

The representation and matching of categorical shape

Ali Shokoufandeh ^{a,*}, Lars Bretzner ^b, Diego Macrini ^c, M. Fatih Demirci ^a,
Clas Jönsson ^b, Sven Dickinson ^c

^a Department of Computer Science, Drexel University, Philadelphia, PA 19104, USA

^b Computational Vision and Active Perception Laboratory, Department of Numerical Analysis and Computer Science, KTH, Stockholm, Sweden

^c Department of Computer Science, University of Toronto, Toronto, Ont., Canada M5S 3G4

Received 18 July 2005; accepted 16 May 2006

Available online 30 June 2006

Abstract

We present a framework for categorical shape recognition. The coarse shape of an object is captured by a multiscale blob decomposition, representing the compact and elongated parts of an object at appropriate scales. These parts, in turn, map to nodes in a directed acyclic graph, in which edges encode both semantic relations (parent/child) as well as geometric relations. Given two image descriptions, each represented as a directed acyclic graph, we draw on spectral graph theory to derive a new algorithm for computing node correspondence in the presence of noise and occlusion. In computing correspondence, the similarity of two nodes is a function of their topological (graph) contexts, their geometric (relational) contexts, and their node contents. We demonstrate the approach on the domain of view-based 3-D object recognition.

© 2006 Elsevier Inc. All rights reserved.

Keywords: Generic object recognition; Shape categorization; Graph matching; Scale-spaces; Spectral graph theory

1. Introduction

Object categorization has long been a goal of the object recognition community, with notable early work by Binford [3], Marr and Nishihara [32], Agin and Binford [1], Nevatia and Binford [35], and Brooks [6] attempting to construct and recognize prototypical object models based on their coarse shape structure. However, the representational gap between low-level image features and the abstract nature of the models was large, and the community lacked the computational infrastructure required to bridge this representational gap [22]. Instead, images were simplified and objects were textureless, so that extracted image features could map directly to salient model features. In the years to follow, such generic object recognition systems gave way to exemplar-based systems, first passing

through highly constrained geometric (CAD) models, and on to today's generation of appearance-based models. Whether the images were moved closer to the models (earlier approaches) or the models moved closer to the images (later approaches), the representational gap has remained largely unaddressed.

The community is now returning to the problem of object categorization, using powerful machine learning techniques and new appearance-based features. However, appearance-based representations are not invariant to significant within-class appearance change, due to texture, surface markings, structural detail, or articulation. As a result, the categories are often very restricted, such as faces, cars, motorcycles, and specific species of animals. Accommodating significant within-class shape variation puts significant demands on a representation: (1) it must capture the coarse structure of an object; and (2) it must be local, in order to accommodate occlusion, clutter, and noise. These criteria point to a structured shape description not unlike the ambitious part-based recognition frameworks

* Corresponding author.

E-mail address: ashokouf@cs.drexel.edu (A. Shokoufandeh).

proposed in the 70s and 80s. However, the representational gap facing these early systems still poses a major obstacle.

We begin by imposing a strong shape prior on the parts and relations making up an object. Specifically, we represent a 2-D object view as a multiscale blob decomposition, whose part vocabulary includes two types, compact blobs and elongated ridges, and whose relations also include two types, semantic (parent/child and sibling) and geometric. The detected qualitative parts and their relations are captured in a directed acyclic graph, called a *blob graph*, in which nodes represent parts and edges capture relations.

Choosing a restricted vocabulary of parts helps us bridge the representational gap. Early generation systems started with low-level features such as edges, regions, and interest points, and were faced with the challenging task of grouping and abstracting them to form high-level parts, such as generalized cylinders. By severely restricting the part vocabulary, we construct a high-level part detector from simple, multiscale filter responses. Although the parts are simple and qualitative, their semantic and geometric relations add rich structure to the representation, yielding a shape representation that can be used to discriminate shape categories.

Any shortcut to extracting high-level part structure from low-level features, such as filter responses, will be prone to error. Thus, the recovered blob graph will be missing nodes/edges and will contain spurious nodes/edges, setting up a challenging inexact graph-matching problem. In our previous work on matching rooted trees [50], we drew on spectral graph theory to represent the coarse “shape” of a tree as a low-dimensional vector based on the eigenvalues of the tree’s symmetric adjacency matrix. Our matching algorithm utilized these vectors in a coarse-to-fine matching strategy that found corresponding nodes. Although successfully applied to *shock trees*, the matching algorithm suffered from a number of limitations: (1) it could not handle the directed acyclic graph structure found in, for example, our multiscale blob graphs, e.g., a blob may have two parents; (2) it was restricted to matching hierarchical parent/child relations, and could not accommodate sibling relations, e.g., the geometric relations between blobs at a given scale; and (3) the matching algorithm was an approximation algorithm that could not ensure that hierarchical and sibling relations were preserved during matching, allowing, for example, two siblings (sharing a parent) in one tree to match two nodes in the other tree having an ancestor/descendant relationship.

We first extend our matching algorithm to handle directed acyclic graphs, drawing on our recent work in indexing hierarchical structures [48], in which we represent the coarse shape of a directed acyclic graph as a low-dimensional vector based on the eigenvalues of the DAG’s anti-symmetric adjacency matrix. Next, we introduce a notion of graph neighborhood context, allowing us to accommodate sibling relations, such as our blob graphs’ geometric relations, into our matching algorithm. Finally, we extend the matching algorithm to ensure that both hierarchical

and sibling relations are enforced during matching, yielding improved correspondence in the presence of noise and occlusion. The result is a far more powerful matching framework that’s ideally suited to our multiscale blob graphs.

Following a review of related work in Section 2, we describe our qualitative shape representation in Section 3. Next, we describe our new matching algorithm in Section 4. In Section 5, we evaluate the approach on the domain of view-based 3-D object recognition. We discuss the limitations of the approach in Section 6, and draw conclusions in Section 7.

2. Related work

There has been considerable effort devoted to both scale space theory and hierarchical structure matching, although much less effort has been devoted to combining the two paradigms. Coarse-to-fine image descriptions are plentiful, including work by Burt [7], Lindeberg [30], Simoncelli et al. [51], Mallat and Hwang [31], and Witkin and Tenenbaum [57]. Some have applied such models to directing visual attention, e.g., Tsotsos [54], Jägersand [20], Olshausen et al. [36], and Takačs and Wechsler [52]. Although such descriptions are well suited for tasks such as compression, attention, or object localization, they often lose the detailed shape information required for object recognition.

Others have developed multi-scale image descriptions specifically for matching and recognition. Crowley and Sanderson [10,9,11] extracted peaks and ridges from a Laplacian pyramid and linked them together to form a tree structure. However, during the matching phase, little of the trees’ topology or geometry was exploited to compute correspondence. Rao et al. [40] correlate a rigid, multiscale saliency map of the target object with the image. However, like any template-based approach, the technique is rather global, offering little invariance to occlusion or object deformation. Boyer and Kak [4] also considered the construction of a structural description from noisy image primitives. In their approach, the structural description of an object consists of set of primitive features and their relationships in the scene, e.g., pairwise orientation among features, distance, and an information-theoretic measure of relational inconsistency. These relationships were subsequently extended to a set of parametric descriptions whose values represented the strengths with which a given pair of primitives participate in the relationships. The resulting descriptors were used to identify an optimal mapping function between the descriptions. In an effort to accommodate object deformation, Wiskott et al. apply elastic graph matching to a planar graph whose nodes are wavelet jets. Although their features are multiscale, their representation is not hierarchical, and matching requires that the graphs be coarsely aligned in scale and image rotation [56]. A similar approach was applied to hand posture recognition by Triesch and von der Malsburg [53].

The representation of image features at multiple scales suggests a hierarchical graph representation, which can accommodate feature attributes in the nodes and relational attributes in the arcs. Although graph matching is a popular topic in the computer vision literature [14], including both inexact and exact graph matching algorithms, there is far less work on dealing with the matching of hierarchical graphs, i.e., DAGs, in which lower (deeper) levels reflect less saliency. Pelillo et al. [38] provided a solution for the matching of hierarchical trees by constructing an association graph using the concept of connectivity and solving a maximal clique problem in this new structure. The latter problem can be formulated as a quadratic optimization problem and they used the so-called replicator dynamical systems developed in theoretical biology to solve it. Pelillo [37] generalized the framework for matching free trees and applied simple payoff-monotonic dynamics from evolutionary game theory.

Pelillo et al. proposed a solution to the many-to-many matching of attributed trees [39]. They defined the notion of an ϵ -morphism between trees, and provided a reduction from the matching problem to a maximum weight clique in an association graph. Their solution to the matching problem used replicator dynamical systems from evolutionary game theory. The problem of matching hierarchical trees has also been studied in the context of edit-distance (see, e.g., [44]). In such a setting, one seeks a minimal set of re-labeling, additions, deletions, merges, and splits of nodes and edges that transform one graph into another.

In recent work [12,13], we presented a framework for many-to-many matching of hierarchical structures, where features and their relations were represented using directed edge-weighted graphs. The method begins by transforming the graph into a metric tree. Next, using graph embedding techniques, the tree was embedded into a normed vector space. This two-step transformation allowed us to reduce the problem of many-to-many graph matching to a much simpler problem of matching weighted distributions of points in a normed vector space. To compute the distance between two weighted distributions, we used a distribution-based similarity measure, known as the Earth Mover's Distance under transformation [8,42].

As mentioned in Section 1, object categorization has received renewed attention from the recognition community. In [16], Fergus et al. present a method to learn and recognize object class models from unlabeled cluttered scenes in a scale invariant manner. They employ a probabilistic representation to simultaneously model shape, appearance, occlusion, and relative scale. They also use expectation maximization for learning the parameters of the scale-invariant object model and use this model in a Bayesian manner to classify images. Fei-Fei et al. [15] improved on this by proposing a method for learning object categories from just a few training images. Their proposed method is based on making use of priors to construct a generative probabilistic model, assembled from object categories that were previously learned.

Lazebnik et al. [25] present a framework for the representation of 3-D objects using multiple composite local affine parts. Specifically, these are 2-D configurations of regions that are stable across a range of views of an object, and also across multiple instances of the same object category. These composite representations provide improved expressiveness and distinctiveness, along with added flexibility for representing complex 3-D objects. Leibe and Schiele [27] propose a new database for comparing different methods for object categorization. The database contains high-resolution color images of 80 objects from eight different categories, for a total of 3280 images and was used to analyze the performance of several appearance- and contour-based methods. The best reported method for categorization on this database is a combination of both contour- and shape-based methods. This new generation of categorization systems is primarily appearance-based, and therefore not well-equipped to handle within-class deformations due to significant appearance change, articulation, and significant changes in minor geometric detail.

Eigenvalue-based methods have been used by several researchers to tackle problems in object recognition. Sengupta and Boyer [45] introduced an eigenvalue-based feature representation for CAD models to capture their gross characteristics. This representation was used to compare objects and partition the dataset into structurally homogeneous groups. Shapiro and Brady [46] computed feature correspondences using the eigenvectors of proximity graphs. The closest integrated framework to that proposed here is due to Shokoufandeh et al. [49], who match multiscale blob representations represented as directed acyclic graphs. The multi-scale description in that work, due to Marsic [33], excluded geometric relations among sibling nodes and did not include ridge features. Moreover, the matching framework had to choose between two algorithms, one targeting structural matching, with the other enforcing both structural and geometrical graph alignment. Our proposed multi-scale image representation is far richer in terms of its underlying features, and resembles that of Bretzner and Lindeberg [5], who explored qualitative, multi-scale hierarchies for object tracking. Our matching framework, on the other hand, offers several orders of magnitude less complexity, handles noise more effectively, and can handle both structural and geometrical matching within the same framework.

There is also a large body of work on solving the model matching problem via establishing direct feature correspondences (without graph matching), and we highlight only a few examples. Sclaroff and Pentland [43] proposed the use of generalized symmetries, as defined by each object's eigenmodes, for object recognition. They formulate shape dissimilarity as the amount of deformation required to align the proposed modal descriptions. In [18], Gdalyahu and Weinshall proposed a dynamic programming approach for curve matching under deformation, scaling, and rigid transformations. Belongie et al. [2] used shape context associated with feature points to

measure similarity between shapes. Their proposed matching algorithm uses the shape context descriptors for solving the correspondence between the two shapes and then uses the correspondences to estimate an aligning transformation. Finally, a two-step procedure consisting of embedding and Earth Mover's Distance (EMD) was proposed by Grauman and Darrell [21] for the matching of 2-D contours represented as shape context-like distributions. However, the abstract nature of the embedding means that the explicit many-to-many correspondences between two feature sets cannot be recovered.

3. Building a qualitative shape feature hierarchy

3.1. Extracting qualitative shape features

As mentioned in Section 1, our qualitative feature hierarchy represents an image in terms of a set of blobs and ridges, captured at appropriate scales. The representation is an extension of the work presented in [5]. Blob and ridge extraction is performed using automatic scale selection, as described in previous work (see [29,28]). We also extract a third descriptor, called the *windowed second moment matrix*, which describes the directional characteristics of the underlying image structure.

A scale-space representation of the image signal f is computed by convolution with Gaussian kernels $g(\cdot; t)$ of different variance t , giving $L(\cdot; t) = g(\cdot; t) * f(\cdot)$. Blob detection aims at locating compact objects or parts in the image. The entity used to detect blobs is the square of the normalized Laplacian operator,

$$\nabla_{\text{norm}}^2 L = t(L_{xx} + L_{yy}). \quad (1)$$

Blobs are detected as local maxima in scale-space. Fig. 1(a) shows an image of a hand with the extracted blobs superimposed. A blob is graphically represented by a circle defining a *support region*, whose radius is proportional to \sqrt{t} .

Elongated structures are localized where the multi-scale ridge detector

$$\mathfrak{R}_{\text{norm}} L = t^{3/2}((L_{xx} - L_{yy})^2 + 4L_{xy}^2) \quad (2)$$

assumes local maxima in scale-space. Fig. 1(a) also shows the extracted ridges, represented as superimposed ellipses, each defining a support region, with width proportional to \sqrt{t} . To represent the spatial extent of a detected image structure, a windowed second moment matrix

$$\Sigma = \int_{\eta \in \mathbb{R}^2} \begin{pmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{pmatrix} g(\eta; t_{\text{int}}) d\eta \quad (3)$$

is computed at the detected feature position and at an integration scale t_{int} proportional to the scale t_{det} of the detected image feature. There are two parameters of the directional statistics that we make use of here: the *orientation* and the *anisotropy*, given from the eigenvalues λ_1 and λ_2 ($\lambda_1 > \lambda_2$) and their corresponding eigenvectors \vec{e}_{λ_1} and \vec{e}_{λ_2} of Σ . The anisotropy is defined as

$$\tilde{Q} = \frac{1 - \lambda_2/\lambda_1}{1 + \lambda_2/\lambda_1}, \quad (4)$$

while the orientation is given by the direction of \vec{e}_{λ_1} .

To improve feature detection in scenes with poor intensity contrast between the image object and background, we utilize color information. This is done by extracting features in the R, G and B color bands separately, along with the intensity image. Re-occurring features are rewarded with respect to significance. Furthermore, if we have advance information on the color of the object, improvements can be achieved by weighting the significance of the features in the color bands differently.

When constructing a feature hierarchy, we extract the N most salient image features, ranked according to the response of the scale-space descriptors used in the feature extraction process. From these features, a *Feature Map* is built as described in the following sections.

3.1.1. Merging multiple feature responses

This step removes multiple overlapping responses originating from the same image structure, the effect of which can be seen in Fig. 1(a). To be able to detect overlapping features, a measure of inter-feature similarity is needed. For this purpose, each feature is associated with a 2-D Gaussian kernel $g(\vec{x}, \Sigma)$, where the covariance is given from the second

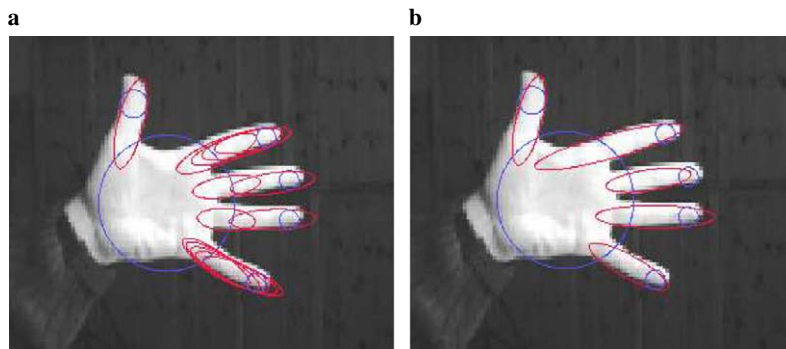


Fig. 1. Feature extraction: (a) extracted blobs and ridges at appropriate scales, (b) extracted features before and after removing multiple responses and linking ridges.

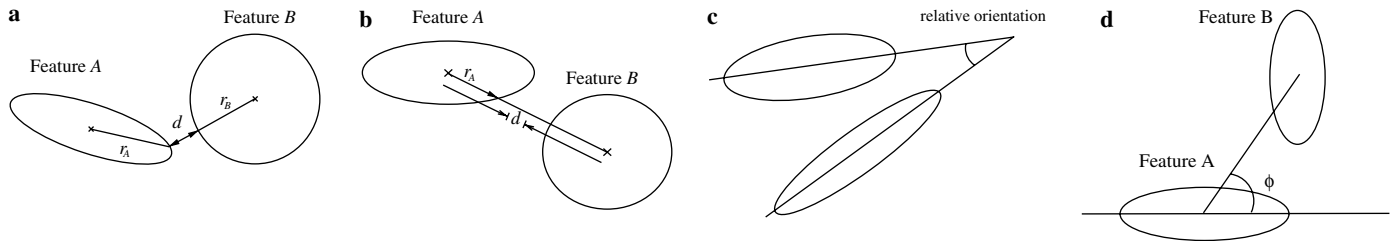


Fig. 2. Four of the relations between features: (a and b) two normalized distance measures, (c) relative orientation, and (d) bearing.

moment matrix in Eq. (3). When two features are positioned near each other, their Gaussian functions will intersect. The similarity measure between two such features is based on the *disjunct volume* D of the two Gaussians [24]. This volume is calculated by integrating the square of the difference between the two Gaussian functions (g_A, g_B) corresponding to the two intersecting features A and B:

$$D(A, B) = \sqrt{\frac{|\Sigma_A| + |\Sigma_B|}{2}} \int_{\mathbb{R}^2} (g_A - g_B)^2 dx. \quad (5)$$

The choice of a Gaussian function is due to the fact that it maximizes the entropy of a random variable given its mean and covariance. The disjunct volume depends on the differences in position, variance, scale and orientation of the two Gaussians. For ridges, the measure is more sensitive to translations in the direction perpendicular to the ridge, which is desirable as the ridge is better localized in this direction.

3.1.2. Linking ridges

The ridge detection will produce multiple responses on a ridge structure that is long compared to its width. These ridges are linked together to form one long ridge, as illustrated in Fig. 1(b). The criteria for when to link two ridges are based on two conditions: (1) they must be aligned, and (2) their support regions must overlap. After the linking is performed, the anisotropy and support region for the resulting ridge is re-calculated. The anisotropy is re-calculated from the new length/width relationship as $1 - (\text{width of structure}) / (\text{length of structure})$.

3.2. Assembling the features into a graph

Once the Feature Map is constructed, the component features are assembled into a directed acyclic graph. Associated with each node (blob/ridge) are a number of attributes, including position, orientation, and support region. The feature at the coarsest scale of the Feature Map is chosen as the root. Next, finer-scale features that overlap with the root become its children through hierarchical edges. These children, in turn, select overlapping features (again through hierarchical edges) at finer scales to be their children, etc. From the unassigned features, the feature at the coarsest scale is chosen as a new root. Children of this root are selected from unassigned as well as assigned features, and the process is repeated until all features are

assigned to a graph. A child node can therefore have multiple parents. To yield a single rooted graph, which is needed in the matching step, a virtual top root node is inserted as the parent of all root nodes in the image. Sibling relationships are introduced between nodes that share a parent and do not have a hierarchical edge between them.¹

Associated with every pair of related features are a number of important geometric attributes. Specifically, for a pair of vertices, \mathcal{V}_A representing feature \mathcal{F}_A and \mathcal{V}_B representing feature \mathcal{F}_B , related through a sibling relation or a hierarchical edge $\mathcal{E} = (\mathcal{V}_A, \mathcal{V}_B)$, we define the following attributes, as shown in Fig. 2:

- *Distance*. For sibling vertices, the smallest distance d from the support region of \mathcal{F}_A to the support region of \mathcal{F}_B , normalized to the largest of the radii r_A and r_B (this distance will be zero for overlapping features). If there is a hierarchical edge \mathcal{E} between vertices \mathcal{V}_A and \mathcal{V}_B , the distance between their centers is normalized to the radius r_A of \mathcal{F}_A in the direction of the distance vector between their centers.
- *Relative orientation*. The relative orientation between \mathcal{F}_A and \mathcal{F}_B .
- *Bearing*. The bearing of a feature \mathcal{F}_B , as seen from a feature \mathcal{F}_A , is defined as the angle of the distance vector $x_B - x_A$ with respect to the orientation of A measured counter-clockwise.
- *Scale ratio*. The scale invariant relation between \mathcal{F}_A and \mathcal{F}_B is a ratio between scales $t_{\mathcal{F}_A}$ and $t_{\mathcal{F}_B}$.

An example of a blob graph for a hand image, showing hierarchical edges, is shown in Fig. 3.

4. Matching problem formulation

Given two images and their corresponding Feature Map graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with $|V_1| = n_1$ and $|V_2| = n_2$, we seek a method for computing their similarity. In the absence of noise, segmentation errors, occlusion, and clutter, computing the similarity of G_1 and G_2 could be formulated as a label-consistent graph isomorphism problem. However, under normal imaging conditions, there may not exist significant subgraphs common to G_1 and G_2 . We

¹ Sibling relationships are not encoded as edges in a structural description of the graph, thereby ensuring that it is a DAG.

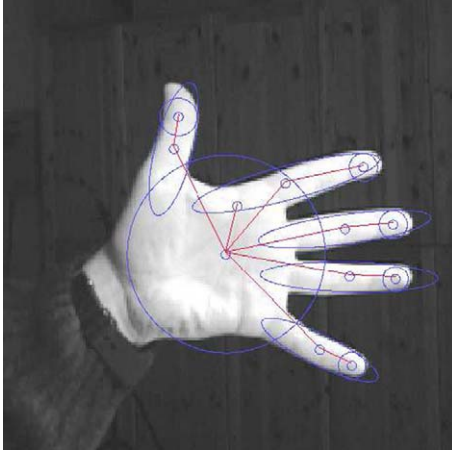


Fig. 3. Example graph of a hand image, with the hierarchical edges shown in green. (For interpretation of the references to colours in this figure legend, the reader is referred to the web version of this paper.)

therefore seek an approximate solution that captures both the structural and geometrical similarity of G_1 and G_2 as well as corresponding node similarity. Structural similarity is a domain-independent measure that accounts for similarity in the “shapes” of two graphs, in terms of numbers of nodes, branching factor distributions, etc. Geometrical similarity, on the other hand, accounts for consistency in the relative positions, orientations, and scales of nodes in the two graphs. In the following subsections, we describe these two signatures and combine them in an efficient algorithm to match two blob graphs.

4.1. Encoding graph structure

As described in Section 1, our previous work on rooted tree matching drew on the eigenvalues of a tree’s symmetric $\{0, 1\}$ adjacency matrix to encode the “shape” of a tree using a low-dimensional vector. The eigenvalues of a graph’s adjacency matrix characterize the graph’s degree distribution, an important structural property of the graph. In extending that approach to DAG matching, we first draw on our recent work in indexing hierarchical (DAG) structures [48], in which the *magnitudes* of the eigenvalues of a DAG’s antisymmetric $\{0, 1, -1\}$ adjacency matrix² are used to encode the shape of a DAG using a low-dimensional vector. Moreover, the eigenvalues are invariant to minor structural perturbation of the graph due to noise and occlusion [45,48].

Let us briefly review the construction of our graph abstraction; details can be found in [48]. Let \mathcal{D} be a DAG whose maximum branching factor is $\Delta(\mathcal{D})$, and let the subgraphs of its root be $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_s$, as shown in Fig. 4. For each subgraph, \mathcal{D}_i , whose root degree is $\delta(\mathcal{D}_i)$, we compute the magnitudes of the (complex) eigen-

values of \mathcal{D}_i ’s submatrix, sort the magnitudes in decreasing order, and let S_i be the sum of the $\delta(\mathcal{D}_i) - 1$ largest magnitudes. The sorted S_i ’s become the components of a $\Delta(\mathcal{D})$ -dimensional vector assigned to the DAG’s root. If the number of S_i ’s is less than $\Delta(\mathcal{D})$, then the vector is padded with zeroes. We can recursively repeat this procedure, assigning a vector to each non-terminal node in the DAG, computed over the subgraph rooted at that node. We call each such vector a *topological signature vector*, or TSV. The TSV assigned to a node allows the structural context of the node (i.e., the subgraph rooted at the node) to be encapsulated in the node as an attribute. Finally, it should be noted that a node may be a member of multiple subgraphs, for a node may have multiple parents in a directed acyclic graph. In this case, the node will contribute to the TSV of each of its parents. Node d in Fig. 4 is an example of such a node.

4.2. Encoding graph geometry

The above encoding of structure suffers from the drawback that it does not capture the geometry of the nodes. For example, two graphs with identical structure may differ in terms of the relative positions of their nodes, the relative orientations of their nodes (for elongated nodes), and the relative scales of their nodes. Just as we derived a topological signature vector, which encodes the “neighborhood” structure of a node, we now seek an analogous “geometrical signature vector”, which encodes the neighborhood geometry of a node. This geometrical signature will be combined with our new topological signature in a new algorithm that computes the distance between two directed acyclic graphs and preserves hierarchical constraints.

Let $G = (V, E)$ be a graph to be recognized (input image). For every pair of vertices $u, v \in V$, if there is an edge $\mathfrak{E} = (u, v)$ between them, we let $R_{u,v}$ denote the attribute vector associated with edge \mathfrak{E} . The entries of each such vector represent the set of relations $\mathfrak{R} = \{\text{distance, relative orientation, bearing, scale ratio}\}$ between u and v , as shown in Fig. 5. For a vertex $u \in V$, we let $N(u)$ denote the set of vertices $v \in V$ such that the pair (u, v) form a sibling relationship. For a relation $p \in \mathfrak{R}$, we will use $\mathfrak{F}(u, p)$ to denote the distribution of values of relation p between vertex u and all the vertices in the set $N(u)$, i.e., $\mathfrak{F}(u, p)$ is a histogram encoding the p th entry of the vectors $R_{u,v}$ for $v \in N(u)$.³

Given two graphs $G = (V, E)$ and $G' = (V', E')$ with vertices $u \in V$ and $u' \in V'$, we compute the similarity between u and u' in terms of their respective distributions $\mathfrak{F}(u, p)$ and $\mathfrak{F}(u', p)$, for all $p \in \mathfrak{R}$. Now, let $d_p(u, u')$ denote the Earth Mover’s Distance (EMD) [42] between two such distributions $\mathfrak{F}(u, p)$ and $\mathfrak{F}(u', p)$, i.e., $d_p(u, u')$ denotes the

² A matrix with 1’s (–1’s) indicating a forward (backward) edge between adjacent nodes in the graph (and 0’s on the diagonal).

³ The exception to this rule is the orientation relation. Rather than using absolute orientation, measured with respect to a reference direction, we instead use the angle from the previous edge in a clockwise ordering of edges emanating from a vertex.

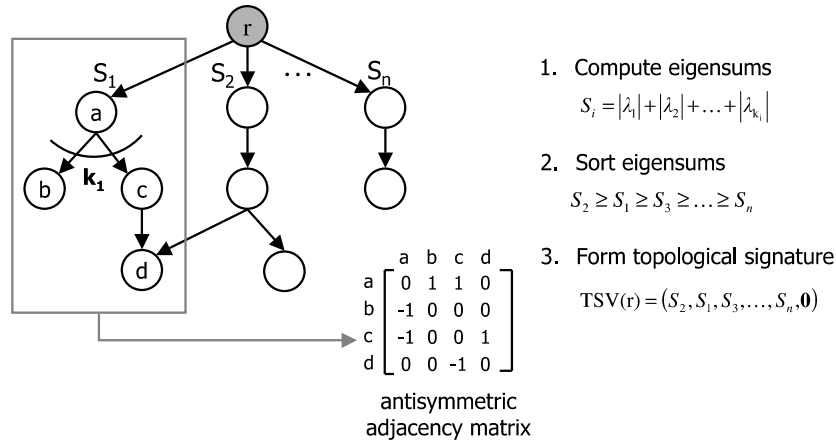


Fig. 4. Forming the structural signature.

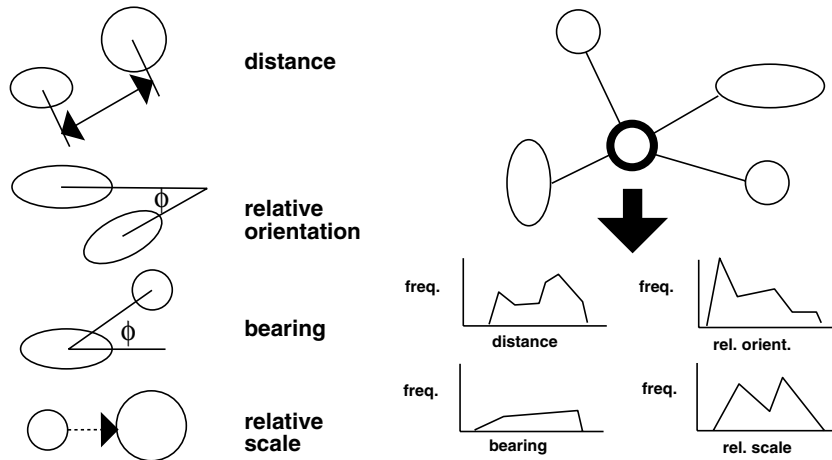


Fig. 5. Forming the geometric signature.

minimum amount of work (defined in terms of displacements of the masses associated with histograms $\mathfrak{P}(u, p)$ and $\mathfrak{P}(u', p)$) it takes to transform one distribution into another. The main advantage of using EMD to compute $d_p(u, u')$ lies in the fact that it subsumes many histogram distances and permits partial matches in a natural way. This important property allows the similarity measure to deal with the case where the masses associated with distributions $\mathfrak{P}(u, p)$ and $\mathfrak{P}(u', p)$ are not equal. Details of the method are presented in [23]. Given the values of $d_p(u, u')$ for all $p \in \mathfrak{R}$, we arrive at a final node similarity function for vertices u and u' :

$$\sigma(u, u') = e^{-\sum_{p \in \mathfrak{R}} d_p(u, u')}$$

4.3. Matching algorithm

As mentioned in Section 1, our previous work addressed the problem of matching rooted trees, and was unable to match directed acyclic graphs, unable to accommodate geometric relations among nodes, and unable to preserve

hierarchical and sibling relations. Still, it serves as the starting point for our new algorithm, and we review it accordingly. The method was a modified version of Reyner’s algorithm [41,55] for finding the largest common subtree. The main idea of the algorithm was to cast the structural matching problem as a set of bipartite graph-matching problems. A similarity matrix between the two graphs’ nodes was constructed with each entry computing the pairwise similarity between a particular node in the first tree and a node in the second tree. This similarity measure was a weighted combination of the distance between the two nodes’ TSVs, reflecting the extent to which their underlying subtrees had similar structure, and the two nodes’ internal attributes were similar.⁴

The algorithm is illustrated in Fig. 6. The best pairwise node correspondence obtained after a maximum cardinality maximum weight (MCMW) bipartite matching is

⁴ For shock graphs, each node encoded a set of medial axis, or shock, points and their similarity was computed based on a Hausdorff distance between these point sets.

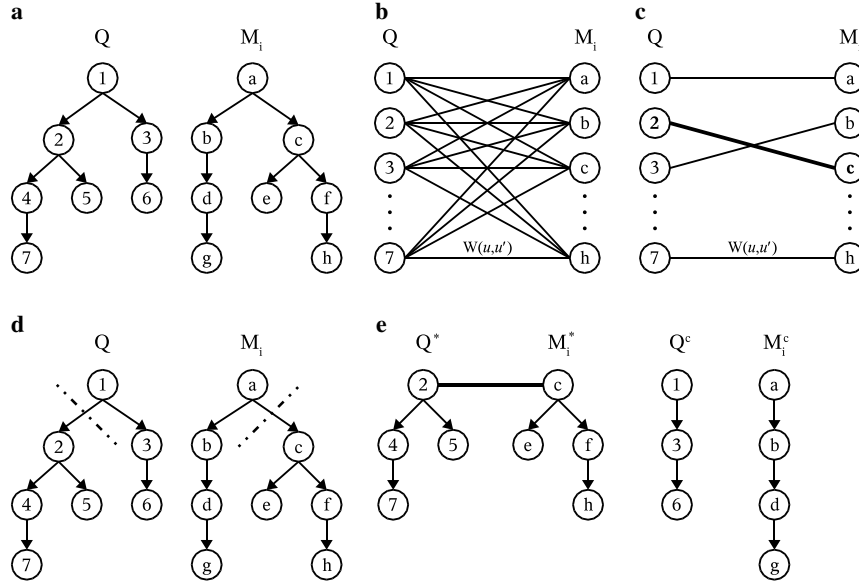


Fig. 6. The DAG matching algorithm. (a) Given a query graph and a model graph, (b) form bipartite graph in which the edge weights are the pair-wise node similarities. Then, (c) compute a maximum matching and add the best edge to the solution set. Finally, (d) split the graphs at the matched nodes, and (e) recursively descend.

extracted and put into the solution set of correspondences. In a greedy fashion, the algorithm recursively matches the two resulting pairs of corresponding forests, at each step computing a maximum matching and placing the best corresponding pair of nodes in the solution set. The key idea in casting a graph-matching problem as a number of MCMW bipartite matching problems is to use the topological signature vectors (TSV) to penalize nodes with different underlying graph structure. This effectively allows us to discard the graphs' edge structure and formulate the problem as an attributed point matching problem, with a node's underlying structural context encoded as a low-dimensional vector node attribute.

To extend this framework to accommodate DAG matching, geometric relations, and hierarchical/sibling constraint satisfaction, we begin by introducing some definitions and notations. Let $\mathfrak{Q} = (V_{\mathfrak{Q}}, E_{\mathfrak{Q}})$ and $\mathfrak{M} = (V_{\mathfrak{M}}, E_{\mathfrak{M}})$ be the two DAGs to be matched, with $|V_{\mathfrak{Q}}| = n_{\mathfrak{Q}}$ and $|V_{\mathfrak{M}}| = n_{\mathfrak{M}}$. Define d to be the maximum degree of any vertex in \mathfrak{Q} and \mathfrak{M} , i.e., $d = \max(\delta(\mathfrak{Q}), \delta(\mathfrak{M}))$. For each vertex v , let $\chi(v) \in R^d$ be the topological signature vector (TSV), introduced in Section 4.1. The bipartite edge-weighted graph $\mathfrak{G} = (V_{\mathfrak{Q}}, V_{\mathfrak{M}}, E_{\mathfrak{G}})$ is represented as a $n_{\mathfrak{Q}} \times n_{\mathfrak{M}}$ matrix \mathbf{W} whose (q,m) th entry has the value:

$$\mathbf{W}_{q,m} = \alpha\sigma(q,m) + (1 - \alpha)(\|\chi(q) - \chi(m)\|), \quad (6)$$

where $\sigma(q,m)$ denotes the node similarity between nodes $q \in \mathfrak{Q}$ and $m \in \mathfrak{M}$, and α is a convexity parameter that weights the relevance of each term. Using the scaling algorithm of Gabow and Tarjan [17], we can efficiently compute the maximum cardinality, maximum weight matching in \mathfrak{G} with complexity $O(|V||E|)$, resulting in a list

of node correspondences between \mathfrak{Q} and \mathfrak{M} , called \mathfrak{Q} , that can be ranked in decreasing order of similarity.

The relative contributions of the topological and geometrical components is clearly task dependent. Assigning a low weight to the geometric term emphasizes only the structural differences between two graphs while abstracting away the details of the parts. Such a weighting is appropriate for very coarse, structural matching. Conversely, increasing the weight of the geometric term accentuates the detailed geometric differences between the features, which is more appropriate for subclass, or exemplar recognition. Thus, changing the weight of α from a low value to a high value effectively yields a coarse-to-fine matching strategy.

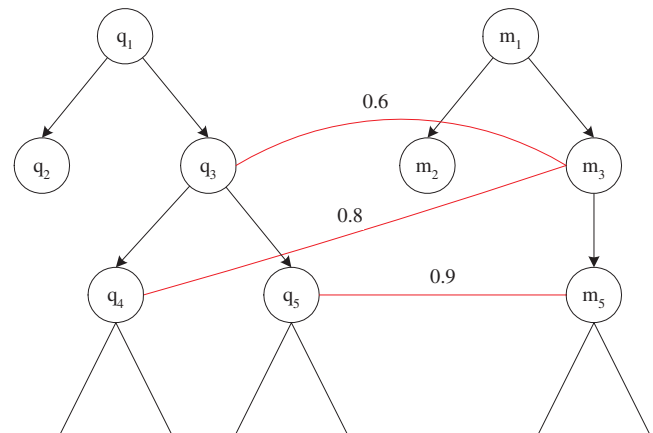


Fig. 7. A case in which the hierarchical constraints between query nodes and model nodes will be violated after two iterations of the algorithm. Note that only non-zero $\mathbf{W}_{q,m}$ values are shown.

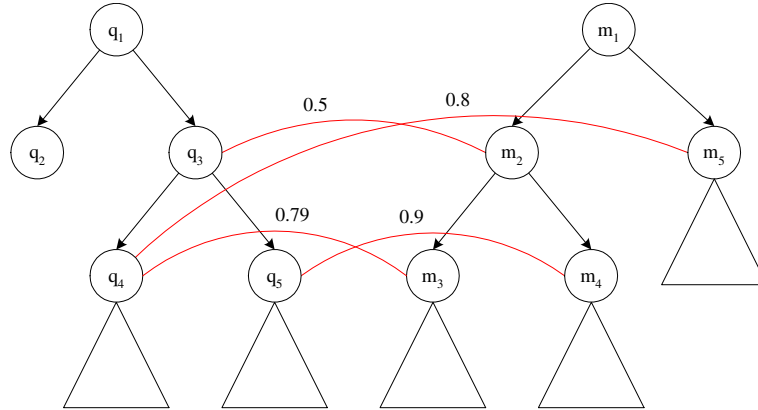


Fig. 8. Preserving sibling relationships by propagating the information from the previous best match. The matching pair (q_4, m_3) is chosen over the slightly better match (q_4, m_5) , because it results in two siblings in the query being matched to two siblings in the model.

This set of node correspondences maximizes the sum of node similarities, but does not enforce any hierarchical constraints other than the implicit ones encoded in $(\|\chi(q) - \chi(m)\|)$. Thus, instead of using all the node correspondences, we take a greedy approach and assume that only the first one is correct, and remove the subgraphs rooted at the selected matched pair of nodes. We now have two smaller problems of graph matching, one for the pair of removed subgraphs, and another for the two remainders of the original graphs. Both subproblems can, in turn, be solved by a recursive call of the above algorithm. The complexity of such a recursive algorithm is $O(n^3)$.

It turns out that splitting subgraphs at nodes with high confidence of being a good correspondence is not a strong enough constraint to guarantee that all the hierarchical relations are satisfied. Consider, for example, the graphs in Fig. 7. After the first iteration of the matching algorithm, nodes (q_5, m_5) will be matched since their similarity is the highest one in \mathcal{Q}_1 . In the next iteration, the subgraph rooted at q_5 , \mathcal{Q}^* , and the subgraph rooted at m_5 , \mathcal{M}^* , as well as their corresponding complement graphs \mathcal{Q}^c and \mathcal{M}^c , will be recursively evaluated.⁵ When matching \mathcal{Q}^c against \mathcal{M}^c , the best node correspondence according to the outlined algorithm will be (q_4, m_3) . It is easy to see that this match violates the hierarchical constraints among nodes because the siblings q_4 and q_5 are mapped to m_3 and m_5 , respectively, with m_3 a parent of m_5 .

Another constraint that arises in several domains is that of preserving sibling relationships. Note that this constraint is not, strictly speaking, a hierarchical constraint, since there are no hierarchical dependencies among sibling nodes. While it may be tempting to enforce this constraint when matching, there is a possibility that a sibling relation is genuinely broken by an occluder. In such a case, we may not want to enforce this constraint or else we will be unable to find meaningful matching subgraphs. A compromise solution would be to penalize the matches that break a sib-

ling relationship so as to favor those that provide a good set of correspondences while maintaining these relationships intact.

Fig. 8 illustrates the problem. Assuming that (q_5, m_4) has just been added to the solution set, the next best correspondence is (q_4, m_5) , which violates a sibling constraint. To avoid this, we can propagate the information provided by the previous best match (q_5, m_4) . This information is used to favor q_4 's sibling, so that (q_4, m_3) can be chosen instead. Since we do not want to become too sensitive to noise in the graph, we shall consider preserving the sibling-or-sibling-descendant relationships instead of the stricter sibling relationship. We will refer to this asymmetric relation between nodes as the SSD relation.⁶ Note that due to the asymmetry of the relation, the desired propagation of information will occur only when the algorithm proceeds in a top-down fashion. In the next section, we will see how to promote a top-down node matching.

Before continuing, let us define the rather intuitive node relationships that we will be working with. Let $\mathcal{G}(V, E)$ be a DAG and let u, v be two nodes in V . We say that u is a parent of v if there is an edge from u to v . Furthermore, let u be the ancestor of v if and only if there is a path from u to v . Similarly, let u be a SSD of v if and only if there exists a parent of v that is also an ancestor of u .

The relations defined above will allow us to express the desired constraints. However, we first need to determine how to make this information explicit, for it is not immediately available from the adjacency matrices of the graphs. A simple method is to compute the transitive closure graphs of our graphs. The transitive closure of a directed graph $\mathcal{G} = (V, E)$ is a directed graph $\mathcal{S} = (V, F)$, with (v, w) in F if and only if there is a path from v to w in \mathcal{G} . The transitive closure can be computed in linear time in the size of the graph $O(|V||E|)$ [19].

⁵ In the recursive call for \mathcal{Q}^* and \mathcal{M}^* , nodes q_5 and m_5 will be in the solution set, and so they will not be evaluated again.

⁶ Note that while the sibling relationship is symmetric, the SSD relationship is not, i.e., if u is the ‘‘nephew’’ of v , then $\text{SSD}(u, v)$ is true, but $\text{SSD}(v, u)$ is false.

It is easy to see, from the above definition, that the transitive closure of a graph is nothing else than the ancestor relation. Computing the SSD relation, on the contrary, requires a bit of extra work. Let $\mathbf{A}_{\mathfrak{G}}$ be the adjacency matrix of the DAG, $\mathfrak{G}(V, E)$, and let $\mathbf{T}_{\mathfrak{G}}$ be the adjacency matrix of the transitive closure graph of \mathfrak{G} . By means of these two matrices, we can now compute the non-symmetric SSD relation by defining $\mathbf{S}_{\mathfrak{G}}$ as the $|V| \times |V|$ matrix, where

$$\mathbf{S}_{\mathfrak{G}}(u, v) = \begin{cases} 1 & \text{if } \exists_{w \in V} \{ \mathbf{A}_{\mathfrak{G}}(w, v) = 1 \& \mathbf{T}_{\mathfrak{G}}(w, u) = 1 \}, \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

Armed with our new matrices $\mathbf{T}_{\mathfrak{Q}}$, $\mathbf{T}_{\mathfrak{M}}$, $\mathbf{S}_{\mathfrak{Q}}$, and $\mathbf{S}_{\mathfrak{M}}$, we can update the similarity matrix, \mathbf{W} , at each iteration of the algorithm, so as to preserve the ancestor relations and to discourage breaking SSD relations. At the first iteration, $n = 0$, we start with $\mathbf{W}^0 = \mathbf{W}$. Next, let (q', m') be the best node correspondence selected at the n th iteration of the algorithm, for $n \geq 0$. The new weights for each entry $\mathbf{W}_{q,m}^{n+1}$ of the similarity matrix, which will be used as edge weights in the bipartite graph at iteration $n + 1$, are updated according to:

$$\mathbf{W}_{q,m}^{n+1} = \begin{cases} 0 & \text{if } \mathbf{T}_{\mathfrak{Q}}(q, q') \neq \mathbf{T}_{\mathfrak{M}}(m, m'), \\ \beta \mathbf{W}_{q,m}^n & \text{else if } \mathbf{S}_{\mathfrak{Q}}(q, q') \neq \mathbf{S}_{\mathfrak{M}}(m, m'), \\ \mathbf{W}_{q,m}^n & \text{otherwise;} \end{cases} \quad (8)$$

where $0 \leq \beta \leq 1$ is a term that penalizes a pair of sibling nodes in the query being matched to a pair of non-sibling nodes in the model. It is sufficient to apply a small penalty to these cases, since the goal is simply to favor siblings over non-siblings when the similarities of the others are comparable to that of the siblings.

It is clear that when q' and m' are the roots of the subgraphs to match, the ancestor and SSD relations will be true for all the nodes in the DAG. Thus, in practice, when matching the q' -rooted and m' -rooted DAGs, we can avoid evaluating the conditions above. In addition, we know that only a few weights will change as the result of new node correspondences, and so we only need to update those entries of the matrix. This can be done efficiently by designing a data structure that simplifies the access to the weights that are to be updated. Alternatively, the update step can also be efficiently implemented with matrices by noticing that the column of $\mathbf{A}_{\mathfrak{G}}$ corresponding to node u tells us all the parents of u , while the row of $\mathbf{T}_{\mathfrak{G}}$ corresponding to node v give us all the descendants of v . Thus, given a node pair (q', m') , the positive entries in the rows of $\mathbf{T}_{\mathfrak{Q}}$ and $\mathbf{T}_{\mathfrak{M}}$ corresponding to the non-zero entries in columns q' and m' of $\mathbf{A}_{\mathfrak{Q}}$ and $\mathbf{A}_{\mathfrak{M}}$, respectively, coincide with the only the entries of $\mathbf{W}_{q,m}^n$ that need to be updated at each iteration of the algorithm.

A careful look at the algorithm as it has been stated so far will reveal that, in general, the first node correspondences found will be those among lower-level nodes in the hierarchy. We can expect this bottom-up behavior of the algorithm because the lower-level nodes carry less

structural information and so their weight will be less affected by the structural difference of the graphs rooted at them. Therefore, nodes at the bottom of the hierarchy will tend to have high similarity values and consequently, they will be chosen to split the graphs, creating small DAG's with few constraints on the nodes.

A solution to this problem is to redefine the way we choose the best edge from the bipartite matching. Instead of simply choosing the edge with greatest weight, we will also consider the order⁷ of the DAG rooted at the matched nodes to select the pair of nodes that have a large similarity weight and are also roots of large subgraphs. We define the mass, $m(v)$, of node v as the order, $n(T)$, of the DAG rooted at v . For a given graph $\mathfrak{G}(V, E)$, the $|V|$ -dimensional mass vector, $\mathbf{M}_{\mathfrak{G}}$, in which each of its dimensions is the mass $m(v)$ of a distinct $v \in V$, can be computed from the transitive closure matrix, $\mathbf{T}_{\mathfrak{G}}$, of the graph by $\mathbf{M}_{\mathfrak{G}} = \mathbf{T}_{\mathfrak{G}} \times \bar{\mathbf{1}}$, where $\bar{\mathbf{1}}$ is the $|V|$ -dimensional vector whose elements are all equal to 1. Thus, $\mathbf{M}_{\mathfrak{G}}$ is a vector in which each element $\mathbf{M}_{\mathfrak{G}}(v)$, for $v \in V$, is the number of nodes in the DAG rooted at v .

Unfortunately, the mass does not give us enough information about the depth of the subgraph rooted at a node since, for example, the path of n nodes has the same mass as the star of n nodes. A better idea is to consider the cumulative mass, \hat{m} . Let $\hat{m}(v)$ be defined as the sum of all the masses of the nodes of the DAG rooted at v . Thus, the cumulative mass vector will be given by $\hat{\mathbf{M}}_{\mathfrak{G}} = \mathbf{T}_{\mathfrak{G}} \times \mathbf{M}_{\mathfrak{G}}$, which can also be written as $\hat{\mathbf{M}}_{\mathfrak{G}} = \mathbf{T}_{\mathfrak{G}}^2 \times \bar{\mathbf{1}}$. This vector can then be used to obtain a relative measure of how tall and wide the rooted subgraphs are with respect to the graph they belong to, by simply normalizing the masses. Let $\tilde{\mathbf{M}}_{\mathfrak{G}}$ be the normalized cumulative mass vector given by

$$\tilde{\mathbf{M}}_{\mathfrak{G}} = \frac{\hat{\mathbf{M}}_{\mathfrak{G}}}{\max_{v \in V} \{ \hat{\mathbf{M}}_{\mathfrak{G}}(v) \}}, \quad (9)$$

where the normalizing factor will correspond to the cumulative mass of the node whose in-degree is zero—a root—and has the greatest cumulative mass in \mathfrak{G} .

The cumulative mass is exactly the piece of information we need, since it should be easy to see that for all the trees with n nodes, the star is the one with smallest cumulative mass, while the path is the one with the greatest. Hence, the cumulative mass, \hat{m} , for the root of a tree of order n satisfies $2n - 1 \leq \hat{m} \leq \frac{1}{2}n(n + 1)$. This measure is a good indicator of how deep and wide a subtree is, and so provides a means to find a compromise between the node similarities and their positions in the graph.

We can then promote a top-down behavior in the algorithm by selecting the match $(q, m)^+$ from the list, \mathfrak{Q} , returned by each MCMW bipartite matching, with the maximum convex sum of the similarity and the relative mass of the matched nodes,

⁷ Here we follow the convention in the graph theory literature that considers the *order* of a graph to be the number of nodes in the graph, and the *size* of the graph to be the number of edges in the graph.

$$(q, m)^+ = \operatorname{argmax}_{(q,m) \in \Omega} \{ \gamma \mathbf{W}_{q,m} + (1 - \gamma) \max(\tilde{\mathbf{M}}_{\Omega}(q), \tilde{\mathbf{M}}_{\mathfrak{M}}(m)) \}, \quad (10)$$

where $0 \leq \gamma \leq 1$ is a real value that controls the influence of the relative cumulative mass in selecting the best match. Since we want to promote a top-down association of nodes without distorting the actual node similarities, we suggest γ to be in the interval $[0.7, 0.9]$. In Fig. 9, we compare the sequences of graph splits using different values for γ . When $\gamma = 1$, we obtain the original equation in [47] that, as can be seen in the figure, tends to produce a bottom-up behavior of the algorithm.

Given the set of node correspondences between two graphs, the final step is to compute an overall measure of graph similarity. The similarity of the query graph to the model graph is given by

$$\sigma_{\phi}(\Omega, \mathfrak{M}) = \frac{(n_{\Omega} + n_{\mathfrak{M}}) \sum_{(q,m)^+ \in \phi} \mathbf{W}_{q,m}}{2n_{\Omega}n_{\mathfrak{M}}}, \quad (11)$$

where n_{Ω} and $n_{\mathfrak{M}}$ are the orders of the query graph and the model graph, respectively.

The graph similarity is given by a weighted average of the number of matched nodes in the query and in the model, where the weights are given by the node similarity of each matched node pair. If all the query nodes are matched with similarity 1, i.e., their attributes are identical, we have $\sum_{(q,m)^+ \in \phi} \mathbf{W}_{q,m} = n_{\Omega}$, and so $\sigma_{\phi}(\Omega, \mathfrak{M}) = \frac{1}{2} \left(\frac{n_{\Omega}}{n_{\mathfrak{M}}} + 1 \right)$. Since all query nodes have been matched, we know that $n_{\mathfrak{M}} \geq n_{\Omega}$, and so $\sigma_{\phi}(\Omega, \mathfrak{M})$ will be one when all the model nodes are mapped, and less than one otherwise. Therefore, the graph similarity is proportional to the quality of each pair of node correspondences, and inversely proportional to the number of unmatched nodes, both in the query and in the model. Hence, a model that contains the query as a relatively small subgraph is not as good a match as

a model for which most of nodes match those of the query graph, and vice versa.

Finally, it should be noted that the relative weighting of 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. Thus, if portions of the model were more geometrically constrained (e.g., articulation was prohibited), those model nodes could have a higher weighting on their geometric similarity component. Similarly, portions of the model that were less constrained could have a higher weighting on the topological similarity component. This is a very powerful feature of the algorithm, allowing the incorporation of local model constraints into the matching algorithm.

The final algorithm is shown in Fig. 10. The first step of the algorithm is to compute a node similarity matrix, the transitive closure matrices, the sibling matrices, and the node TSVs for both graphs. Assuming a linear algorithm for the pairwise node similarities, the former matrix can be computed in $O(n^3)$. The other matrices can, in turn, be obtained in linear time and in quadratic time, respectively. At each iteration of the algorithm, we have to compute a MCMW bipartite matching, sort its output, and update the similarity matrix. The complexity at each step is then determined by that of the bipartite matching algorithm, $O(|V||E|)$, since it is the most complex operation of the three. The number of iterations is bounded by $\min(n_{\Omega}, n_{\mathfrak{M}})$, and so the overall complexity of the algorithm is $O(n^3)$. Hence, we have provided the algorithm with important properties for the matching process while maintaining its original complexity. An example of the blob correspondence computed over two hand exemplars is shown in Fig. 11.

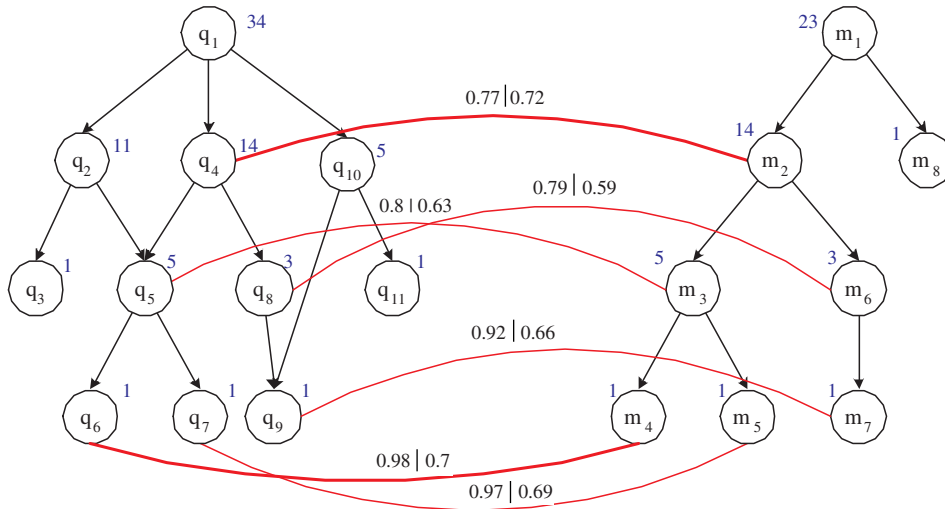


Fig. 9. An example in which $\gamma < 1$ can promote a top-down behavior in the algorithm. The cumulative mass of each node is shown in blue. Edge weights are computed according to Eq. 10, for $\gamma = 1$ and $\gamma = 0.7$. For the given set of node similarities and $\gamma = 1$, the best node correspondence at this iteration is the pair of leaves (q_6, m_4) , whereas for $\gamma = 0.7$, the best node correspondence is the pair of non-terminal nodes (q_4, m_2) . (For interpretation of the references to colours in this figure legend, the reader is referred to the web version of this paper.)

```

procedure isomorphism( $\Omega, \mathfrak{M}$ )
   $\Phi(\Omega, \mathfrak{M}) \leftarrow \emptyset$ ; solution set
  compute the  $n_\Omega \times n_{\mathfrak{M}}$  weight matrix  $\mathbf{W}^0$  from Eq. 6
   $\mathbf{T}_\Omega =$  compute transitive closure matrix from  $\mathbf{A}_\Omega$ 
   $\mathbf{T}_{\mathfrak{M}} =$  compute transitive closure matrix from  $\mathbf{A}_{\mathfrak{M}}$ 
   $\mathbf{S}_\Omega =$  compute SSD matrix from  $\mathbf{A}_\Omega$  and  $\mathbf{T}_\Omega$ 
   $\mathbf{S}_{\mathfrak{M}} =$  compute SSD matrix from  $\mathbf{A}_{\mathfrak{M}}$  and  $\mathbf{T}_{\mathfrak{M}}$ 
  compute the TSV of each nonterminal node in  $\Omega$  and  $\mathfrak{M}$ 
  unmark all nodes in  $\Omega$  and in  $\mathfrak{M}$ 
  call match(root( $\Omega$ ), root( $\mathfrak{M}$ ))
  return( $\sigma_\Phi(\Omega, \mathfrak{M})$ )
end

procedure match( $u, v$ )
  do
    {
      let  $\Omega_u \leftarrow u$  rooted unmarked subgraph of  $\Omega$ 
      let  $\mathfrak{M}_v \leftarrow v$  rooted unmarked subgraph of  $\mathfrak{M}$ 
       $\mathcal{L} \leftarrow$  max cardinality, max weight bipartite matching between
        unmarked nodes in  $\mathcal{G}(V_{\Omega_u}, V_{\mathfrak{M}_v})$  with weights from  $\mathbf{W}^{n+1}$  (see[17])
      ( $u', v'$ )  $\leftarrow$  choose max weight pair in  $\mathcal{L}$  from Eq. 10
       $\Phi(\Omega, \mathfrak{M}) \leftarrow \Phi(\Omega, \mathfrak{M}) \cup \{(u', v')\}$ 
      update the similarity matrix  $\mathbf{W}^{n+1}$  according to Eq. 8
      mark  $u'$ 
      mark  $v'$ 
      call match( $u', v'$ )
      call match( $u, v$ )
    }
  while ( $\Omega_u \neq \emptyset$  and  $\mathfrak{M}_v \neq \emptyset$ )

```

Fig. 10. Algorithm for matching two directed acyclic graphs.

5. Experiments

We evaluate our framework on the domain of view-based 3-D object recognition where the objective is to choose the right object (identification) for a particular query view and also to determine its correct pose (pose estimation). To provide a comprehensive evaluation, we used two popular image libraries; the Columbia University COIL-20 (20 objects, 72 views per object) [34] and the ETH Zurich ETH-80 (8 categories, 10 exemplars per category, 41 views per exemplar) [26]. Sample views of objects from these two libraries are shown in Fig. 12. Note that in the ETH-80 library, some categories have very similar shape, differing only in their appearance. Thus, the horse, dog, and cow categories were collapsed to form a 4-legged animal category, the apple and tomato categories were collapsed to form a spherical fruit category, and the two car instance categories were collapsed to form a single car category.

We applied the following leave-one-out procedure to each database to evaluate the proposed framework. Specifically, we initially removed the first entry from the database, used it as a query, and computed its similarity with each of the remaining views in the database. We then returned the query back to the database and repeated the same process for each of the other database entries. This process results in an $n \times n$ similarity matrix, where the entry (i, j) indicates how similar views i and j are. For a particular query, we classify its identification as correct if the maximum similarity is obtained with a view which belongs to the same object as the query. Consequently, pose estima-

tion is correct if view i of object j , $v_{i,j}$ matches most closely with $v_{n,j}$, where n is one of i 's neighboring views.

Our overall recognition rates for COIL-20 and ETH-80 datasets are 93.5% and 97.1%, respectively. We show a part of the matching results in Table 1. Upon investigation as to why the COIL-20 dataset yields poorer performance, we found that most of the mismatches were between three different car objects: column three of the first row, column one of the second row, and column four of the fourth row, as shown in the left of Fig. 12. Despite being different objects with different appearance, their coarse shape structure is similar and their blob graphs are indeed similar. If we group these three exemplars into the same category and count these matches as correct, our recognition rate rises to 96.5%. Our recognition framework is clearly suited to coarse shape categorization as opposed to exemplar matching.

For pose estimation, we observe that in all but 9.8% and 14.6% of the COIL-20 and ETH-80 experiments, respectively, the closest match selected by our algorithm was a neighboring view. Note that if the closest matching view was not an immediate neighbor drawn from the *same exemplar*, the match was deemed incorrect, despite the fact that the matching view might be a neighboring view of a different exemplar from the same category. This is perhaps overly harsh, as reflected by the 14.6% results, but view alignment between exemplars belonging to the same category was not provided. These results can be considered worst case for several additional reasons. Given the high similarity among neighboring views, it could be argued that

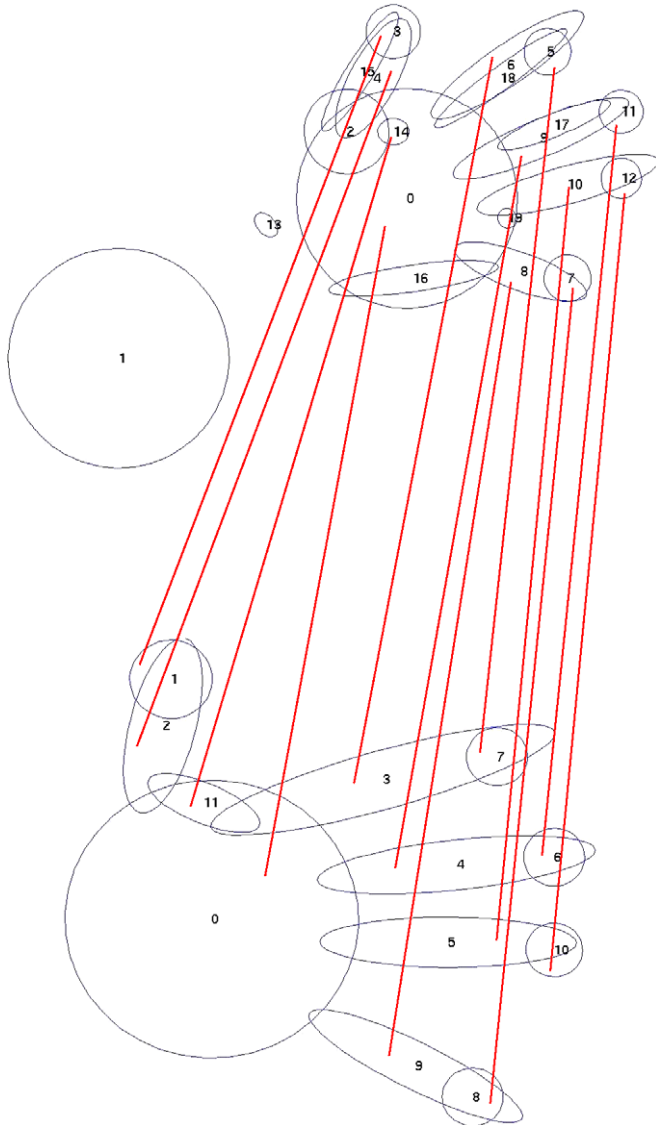


Fig. 11. Example correspondence computed between two blob graphs.

our pose estimation criterion is overly harsh, and that perhaps a measure of “viewpoint distance”, i.e., “how many views away was the closest match” would be less severe.

In any case, we anticipate that with fewer samples per object, neighboring views would be more dissimilar, and our matching results would improve. More importantly, many of the objects are rotationally symmetric, and if a query has an identical view elsewhere in the dataset, that view might be chosen (with equal similarity) and scored as an error.

To demonstrate the framework’s robustness, we performed five perturbation experiments on both datasets. The experiments were identical to the experiments above, except that we randomly chose a node, v , in the query graph, if the number of nodes in the directed acyclic subgraph rooted at v was less than 10% of the number of nodes in the original graph, we deleted the rooted subgraph from the query. We then repeated the same process for maximum ratios of 20%, 30%, 40%, and 50%. The results are shown in Table 2, and reveal that the error rate increases gracefully as a function of increased perturbation. Although not a true occlusion experiment, which would require that we replace the removed subgraph with an occluder subgraph, these results demonstrate the framework’s ability to match local structure, a property essential for handling occlusion.

The above experiments clearly establish the efficacy of the proposed generic object recognition framework. But one might wonder how other graph-matching frameworks perform on the same features, or how non-graph-matching frameworks might perform on the same features. We begin by comparing our proposed recognition framework to a different graph-matching framework operating on the same features. Specifically, we compare our framework to its predecessor, a state-of-the-art inexact graph-matching algorithm first demonstrated on the problem of shock graph matching [50]. Unlike its predecessor, our new algorithm enforces a number of hierarchical and sibling constraints that result in a strengthened graph-matching framework. Table 3 compares the two graph matchers (“current” and “Siddiqi et al.”), in terms of both recognition and pose estimation, on the COIL-20 and ETH-80 datasets. The improvement due to the new constraint checking is dramatic for both tasks on both datasets, clearly demonstrating the improvement in graph matching.

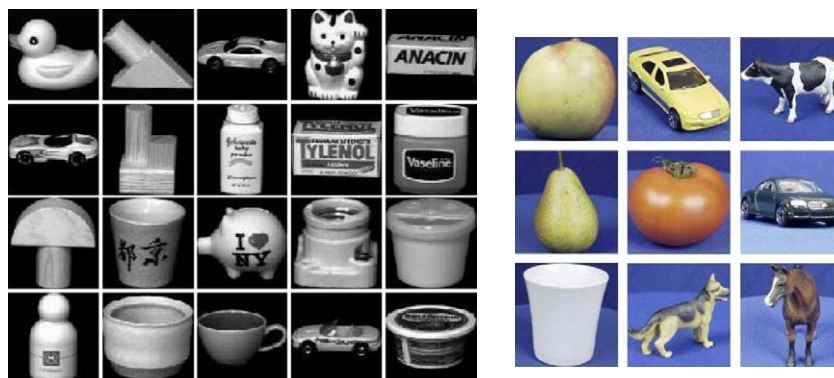


Fig. 12. Views of sample objects from the Columbia University Image Library (COIL-20) and the ETH Zurich (ETH-80) Image Set.

Table 1
Top matched models are sorted by the similarity to the query







































































































































































































































Query	Top 9 Matched Objects								
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									
									

Table 2
Recognition rate as a function of increasing perturbation in the form of missing data

Perturbation	10%	20%	30%	40%	50%
Recognition rate (COIL-20)	91.2%	89.5%	87.3%	83.7%	78.6%
Recognition rate (ETH-80)	94.2%	91.5%	89.3%	84.7%	82.6%

Percentages indicate how much of the query graph was removed prior to matching.

Table 3
Comparison of proposed graph-matching framework to a competing graph-matching framework, proposed by Siddiqi et al. [50]

Algorithm	COIL-20		ETH-80	
	Rec. (%)	Pose est. (%)	Rec. (%)	Pose est. (%)
Current	93.5	90.2	97.1	85.4
Siddiqi et al.	79.9	58.4	87.4	77.5

Significant improvement is seen on both recognition and pose estimation tasks for both the COIL-20 and ETH-80 databases.

Next, we compare our recognition framework to a non-graph-based matching algorithm. Specifically, we compare our framework to an algorithm that retains the

same features, but discards the graph structure. Blobs and ridges are matched using the Earth Mover’s Distance algorithm [8], constrained to ensure a one-to-one match-

Table 4

Comparison of proposed graph-matching framework to a competing non-graph-matching framework, based on the Earth Mover's Distance [8]

Algorithm	COIL-20		ETH-80	
	Rec. (%)	Pose est. (%)	Rec. (%)	Pose est. (%)
Current	93.5	90.2	97.1	85.4
Constrained EMD	94.1	67.7	80.1	58.1

Significant improvement is seen on three of four experiments.

ing of features (to be consistent with our proposed framework). The results of the comparison are shown in Table 4. With the exception of the COIL-20 recognition experiment, the improvement of our proposed algorithm is significant, indicating the representational power of capturing the topology of the parts. One possible explanation for the less dramatic improvement on the COIL database might be that part articulation among the exemplars in a category is minimal; in fact, few of the objects have very definitive part structure at all. Whereas the structure of our blob/ridge graph is invariant to articulation of its parts (although such transformations can be penalized in the contextual node similarity function), a purely geometric matching scheme cannot accommodate such transformations.

6. Limitations

Both the representation and matching components of our integrated framework have limitations. Since it is based on image gradients, the blob and ridge decomposition does not perform well in the presence of textured surfaces, and spurious and missing blobs may result. Although the matching algorithm can accommodate both noise and occlusion, it does rely on there being a sufficient number of one-to-one correspondences to discriminate the correct model from other models. If blobs are highly over- or under-segmented, matching may fail as too few one-to-one correspondences may exist.

7. Conclusions

Matching two images whose similarity exists at the coarse shape level is critical to object categorization. Blobs and ridges provide an ideal multiscale part vocabulary for coarse shape modeling which, when combined with an array of geometric relations in the form of a graph, yield a powerful categorical shape representation, providing a powerful, hierarchical characterization of an object's coarse shape. Our inexact graph-matching framework exploits both the topological as well as the geometrical relations in a directed acyclic graph to yield an efficient algorithm for coarse-to-fine shape matching. We have demonstrated the generality of the framework by applying it to two different domains (without domain-specific tuning), with very encouraging results in each domain.

Acknowledgments

The authors gratefully acknowledge the support of the ONR, NSERC, NSF, CITO, NSERC-IRIS, CITO, and MD Robotics.

References

- [1] G. Agin, T. Binford, Computer description of curved objects, *IEEE Trans. Comput.* C-25 (4) (1976) 439–449.
- [2] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, *IEEE Trans. Pattern Anal. Mach. Intell.* 24 (4) (2002) 509–522.
- [3] T. Binford, Visual perception by computer, in: *Proceedings of the IEEE Conference on Systems and Control*, Miami, FL, 1971.
- [4] K.L. Boyer, A.C. Kak, Structural stereopsis for 3-D vision, *IEEE Trans. Pattern Anal. Mach. Intell.* 10 (2) (1988) 144–166.
- [5] L. Bretzner, T. Lindeberg, Qualitative multi-scale feature hierarchies for object tracking, *J. Vis. Commun. Image Representation* 11 (2000) 115–129.
- [6] R. Brooks, Model-based 3-D interpretations of 2-D images, *IEEE Trans. Pattern Anal. Mach. Intell.* 5 (2) (1983) 140–150.
- [7] P.J. Burt, Attention mechanisms for vision in a dynamic world, in: *Proceedings of the International Conference on Pattern Recognition*, vol.1, The Hague, The Netherlands, 1988, pp. 977–987.
- [8] S.D. Cohen, L.J. Guibas, The earth mover's distance under transformation sets, in: *Proceedings of the Seventh International Conference on Computer Vision*, Kerkyra, Greece, 1999, pp. 1076–1083.
- [9] J. Crowley, A. Parker, A representation for shape based on peaks and ridges in the difference of low-pass transform, *IEEE Trans. Pattern Anal. Mach. Intell.* 6 (2) (1984) 156–169.
- [10] J.L. Crowley, A multiresolution representation for shape, in: *Azriel Rosenfeld (Ed.), Multiresolution Image Processing and Analysis*, Springer Verlag, Berlin, 1984, pp. 169–189.
- [11] J.L. Crowley, Arthur C. Sanderson, Multiple resolution representation and probabilistic matching of 2-D gray-scale shape, *IEEE Trans. Pattern Anal. Mach. Intell.* 9 (1) (1987) 113–121.
- [12] M.F. Demirci, A. Shokoufandeh, S. Dickinson, Y. Keselman, L. Bretzner, Many-to-many matching of scale-space feature hierarchies using metric embedding, in: *Proceedings of the Fourth International Conference on Scale Space Methods in Computer Vision*, 2003, pp. 17–32.
- [13] M.F. Demirci, A. Shokoufandeh, S. Dickinson, Y. Keselman, L. Bretzner, Many-to-many feature matching using spherical coding of directed graphs, in: *Proceedings of the Eighth European Conference on Computer Vision*, 2004, pp. 332–335.
- [14] S. Dickinson, M. Pelillo, R. Zabih, Special section on graph algorithms and computer vision, *IEEE Trans. Pattern Anal. Mach. Intell.* 23 (10) (2001).
- [15] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples, in: *Workshop on Generative-Model Based Vision*, 2004.
- [16] R. Fergus, P. Perona, A. Zisserman, Object class recognition by unsupervised scale-invariant learning, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2003, pp. 264–271.

- [17] H.N. Gabow, R.E. Tarjan, Faster scaling algorithms for general graph matching problems, *J. ACM* 38 (1991) 815–853.
- [18] Y. Gdalyahu, D. Weinshall, Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes, *IEEE Trans. Pattern Anal. Mach. Intell.* 21 (12) (1999) 1313–1328.
- [19] A. Goralcikova, V. Konbek, A reduct and closure algorithm for graphs, *Math. Found. Comput. Sci., Lect. Notes Comput. Sci.* 74 (1979) 301–307.
- [20] M. Jägersand, Saliency maps and attention selection in scale and spatial coordinates: an information theoretic approach, in: *Proceedings of the Fifth International Conference on Computer Vision*, Boston, MA, 1995, pp. 195–202.
- [21] K. Grauman, T. Darrell, Fast contour matching using approximate earth mover's distance, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society, 2004, pp. 220–227.
- [22] Y. Keselman, S. Dickinson, Generic model abstraction from examples, *IEEE PAMI* 27 (7) (2005).
- [23] Y. Keselman, A. Shokoufandeh, M.F. Demirci, S. Dickinson, Many-to-many graph matching via metric embedding, in: *Proceedings of the IEEE Computer Vision and Pattern Recognition*, Madison, WI, 2003.
- [24] I. Laptev, T. Lindeberg, Tracking of multi-state hand models using particle filtering and a hierarchy of multi-scale image features, in: *Proceedings of the Scale-Space'01*, Vancouver, Canada, 2001.
- [25] S. Lazebnik, C. Schmid, J. Ponce, Learning local affine-invariant part models for object class recognition, in: *Workshop on Learning*, Snowbird, Utah, 2004.
- [26] B. Leibe, B. Schiele, Analyzing appearance and contour based methods for object categorization, in: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, WI, 2003.
- [27] B. Leibe, B. Schiele, Analyzing appearance and contour based methods for object categorization, in: *CVPR vol. 2*, 2003, pp. 409–415.
- [28] T. Lindeberg, Edge detection and ridge detection with automatic scale selection, *Int. J. Comput. Vis.* 30 (1998) 117–154.
- [29] T. Lindeberg, Feature detection with automatic scale selection, *Int. J. Comput. Vis.* 30 (1998) 77–116.
- [30] T. Lindeberg, 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.* 11 (3) (1993) 283–318.
- [31] S. Mallat, Wen Liang Hwang, Singularity detection and processing with wavelets, *IEEE Trans. Inf. Theory* 38 (2) (1992) 617–643.
- [32] D. Marr, H. Nishihara, Representation and recognition of the spatial organization of three-dimensional shapes, *R. Soc. Lond. B* 200 (1978) 269–294.
- [33] I. Marsic, Data-Driven Shifts of Attention in Wavelet Scale Space. Technical Report CAIP-TR-166, CAIP Center, Rutgers University, Piscataway, NJ, 1993.
- [34] H. Murase, S. Nayar, Visual learning and recognition of 3-D objects from appearance, *Int. J. Comput. Vis.* 14 (1995) 5–24.
- [35] R. Nevatia, T. Binford, Description and recognition of curved objects, *Artif. Intell.* 8 (1977) 77–98.
- [36] B. Olshausen, C. Anderson, D. Van Essen, A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information, *J. Neurosci.* 13 (11) (1992) 4700–4719.
- [37] M. Pelillo, Matching free trees, maximal cliques, and dynamic monotone game dynamics, *IEEE Trans. Pattern Anal. Mach. Intell.* 24 (11) (2002) 1535–1541.
- [38] M. Pelillo, K. Siddiqi, S. Zucker, Matching hierarchical structures using association graphs, *IEEE Trans. Pattern Anal. Mach. Intell.* 21 (11) (1999) 1105–1120.
- [39] M. Pelillo, K. Siddiqi, S.W. Zucker, Many-to-many matching of attributed trees using association graphs and game dynamics, in: *Proceedings of the International Workshop on Visual Form (IWVF)*, 2001, pp. 583–593.
- [40] R.P.N. Rao, G.J. Zelinsky, M.M. Hayhoe, Dana H. Ballard, Modeling saccadic targeting in visual search, in: D. Touretzky, M. Mozer, M. Hasselmo (Eds.), *Advances in Neural Information Processing Systems*, 8, MIT Press, Cambridge, MA, 1996, pp. 830–836.
- [41] S.W. Reyner, An analysis of a good algorithm for the subtree problem, *SIAM J. Comput.* 6 (1977) 730–732.
- [42] Y. Rubner, C. Tomasi, L.J. Guibas, The earth mover's distance as a metric for image retrieval, *Int. J. Comput. Vis.* 40 (2) (2000) 99–121.
- [43] S. Sclaroff, A.P. Pentland, Modal matching for correspondence and recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* 17 (6) (1995) 545–561.
- [44] T.B. Sebastian, P.N. Klein, B.B. Kimia, Recognition of shapes by editing shock graphs, in: *IEEE International Conference on Computer Vision*, 2001, pp. 755–762.
- [45] K. Sengupta, K. Boyer, Modelbase partitioning using property matrix spectra, *Comput. Vis. Image Understanding* 70 (2) (1998) 177–196.
- [46] L. Shapiro, M. Brady, Feature-based correspondence: an eigenvector approach, *Image Vis. Comput.* 10 (5) (1992) 283–288.
- [47] A. Shokoufandeh, S. Dickinson, A unified framework for indexing and matching hierarchical shape structures, in: *Proceedings of the Fourth International Workshop on Visual Form*, Capri, Italy, 2001, pp. 28–30.
- [48] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, S. Zucker, Indexing hierarchical structures using graph spectra, *IEEE PAMI* 27 (7) (2005) 1125–1140.
- [49] A. Shokoufandeh, I. Marsic, S. Dickinson, View-based object recognition using saliency maps, *Image Vis. Comput.* 17 (5–6) (1999) 445–460.
- [50] K. Siddiqi, A. Shokoufandeh, S. Dickinson, S. Zucker, Shock graphs and shape matching, *Int. J. Comput. Vis.* 30 (1999) 1–24.
- [51] E.P. Simoncelli, W.T. Freeman, E.H. Adelson, D.J. Heeger, Shiftable multiscale transforms, *IEEE Trans. Inf. Theory* 38 (2) (1992) 587–607.
- [52] Barnabas Takaçs, Harry Wechsler, A dynamic and multiresolution model of visual attention and its application to facial landmark detection, *Comput. Vis. Image Understanding*. 70 (1) (1998) 63–73.
- [53] J. Triesch, C. von der Malsburg, Robust classification of hand postures against complex background, in: *Proceedings of the International Conference on Face and Gesture Recognition*, Killington, Vermont, 1996, pp. 70–175.
- [54] J. Tsotsos, An inhibitory beam for attentional selection, in: L. Harris, M. Jenkin (Eds.), *Spatial Vision in Humans and Robots*, Cambridge University Press, 1993.
- [55] R. Verma, Steven W. Reyner, An analysis of a good algorithm for the subtree problem, corrected, *SIAM J. Comput.* 18 (5) (1989) 906–908.
- [56] L. Wiskott, J. Fellous, N. Krüger, C. von der Malsburg, Face recognition by elastic bunch graph matching, *IEEE Trans. Pattern Anal. Mach. Intell.* 19 (7) (1997) 775–779.
- [57] A. Witkin, J. Tenenbaum, On the role of structure in vision, in: J. Beck, A. Rosenfeld (Eds.), *Human Machine Vision*, Academic Press, New York, 1983.