jagota (1995) Feasible and infeasible maxima in a quadratic program for maximum clique. *Journal of Artificial Neural Networks* 2:411–420
66. Rosenfeld A, Hummel RA, Zucker SW (1976) Scene labelling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics* 6(6):420–433
67. Pelillo M (1998) A unifying framework for relational structure matching. *Proceedings of 14th International Conference on Pattern Recognition*, pp. 1316–1319
68. Pelillo M, Siddiqi K, Zucker SW (1999) Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(11):1105–1120
69. Branca A, Stella E, Distanto A (1999) Feature matching by searching maximum clique on high order association graph. *Proceedings of the 10th International Conference on Image Analysis and Processing*, pp. 642–658
70. Medasani S, Krishnapuram R (1999) A fuzzy approach to content-based image retrieval. *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 1251–1260
71. Medasani S, Krishnapuram R, Choi YS (2001) Graph matching by relaxation of fuzzy assignments. *IEEE Transactions on Fuzzy Systems* 9(1):173–182
72. van Wyk BJ, van Wyk MA (2002) Non-bayesian graph matching without explicit compatibility calculations. *Proceeding of the Joint IAPR International Workshops SSPR and SPR*, pp. 74–82
73. van Wyk BJ, van Wyk MA, Hanrahan HE (2002) Successive projection graph matching. *Proceeding of the Joint IAPR International Workshops SSPR and SPR*, pp. 263–271
74. Umeyama S (1988) An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10(5):695–703
75. Xu L, King I (2001) A PCA approach for fast retrieval of structural patterns in attributed graphs. *IEEE Transactions on Systems, Man and Cybernetics, Part B* 31(5):812–817
76. Carcassoni M, Hancock ER (2001) Weighted graph-matching using modal clusters. *Proceedings of the Third IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 260–269
77. Kosinov S, Caelli T (2002) Inexact multisubgraph matching using graph eigenspace and clustering models. *Proceedings of the Joint IAPR International Workshops SSPR and SPR*, pp. 133–142
78. Shokoufandeh A, Dickinson S (2001) A unified framework for indexing and matching hierarchical shape structures. In: Arcelli C, Cordella LP, Sanniti di Baja G (eds), *Workshop on Visual Form, LNCS 2059*, pp. 67–84
79. Jagota A, Pelillo M, Rangarajan A (2000) A new deterministic annealing algorithm for maximum clique. *Proceedings of the International Joint Conference on Neural Networks* 6:505–508
80. Gendreau M, Salvail L, Soriano P (1993) Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research* 41:385–403
81. Williams ML, Wilson RC, Hancock ER (1999) Deterministic search for relational graph matching. *Pattern Recognition* 32(7):1255–1271
82. Eshera MA, Fu KS (1986) An image understanding system using attributed symbolic representation and inexact graph-matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8(5):604–618
83. Hong P, Wang R, Huang T (2000) Learning patterns from images by combining soft decisions and hard decisions. *Proceedings of the 2000 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 79–83
84. Jiang X, Bunke H (1998) Marked subgraph isomorphism of ordered graphs. *Proceeding of the Joint IAPR International Workshops SSPR and SPR*, pp. 122–131

85. Seong D, Kim HS, Park KH (1993) Incremental clustering of attributed graphs. *IEEE Transactions on Systems, Man and Cybernetics* 23(5):1399–1411
86. Zhang H, Yan H (1999) Graphic matching based on constrained Voronoi diagrams. *Proceedings of the Fifth International Symposium on Signal Processing and its Applications*, pp. 431–434
87. Luo B, Robles-Kelly A, Torsello A, Wilson RC, Hancock ER (2001) Clustering shock trees. *Proceedings of the Third IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 217–228
88. Suganthan PN, Teoh EK, Mital D (1995) Pattern recognition by graph matching using the potts MFT neural networks. *Pattern Recognition* 28(7):997–1009
89. Torsello A, Hancock ER (2002) Learning structural variations in shock trees. *Proceeding of the Joint IAPR International Workshops SSPR and SPR*, pp. 113–122
90. Perchant A, Boeres C, Bloch I, Roux M, Ribeiro C (1999) Model-based scene recognition using graph fuzzy homomorphism solved by genetic algorithm. *Proceedings of the Second IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 61–70
91. Boeres MC, Ribeiro CC, Bloch I (2004) A Randomized Heuristic for Scene Recognition by Graph Matching. In: Ribeiro CC, Martins SL (eds) *Experimental and Efficient Algorithms: Third International Workshop*, LNCS 3059, pp. 100–113
92. Wilson RC, Hancock ER (1999) Graph matching with hierarchical discrete relaxation. *Pattern Recognition Letters* 20(10):1041–1052
93. Neuhaus M, Bunke H (2003) An error-tolerant approximate matching algorithm for attributed planar graphs and its application to fingerprint classification. In: Fred A, Caelli T, Camphilho A (eds), *SSPR 2004*, LNCS 3138
94. Jia J, Abe K (1998) Automatic generation of prototypes in 3D structural object recognition. *Proceedings of the Fourteenth International Conference on Pattern Recognition*, pp. 697–700
95. Shasha D, Wang JTL, Zhang K, Shih FY (1994) Exact and approximate algorithms for unordered tree matching. *IEEE Transactions on Systems, Man and Cybernetics* 24(4):668–678
96. Sanfeliu A, Serratoso F, Alquezar R (2004) Second-order random graphs for modeling sets of attributed graphs and their application to object learning and recognition. *International Journal of Pattern Recognition and Artificial Intelligence* 18(3):375–396
97. Robles-Kelly A, Hancock ER (2005) Graph edit distance from spectral seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(3):365–378
98. Englert R, Cremers AB, Seelmann-Eggebert J (1997) Recognition of polymorphic pattern in parameterized graphs for 3D building reconstruction. *Computing. Supplement* 12:11–20
99. Meth R, Chellappa R (1996) Target indexing in synthetic aperture radar imagery using topographic features. *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2152–2155
100. Li WJ, Lee T (2000) Object recognition by sub-scene graph matching. *IEEE International Conference on Robotics and Automation*, pp. 1459–1464
101. Belongie S, Malik J (2000) Matching with shape contexts. *Proceedings of IEEE Workshop on Content-based Access of Image and Video Libraries*, pp. 20–26
102. Sebastian TB, Klein PN, Kimia BB (2005) Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(5): 550–571

103. Koo JH, Yoo SI (1998) A structural matching for two-dimensional visual pattern inspection. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 4429–4434
104. Sharvit D, Chan J, Tek H, Kimia BB (1998) Symmetry-based indexing of image databases. *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp. 56–62
105. Bauckhage C, Wachsmuth S, Sagerer G (2001) 3D assembly recognition by matching functional subparts. *Proceedings of the Third IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 95–104
106. Olatunbosun S, Dowling GR, Ellis TJ (1996) Topological representation for matching coloured surfaces. *Proceedings of the International Conference on Image Processing*, pp. 1019–1022
107. Fuchs F, Le Men H (2000) Efficient subgraph isomorphism with ‘A Priori’ knowledge. *Proceeding of the Joint IAPR International Workshops SSPR and SPR*, pp. 427–436
108. Fuchs F, Le Men H (1999) Building reconstruction on aerial images through multi-primitive graph matching. *Proceedings of the Second IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 21–30
109. Messmer BT, Bunke H (1998) A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(5):493–504
110. Bron C, Kerbosch J (1973) Finding all cliques of an undirected graph. *Communications of ACM* 16(9):575–577
111. Foggia P, Sansone C, Tortorella F, Vento M (1999) Definition and validation of a distance measure between structural primitives. *Pattern Analysis and Applications* 2: 215–227
112. Foggia P, Genna R, Vento M (2001) Symbolic vs. connectionist learning: an experimental comparison in a structured domain. *IEEE Transactions on Knowledge and Data Engineering* 13(2):176–195
113. Chan KP (1996) Learning templates from fuzzy examples in structural pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics, Part B* 26(1):118–123
114. Lee RST, Liu JNK (1999) An oscillatory elastic graph matching model for recognition of offline handwritten Chinese characters. *Third International Conference on Knowledge-Based Intelligent Information Engineering Systems*, pp. 284–287
115. Suganthan PN, Yan H (1998) Recognition of handprinted Chinese characters by constrained graph matching. *Image and Vision Computing* 16(3):191–201
116. Liu JZ, Ma K, Cham WK, Chang MMY (2000) Two-layer assignment method for online Chinese character recognition. *IEEE Proceedings Vision, Image and Signal Processing* 147(1):47–54
117. Lu SW, Ren Y, Suen CY (1991) Hierarchical attributed graph representation and recognition of handwritten Chinese characters. *Pattern Recognition* 24:617–632
118. Chen LH, Lieh JR (1990) Handwritten character recognition using a 2-layer random graph model by relaxation matching. *Pattern Recognition* 23:1189–1205
119. Suganthan PN, Teoh EK, Mital D (1995) A self organizing Hopfield network for attributed relational graph matching. *Image and Vision Computing* 13(1):61–73
120. Pavlidis T, Sakoda WJ, Shi H (1995) Matching graph embeddings for shape analysis. *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 729–733
121. Filatov A, Gitis A, Kil I (1995) Graph-based handwritten digit string recognition. *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 845–848



122. Lladós J, López-Krahe J, Martí E (1996) Hand drawn document understanding using the straight line Hough transform and graph matching. *Proceedings of the 13th International Conference on Pattern Recognition*, pp. 497–501
123. Changhua L, Bing Y, Weixin X (2000) Online hand-sketched graphics recognition based on attributed relational graph matching. *Proceedings of the Third World Congress on Intelligent Control and Automation*, pp. 2549–2553
124. Jiang X, Munger A, Bunke H (1999) Synthesis of representative symbols by computing generalized median graphs. *Proceedings of the International Workshop on Graphics Recognition GREC'99*, pp. 187–194
125. Lades M, Vorbruggen JC, Buhmann J, Lange J, von der Malsburg C, Wurz RP, Konen W (1993) Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers* 42(3):300–311
126. Wiskott L, Fellous JM, Kruger N, von der Malsburg C (1997) Face recognition by elastic bunch graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19(7):775–779
127. Wiskott L (1997) Phantom faces for face analysis. *Proceedings of the International Conference on Image Processing*, pp. 308–311
128. Lim R, Reinders MJT (2001) Facial landmarks localization based on fuzzy and gabor wavelet graph matching. *The Tenth IEEE International Conference on Fuzzy Systems*, pp. 683–686
129. Kotropoulos C, Tefas A, Pitas I (2000) Frontal face authentication using morphological elastic graph matching. *IEEE Transactions on Image Processing* 9(4):555–560
130. Tefas A, Kotropoulos C, Pitas I (2001) Using support vector machines to enhance the performance of elastic graph matching for frontal face authentication. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(7):735–746
131. Duc B, Fischer S, Bigun J (1999) Face authentication with Gabor information on deformable graphs. *IEEE Transactions on Image Processing* 8(4):504–516
132. Lyons MJ, Budynek J, Akamatsu S (1999) Automatic classification of single facial images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(12):1357–1362
133. Wang M, Iwai Y, Yachida M (1998) Expression recognition from time-sequential facial images by use of expression change model. *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 354–359
134. Elagin E, Steffens J, Neven H (1998) Automatic pose estimation system for human faces based on bunch graph matching technology. *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 136–141
135. Triesch J, von der Malsburg C (2001) A system for person-independent hand posture recognition against complex backgrounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(12):1449–1453
136. Burge M, Burger W (2000) Ear biometrics in computer vision. *Proceedings of 15th International Conference on Pattern Recognition*, pp. 822–826
137. Maio D, Maltoni D (1996) A structural approach to fingerprint classification. *Proceedings of the 13th International Conference on Pattern Recognition*, pp. 578–585
138. Fan KC, Liu CW, Wang YK (1998) A fuzzy bipartite weighted graph matching approach to fingerprint verification. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 4363–4368
139. Petrakis GM, Faloutsos C (1997) Similarity searching in medical image databases. *IEEE Transaction on Knowledge and Data Engineering* 9(3):435–447

140. Cho SJ, Yoo SJ (1998) Image retrieval using topological structure of user sketch. *IEEE International Conference on Systems, Man and Cybernetics*, pp. 4584–4588
141. de Mauro C, Diligenti M, Gori M, Maggini M (2003) Similarity learning for graph-based image representations. *Pattern Recognition Letters* 24(8):1115–1122
142. Doulamis N, Doulamis A, Kollias S (1999) Efficient content-based retrieval of humans from video databases. *Proceedings of the International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pp. 89–95
143. Folkers A, Samet H, Soffer A (2000) Processing pictorial queries with multiple instances using isomorphic subgraphs. *Proceedings of the 15th International Conference on Pattern Recognition*, pp. 51–54
144. Hlaoui A, Wang S (2002) A new algorithm for graph matching with application to content-based image retrieval. *Proceeding of the Joint IAPR International Workshops SSPR and SPR*, pp. 291–300
145. Huet B and Hancock ER (1998) Fuzzy relational distance for large-scale object recognition. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 138–143
146. Huet B, Kern NJ, G Guarascio, B Merialdo (2001) Relational skeletons for retrieval in patent drawings. *Proceedings of the International Conference on Image Processing*, pp. 737–740
147. Park IK, Yun ID, Lee SU (1997) Models and algorithms for efficient color image indexing. *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp. 36–41
148. Shearer K, Venkatesh S, Bunke H (2001) Video sequence matching via decision tree path following. *Pattern Recognition Letters* 22(5):479–492
149. Ozer B, Wolf W, Akansu AN (1999) A graph based object description for information retrieval in digital image and video libraries. *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, pp. 79–83
150. Chen HT, Lin H, Liu TL (2001) Multi-object tracking using dynamical graph matching. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 210–217
151. Gomila C, Meyer F (2001) Tracking objects by graph matching of image partition sequences. *Proceedings of the Third IAPR-TC15 Workshop on Graph-Based Representations in Pattern Recognition*, pp. 1–11
152. Conte D, Foggia P, Guidobaldi C, Limongiello A, Vento M (2004) An object tracking algorithm combining different cost functions. In: Campilho A, Kamel M (eds) *ICIAR 2004*, LNCS 3212, pp. 614–622
153. Conte D, Foggia P, Jolion JM, Vento M (2006) A graph-based, multi-resolution algorithm for tracking objects in presence of occlusions. *Pattern Recognition* 39(4):562–572
154. Salotti M, Laachfoubi N (2001) Topographic graph matching for shift estimation. *Proceedings of the Third IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pp. 54–63
155. Haris K, Efstradiatis SN, Maglaveras N, Pappas C, Gourassas J, Louridas G (1999) Model-based morphological segmentation and labeling of coronary angiograms. *IEEE Transactions on Medical Imaging* 18(10):1003–1015
156. Charnoz A, Agnus V, Malandain G, Soler L, Tajine M (2005) Tree matching applied to vascular system. In: Brun L, Vento M (eds) *Graph-based Representations in Pattern Recognition*, LNCS 3434, pp. 183–192
157. Fischer S, Gilomen K, Bunke H (2002) Identification of diatoms by grid graph matching. *Proceeding of the Joint IAPR International Workshops SSPR and SPR*, pp. 94–103

158. Ambauen R, Fischer S, Bunke H (2003) Graph Edit Distance with Node Splitting and Merging, and Its Application to Diatom Identification. In: Hancock ER, Vento M (eds) Graph Based Representations in Pattern Recognition, LNCS 2726, pp. 95–106
159. Bunke H, Vento M (1999) Benchmarking of graph matching algorithms, Proceedings of Second IAPR TC-15 GbR Workshop, Haindorf, pp. 109–114
160. Kropatsch W (2001) Benchmarking graph matching algorithms – A complementary view. Proceedings of Third IAPR – TC15 Workshop on Graph-Based Representations, Italy, pp. 170–175
161. Irniger C, Bunke H (2003) Theoretical analysis and experimental comparison of graph matching algorithms for database filtering. In: Hancock ER, Vento M (eds) Graph Based Representations in Pattern Recognition, LNCS 2726, pp. 118–129
162. Kalviainen H, Oja E (1990) Comparisons of attributed graph matching algorithms for computer vision. In: Proceedings of STEP-90, Finnish Artificial Intelligence Symposium, pp. 354–368
163. Bunke H, Foggia P, Guidobaldi C, Vento M (2003) Graph clustering using the weighted minimum common supergraph. In: Hancock ER, Vento M (eds) Graph Based Representations in Pattern Recognition, LNCS 2726, pp. 235–246
164. Conte D, Guidobaldi C, Sansone C (2003) A comparison of three maximum common subgraph algorithms on a large database of labeled graphs. In: Hancock ER, Vento M (eds) Graph Based Representations in Pattern Recognition, LNCS 2726, pp. 130–141
165. Foggia P, Sansone C, Vento M (2001) A database of graphs for isomorphism and subgraph isomorphism benchmarking. Proceedings of the Third IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition, Ischia, May 23–25, pp. 176–187
166. Hagenbuchner M, Gori M, Bunke H, Tsoi AC, Irniger C (2003) Using attributed plex grammars for the generation of image and graph databases. *Pattern Recognition Letters* 24(8):1081–1087
167. Feder J, (1971) Plex languages. *Information Science* 3:225–241
168. Cordella LP, Foggia P, Sansone C, Vento M (2002) Learning structural shape descriptions from examples. *Pattern Recognition Letters* 23(12):1427–1437
169. McGregor JJ (1982) Backtrack search algorithm and the maximal common subgraph problem. *Software – Practice and Experience* 12(1):23–34
170. Balas E, Yu CS (1986) Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing* 15(4):1054–1068