Published in IET Computer Vision
Received on 8th February 2012
Revised on 23rd July 2012
doi: 10.1049/iet-cvi.2012.0030

ISSN 1751-9632

Many-to-many feature matching in object recognition: a review of three approaches

A. Shokoufandeh[1] *Y. Keselman*[2] *M.F. Demirci*[3] *D. Macrini*[4] *S. Dickinson*[5]

[1]*Computer Science Department, Drexel University, 3200 Chestnut St., Philadelphia, PA 19104, USA*
[2]*The Walt Disney Company, 925 4th Avenue, Seattle, WA 98104-2343, USA*
[3]*TOBB University of Economics and Technology, Sogutozu, Ankara 06560, Turkey*
[4]*School of Electrical Engineering and Computer Science, University of Ottawa, 800 Avenue King Edward, Ottawa, Ontario K1N 6N5, Canada*
[5]*Department of Computer Science, University of Toronto, 6 King's College Rd., Toronto, Ontario M5S 3G4, Canada*
E-mail: sven@cs.toronto.edu

Abstract: The mainstream object categorisation community relies heavily on object representations consisting of local image features, due to their ease of recovery and their attractive invariance properties. Object categorisation is therefore formulated as finding, that is, 'detecting', a one-to-one correspondence between image and model features. This assumption breaks down for categories in which two exemplars may not share a single local image feature. Even when objects are represented as more abstract image features, a collection of features at one scale (in one image) may correspond to a single feature at a coarser scale (in the second image). Effective object categorisation therefore requires the ability to match features many-to-many. In this paper, we review our progress on three independent object categorisation problems, each formulated as a graph matching problem and each solving the many-to-many graph matching problem in a different way. First, we explore the problem of learning a shape class prototype from a set of class exemplars which may not share a single local image feature. Next, we explore the problem of matching two graphs in which correspondence exists only at higher levels of abstraction, and describe a low-dimensional, spectral encoding of graph structure that captures the abstract shape of a graph. Finally, we embed graphs into geometric spaces, reducing the many-to-many graph-matching problem to a weighted point matching problem, for which efficient many-to-many matching algorithms exist.

1 Introduction

One of the fundamental obstacles to object categorisation is the common assumption that saliency in the image implies saliency in the model; in other words, for every salient image feature, there is a corresponding salient model feature [1]. In a community enamored with local image features, this one-to-one feature correspondence assumption constrains successful object categorisation to classes whose exemplars share the same local image features in the same configurations; popular examples include faces, cars, motorcycles, etc. When image features are represented as graphs, this one-to-one feature correspondence assumption translates into a one-to-one node correspondence assumption, allowing a variety of exact and inexact graph-matching techniques to be applied.

Despite the appeal of this one-to-one correspondence assumption, there exist a variety of conditions that may lead to graphs that represent visually similar objects yet do not contain a single one-to-one node correspondence. For example, due to noise or segmentation errors, a single feature (node) in one graph may map to a collection of broken features (nodes) in another graph. Or, due to scale differences, a single, coarse-grained feature in one graph may map to a collection of fine-grained features in another graph. Finally, although there may be classes in which two exemplars belonging to the class may not share a single local image feature, correspondence may exist at the level of image feature subcollections, for example, three regions in one image may map to five regions in another image with no two regions in one-to-one correspondence. In general, we seek not a one-to-one correspondence between image features (nodes), but rather a many-to-many correspondence.

In this paper, we review three of our approaches to the problem of many-to-many graph matching for object recognition, expanding on an earlier review presented in [2]. In the first approach, described in more detail in [3], we explore the problem of abstracting a categorical model from a set of images containing exemplars belonging to a known category. Specifically, we present to the system a set of region adjacency graphs, representing region segmentations of the exemplar images. While a single region (node-to-node) correspondence may not exist across the input graphs, correspondences may exist at more abstract levels. For example, the region formed by merging thee adjacent regions (nodes) in one graph may correspond to the region formed by merging five adjacent regions (nodes) in another graph, etc. For each input graph, we first define a lattice of all possible such graph abstractions, that is, graphs formed by merging adjacent regions (region merges). We then

formulate the problem as finding the most informative common abstraction across the set of lattices, that is, the largest graph common to each lattice. The key challenge we face is managing the complexity of generating the lattices and searching for a common abstraction.

In the second approach, described in more detail in [4, 5], we explore the problem of matching hierarchical graphs that represent the scale-space, coarse-to-fine, or part/whole structure of an image. Specifically, we draw on spectral graph theory to map the structure of a directed acyclic graph (DAG) to a low-dimensional vector that characterises the topological properties, or 'shape', of the DAG. This allows the similarity of two DAGs (or sub-DAGs) to be computed as the distance between their respective shape vectors. Our matching algorithm uses these shape vectors in a coarse-to-fine manner, using matches at coarser levels to efficiently constrain the matching at finer levels. If feature (node) correspondence exists at coarser, more abstract levels, the algorithm will find such correspondences, each implicitly defining a many-to-many matching of the subgraphs rooted at these coarser nodes. If correspondences also exist at finer scales, the algorithm continues in a top-down manner, using the coarse-level correspondences to constrain the search for fine-level correspondences.

Finally, in the third approach, described in more detail in [6], we explore the problem of many-to-many graph matching from a different perspective. Unlike the two above approaches, which search for correspondences at higher levels of abstraction, we explicitly match the nodes in two graphs many-to-many. Using low-distortion graph-embedding techniques, we first embed the nodes of a graph into a geometric space, where nodes become points and edge weights reflect the distances between the embedded nodes. Given two graphs embedded into the same geometric space, many-to-many graph matching is reformulated as many-to-many point matching in the geometric space, for which efficient, polynomial-time algorithms exist. Specifically, we use the Earth Mover's Distance (EMD) algorithm to compute the many-to-many correspondences between points which, in turn, defines the many-to-many correspondences between the nodes in the original graphs.

The ability to match features (or graphs) many-to-many is essential for object categorisation, in which shape similarity between two objects exists not at the level of low-level features that appear explicitly in the image, but rather at the level of feature (graph) abstractions. The three approaches reviewed in this paper provide three different perspectives on this important problem. Each addresses the critical subproblem of image abstraction. In the first case, the process of abstraction is guided by evidence from similar objects. In the second case, a restricted set of abstractions is computed directly without such supporting evidence. Although meaningful abstractions are not explicitly computed in the third case, the process of abstraction is tightly linked to the matching process and is critical to grouping features to yield many-to-many correspondences. The abstraction of image data is an important problem that has been largely ignored by the object recognition community, and is arguably the single greatest challenge to successful object categorisation [7].

2 Related work

Over the past three decades, graphs and their derivations have played a pivotal role in shape representation and multi-scale image analysis. Crowley and Parker [8] proposed one of the earliest hierarchical frameworks for capturing peaks and ridges at multiple scales, and used it to match images. During the 1990s, Lindeberg [9] examined the behaviour of image structures over scales, measured the saliency of image structures from stability properties, and extracted relations between structures at different scales. Lindeberg [10] also described a method for edge and ridge detection using such features as one-dimensional curves in the three-dimensional scale-space representation of the image. This work was later followed by Dufournaud *et al.* [11] who proposed using the Harris operator for detecting salient points. Although most of these frameworks capture qualitative shape features in a hierarchical graph, they formulated object recognition as finding the one-to-one node correspondence between two graphs.

The multiscale blob feature introduced by Lindeberg, in fact, forms the basis for the popular Scale-invariant feature transform (SIFT) framework, proposed by Lowe [12]. Another blob detector, inspired by SIFT, is the SURF feature developed by Kadir and Brady [13]. Bay *et al.* [14] also proposed a multi-scale model for the selection of salient regions of an image by detecting scale localised interest points akin to blob-like features. Many existing approaches for capturing scale space features, such as [15–17], have demonstrated the power of explicitly encoding the relational information among image features in the form a graph. Although graphs over such features are common, matching still assumes one-to-one feature correspondence. When distinctive, appearance-based features are assigned to blobs, as in SIFT, the one-to-one correspondence assumption holds only for highly restricted categories whose members share the same distinctive local features, for example, faces, cars, humans and bicycles. For more general categories, two exemplars may not share a single SIFT feature, motivating the extraction of more shape-based categorical features that cannot necessarily be matches one-to-one.

In recent years, the recognition community has moved from sparse local features to more complete representations, although most approaches are typically restricted to the detection of specific categories, for example, the detection of humans in cluttered scenes from a learned appearance model [18], or using learned categorical models for the detection of heads, leaves and cars [19]. There have also been attempts at combining appearance and shape to model and recognise isolated objects [20] and learning probabilistic indices for capturing part structure [21]. Constructing abstract models for matching and recognition is also related to generating a prototypical graph from a set of exemplars. One such abstraction is the median graph, defined as a graph, chosen among a set of sample graphs, whose sum distance to all other graphs in the set is minimised. Jiang *et al.* [22] developed a heuristic method for computing the generalised median graphs of given sample sets and explored their application to visual recognition. Torsello and Hancock [23] also introduced an abstraction model known as the union tree for a class of samples, from which class members can be derived using a minimal number of graph edit operations. In addition to model abstraction for recognition, Todorovic and Ahuja [24] used union trees for learning region-based hierarchical object models from sample segmentations.

Most of the early work on many-to-many matching in graphs is based on the edit-distance, formulated as finding the minimal set of re-labellings, additions, deletions, merges and splits of nodes and edges that transform one graph into

doi: 10.1049/iet-cvi.2012.0030

another [25–28]. These methods suffer from high computational complexity, especially when dealing with large occlusions and scene clutter. Many-to-many matching has also been studied as a variation of inexact and partial matching, most often in the context of point-set matching. Gold and Rangarajan [29] proposed one of the earliest partial matching algorithms for attributed sparse graphs. They used an EM-like graduated assignment approach that revises earlier mappings based on local constraints encoded by the graphs. The final outcome of the algorithm is a one-to-one matching, but it also maintains a stochastic matrix that closely resembles a many-to-many assignment matrix.

Carcassoni and Hancock [30] also proposed a fractional point matching algorithm that combines point clustering, cluster matching and matching points from matching clusters. These subproblems were jointly solved using a combination of the eigenvector technique and the EM approach for likelihood computation. The proposed approach proved to be very robust to noise and occlusion. Caelli [31] showed that inexact graph matching could be solved using the re-normalisation of vector representations of vertices into the eigenspaces of graphs combined with a form of relative clustering. Their method also cannot accommodate occlusion, directed graphs or node attributes, and may yield high embedding distortion.

3 Generic model acquisition from examples

Assume that we are presented with a collection of images, such that each image contains a single exemplar, all exemplars belong to a single known class, and the viewpoint with respect to the exemplar in each image is similar. We assume that some standard feature extraction process is applied to each image to yield a set of input features, organised in a graph in which nodes represent features and edges represent the relations between nearby features. However, we do not assume one-to-one feature correspondence at this input feature level. In fact, there may not exist a single input feature in common between the exemplars. Still, at some higher level of abstraction, the exemplars look similar and commonality emerges. Fig. 1*a* illustrates a simple example in which three different images, each containing a block in a similar orientation, are presented to the system. In each case, a standard region segmentation algorithm yields a region adjacency graph. Although there may not be a single node that is common to each of the three graphs, the common structure between the three exemplars arises once certain regions are grouped, defining a many-to-many matching across the input graphs.

Our task is to find the common structure in these images, under the assumption that structure that is common across many exemplars of a known class must be definitive of that class. Fig. 1*b* illustrates the class 'abstraction' that is derived from the input examples. Each region in the abstraction maps to a different group of regions in each exemplar, defining a many-to-many matching of the regions across the set of exemplars. In this example, the domain of input examples is rich enough to 'intersect out' irrelevant structure (or appearance) of the block. However, had many or all the exemplars had vertical stripes, the approach would be expected to include vertical stripes in that view of the abstracted model. Hence, the approach assumes that regularities across the exemplars are nonaccidental and should be captured in the final model.

The output of the model abstraction process will yield a region adjacency graph containing the 'meta-regions' that define a particular view of the generic model. Other views of the same exemplars would similarly yield other categorical views of the generic model. The integration of these categorical views into an optimal partitioning of the viewing sphere, or the recovery of three-dimensional parts from these views (e.g. see [32–34]), is beyond the scope of this paper. For now, the result will be a collection of categorical two-dimensional views that describe a categorical three-dimensional object. The collection of categorical object views for a set of object categories would constitute a database to support object categorisation.

3.1 Problem formulation

Returning to Fig. 1, let us now formulate our problem more concretely. As we stated, each input image is processed to form a region adjacency graph; we employ the region segmentation algorithm of Felzenzwalb and Huttenlocher [35]. Let us now consider the region adjacency graph corresponding to one input image. We will assume, for now, that our region adjacency graph represents an oversegmentation of the image; in [3], we discuss the problem of undersegmentation, and how our approach can accommodate it. The space of all possible region adjacency graphs formed by any sequence of merges of adjacent regions will form a lattice, as shown in Fig. 2. The lattice size is exponential in the number of regions obtained after initial oversegmentation. Indeed, considering the simple case of a long rectangular strip subdivided into $n+1$ adjacent rectangles, the first pair of adjacent regions able to be merged can be selected in n ways, the second in $n-1$, and so on, giving a lattice size of $n!$.

Each of the input images will yield its own lattice. The bottom node in each lattice will be the original region adjacency graph. In all likelihood, if the exemplars have different shapes (within-class deformations) and/or surface markings, the graphs forming the bottom of their corresponding lattices may bear little or no resemblance to each other. Clearly, similarity between the exemplars cannot be ascertained at this level, for there does not exist a one-to-one correspondence between the 'salient' features (i.e. regions) in one graph and the salient features in another. On the other hand, the top of each exemplar's lattice, representing a silhouette of the object (where all regions have been merged into one region), carries little information about the salient surfaces of the object.

We can now formulate our problem more precisely, recalling that a lattice consists of a set of nodes, with each node corresponding to an entire region adjacency graph. Given N input image exemplars, $E_1, E_2, \ldots, E_N$, let $L_1, L_2, \ldots, L_N$ be their corresponding lattices, and for a given lattice, L_i, let $L_i n_j$ be its constituent nodes, each representing a region adjacency graph, G_{ij}. We define a 'common abstraction' (CA), as a set of nodes (one per lattice)

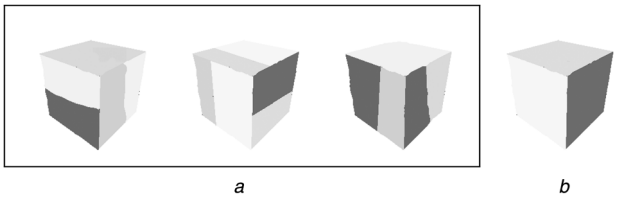


Fig. 1 *Illustrative example of generic model acquisition*

a Input exemplars belonging to a single known class
b Generic model abstracted from examples

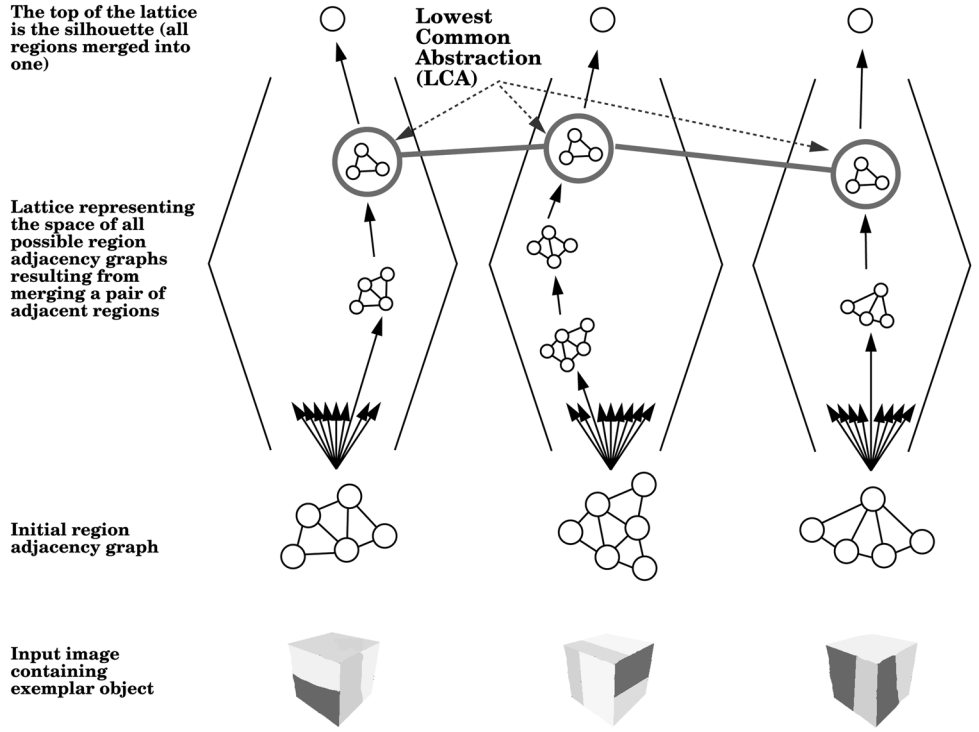


Fig. 2 *Lowest common abstraction of a set of input exemplars*

$L_1 n_{j_1}, L_2 n_{j_2}, \ldots, L_N n_{j_N}$ such that for any two nodes $L_p n_{j_p}$ and $L_q n_{j_q}$, their corresponding graphs G_{pj_p} and G_{qj_q} are isomorphic. Thus, the root node (whose graph consists of one node representing the silhouette region) of each lattice is a common abstraction. We define the 'lowest common abstraction' (LCA), as the common abstraction whose underlying graph has maximal size (in terms of number of nodes). Given these definitions, our problem can be simply formulated as finding the LCA of N input image exemplars.

Intuitively, we are searching for a node (region segmentation) that is common to every input exemplar's lattice and that retains the maximum amount of structure common to all exemplars. In other words, if we intersect all the lattices, the LCA would be the largest graph in the 'intersection lattice'. Unfortunately, there are two main problems with this formulation: (i) generating a single lattice, let alone one for each input exemplar, is intractable; and (ii) the presence of a single, heavily undersegmented exemplar (a single-node silhouette in the extreme case) will drive the LCA toward the trivial silhouette CA. We will address the complexity problem in two ways. First, we compute a weak approximation to the intersection lattice as the set of pairwise LCA's (weak in the sense that what is common to two exemplars may not be common to all exemplars). Second, we compute the LCA of two exemplars without computing their lattices; rather, we directly compute only that portion of the intersection of the two lattices that contains the LCA. We address the undersegmentation problem by relaxing our LCA definition to make it less sensitive to outliers.

3.2 LCA of two examples

We begin by focusing our attention on finding the LCA of two lattices, whereas in the next section, we will accommodate any number of lattices. As the input lattices are exponential in the number of regions, actually computing the lattices is intractable. Clearly, we need a means for focusing the search for the LCA that avoids significant lattice generation. Our approach will be to restrict the search for the LCA to the 'intersection' of the lattices. Typically, the intersection of two lattices is much smaller than either lattice (unless the images are very similar), and leads to a tractable search space. But how do we generate this new 'intersection' search space without enumerating the lattices?

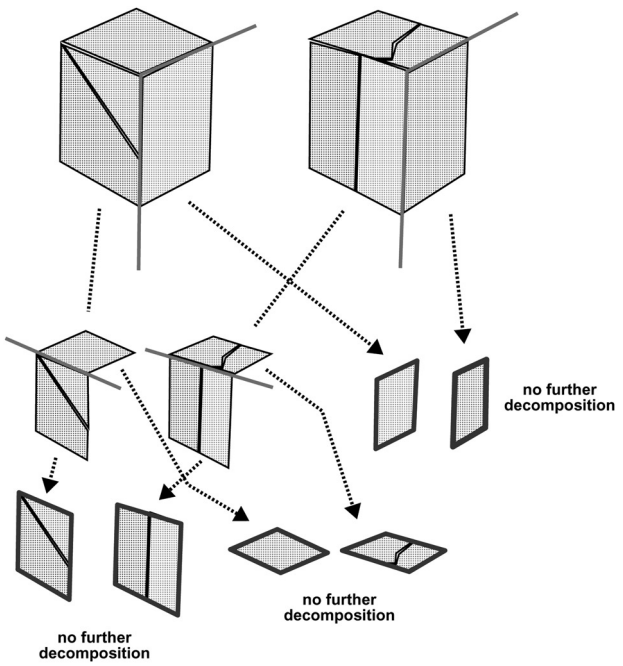


Fig. 3 *Finding the lowest common abstraction between two exemplars through a coordinated, recursive decomposition of their silhouettes*

doi: 10.1049/iet-cvi.2012.0030

Our solution, as illustrated in Fig. 3, is to work top-down, beginning with a node known to be in the intersection lattice – the root node, representing a single region (silhouette). If the intersection lattice contains only this one node, that is, one or both of the region segmented images contain a single region, then the process stops and the LCA is simply the root (silhouette). However, in most cases, the root of each input lattice is derived from an input region adjacency graph containing multiple regions. So, given two silhouettes, each representing the apex of a separate, non-trivial lattice, we have the opportunity to search for a lower abstraction (than the root) common to both lattices. Our approach will be to find a decomposition of each silhouette region into two subregions, such that (i) the shapes of the corresponding subregions are similar (for this, we employ the shape matching method outlined in [4]); and (ii) the relations among the corresponding subregions are similar. Since there is an infinite number of possible decompositions of a region into two component regions, we will restrict our search to the space of decompositions along region boundaries in the original region adjacency graphs. Note that there may be multiple two-region decompositions that are common to both lattices; each is a member of the intersection set.

Assuming that we have some means for ranking the matching decompositions (if more than one exist), we pick the best one (the remainder constituting a set of backtracking points), and recursively apply the process to each pair of isomorphic component subregions. [Each subregion corresponds to the union of a set of regions corresponding to nodes belonging to a connected subgraph of the original region adjacency graph.] The process continues in this manner, 'pushing' its way down the intersection lattice, until no further decompositions are found. This lower 'fringe' of the search space represents the LCA of the original two lattices.

The specific algorithm for choosing the optimal pair of decompositions is illustrated in Fig. 4, described in [3], and can be summarised as follows:

1. Map each region adjacency graph to its dual 'boundary segment graph', in which boundary segments become nodes and edges capture segment adjacency. Therefore, a cut in a region adjacency graph maps to a path in its dual boundary segment graph.
2. Form the association graph of the two boundary segment graphs. Nodes and arcs in the association graph correspond to pairs of nodes and arcs, respectively, in the boundary segment graphs. Therefore a path in the association graph corresponds to a pair of paths in the boundary segment graphs which, in turn, correspond to a pair of decompositions (cuts) of the region adjacency graphs.
3. With appropriate edge weights, the optimal pair of corresponding decompositions corresponds to the shortest path in the association graph. The edge weights, described in detail in [3], reflect the shape 'distance' between pairs of corresponding contours making up the cuts.
4. The optimal pair of decompositions is verified in terms of satisfying the criteria of region shape similarity [4] and region relation consistency [3], and the process is applied recursively to the corresponding region adjacency subgraphs resulting from the pair of cuts.

3.3 LCA of multiple examples

So far, we have addressed only the problem of finding the LCA of two examples. How, then, can we extend our approach to find the LCA of multiple examples? Furthermore, when moving toward multiple examples, how do we prevent a 'noisy' example, such as a single, heavily undersegmented silhouette, from driving the solution towards the trivial silhouette? To extend our two-exemplar LCA solution to a robust, multi-exemplar solution, we begin with two important observations. First, the LCA of two exemplars lies in the intersection of their abstraction lattices. Thus, both exemplar region adjacency graphs can be transformed into their LCA by means of sequences of region merges. Second, the total number of merges required to transform the graphs into their LCA is minimal among all elements of the intersection lattice, that is, the LCA lies at the lower fringe of the lattice.

Our solution begins by constructing an approximation to the intersection lattice of multiple examples. Consider the closure of the set of the original region adjacency graphs under the operation of taking pairwise LCAs. In other words, starting with the initial region adjacency graphs, we find their pairwise LCAs, then find pairwise LCAs of the resulting abstraction graphs, and so on (note that duplicate graphs are removed). We take all graphs, original and LCA, to be nodes of a new 'closure' graph. The closure graph can

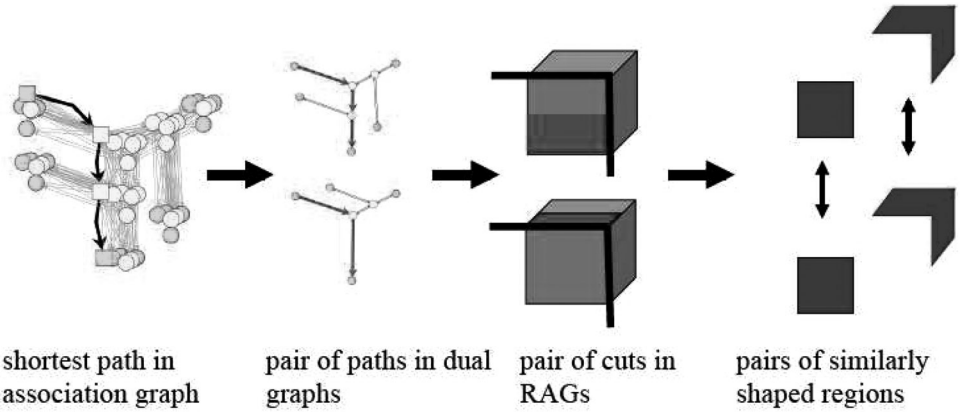


Fig. 4 *LCA formulated as the shortest path in the association graph formed by the two region adjacency graphs: (left to right)*

(i) Association graph is the product of the two boundary segment graphs, each the dual of a region adjacency graph; (ii) Given an appropriate assignment of edge weights, the shortest path in the association graph corresponds to a pair of paths in the boundary segment graphs; (iii) The pair of boundary segment graphs, in turn, represent a pair of cuts in the original region segmentation graphs; (iv) The cuts define two pairs of corresponding regions. Provided their shapes and relations are similar, the shortest path is accepted, and the process is recursively applied to each pair of component regions

be considered as a weak approximation to the true intersection lattice, as the LCA of a pair of graphs may not be the LCA of all graphs. If graph H was obtained as the LCA of graphs G_1 and G_2, then directed arcs go from nodes corresponding to G_1, G_2 to the node corresponding to H in the closure graph.

Next, we will relax the first property above to accommodate 'outlier' exemplars, such as undersegmented, input silhouettes. Specifically, we will not enforce that the LCA of multiple exemplars lies in the intersection set of all input exemplars. Rather, we will choose a node in our approximate intersection lattice that represents a 'low abstraction' for many (but not necessarily all) input exemplars. More formally, we will define the LCA of a set of exemplar region adjacency graphs to be that element in the intersection of two or more abstraction lattices that minimises the total number of edit operations (merges or splits) required to obtain the element from all the given exemplars. If a node in the intersection lattice lies along the lower fringe with respect to a significant number of input exemplars, then its sum distance to all exemplars is small. Conversely, the sum distance between the silhouette outlier (in fact, the true LCA) and the input exemplars will be large, eliminating that node from contention. Our algorithm for computing this 'median' of the closure graph, along with an analysis of its complexity, is given in [3].

3.4 Demonstration

In Fig. 5, we illustrate the results of our pairwise LCA approach applied to a set of three coffee cup images, respectively. The lower row represents the original images, the next row up represents the input region segmented images (with black borders), while the LCA is shown with an orange border. In this case, the pairwise LCA of each pair of input images is the same. The solution captures our intuitive notion of the cup's surfaces, with a handle, a body, a top, and the hole in the handle (the algorithm cannot distinguish a surface from a hole).

A second example illustrates how restricting the decomposition to the space of pairs of cuts in two region adjacency graphs can fail to find the correct abstraction. In Fig. 6, region undersegmentation near the junction of the handle and body of the middle cup prevents appropriate cuts that separate handle from body in the other two cups from being chosen when a pairwise LCA is computed with the middle cut (a similar undersegmentation problem prevents the first and third cups from finding an appropriate cut separating handle from body). The decomposition process is therefore forced to terminate early, yielding a LCA which is non-intuitive.

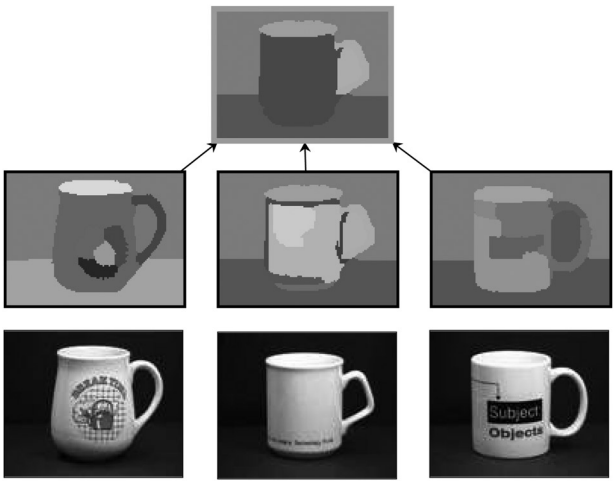


Fig. 5 *Computed correct LCA (topmost image) of three examples*

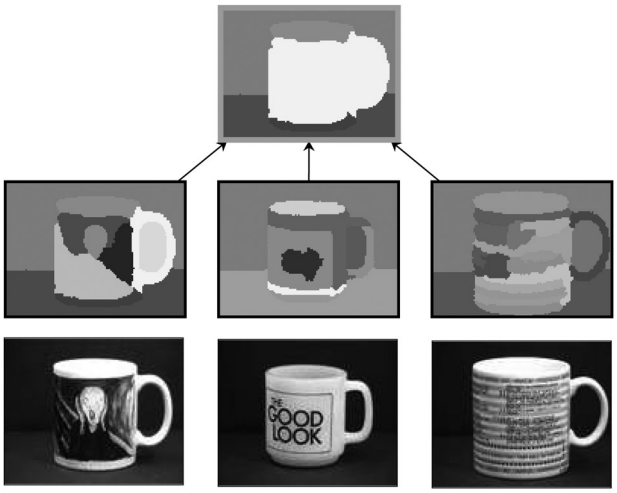


Fig. 6 *Computed problematic LCA (orange border) of three examples (see text for explanation)*

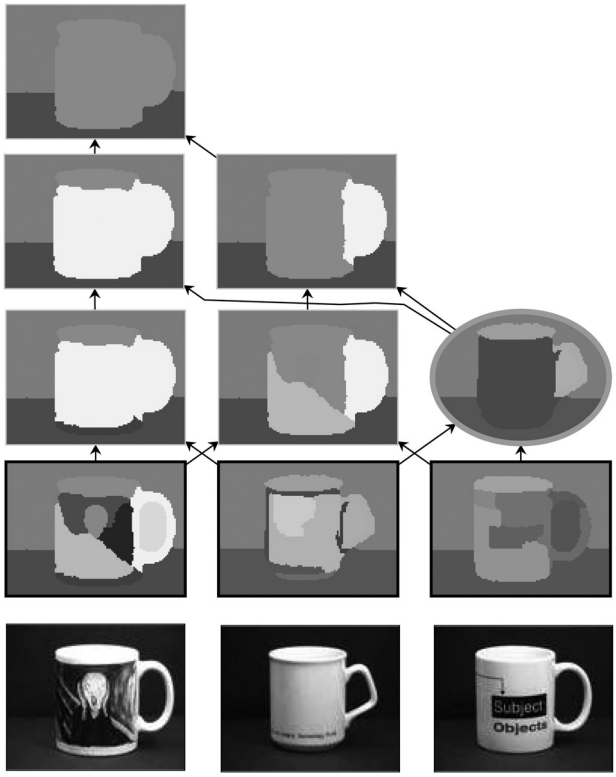


Fig. 7 *Searching the intersection lattice for the LCA of three examples*

A final example, shown in Fig. 7, illustrates the weak intersection lattice computed for three input exemplars. The edges in the closure graph are shown, indicating which abstractions are derived from which region adjacency graphs; edge weight, representing edit distances, are not shown. The LCA, indicated in orange, is that node in the closure graph, whose sum distance to all exemplars is minimum. In this case, the computed LCA represents the correct common abstraction.

doi: 10.1049/iet-cvi.2012.0030

4 Graph matching using a spectral encoding of graph structure

We now turn to the problem of inexact graph matching where, due to noise and occlusion, two graphs representing very similar objects may not be isomorphic. In a coarse-to-fine feature hierarchy, we would like to find correspondences in a coarse-to-fine fashion, requiring that we have some way of compactly, robustly, and efficiently describing the underlying structure rooted at a node. Given such a low-dimensional encoding of DAG structure, we could compare the 'shapes' of two graphs by comparing their encodings. Although correspondence may be established between two nodes in two graphs, according to the similarity of their encodings, it is important to note that these encodings describe the entire underlying structures rooted at these two nodes. Since these structures may have different numbers of nodes and may have slight variation in branching structure, the resulting matching can be seen as many-to-many, effectively matching two graph abstractions.

4.1 Eigen-decomposition of structure

To describe the topology of a DAG, we turn to the domain of eigenspaces of graphs, first noting that any graph can be represented as a 0, 1, −1 node-adjacency matrix (which we will subsequently refer to as an adjacency matrix), with 1's (-1's) indicating a forward (backward) edge between adjacent nodes in the graph (and 0's on the diagonal). The eigenvalues of a graph's adjacency matrix encode important structural properties of the graph relating to node degree distribution. Furthermore, the eigenvalues of the adjacency matrix $\boldsymbol{A}$ are invariant to any orthonormal transformation of the form $\boldsymbol{P}^t\boldsymbol{AP}$. Since a permutation matrix is orthonormal, the eigenvalues of a graph are invariant to any consistent re-ordering of the graph's branches. In [5], we established the stability of a graph's eigenvalues to minor perturbation due to noise and occlusion. Specifically, we have shown that such structural changes to a DAG can be modelled as a two-step transformation of its adjacency matrix. The first step transforms the adjacency matrix to a new matrix having the same spectral properties as the original matrix. The second step adds a noise matrix to this new matrix, representing the structural changes. These changes take the form of addition/deletion of nodes/arcs to/from the original DAG. We draw on results that relate the distortion of the spectral elements of the adjacency matrix resulting from the first step to the magnitude of the noise added in the second step. Since the spectrum of the original matrix is the same as that of the transformed matrix, the noise-dependent bounds therefore apply to the original matrix. These results establish the stability of a DAG's spectrum to minor topological changes.

4.2 Formulating an index

We can now proceed to define an index based on the eigenvalues. We could, for example, define a vector to be the sorted eigenvalues of a DAG, with the resulting index used to retrieve nearest neighbours in a model DAG database having similar topology. However, for large DAGs, the dimensionality of the index (and model DAG database) would be prohibitively large. Our solution to this problem will be based on eigenvalue sums rather than on the eigenvalues themselves. Still, one more problem exists. The eigenvalues are a global descriptor, and occlusion and/or scene clutter may lead to heavy eigenvalue (index) distortion. Therefore we need a mechanism for describing the structure of graph 'parts' that survive occlusion/clutter.

Specifically, let T be a DAG whose maximum branching factor is $\Delta(T)$, and let the subgraphs of its root be T_1, T_2, …, T_S, as shown in Fig. 8. For each subgraph, T_i, whose root degree is $\delta(T_i)$, we compute the eigenvalues of T_i's submatrix, sort them in decreasing order by absolute value of their magnitude, and let S_i be the sum of the $\delta(T_i) - 1$ largest magnitudes. The sorted S_is become the components of a $\Delta(T)$-dimensional vector assigned to the DAG's root. If the number of S_i's is less than $\Delta(T)$, then the vector is padded with zeroes. We can recursively repeat this procedure, assigning a vector to each nonterminal node in the DAG, computed over the subgraph rooted at that node. The reasons for computing a description for each node, rather than just the root, will become clear in the next section.

Although the eigenvalue sums are invariant to any consistent re-ordering of the DAG's branches, we have given up some uniqueness (due to the summing operation) in order to reduce dimensionality. We could have elevated only the largest eigenvalue from each subgraph (non-unique but less ambiguous), but this would be less representative of the subgraph's structure. We choose the $\delta(T_i) - 1$ largest eigenvalues for two reasons: (i) the largest eigenvalues are more informative of subgraph structure, and (ii) by

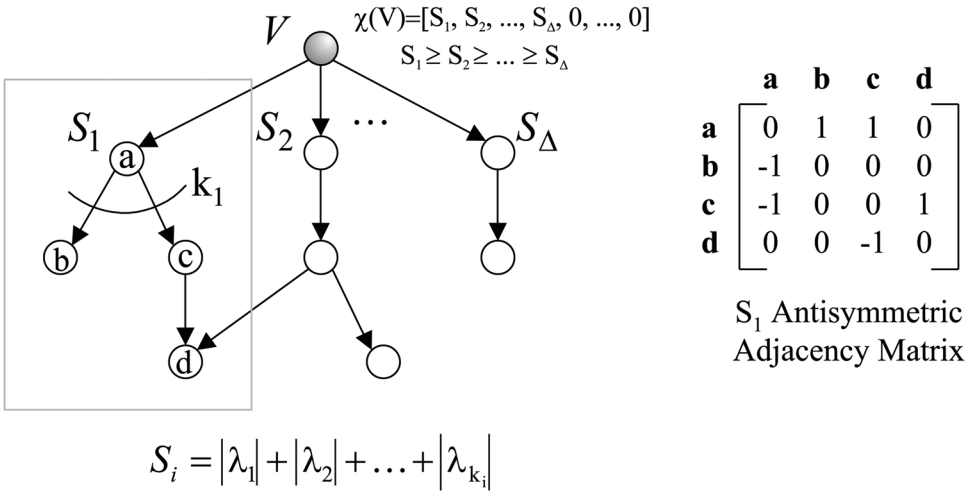


Fig. 8 *Forming a low-dimensional vector description of graph structure*

summing $\delta(T_i) - 1$ elements, we effectively normalise the sum according to the local complexity of the subgraph root.

We note that the proposed topological index satisfies a series of criteria, making it efficient and robust for structural characterisation of DAGs. The eigen-decomposition yields a low-dimensional vector assigned to each node in the DAG, which captures the local topology of the DAG rooted at that node. Furthermore, a node's vector is invariant to any consistent reordering of the node's subgraph. The components of a node's vector are based on summing the largest eigenvalues of its subgraph's adjacency submatrix. Although our dimensionality-reducing summing operation has cost us some uniqueness, as the elements of each sum are positive values and are monotonic with respect to structural complexity, the partial sums still have relatively low ambiguity [36]. Also, as discussed in Section 4.1, we have shown our index to be stable to minor perturbations of the DAG's topology. As shown in [4], these sums can be computed even more efficiently than the eigenvalues themselves.

4.3 Matching hierarchical structures

Each of the top-ranked candidates emerging from the indexing process must be verified to determine which is most similar to the query. If there were no clutter, occlusion or noise, our problem could be formulated as a graph isomorphism problem. If we allowed clutter and limited occlusion, we would search for the largest isomorphic subgraphs between query and model. Unfortunately, with the presence of noise, in the form of the addition of spurious graph structure and/or the deletion of salient graph structure, large isomorphic subgraphs may simply not exist. It is here that we call on our eigen-characterisation of graph structure to help us overcome this problem.

Each node in our graph (query or model) is assigned a topical signature vector (TSV), which reflects the underlying structure in the subgraph rooted at that node. If we simply discarded all the edges in our two graphs, we would be faced with the problem of finding the best correspondence between the nodes in the query and the nodes in the model; two nodes could be said to be in close correspondence if the distance between their TSVs (and the distance between their domain-dependent node labels) was small. In fact, such a formulation amounts to finding the maximum cardinality, minimum weight matching in a bipartite graph spanning the two sets of nodes. At first glance, such a formulation might seem like a bad idea (by throwing away all that important graph structure!) until one recalls that the graph structure is really encoded in the node's TSV. Is it then possible to reformulate a noisy, largest isomorphic subgraph problem as a simple bipartite matching problem?

Unfortunately, in discarding all the graph structure, we have also discarded the underlying hierarchical structure. There is nothing in the bipartite graph-matching formulation that ensures that hierarchical constraints among corresponding nodes are obeyed, that is, that parent/child nodes in one graph do not match child/parent nodes in the other. This reformulation, although softening the overly strict constraints imposed by the largest isomorphic subgraph formulation, is perhaps too weak. We could try to enforce the hierarchical constraints in our bipartite matching formulation, but no polynomial-time solution is known to exist for the resulting formulation. Clearly, we seek an efficient approximation method that will find corresponding nodes between two noisy, occluded DAGs, subject to hierarchical constraints.

Our algorithm, a modification to Reyner's algorithm [37], combines the above bipartite matching formulation with a greedy, best-first search in a recursive procedure to compute the corresponding nodes in two rooted DAGs. As in the above bipartite matching formulation, we compute the maximum cardinality, minimum weight matching in the bipartite graph spanning the two sets of nodes. Edge weight will encode a function of both topological similarity as well as domain-dependent node similarity. The result will be a selection of edges yielding a mapping between query and model nodes. As mentioned above, the computed mapping may not obey hierarchical constraints. We therefore greedily choose only the best edge (the two most similar nodes in the two graphs, representing in some sense the two most similar subgraphs), add it to the solution set, and recursively apply the procedure to the subgraphs defined by these two nodes. Unlike a traditional depth-first search which backtracks to the next statically determined branch, our algorithm effectively recomputes the branches at each node, always choosing the next branch to descend in a best-first manner. In this way, the search for corresponding nodes is focused in corresponding subgraphs (rooted DAGs) in a top-down manner, thereby ensuring that hierarchical constraints are obeyed. The algorithm is illustrated in Fig. 9.

4.4 Demonstration

To demonstrate our approach to matching, we turn to the domain of two-dimensional object recognition [4, 39]. We adopt a representation for two-dimensional shape that is based on a colouring of the shocks (singularities) of a curve evolution process acting on simple closed curves in the plane [40]. Any given two-dimensional shape gives rise to a rooted 'shock tree', in which nodes represent parts (whose labels are drawn from four qualitatively defined classes) and arcs represent relative time of formation (or relative size). Fig. 10 illustrates a two-dimensional shape, its shock structure, and its resulting shock graph, whereas Fig. 11 illustrates a matching example of two similar shapes, showing part correspondences. Extensive examples and performance evaluation can be found in [4, 41, 42].

The proposed matching algorithm has also been applied to the categorical shape recognition problem. In this application, first the coarse shape of an object will be captured using a multi-scale blob and ridge decomposition that captures the compact and elongated subparts of a given object at relevant scales [43]. This part structure will, in turn, be represented as a DAG, where nodes encode the coarse shape components in terms of blobs and ridges and edges capture both spatial and topological relationships among nodes in scale-space [38]. An example of a blob graph for a hand image, showing hierarchical edges, is shown in Fig. 12*a*. In a generalisation [38] of the original DAG matching algorithm [4], we devised a weighting scheme in which the topological and geometric terms in the bipartite graph edge weights need not be constant for all edges. Since each edge spans an image node and a model node, the model can be used to define an a priori weighting scheme that is edge-dependent. As a result, if portions of the model have additional geometric constraints (e.g. articulation was prohibited), those model nodes could have a higher weighting on their geometric similarity term. This extension has proven to be very robust in dealing with

doi: 10.1049/iet-cvi.2012.0030

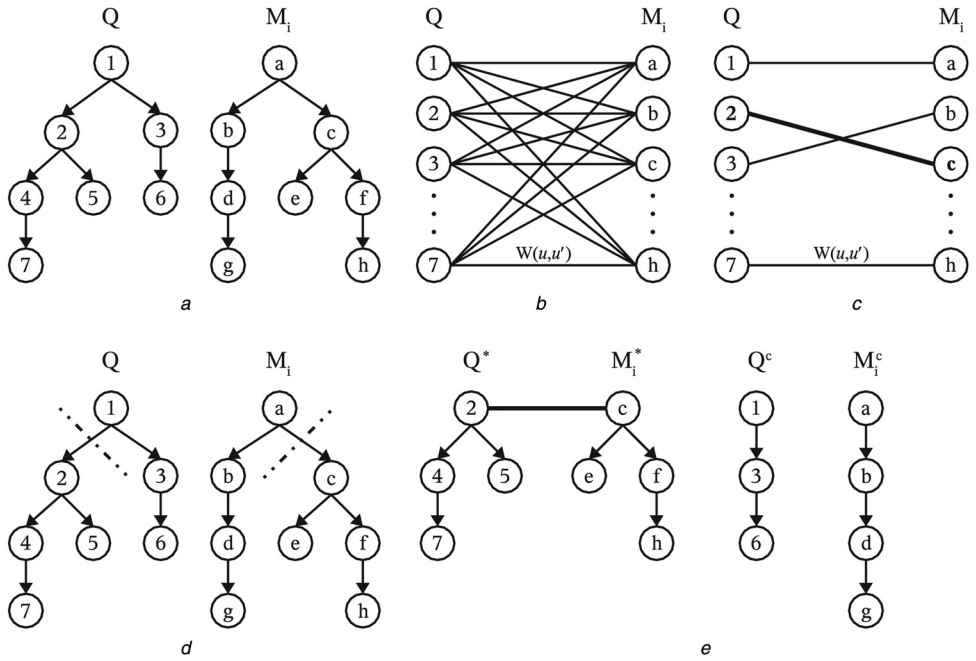


Fig. 9 *Hierarchical matching algorithm [38]*

a Given a query graph and a model graph
b We form a bipartite graph in which the edge weights are the pair-wise node similarities of query and model graphs Then,
c Then we compute a maximum matching and add the best edge to the solution set
d Finally, split the graphs at the matched nodes
e Recursively descend

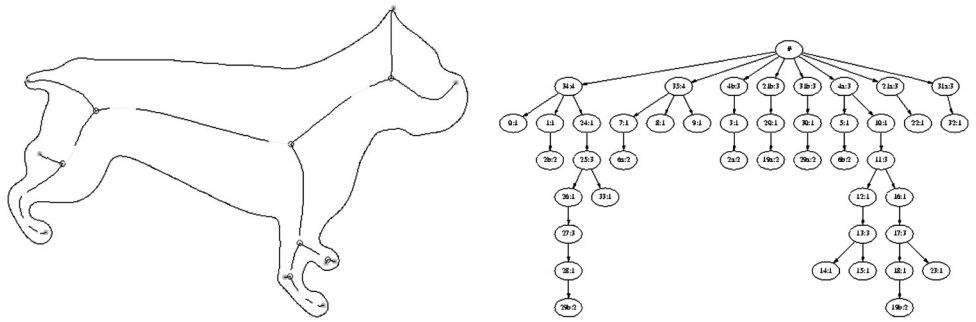


Fig. 10 *Illustrative example of a shape and its corresponding shock graph*

scale-space graphs, allowing the incorporation of local model constraints into the matching algorithm. An example of the blob correspondence computed over two hand exemplars is shown in Fig. 12*b*.

5 Many-to-many graph matching using graph embedding

The main objective of the many-to-many graph matching problem is to establish a minimum cost mapping between the vertices of two attributed, edge-weighted graphs. The general formulation of the many-to-many graph matching problem results in an intractable computational problem. First, due to the exponential size of power-sets of underlying graphs in terms of number of vertices in each graph, the size of the search space for the many-to-many matching problem is exponential. Even simplifying the problem to one-to-one mappings, by replacing the power-sets with actual vertex sets, will result in the multidimensional matching problem that is known to be NP-complete for arbitrary labelled graphs. Several existing heuristic approaches to the problem of many-to-many graph matching suffer from computational inefficiency and/or from an inability to handle small perturbations in graph structure. We seek a solution to this problem while addressing drawbacks of existing approaches. Drawing on recently developed techniques from the domain of low-distortion graph embedding, we have explored an efficient method for mapping a graph's structure to a set of vectors in a low-dimensional space. This mapping not only simplifies the original graph representation, but it retains important information about both local (neighbourhood) as well as global graph structure. Moreover, the mapping is stable with respect to noise in the graph.

Armed with a low-dimensional, robust vector representation of an input graph's structure, many-to-many graph matching can now be reduced to the much simpler problem of matching weighted distributions of points in a normed vector space, using a 'distribution-based' similarity measure. We consider one such similarity measure, known

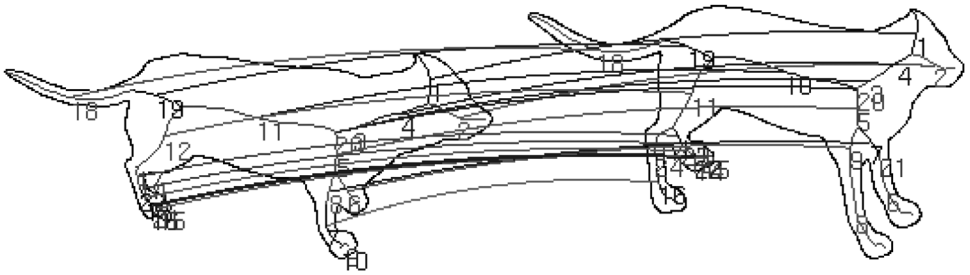
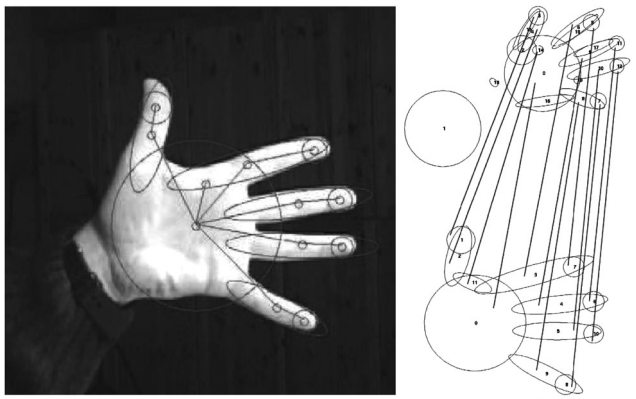


Fig. 11 *Example part correspondences computed by the matching algorithm*



a *b*

Fig. 12 *Example graph of blob and ridge representation of*

a Hand image with the hierarchical edges of the scale-space DAG shown in grey [38]
b Computed correspondence [38]

as the EMD, and show that the many-to-many vector mapping that realizes the minimum EMD corresponds to the desired many-to-many matching between nodes of the original graphs. The result is a more efficient and more stable approach to many-to-many graph matching that, in fact, includes the special case of one-to-one graph matching. The overview of the approach is presented in Fig. 13. Fig. 14 illustrates an example domain in which we have successfully applied the approach to many-to-many matching between two hand images represented as hierarchical blob and ridge decompositions; note that the number of blobs used to model the fingers or palm differs between the two decompositions. The algorithm computes the many-to-many correspondences indicated by the similar colouring.

5.1 Low-distortion embedding

Our interest in low-distortion embedding is motivated by its ability to transform the problem of many-to-many matching in finite graphs to geometrical problems in low-dimensional vector spaces. Specifically, let $G_1 = (\mathcal{A}_1, E_1, \mathcal{D}_1)$, $G_2 = (\mathcal{A}_2, E_2, \mathcal{D}_2)$ denote two graphs on vertex sets $\mathcal{A}_1$ and $\mathcal{A}_2$, edge sets E_1 and E_2, under distance metrics $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively ($\mathcal{D}_i$ represents the distances between all pairs of nodes in G_i). Ideally, we seek a single embedding that can map each graph to the same vector space, in which the two embeddings can be directly compared. However, in general, this is not possible without introducing unacceptable distortion.

We will therefore tackle the problem in two steps. First, we will seek low-distortion embeddings f_i that map sets $\mathcal{A}_i$ to normed spaces $(\mathcal{B}_i, ||.||_k)$, $i \in 1, 2$. Next, we will align the normed spaces, so that the embeddings can be directly compared. Using these mappings, the problem of many-to-many vertex matching between G_1 and G_2 is therefore reduced to that of computing a mapping $\mathcal{M}$ between subsets of $\mathcal{B}_1$ and $\mathcal{B}_2$.

In practice, the robustness and efficiency of mapping $\mathcal{M}$ will depend on several parameters, such as the magnitudes of distortion of the $\mathcal{D}_i$'s under the embeddings, the computational complexity of applying the embeddings, the efficiency of computing the actual correspondences (including alignment) between subsets of $\mathcal{B}_1$ and $\mathcal{B}_2$, and the quality of the computed correspondence. The latter issue will be addressed in Section 5.2.

The problem of low-distortion embedding has a long history for the case of planar graphs, in general, and trees, in particular. More formally, the most desired embedding is the subject of the following conjecture:

Conjecture 1 [44]: Let $G = (\mathcal{A}, E)$ be a planar graph, and let $M = (\mathcal{A}, \mathcal{D})$ be the shortest-path metric for the graph G. Then there is an embedding of M into $\|.\|_p$ with $O(1)$ distortion.

This conjecture has only been proven for the case in which G is a tree. Although the existence of such a distortion-free embedding under $\|.\|_k$-norms was established in [45], no

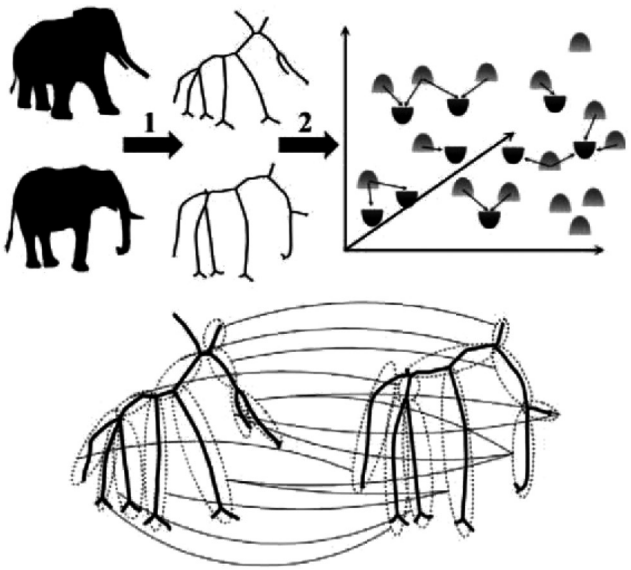


Fig. 13 *Overview of many-to-many matching procedure*

Pair of views are represented by undirected, rooted, weighted graphs (transition 1). The graphs are mapped into a low-dimensional vector space using a low-distortion graph embedding (transition 2). A many-to-many point (graph node) correspondence is computed by the Earth Mover's Distance under transformation (shown at the bottom). The many-to-many point correspondences define the many-to-many nodes correspondences in the two original graphs

doi: 10.1049/iet-cvi.2012.0030

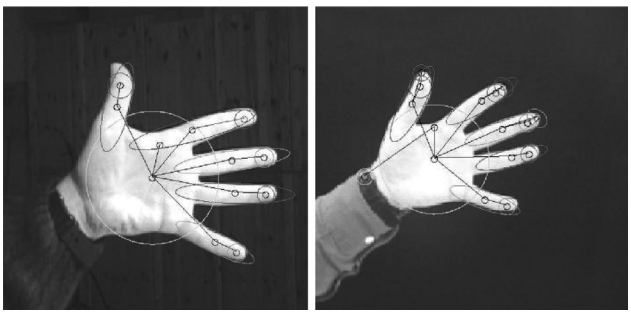


Fig. 14 *Many-to-many graph matching applied to Blob graphs*

Many-to-many feature correspondences have been coloured the same. Corresponding groups of nodes (whose cardinalities may be different) are coloured the same (courtesy of [6])

deterministic construction was provided. One such deterministic construction was given by Matoušek [46], suggesting that if we could somehow map our graphs into trees, with small distortion, we could adopt Matoušek's framework.

5.1.1 Shortest path metric: Before we can proceed with Matoušek's embedding, we must choose a suitable metric for our graphs, that is, we must define a distance between any two vertices. Let $G = (\mathcal{A}, E)$ denote an edge-weighted graph with real edge weights $\mathcal{W}(e)$, $e \in E$. We will say that $\mathcal{D}$ is a metric for G if, for any three vertices u, v, $w \in \mathcal{A}$, $\mathcal{D}(u, v) = \mathcal{D}(v, u) \geq 0$, $\mathcal{D}(u, u) = 0$ and $\mathcal{D}(u, v) \leq \mathcal{D}(u, w) + \mathcal{D}(w, v)$. In general, there are many ways to define metric distances on a weighted graph. The best-known metric is the shortest-path metric $\delta(., .)$, that is, $\mathcal{D}(u, v) = \delta(u, v)$, the shortest path distance between u and v for all u, $v \in \mathcal{A}$. In fact, if the weighted graph G is a tree, the shortest path between any two vertices is unique, and the weights of the shortest paths between vertices will define a metric $\mathcal{D}(., .)$.

In the event that G is not a tree, $\mathcal{D}(., .)$ can be defined through a special representation of G, known as the 'centroid metric tree' $\mathfrak{T}$ [47]. The path-length between any two vertices u, v in $\mathfrak{T}$ will mimic the metric $\delta(u, v)$ in G. A metric $\mathcal{D}(., .)$ on n objects $\{v_1, \ldots, v_n\}$ is a centroid metric if there exist labels $\ell_1, \ldots, \ell_n$ such that for all $i \neq j$, $\mathcal{D}(v_i, v_j) = \ell_i + \ell_j$. If G is not a tree, its centroid metric tree $\mathfrak{T}$ is a star on vertex-set $\mathcal{A} \cup \{c\}$ and weighted edge-set $\{(c, v_i) | \mathcal{W}(c, v_i) = \ell_i, v_i \in \mathcal{A}\}$. It is easy to see that the path-lengths between v_i and v_j in $\mathfrak{T}$ will correspond to $\mathcal{D}(v_i, v_j)$ in G. For details on the construction of a metric labelling ℓ_i of a metric distance $\mathcal{D}(., .)$ (see [47]).

5.1.2 Path partition of a graph: The construction of the embedding depends on the notion of a path partition of a graph. In this subsection, we introduce the path partition, and then use it in the next subsection to construct the embedding. Given a weighted graph $G = (\mathcal{A}, E)$ with metric distance $\mathcal{D}(., .)$, let $\mathfrak{T} = (\mathcal{A}, \mathfrak{E})$ denote a tree representation of G, whose vertex distances are consistent with $\mathcal{D}(., .)$. In the event that G is a tree, $\mathfrak{T} = G$; otherwise $\mathfrak{T}$ is the centroid metric tree of G. To construct the embedding, we will assume that $\mathfrak{T}$ is a rooted tree. It will be clear from the construction that the choice of the root does not affect the distortion of the embedding.

The dimensionality of the embedding of $\mathfrak{T}$ depends on the caterpillar dimension, denoted by $\mathrm{c\,dim}(\mathfrak{T})$, and is recursively defined as follows [46]. If $\mathfrak{T}$ consists of a single vertex, we set $\mathrm{c\,dim}(\mathfrak{T}) = 0$. For a tree $\mathfrak{T}$ with at least 2 vertices, $\mathrm{c\,dim}(\mathfrak{T}) \leq k + 1$ if there exist paths $P_1, \ldots, P_r$ beginning at the root and otherwise pairwise disjoint, such that each component $\mathfrak{T}_j$ of $\mathfrak{T} - \mathfrak{E}(P_1) - \mathfrak{E}(P_2) - \cdots - \mathfrak{E}(P_r)$ satisfies $\mathrm{c\,dim}(\mathfrak{T}_j) \leq k$. Here $\mathfrak{T} - \mathfrak{E}(P_1) - \mathfrak{E}(P_2) - \cdots - \mathfrak{E}(P_r)$ denotes the tree $\mathfrak{T}v_i$ with the edges of the P_i's removed, and the components $\mathfrak{T}_j$ are rooted at the single vertex lying on some P_i. The caterpillar dimension can be determined in linear time for a rooted tree $\mathfrak{T}$, and it is known that $\mathrm{c\,dim}(\mathfrak{T}) \leq \log(|\mathcal{A}|)$ (see [46]).

The construction of vectors $f(v)$, for $v \in \mathcal{A}$, depends on the notion of a 'path partition' of $\mathfrak{T}$. The path partition $\mathfrak{P}$ of $\mathfrak{T}$ is empty if $\mathfrak{P}$ is a single vertex; otherwise $\mathfrak{P}$ consists of some paths $P_1, \ldots, P_r$ as in the definition of $\mathrm{c\,dim}(\mathfrak{T})$, plus the union of path partitions of the components of $\mathfrak{T} - \mathfrak{E}(P_1) - \mathcal{M}(P_2) - \cdots - \mathfrak{E}(P_r)$. The paths $P_1, \ldots, P_r$ have level 1, and the paths of level $k \geq 2$ are the paths of level $k - 1$ in the corresponding path partitions of the components of $\mathfrak{T} - \mathfrak{E}(P_1) - \mathfrak{E}(P_2) - \cdots - \mathfrak{E}(P_r)$. Note that the paths in a path partition are edge-disjoint, and their union covers the edge-set of $\mathfrak{T}$.

To illustrate these concepts, consider the tree shown in Fig. 15. The three darkened paths from the root represent three level 1 paths. Following the removal of the level 1 paths, we are left with six connected components that, in turn, induce seven level 2 paths, shown with lightened edges. [Note that the third node from the root in the middle level 1 branch is the root of a tree-component consisting of five nodes that will generate two level 2 paths.] Following the removal of the seven level 2 paths, we are left with an empty graph. Hence, the caterpillar dimension ($\mathrm{c\,dim}(\mathfrak{T})$) is two. It is easy to see that the path partition $\mathfrak{P}$ can be constructed using a modified depth-first search in $O(|\mathcal{A}|)$ time.

5.1.3 Construction of the embedding: Given a path partition $\mathfrak{P}$ of $\mathfrak{T}$, we will use m to denote the number of levels in $\mathfrak{P}$, and let $P(v)$ represent the unique path between the root and a vertex $v \in \mathcal{A}$. The first segment of $P(v)$ of weight l_1 follows some path P^1 of level 1 in $\mathfrak{P}$, the second segment of weight l_2 follows a path P^2 of level 2, and the last segment of weight l_α follows a path P^α of level $\alpha \leq m$. The sequences $\langle P^1, \ldots, P^\alpha \rangle$ and $\langle l_1, \ldots, l_\alpha \rangle$ will be referred to as the 'decomposition sequence and the weight sequence' of $P(v)$, respectively.

To define the embedding $f: \mathcal{A} \to \mathcal{B}$ under $\|.\|_2$, we let the relevant coordinates in $\mathcal{B}$ be indexed by the paths in $\mathfrak{P}$. The vector $f(v)$, $v \in \mathcal{A}$, has non-zero coordinates corresponding to the paths in the decomposition sequence of $P(v)$.

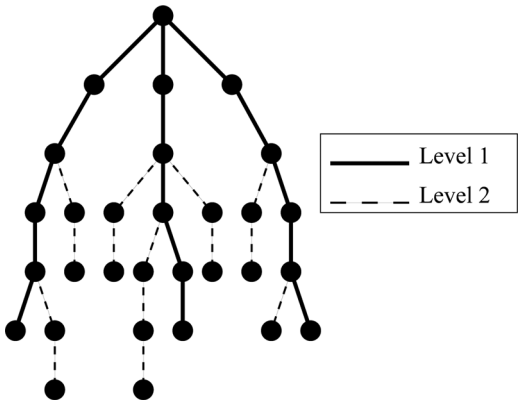


Fig. 15 *Path partition of a tree*

Returning to Fig. 15, the vector $\boldsymbol{f(v)}$ will have ten components (defined by three level 1 paths and seven level 2 paths). Furthermore, every vector $\boldsymbol{f(v)}$ will have at most two non-zero components. Consider, for example, the lowest leaf node in the middle branch. Its path to the root will traverse three level 2 edges corresponding to the fourth level 2 path, as well as three level 1 edges corresponding to the second level 1 path.

Such embedding functions have become fairly standard in the metric space representation of weighted graphs [46, 48]. In fact, Matoušek [46] has proven that setting the i-th coordinate of $f(v)$, corresponding to path P^k, $1 \leq k \leq \alpha$, in decomposition sequence $\langle P^1, \ldots, P^\alpha \rangle$, to

$$f(v)_i = \sqrt{l_k \left[l_k + \sum_{j=1}^{\alpha} \max(0, l_j - l_k/2m) \right]}$$

will result in a small distortion of at most $\sqrt{\log \log |\mathcal{A}|}$. It should be mentioned that although the choice of path decomposition $\mathfrak{P}$ is not unique, the resulting embeddings are isomorphic up to a transformation. Computationally, constructions of $\mathfrak{T}$, $\mathfrak{P}$ and $\mathcal{B}$ are all linear in terms of $|\mathcal{A}|$ and $|\mathfrak{E}|$.

The above embedding has preserved both graph structure and edge weights, but has not accounted for node information. To accommodate node information in our embedding, we will associate a weight w_v to each vector $\boldsymbol{f(v)}$, for all $v \in \mathcal{A}$. These weights will be defined in terms of vertex labels which, in turn, encode image feature values. Note that nodes with multiple feature values give rise to a vector of weights assigned to every point. We will present an example of one such distribution in Section 5.3.

5.2 Distribution-based matching

By embedding vertex-labelled graphs into normed spaces, we have reduced the problem of many-to-many matching of graphs to that of many-to-many matching of weighted distributions of points in normed spaces. However, before we can match two point distributions, we must map them into the same normed space. This involves reducing the dimensionality of the higher-dimensional distribution and aligning the two distributions. Given a pair of weighted distributions in the same normed space, the EMD framework [49] is then applied to find an optimal match between the distributions. The EMD approach computes the minimum amount of work (defined in terms of displacements of the masses associated with points) it takes to transform one distribution into another.

5.2.1 Embedding point distributions in the same normed space: Embeddings produced by the graph embedding algorithm can be of different dimensions and are defined only up to a distance-preserving transformation (a translated and rotated version of a graph embedding will also be a graph embedding). Therefore in order to apply the EMD framework, we perform a PCA-based 'registration' step, whose objective is to project the two distributions into the same normed space. Intuitively, the projection of the original vectors associated with each graph embedding onto the subspace spanned by the first K right singular vectors of the covariance matrix retains the maximum information about the original vectors among all projections onto subspaces of dimension K. Hence, if K is the minimum of the two vector dimensions, projecting the two embeddings onto the first K right singular vectors of their covariance matrices will equalise their dimensions while losing minimal information [50]. The resulting transformation is expected to minimise the initial EMD between the distributions.

5.2.2 Earth mover's distance: The EMD [49, 51] is designed to evaluate the dissimilarity between two multi-dimensional distributions in some feature space. The EMD approach assumes that a distance measure between single features, called the 'ground distance', is given. The EMD then 'lifts' this distance from individual features to full distributions. Computing the EMD is based on a solution to the well-known 'transportation problem' [52], whose optimal value determines the minimum amount of 'work' required to transform one distribution into the other. Moreover, if the weights of the distributions are the same, and the ground distance is a metric, EMD induces a metric distance [49].

Recall that a translated and rotated version of a graph embedding will also be a graph embedding. To accommodate pairs of distributions that are 'not rigidly embedded', Cohen and Guibas [51] extended the definition of EMD, originally applicable to pairs of fixed sets of points, to allow one of the sets to undergo a transformation. They also suggested an iterative process (which they call FT, short for 'an optimal flow and an optimal transformation') that achieves an infimum of the objective function for EMD. The iterative process stops when the improvement in the objective function value falls below a threshold. For our application, the set of allowable transformations consists of only those transformations that preserve distances. Therefore we use a weighted version of the least squares estimation algorithm [53] to compute an optimal distance-preserving transformation given a flow between the distributions. The main advantage of using EMD lies in the fact that it subsumes many histogram distances and permits partial and many-to-many matches under transformation in a natural way. This important property allows the similarity measure to deal with uneven clusters and noisy datasets.

5.3 Demonstration

To demonstrate our approach to many-to-many matching, we turn to the domain of view-based object recognition using silhouettes. For a given view, an object's silhouette is first represented by an undirected, rooted, weighted graph, in which nodes represent 'shocks' [4] (or, equivalently, skeleton points) and edges connect adjacent shock points. [Note that this representation is closely related to Siddiqi et al.'s shock graph [4], except that our nodes (shock points) are neither clustered nor are our edges directed.] We will assume that each point p on the discrete skeleton is labelled by a four-dimensional vector $\boldsymbol{v(p)} = (x, y, r, \alpha)$, where (x, y) are the Euclidean coordinates of the point, r is the radius of the maximal bi-tangent circle centred at the point, and α is the angle between the normal to either bitangent and the linear approximation to the skeleton curve at the point. [Note that this four-tuple is slightly different from Siddiqi *et al.*'s shock point four-tuple, where the latter's radius is assumed normal to the axis.] This four-tuple can be thought of as encoding local shape information of the silhouette.

doi: 10.1049/iet-cvi.2012.0030

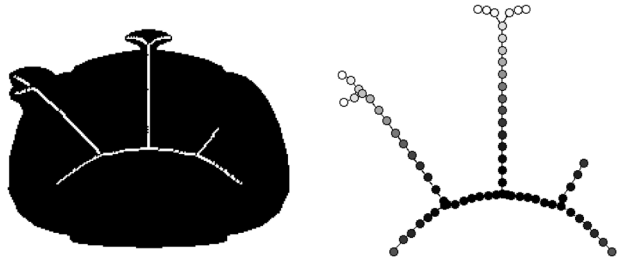


Fig. 16 *Left: the silhouette of a teapot and its medial axis; right: the medial axis tree constructed from the medial axis*
Darker nodes reflect larger radii

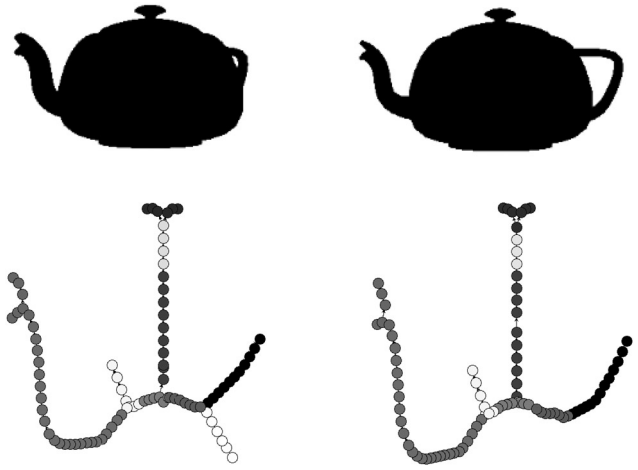


Fig. 17 *Illustration of the many-to-many correspondences computed for two adjacent views of the teapot*
Matched point clusters are shaded with the same colour

To convert our shock graphs to shock trees, we compute the minimum spanning tree of the weighted shock graph. As the edges of the shock graph are weighted based on Euclidean distances of corresponding nodes, the minimum spanning tree will a generate suitable tree approximation for shock graphs. The root of the tree is the node that minimises the sum of distances to all other nodes. Finally, each node is weighted proportionally to its average radius, with the total tree weight being 1. An illustration of the procedure is given in Fig. 16. The left portion shows the initial silhouette and its shock points (skeleton). The right portion depicts the constructed shock tree. Darker, heavier nodes correspond to fragments whose average radii are larger. Fig. 17 illustrates the many-to-many correspondences that our matching algorithm yields for two adjacent views (30° and 40°) of the teapot. Corresponding clusters (many-to-many mappings) have been shaded with the same colour. Note that the extraneous branch in the left view was not matched in the right view, reflecting the method's ability to deal with noise. Further details and additional experiments can be found in [50].

6 Conclusions

Within-class shape and appearance variation, scale change, articulation, noise and segmentation errors can all conspire to violate the common assumption that there exists a one-to-one correspondence between image and model features. If categorisation is to move beyond those restrictive categories where the same local features can be found on each exemplar belonging to the category, features must be matched many-to-many. We have reviewed three different graph-theoretic approaches to the problem of many-to-many feature matching. Establishing a many-to-many correspondence requires either that features are grouped and abstracted so that one-to-one correspondence exist between the abstractions or that many-to-many correspondences are explicitly found between groups of features.

Our first approach addresses the problem of learning a category model from a set of exemplars by searching the space of all possible abstractions of an image, and using support from other objects drawn from the same category to help constrain the search. Our second approach addresses the problem of hierarchical graph matching, drawing on spectral graph theory to generate and match structural abstractions of DAGs. Our third approach explicitly addresses the problem of many-to-many graph matching by first embedding the graphs into a geometric domain, and then employing efficient algorithms for many-to-many point matching. Although each of these approaches addresses a different problem, they all face the common challenge of perceptual grouping and image abstraction. Matching groups of features many-to-many requires that the features be nonaccidentally grouped, and it is only when the perceptual groups are abstracted can they be directly compared. In the wake of the community's formulation of recognition as detection, these two critical problems have unfortunately not received the attention they deserve [7].

7 Acknowledgments

The authors gratefully acknowledge the support of NSERC, IRIS, NSF, ONR, DARPA, and PREA.

8 References

1 Lowe, D.: 'Distinctive image features from scale-invariant keypoints', *IJCV*, 2004, **60**, (2), pp. 91–110
2 Shokoufandeh, A., Keselman, Y., Demirci, F., Macrini, D., Dickinson, S.J.: 'Many-to-many feature matching in object recognition', in Christensen, H., Nagel, H.-H. (Eds.): 'Cognitive vision systems: sampling the spectrum of approaches' (Springer-Verlag, Berlin, 2006), pp. 107–125
3 Keselman, Y., Dickinson, S.J.: 'Generic model abstraction from examples', *IEEE PAMI*, 2005, **27**, (7), pp. 1141–1156
4 Siddiqi, K., Shokoufandeh, A., Dickinson, S., Zucker, S.: 'Shock graphs and shape matching', *Int. J. Comput. Vis.*, 1999, **30**, pp. 1–24
5 Shokoufandeh, A., Macrini, D., Dickinson, S., Siddiqi, K., Zucker, S.: 'Indexing hierarchical structures using graph spectra', *IEEE PAMI*, 2005, **27**, (7), pp. 1125–1140
6 Demirci, F., Shokoufandeh, A., Keselman, Y., Bretzner, L., Dickinson, S.: 'Object recognition as many-to-many feature matching', *IJCV*, 2006, **69**, (2), pp. 203–222
7 Dickinson, S.: 'The evolution of object categorization and the challenge of image abstraction', in Dickinson, S., Leonardis, A., Schiele, B., Tarr, M. (Eds.): 'Object categorization: computer and human vision perspectives' (Cambridge University Press, New York, 2009), pp. 1–37
8 Crowley, J., Parker, A.: 'A representation for shape based on peaks and ridges in the difference of low-pass transform', *IEEE PAMI*, 1984, **6**, (2), pp. 156–170
9 Lindeberg, T.: 'Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus of attention', *Int. J. Comput. Vis.*, 1993, **1**, (3), pp. 283–318
10 Lindeberg, T.: 'Edge detection and ridge detection with automatic scale selection', *Int. J. Comput. Vis.*, 1998, **30**, (2), pp. 117–156
11 Dufournaud, Y., Schmid, C., Horaud, R.: 'Matching images with different resolutions'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Los Alamitos, 2000, pp. 612–618
12 Lowe, D.: 'Object recognition from local scale-invariant features'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Washington, 1999, pp. 1150–1157

13 Kadir, T., Brady, M.: 'Saliency, scale and image description', *Int. J. Comput. Vis.*, 2001, **45**, (2), pp. 83–105
14 Bay, H., Ess, A., Tuytelaars, T., van Gool, L.: 'Speeded-up robust features (surf)', *Comput. Vis. Image Underst.*, 2008, **110**, (3), pp. 346–359
15 Shokoufandeh, A., Marsic, I., Dickinson, S.: 'View-based object recognition using saliency maps', *Image Vis. Comput.*, 1999, **17**, (5/6), pp. 445–460
16 Bretzner, L., Lindeberg, T.: 'Qualitative multiscale feature hierarchies for object tracking', *J. Vis. Commun. Image Represent.*, 2000, **11**, (2), pp. 115–129
17 Shokoufandeh, A., Bretzner, L., Macrini, D., Demirci, M.F., Jönsson, C., Dickinson, S.: 'The representation and matching of categorical shape', *Comput. Vis. Image Understand*, 2006, **103**, (2), pp. 139–154
18 Mohan, A., Papageorgiou, C., Poggio, T.: 'Example-based object detection in images by components', *IEEE PAMI*, 2001, **23**, (4), pp. 349–361
19 Weber, M., Welling, M., Perona, P.: 'Unsupervised learning of models for recognition'. Proc. Sixth European Conf. Computer Vision, 2000, pp. 18–32
20 Leibe, B., Schiele, B.: 'Analyzing appearance and contour based methods for object categorization'. Proc. Sixth European Conf. Computer Vision, 2003, pp. 409–415
21 Winn, J., Jojic Locus, N.: 'Learning object classes with unsupervised segmentation'. Proc. of ICCV, 2005, pp. 756–763
22 Jiang, X., Munger, A., Bunke, H.: 'On median graphs: properties, algorithms, and applications', *IEEE PAMI*, 2001, **23**, (10), pp. 1144–1151
23 Torsello, A., Hancock, E.R.: 'Learning shape-classes using a mixture of tree-unions', *IEEE PAMI*, 2006, **28**, (6), pp. 954–967
24 Todorovic, S., Ahuja, N.: 'Unsupervised category modeling, recognition, and segmentation in images', *IEEE PAMI*, 2008, **30**, (12), pp. 2158–2174
25 Messmer, B., Bunke, H.: 'Efficient error-tolerant subgraph isomorphism detection', in Dori, D., Bruckstein, A. (Eds.): 'Shape, structure and pattern recognition' (World Scientific Publ. Co, 1995), pp. 231–240
26 Liu, T.L., Geiger, D.: 'Approximate tree matching and shape similarity'. Proc. Seventh Int. Conf. on Computer Vision, Kerkyra, Greece, 1997
27 Myers, R., Wilson, R., Hancock, E.: 'Bayesian graph edit distance', *IEEE PAMI*, 2000, **22**, (6), pp. 628–635
28 Sebastian, T., Klein, P., Kimia, B.: 'Recognition of shapes by editing shock graphs'. Proc. CVPR, 2001
29 Gold, S., Rangarajan, A.: 'Agraduated assignment algorithm for graph matching', *IEEE PAMI*, 1996, **18**, (4), pp. 377–388
30 Carcassoni, M., Hancock, E.R.: 'Correspondence matching with modal clusters', *IEEE PAMI*, 2003, **25**, (12), pp. 1609–1614
31 Caelli, T.: 'Correspondence matching with modal clusters', *IEEE PAMI*, 2004, **26**, (4), pp. 515–519
32 Dickinson, S., Davis, L.: 'A flexible tool for prototyping alv road following algorithms', *IEEE J. Robot. Autom.*, 1990, **6**, (2), pp. 232–242
33 Dickinson, S., Pentland, A., Rosenfeld, A.: '3-D shape recovery using distributed aspect matching', *IEEE PAMI*, 1992, **14**, (2), pp. 174–198
34 Dickinson, S., Pentland, A., Rosenfeld, A.: 'From volumes to views: an approach to 3-D object recognition', *CVGIP, Image Understanding*, 1992, **55**, (2), pp. 130–154
35 Felzenszwalb, P., Huttenlocher, D.: 'Image segmentation using local variation'. IEEE Conf. on Computer Vision and Pattern Recognition, Santa Barbara, CA, 1998, pp. 98–104
36 Stewart, G., Sun, J.: 'Matrix perturbation theory' (Academic Press, San Diego, CA, 1990)
37 Reyner, S.W.: 'An analysis of a good algorithm for the subtree problem', *SIAM J. Comput.*, 1977, **6**, pp. 730–732
38 Shokoufandeh, A., Bretzner, L., Macrini, D., Demirci, M.F., Jonsson, C., Dickinson, S.: 'The representation and matching of categorical shape', *CVIU*, 2006, **103**, (2), pp. 139–154
39 Shokoufandeh, A., Dickinson, S., Siddiqi, K., Zucker, S.: 'Indexing using a spectral encoding of topological structure'. IEEE Conf. on Computer Vision and Pattern Recognition, Fort Collins, CO, June 1999, pp. 491–497
40 Kimia, B.B., Tannenbaum, A., Zucker, S.W.: 'Shape, shocks, and deformations I: the components of two-dimensional shape and the reactiondiffusion space', *Int. J. Comput. Vis.*, 1995, **15**, pp. 189–224
41 Macrini, D., Shokoufandeh, A., Dickinson, S., Siddiqi, K., Zucker, S.: 'View-based 3-D object recognition using shock graphs'. Proc. Int. Conf. on Pattern Recognition, Quebec, 2002, vol. 3, pp. 24–28
42 Macrini, D.: 'Indexing and matching for view-based 3-D object recognition using shock graphs'. Master's thesis, Department of Computer Science, University of Toronto, 2003
43 Shokoufandeh, A., Dickinson, S.J., Bretzner, J.L., Lindeberg, T.: 'On the representation and matching of qualitative shape at multiple scales'. Proc. Seventh European Conf. on Computer Vision, 2002, vol. 3, pp. 759–775
44 Gupta, A., Newman, I., Rabinovich, Y., Sinclair, A.: 'Cuts, trees and l_1 embeddings'. Proc. Symp. on Foundations of Computer Science, 1999, pp. 399–409
45 Linial, N., London, E., Rabinovich, Y.: 'The geometry of graphs and some of its algorithmic applications'. Proc. 35th Annual Symp. on Foundations of Computer Science, 1994, pp. 557–591
46 Matoušek, J.: 'On embedding trees into uniformly convex Banach spaces', *Israel J. Math.*, 1999, **237**, pp. 221–237
47 Agarwala, R., Bafna, V., Farach, M., Paterson, M., Thorup, M.: 'On the approximability of numerical taxonomy (fitting distances by tree metrics)', *SIAM J. Comput.*, 1999, **28**, (2), pp. 1073–1085
48 Linial, N., Magen, A., Saks, M.E.: 'Trees and Euclidean metrics'. Proc. 30th Annual ACM Symp. on Theory of Computing, 1998, pp. 169–175
49 Rubner, Y., Tomasi, C., Guibas, L.J.: 'The earth mover's distance as a metric for image retrieval', *Int. J. Comput. Vis.*, 2000, **40**, (2), pp. 99–121
50 Keselman, Y., Shokoufandeh, A., Demirci, M., Dickinson, S.: 'Many-tomany graph matching via metric embedding'. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Madison, WI, June 2003, pp. 850–857
51 Cohen, S.D., Guibas, L.J.: 'The earth mover's distance under transformation sets'. Proc. Seventh Int. Conf. on Computer Vision, Kerkyra, Greece, 1999, pp. 1076–1083
52 Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: 'Network flows: theory, algorithms, and applications' (Prentice Hall, Englewood Cliffs, New Jersey, 1993), pp. 4–7
53 Umeyama, S.: 'Least-squares estimation of transformation parameters between two point patterns', *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1991, **13**, (4), pp. 376–380

doi: 10.1049/iet-cvi.2012.0030