International Journal of Computer Vision 69(2), 203–222, 2006

DOI: 10.1007/s11263-006-6993-y

Object Recognition as Many-to-Many Feature Matching

M. FATIH DEMIRCI AND ALI SHOKOUFANDEH
Department of Computer Science, Drexel University, Philadelphia, PA 19104, USA

YAKOV KESELMAN
School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, IL 60604, USA

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

SVEN DICKINSON
Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G4

Received July 27, 2005; Revised December 20, 2005; Accepted December 28, 2005

First online version published in May, 2006

Abstract. Object recognition can be formulated as matching image features to model features. When recognition is exemplar-based, feature correspondence is one-to-one. However, segmentation errors, articulation, scale difference, and within-class deformation can yield image and model features which don't match one-to-one but rather many-to-many. Adopting a graph-based representation of a set of features, we present a matching algorithm that establishes many-to-many correspondences between the nodes of two noisy, vertex-labeled weighted graphs. Our approach reduces the problem of many-to-many matching of weighted graphs to that of many-to-many matching of weighted point sets in a normed vector space. This is accomplished by embedding the initial weighted graphs into a normed vector space with low distortion using a novel embedding technique based on a spherical encoding of graph structure. Many-to-many vector correspondences established by the Earth Mover's Distance framework are mapped back into many-to-many correspondences between graph nodes. Empirical evaluation of the algorithm on an extensive set of recognition trials, including a comparison with two competing graph matching approaches, demonstrates both the robustness and efficacy of the overall approach.

Keywords: graph matching, graph embedding, Earth Mover's Distance (EMD), object recognition

1. Introduction

The problem of object recognition is often formulated as that of matching configurations of image features to configurations of model features. Such configurations are often represented as vertex-labeled graphs, whose nodes represent image features (or their abstractions), and whose edges represent relations (or constraints) between the features. For scale-space structures, represented as directed graphs, relations can represent both parent/child relations as well as sibling relations. To match two graph representations (hierarchical

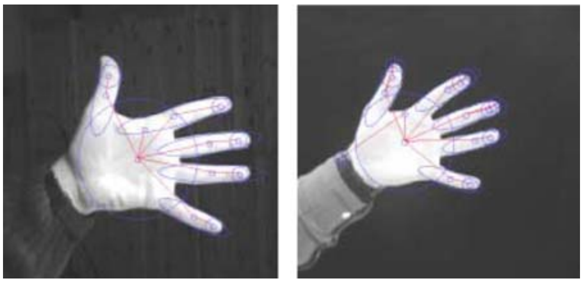

Figure 1. The need for many-to-many matching. In the two images, the two objects are similar, but the extracted features are not necessarily one-to-one. Specifically, the ends of the fingers in the left hand have been over-segmented in the right hand.

or otherwise) means to establish correspondences between their nodes. To evaluate the quality of a match, an overall distance measure is defined, whose value depends on both node and edge similarity.

Previous work on graph matching has typically focused on the problem of finding a one-to-one correspondence between the vertices of two graphs. However, the assumption of one-to-one correspondence is a very restrictive one, for it assumes that the primitive features (nodes) in the two graphs agree in their level of abstraction. Unfortunately, there are a variety of conditions that may lead to graphs that represent visually similar image feature configurations yet do not contain a single one-to-one node correspondence.

The limitations of the one-to-one assumption are illustrated in Fig. 1, in which an object is decomposed into a set of ridges and blobs extracted at appropriate scales (Shokoufandeh et al., 2002). The ridges and blobs map to nodes in a directed acyclic graph, with parent/child edges directed from coarser scale nodes to overlapping finer scale nodes. Although the two images clearly contain the same object, the decompositions are not identical. Specifically, the ends of the fingers in the right hand have been over-segmented with respect to the left hand. It is quite common that due to noise or segmentation errors, a single feature (node) in one graph can correspond to a collection of broken features (nodes) in another graph. Or, due to scale differences, a single, coarse-grained feature in one graph can correspond to a collection of fine-grained features in another graph. Hence, we seek not a one-to-one correspondence between image features (nodes), but rather a many-to-many correspondence. The space of possible many-to-many correspondences between two graphs is exponential. In the worst case, without restricting a subset of nodes to be connected, any subset of nodes in one graph can match any subset of nodes in the other. For a given graph, the number of partitions of the graph into subsets is the Bell number (Bell, 1934), which is exponential. Since any partition in one graph can match any partition in the other graph, the size of the space of possible correspondences is bounded by the product of the Bell numbers of the two graphs. We formulate our problem as finding the pair of partitions, one per graph, along with a mapping from the elements of one partition to the elements of the other, that optimizes a measure of similarity of the corresponding subsets defined by the mapping. Our approach searches for corresponding partitions that maintain coherent (i.e., connected) subsets while preserving the relational structure among the subsets.

In our approach, we will transform the graphs into an alternative domain in which approximating the many-to-many matching becomes tractable. We draw on recent low-distortion graph embedding techniques which embed the nodes in a graph into points in a low-dimensional geometric space. Although each point in this embedding space represents a node in the original graph, the edges in the original graph do not explicitly find their way to this new space. Instead, the points are positioned in the embedded space such that the Euclidean distance between pairs of points reflects the shortest-path distances between their corresponding nodes in the original graph.

Matching two graphs can now be formulated as the problem of matching their two embeddings, which we'll assume are roughly aligned in space. However, this has not solved our original problem, for any subset of points in one embedding may correspond to any set of points in the second. If we make the assumption that the weight of a graph edge reflects the probability that the two nodes the edge spans should be grouped as part of a many-to-many correspondence, then the embedding process ensures that nodes to be grouped get mapped to points in close geometric proximity. Furthermore, if we assume that a node's attributes can be mapped to a vector of "masses", then our many-to-many point matching problem can be formulated as finding corresponding subsets of points, such that the points are in close proximity and the subsets have similar mass.

Such a formulation is ideally suited to the Earth Mover's Distance (Cohen and Guibas, 1999) algorithm, whose solution computes the mass flows from one weighted point set to another that minimize the total

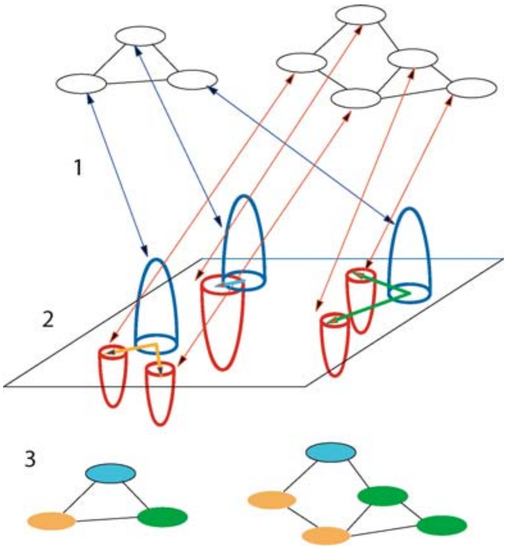

Figure 2. Overview of the approach: (1) two graphs to be matched (top) are embedded into the same normed space, with point mass (size of pile or hole) a function of a node's attributes; (2) the points are matched many-to-many using the EMD algorithm; and finally (3) the computed flows define explicit many-to-many node correspondences between the original graphs.

work. The computed flows, in turn, explicitly define the many-to-many node correspondences between the original graphs. The approach is illustrated in Fig. 2, in which two graphs are embedded into the same space. EMD is used to match the embedded point masses many-to-many, and the computed flows define a many-to-many correspondence between the two graphs.

Following a review of related work, we review the basics of low-distortion graph embedding in Section 3, leading to our first solution based on the graph embedding work of Matoušek (1999), described in Section 3.1. The method begins by transforming a graph into a metric tree which, in turn, is embedded into a normed vector space. Although effective, the approach suffers from a significant limitation. Namely, each graph is embedded into a vector space whose dimensionality is a property of the graph. Thus before two embeddings can be matched, a dimensionality reduction step (on the space of larger dimension) is required, which is both costly and prone to error. In Section 3.2, we present a novel embedding method based on a spherical coding algorithm. This efficient (linear-time) method embeds metric trees into vector spaces of prescribed dimensionality, precluding the need for a dimensionality reduction step. In Section 4, we address the problem of dealing with directed edges, allowing the framework to be applied to hierarchical graphs, such as scale space structures. We evaluate and compare both embedding functions on two different view-based recognition domains in Section 6. Moreover, we perform a direct head-to-head comparison of our approach with two competing graph matching algorithms: a one-to-one algorithm and a many-to-many algorithm. We close the paper with a discussion of limitations of the approach, in Section 7, and our conclusions, in Section 8.

2. Related Work

The problem of many-to-many graph matching has been studied most often in the context of edit-distance (see, e.g., Messmer and Bunke 1995; Liu and Geiger 1999; Myers et al. 2000; Sebastian et al. 2001). In such a setting, one seeks a minimal set of re-labelings, additions, deletions, merges, and splits of nodes and edges that transform one graph into another. However, the edit-distance approach has its drawbacks: (1) it is computationally expensive (polynomial-time algorithms are available only for trees); (2) the method does not deal well with occlusion and scene clutter, resulting in much effort spent in "editing out" extraneous graph structure; and (3) the cost of an editing operation often fails to reflect the underlying visual information (for example, the visual similarity of a contour and its corresponding broken fragments should not be penalized by the high cost of merging the many fragments).

Keselman and Dickinson (2005) explored the problem of many-to-many region matching in the context of model-based abstraction from images. Given a set of region adjacency graphs representing region segmented images of different exemplars belonging to a known class, they search the space of pairwise region groupings in each graph that lead to an abstract region adjacency graph which has isomorphic counterparts in the other exemplars' spaces. The resulting *lowest common abstraction*, or LCA, represents the most informative abstraction common to each exemplar. Although effective for supervised structure learning, the technique, in its current form, is not applicable to the problem of matching two graphs for which a common abstraction does not exist.

A spectral abstraction of hierarchical graph structure was originally proposed by Siddiqi et al. (1999) for the problem of matching hierarchical trees, and latter extended by Shokoufandeh et al. (2005) for the problem of structural indexing. Given a rooted directed graph, the eigenvalues of its subgraphs' adjacency matrices were used to derive a low-dimensional vector encoding of the graph structure rooted at that node. Not only could this vector be used for fast structural indexing, the vector formed the heart of a matching algorithm that found corresponding graphs (or subgraphs). Although the recursive algorithm could be used to flesh out explicit one-to-one node correspondences further down the hierarchies, one-to-one node correspondences at higher levels effectively defined a many-to-many matching between their underlying nodes. Belongie et al. (2002) used a similar idea to encode the qualitative shape occupancy characteristics of a neighborhood surrounding a point. In a bipartite matching framework, correspondences were formed between points with similar *shape contexts*, despite the fact that the neighborhoods could have differing numbers of points.

Carcassoni and Hancock (2003) decomposed the problem of matching two point sets of different cardinality into three related subproblems: point clustering, cluster matching, and matching points from matching clusters. These subproblems were jointly solved by an iterative method that combined the eigenvector technique for matching sets of the same cardinality with the EM approach for likelihood computation. The resulting hierarchical matching approach proved to be very robust to noise and occlusion. Gold and Rangarajan (1996) addressed the problem of partial matching of attributed sparse graphs by an EM-like graduated assignment approach that efficiently revised earlier mappings based on local constraints encoded by the graphs. While the final result is a one-to-one matching, the algorithm maintains an intermediate doubly-stochastic matrix that closely resembles a many-to-many matching matrix. The resulting framework was successfully applied to non-rigid matching of image feature sets.

In a novel generalization of Scott and Longuet-Higgins (1991), Kosinov and Caelli (2002) showed how inexact graph matching could be solved using the re-normalization of projections of vertices into the eigenspaces of graphs combined with a form of relational clustering. However, the framework cannot accommodate occlusion, directed graphs, or node attributes, and may yield high embedding distortion. Low-distortion embedding techniques haven proven to be useful in a number of graph algorithms, including clustering and, most recently, on-line algorithms. Gupta (1999) proposed a randomized procedure for embedding metric trees into a vector space of prescribed dimensions. However, if the goal is for two structurally similar graphs to yield embeddings that are easily aligned or matched, such a randomized procedure is problematic. Athitsos et al. (2004) developed an approach for learning a low-distortion, high-dimensional embedding as a combination of several one-dimensional (potentially high-distortion) embeddings. While the approach proved to be valuable for fast database retreival, its usefulness for matching is less clear, since one-dimensional embeddings chosen by the approach for two different objects may differ significantly. Indyk (2001) provides a comprehensive survey of recent advances and applications of low-distortion graph embedding. For recent results related to the properties of low-distortion tree embedding, see (Agarwala et al., 1999; Matoušek, 1999).

Grauman and Darrell (2004) developed an embedding procedure in support of image retrieval. Image feature distributions, like color histograms, do not provide a convenient mechanism for indexing into large image databases. In a two-step procedure, they first embed the feature distribution to a vector and then use the Locality Sensitive Hashing (LSH) algorithm of Gionis et al. (1999) to retrieve nearby candidates. The embedding method was designed so that the distance between two such embeddings mimics the Earth Movers Distance (EMD) between their respective feature distributions. However, the abstract nature of the embedding means that the explicit many-to-many correspondences between two feature sets cannot be recovered. Grauman and Darrell (2004) applied the framework to matching 2D contours represented as shape context-like distributions. Our earlier work that combines low-distortion embedding and EMD is reported in Demirci et al. (2004) and Keselman et al. (2003).

3. Metric Embedding of Graphs

The problem of many-to-many graph matching is an intractable one, for any subgraph of one graph can be assigned to any subgraph of another. Our goal in graph embedding is to map the graphs to an alternative space in which approximating the many-to-many matching of the mapped graphs is computationally tractable. This view of embedding is consistent with the assumption that the weight of a graph edge reflects the probability that the two nodes the edge spans should be grouped as part of a many-to-many correspondence.

One such transformation is a mapping of the nodes of a graph to points in a geometric space—a process known as graph embedding. Then, as we shall see in Section 5, the points (embedded graph nodes) can be matched many-to-many in polynomial time. To ensure that the solution of the many-to-many point matching problem in the embedded space reflects a solution to the many-to-many graph matching problem in the original graph space, the geometric structure of the points must somehow reflect the topological structure of the graph. This requires the embedding process to ensure that nodes to be grouped get mapped to points in close geometric proximity.

During the last decade, low-distortion embedding has become recognized as a very powerful tool for designing efficient algorithms. In low-distortion embedding of metric spaces into normed spaces, we consider mappings $f : \mathcal{V} \to \mathcal{B}$, where $\mathcal{V}$ is a set of points in the original metric space, with distance function $\mathcal{D}(.,.)$, $\mathcal{B}$ is a set of points in the (host) d-dimensional normed space $|| \cdot ||_k$, and for any pair $p, q \in \mathcal{V}$ we have

$$\frac{1}{c}\mathcal{D}(p, q) \leq ||f(p) - f(q)||_k \leq \mathcal{D}(p, q) \qquad (1)$$

for a certain parameter c, known as the *distortion*. Intuitively, such an embedding will enable us to reduce problems defined over *difficult* metric spaces, $(\mathcal{V}, \mathcal{D})$, to problems over *easier* normed spaces, $(\mathcal{B}, ||.||_k)$. Clearly, the closer c is to 1, the better the target set $\mathcal{B}$ mimics the original set $\mathcal{V}$. Consequently, the distortion parameter c is a critical characteristic of the embedding f.

The above definition of low-distortion embedding maps a set of points in the original metric space to a set of points in the target space. Since our domain is graphs, we must choose a suitable metric for our graphs, i.e., we must define a distance between any two vertices. Let $G = (\mathcal{V}, E)$ 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 $u, v, w \in \mathcal{V}$, $\mathcal{D}(u, v) = \mathcal{D}(v, u) \geq 0$, $\mathcal{D}(u, u) = 0$, 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 $\delta(\cdot, \cdot)$, i.e., $\mathcal{D}(u, v) = \delta(u, v)$, the length of the shortest path between u and v for all $u, v \in \mathcal{V}$.

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. (Gupta, 1999). Let $G = (\mathcal{V}, E)$ be a planar graph, and let $M = (\mathcal{V}, \mathcal{D})$ be the shortest-path metric for the graph G. Then there is an embedding of M into $|| \cdot ||_k$ with $O(1)$ distortion.

This conjecture has only been proven for the case in which G is a tree. One such deterministic tree embedding algorithm was given by Matoušek (1999), suggesting that if we could somehow map our graphs into trees, with small distortion, we could adopt Matoušek's framework.

The problem of approximating (or fitting) an $n \times n$ distance matrix $\mathcal{D}$ by a tree metric $\mathfrak{T}$ is known as the *Numerical Taxonomy* problem. Since the numerical taxonomy problem is an open problem for general distance metrics, we must explore approximation methods. The numerical taxonomy problem can be approximated by converting the distance matrix $\mathcal{D}$ to the weaker ultra-metric distance matrix. An *ultra-metric* is a special type of tree metric defined on rooted trees, where the distance to the root is the same for all leaves in the tree, an approximation that introduces small distortion. A metric $\mathcal{D}$ is an ultra-metric if, for all points x, y, z, we have $\mathcal{D}[x, y] \leq \max\{\mathcal{D}[x, z], \mathcal{D}[y, z]\}$. Unfortunately, an ultra-metric does not satisfy all the properties of a metric distance. To create a general tree metric from an ultra-metric, we need to satisfy the *4-point* condition (see Buneman 1971):

$$\begin{aligned}&\mathcal{D}[x, y] + \mathcal{D}[z, w]\\ &\quad \leq \max\{\mathcal{D}[x, z] + \mathcal{D}[y, w], \mathcal{D}[x, w] + \mathcal{D}[y, z]\},\end{aligned} \qquad (2)$$

for all x, y, z, w. A metric that satisfies the 4-point condition is called an *additive metric*, and a metric $\mathcal{D}$ is additive if and only if it is a tree metric (see Buneman 1971).

Our construction of a tree metric will therefore consist of: (1) constructing an ultra-metric from $\mathcal{D}$, and (2) modifying the ultra-metric to satisfy the 4-point condition. One such approximation framework, called the *centroid metric tree* $\mathfrak{T}$, has been proposed by Agarwala et al. (1999). Their algorithm, which follows the two-step procedure outlined above, generates $\mathfrak{T}$ in time $O(n^2)$. Specifically, the path-length between any two vertices u, v in $\mathfrak{T}$ will mimic the metric $\delta(u, v)$ in G. A metric $\mathcal{D}(\cdot, \cdot)$ 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 $\mathcal{V} \cup \{c\}$ and

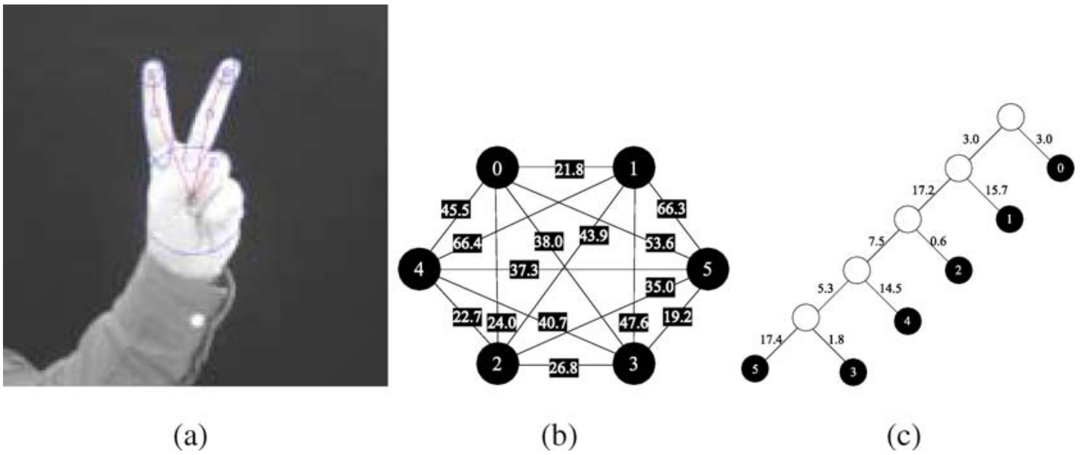

Figure 3. Metric tree representation of the Euclidean distances between nodes in a graph. The gesture image (a) consists of 6 regions (the region representing the entire hand is not shown). The complete graph in (b) captures the Euclidean distances between the centroids of the regions, while (c) is the metric tree representation of the multi-scale decomposition (with additional vertices).

weighted edge-set $\{(c, v_i) | \mathcal{W}(c, v_i) = \ell_i,\ v_i \in \mathcal{V}\}$. 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 labeling ℓ_i of a metric distance $\mathcal{D}(\cdot, \cdot)$, (see Agarwala et al. 1999). It should be noted that this construction does not necessarily maintain the vertex set of G invariant. We will have to ensure that in the embedding process, the extra vertices generated during the metric tree construction are eliminated.

An example of constructing a metric tree from a graph is shown in Fig. 3, in which a hierarchical blob decomposition of an image, shown in (a), yields a graph whose edge weights reflect the Euclidean distances between the nodes (centroids of their corresponding regions), shown in (b). The metric tree representation of the graph is shown in (c). Note that the additional vertices introduced by the construction (shown in white) will be later removed.

Given a metric tree approximation of our original graph, we use the concept of *caterpillar decomposition* and *caterpillar dimension* to capture its topological structure. We illustrate the caterpillar decomposition of a rooted tree with no edge weights in Fig. 4. The three darkened paths from the root represent three edge-disjoint paths, called level 1 paths. If we remove these three level 1 paths from the tree, we are left with the 7 dashed, edge-disjoint paths. These are the level 2 paths, and if removing them had left additional connected components, the process would be repeated until all the edges in the tree had been removed. The union of the paths is called the caterpillar decomposition, denoted by $\mathfrak{P}$, and the number of levels in $\mathfrak{P}$ is called the caterpillar dimension, denoted by cdim($\mathfrak{T}$).

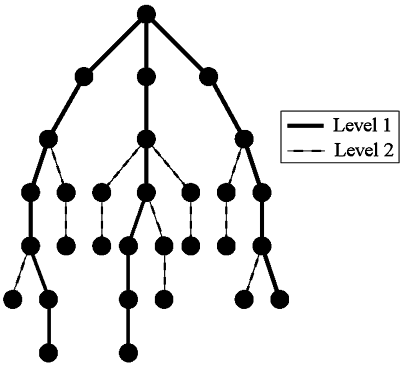

Figure 4. Path partition of a tree.

The caterpillar decomposition $\mathfrak{P}$ can be constructed using a modified depth-first search in linear time. Given a caterpillar decomposition $\mathfrak{P}$ of $\mathfrak{T}$, we will use L to denote the number of leaves of $\mathfrak{T}$, and let $P(v)$ represent the unique path between the root and a vertex $v \in \mathcal{V}$. 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.

3.1. Matoušek's Embedding

The metric tree approximation of our original graph, along with its caterpillar decomposition, are the prerequisites for Matoušek's embedding technique, which maps the nodes in the metric tree to points in some low-dimensional Euclidean space. Given a metric tree, the dimensionality of the embedding is specified by the number of paths in its caterpillar decomposition. To define the embedding $f : \mathcal{V} \rightarrow \mathcal{B}$ under $|| \cdot ||_2$, we let the relevant coordinates in $\mathcal{B}$ be indexed by the paths in $\mathfrak{P}$. The vector $f(v)$, $v \in \mathcal{V}$, has non-zero coordinates corresponding to the paths in the decomposition sequence of $P(v)$. Returning to Fig. 4, the vector $f(v)$ will have 10 components (defined by three level 1 paths and seven level 2 paths). Furthermore, due to the structure of the caterpillar decomposition, every vector $f(v)$ will have at most two non-zero components. Details of this method can be found in Matoušek (1999).

It is important to note that embeddings produced by the above 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 match the two embeddings, we must first perform a "registration" step, whose objective is to project the two distributions into the same normed space. The resulting transformation is expected to minimize the initial EMD between the distributions. Details of a PCA-based alignment technique applied to weighted point sets are presented in Keselman et al. (2003).

3.2. Spherical Coding

The previous embedding procedure suffers from a significant drawback. Namely, each graph is embedded into a vector space whose dimensionality is graph-dependent. Before the embeddings can be matched, a dimensionality reduction step is required, which is both costly and prone to error. In this section, we introduce a novel, linear-time method to embed metric trees into vector spaces of prescribed dimensionality, thereby avoiding the need for a dimensionality reduction step. Like Matoušek's embedding, our embedding is based on the caterpillar decomposition of the metric tree. The paths of this decomposition will be embedded along maximally spaced rays in some fixed-dimension metric space. In this construction, the rays share the origin as their end-points. The main step of the embedding is to identify the principal direction for each ray to guarantee that the rays are maximally apart. In practice, this can be achieved by placing maximally spaced points on the surface of a unit sphere and using the unit-length vectors between the origin and these points as the principle directions of the rays. One may observe that Matoušek's method is a special case of this embedding when the dimension of the embedding space is equal to the number of paths in the decomposition and the corresponding rays form an orthogonal basis for the embedding space.

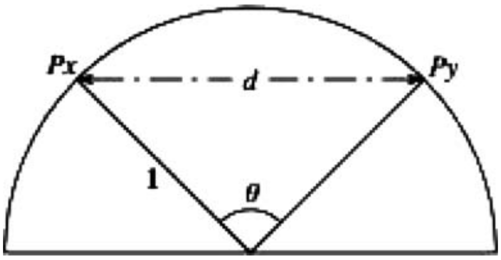

Figure 5. The minimum distance d and minimum angle θ between 2 points.

A *spherical code* is a finite set of n points on the surface of a multi-dimensional unit radius sphere. For our purposes, we identify a spherical code with an embedding of a parent node (the center of the sphere) and its child nodes (points on the sphere). To minimize the distortion of distances between child nodes, we are interested in positioning the points on the sphere so as to maximize the minimum distance between any pair of points (Conway and Sloane, 1998). Equivalently, one can try to minimize the radius r of a multi-dimensional sphere such that n points can be placed on the surface, where any two of the points are at angular distance 2 from each other. Recall that the angular distance between two points is the acute angle subtended by them at the origin. Figure 5 shows the relationship between the minimum distance and minimum angle between 2 points.

The embedding framework is best illustrated through an example, in which a weighted tree is embedded into $\mathbb{R}^2$, as shown in Fig. 6. To ease visualization, we will limit the discussion to the first quadrant. The weighted tree contains 4 paths $\langle a, b, c\rangle$, $\langle a, d, f, h\rangle$, $\langle d, e\rangle$, and $\langle f, g\rangle$ in its caterpillar decomposition. In the embedding, the root is assigned to the origin. Next, we seek a set of 4 vectors, one for each path in the caterpillar decomposition, such that their inner products are minimized, i.e., their endpoints are maximally apart. These

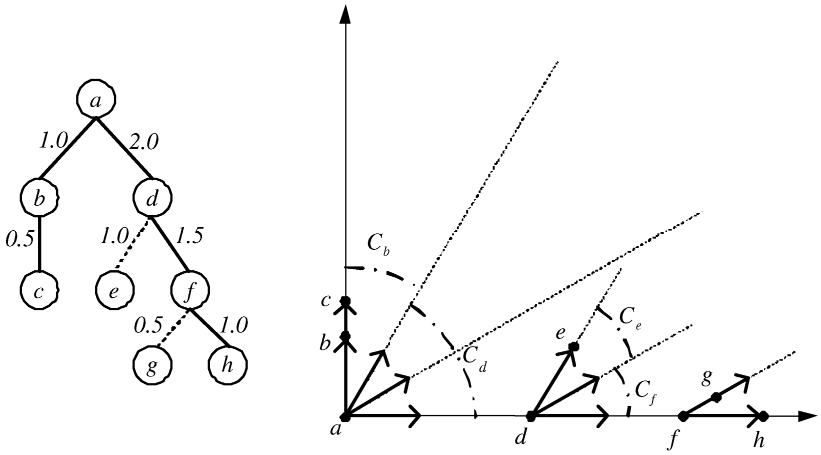

Figure 6. An edge weighted tree and its spherical code in 2D. The Cartesian coordinates of the vertices are: $a = (0, 0)$, $b = (0, 1.0)$, $c = (0, 1.5)$, $d = (2.0, 0)$, $e = (2.5, 0.87)$, $f = (3.5, 0)$, $g = (3.93, 0.25)$, and $h = (4.5, 0)$.

vectors define the general directions in which the vertices on each path in the caterpillar decomposition will be embedded.

Three of the four vectors will be used by the caterpillar paths belonging to the subtree rooted at vertex d, and one vector will be used by the path belonging to the subtree rooted at vertex b. This effectively subdivides the first quadrant into two cones, C_b and C_d. The volume of these cones is a function of the number of caterpillar paths belonging to the subtrees rooted at b and d. The cone C_d, in turn, will be divided into two smaller cones, C_e and C_f, corresponding to the subtrees rooted at e and f, respectively. The extreme rays of sub-cones C_b, C_e, and C_f will correspond to the four directions defining the embedding. Finally, to complete the embedding, we translate the sub-cones away from the origin along their directional rays to positions defined by the path lengths in the tree. For example, to embed point b, we will move along the extremal ray of C_b and will embed b at (0, 1.0). Similarly, the subcone C_d will be translated along the other extremal ray, embedding d at (2.0, 0).

In d-dimensional Euclidean space $\mathbb{R}^d$, computing the embedding $f : \mathcal{V} \to \mathcal{B}$ under $|| \cdot ||_2$ is more involved. Let L denote the number of paths in the caterpillar decomposition. The embedding procedure defines L vectors in $\mathbb{R}^d$ that have a large angle with respect to each other on the surface of a hypersphere S_d of radius r. These vectors are chosen in such a way that any two of their endpoints on the surface $\sum_d$ are at least spherical distance 2 from each other. We will refer to such vectors as *well-separated*. Consider the set of hyperplanes $H_i = (0, 2, 4, \ldots, 2i)$, and let $\sum_d(i) = H_i \cap \sum_d$. Since each of the $\sum_d(i)$ are hypercircles, i.e., surfaces of spheres in dimension $d - 1$, we can recursively construct well-separated vectors on each hypercircle $\sum_d(i)$. Our construction stops when the sphere becomes a circle and the surface becomes a point in 2 dimensions. It is known that taking r to be $O(dL^{1/d-1})$, and the minimum angle between two vectors to be $2/r$, provides us with L well-separated vectors (Conway and Sloane, 1998). In Fig. 6, we have 4 such vectors emanating from the origin.

Now that the embedding directions have been established, we can proceed with the embedding of the vertices. The embedding procedure starts from the root (always embedded at the origin) and embeds vertices following the embedding of their parents. For each vertex in the metric tree $\mathfrak{T}$, we associate with every subtree $\mathfrak{T}_v$ a set of vectors C_v, such that the number of vectors in C_v equals the number of paths in the caterpillar decomposition of $\mathfrak{T}_v$. Initially, the root has the entire set of L vectors. Consider a subtree rooted at vertex v, and let us assume that vertex v has k children, $v_1, \ldots, v_k$. We partition the set of vectors into k subsets, such that the number of vectors in each subset, S_v, equals the number of leaves in $\mathfrak{T}_v$. We then embed the vertex v_l $(1 \leq l \leq k)$ at the position $f(v) + w_l * x_l$, where w_l is the length of the edge (v, v_l) and x_l is some vector in C_v. We recursively repeat the same process for each subtree rooted at every child of v, and stop when there are no more subtrees to consider.

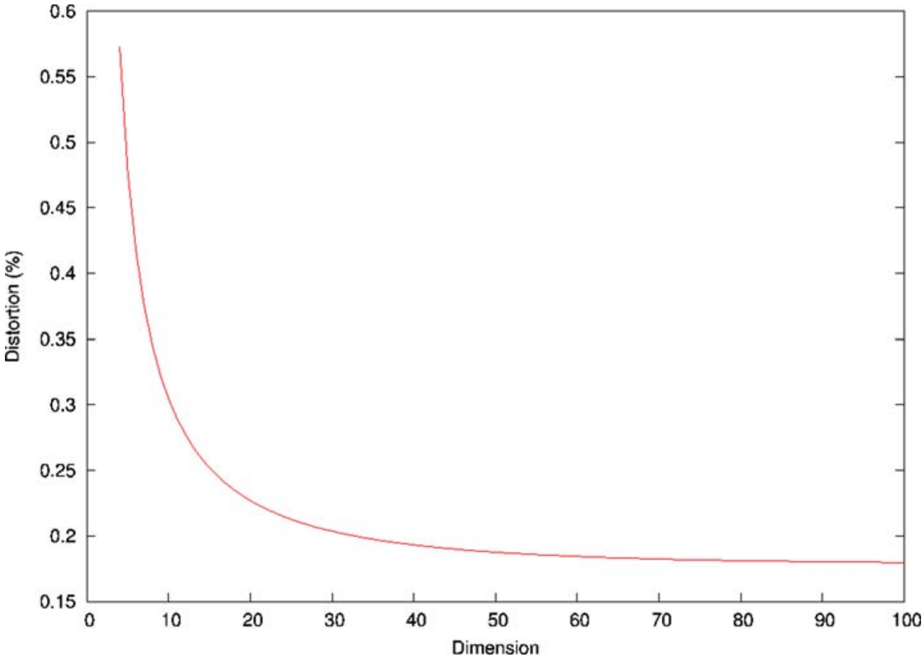

Figure 7. Trade-off between distortion and dimension. Increasing the dimensionality of the embedding space will reduce the the distortion. However, this trend does not continue indefinitely to produce isometric embeddings.

A natural question that one may ask is what dimensionality of Euclidean space should be chosen so that the embedding preserves pairwise distances with minimum distortion. To answer this question, we conducted a set of experiments in which 200 randomly selected edge-weighted trees were embedded into Euclidean spaces of varying dimensions, and measured the average distortion in each dimension. For a given tree, we used the following method to measure its average distortion in one particular dimension. First, we computed all of its pairwise node distances before and after the embedding. We then measured the maximum factor by which any pairwise distance was changed by the embedding algoritm. After repeating this procedure for all trees, the average distortion for one particular dimension was calculated. The trade-off between distortion and dimension is shown in Fig. 7. It should be noted that, while increasing the dimensionality of the embedding space will improve the quality by decreasing the distortion, this trend does not continue indefinitely to produce isometric embeddings. This can be attributed to the fact that the original distances are non-additive, making an isometric embedding impossible.

4. Encoding Directed Edges

The distance metric defined on the graph structure is based on the undirected edge weights. While the above embedding has preserved the distance metric, it has failed to preserve any oriented relations, such as the hierarchical relations common to scale-space or coarse-to-fine structures. This is due to the fact that oriented relations do not satisfy the symmetry property of a metric. We can retain this important information in our embedding by moving it into the nodes as node attributes, a technique used in the encoding of directed topological structure in Siddiqi et al. (1999), directed geometric structure in Shokoufandeh et al. (2002), and shape context in Belongie et al. (2002). Encoding in a node the attributes of the oriented edges incident to the node requires computing distributions on the attributes and assigning them to the node. For example, a node with a single parent at a coarser scale and two children at a finer scale might encode a relative scale distribution (histogram) as a node attribute. The resulting attribute provides a contextual signature for the node which will be used by the matcher (Section 5) to reduce matching ambiguity.

We will motivate this encoding in the context of directed graphs for qualitative shape representation using a blob/ridge decomposition, two example graphs are shown in Fig. 2; details can be found in Shokoufandeh et al. (2002). Blobs (compact regions) and ridges (elongated structures) are extracted as scale-space maxima of

$$\nabla^2_{\text{norm}} L = t\,(L_{xx} + L_{yy}). \tag{3}$$

and

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

respectively (Lindeberg, 1998a, b). For color images, the feature detection is performed in the R, G and B channels. To represent the spatial extent of a detected image structure, a windowed second moment matrix

$$\Sigma = \int_{\eta\in\mathbb{R}^2} \begin{pmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{pmatrix} g(\eta; t_{\text{int}})\, d\eta \tag{5}$$

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. The orientation and the anisotropy of the feature are estimated from the eigenvalues of Σ and the corresponding eigenvectors. The spatial extent of the feature is thus given by the scale, the anisotropy and the orientation. Figure 8 shows an image of a hand with the extracted features superimposed. Blobs and ridges are graphically represented by circles and ellipses defining *support regions*, with radii (for ridges the minor radius) proportional to $\sqrt{t}$.

Spatially overlapping and aligned ridges are linked and spatially overlapping blobs are merged. Directed acyclic graphs are then built in a coarse-to-fine manner. Specifically, let $G = (\mathcal{V}, E)$ be a graph to be embedded. Each feature will be a node in the graph and associated with each node (blob/ridge) are a number of attributes, including position, orientation, and support region. A feature at the coarsest scale 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 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, in this way, have multiple parents. To yield one 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.

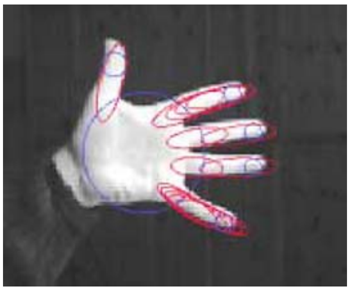

Figure 8. Feature extraction: Extracted blobs and ridges at appropriate scales.

Associated with each edge are a number of important geometric attributes. For an edge $\mathcal{E}$, directed from a vertex $\mathcal{V}_A$ representing feature $\mathcal{F}_A$, to a vertex $\mathcal{V}_B$ representing feature $\mathcal{F}_B$, we define the following attributes, as shown in Fig. 9:

- *Distance.* Two measures of inter-feature distance are associated with the edge: (1) the smallest distance d from the support region of $\mathcal{F}_A$ to the support region of $\mathcal{F}_B$, normalized to the the largest of the radii r_A and r_B; and (2) the distance between their centers 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 $t_{\mathcal{F}_B}$.

Examples of graphs for hand images, showing hierarchical edges, are shown in Fig. 1.

For every pair of vertices, (u, v), we let $R_{u,v}$ denote the attribute vector associated with the pair. The entries of each such vector represent the set of oriented relations; for the example domain of blob graphs, R = {distance, relative orientation, bearing, scale ratio} between u, v, as shown in Fig. 9 (Shokoufandeh et al., 2002). For a vertex $u \in \mathcal{V}$, we let $N(u)$ denote the set of vertices $v \in \mathcal{V}$ adjacent to u. For a relation $p \in R$, we will denote $\mathcal{P}(u, p)$ as the set of values for relation p between u and all vertices in $N(u)$, i.e., $\mathcal{P}(u, p)$ corresponds to entry p of vector $R_{u,v}$ for $v \in N(u)$. Fea-

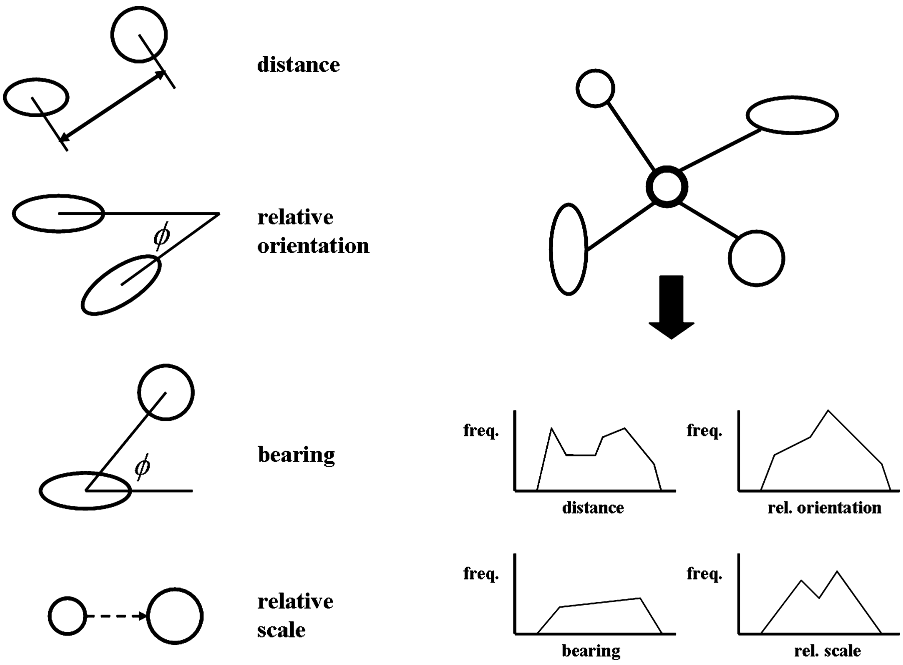

Figure 9. Directed graph relations and their resulting node-centric distributions.

ture vector $\mathcal{P}_u$ for point u is the set of all $\mathcal{P}(u, p)$'s for $p \in R$. Observe that every entry $\mathcal{P}(u, p)$ of vector $\mathcal{P}_u$ can be considered as a local distribution (*histogram*) of feature p in the neighborhood $N(u)$ of u. We adopt the method of Shokoufandeh et al. (2002), in which the distance function for two such vectors $\mathcal{P}_u$ and $\mathcal{P}_p$ is computed through a weighted combination of Hausdorff distances between $\mathcal{P}(u, p)$ and $\mathcal{P}(u', p)$ for all values of p.

5. Distribution-Based Many-to-Many Matching

By embedding vertex-labeled 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. Given a pair of weighted distributions in the same normed space, the Earth Mover's Distance (EMD) framework (Rubner et al., 2000) 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. 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. The main advantage of using EMD 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 uneven clusters and noisy datasets. Details of the method, along with an extension, are presented in Rubner et al. (2000) and Keselman et al. (2003).

The standard EMD formulation assumes that the two distributions have been aligned. However, 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 (1999) 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 **F**low and an optimal **T**ransformation") that achieves a local minimum of the objective function. Details on how we compute the optimal transformation can be found in Keselman et al. (2003) and Demirci et al. (2003).

5.1. The Final Algorithm

Our algorithm for many-to-many graph matching is a combination of the previous procedures, and is summarized as follows:

Algorithm 1: Many-to-many graph matching

```
1: Compute the metric tree $\mathfrak{T}_i$ corresponding to $G_i$ according to Section 3 (see Agarwala et al. 1999 for details).
2: Construct low-distortion embeddings $\mathcal{E}_i = f_i(\mathfrak{T}_i)$ of $\mathfrak{T}_i$ into $(\mathcal{B}_i, || \cdot ||_2)$ according to one of the algorithms presented in Sections 3.1 and 3.2.
3: Compute the EMD between $\mathcal{E}_i$'s by applying the FT iteration, computing the optimal transformation $T$ according to Section 5 (see Keselman et al. 2003 for details).
4: Interpret the resulting optimal flow between $\mathcal{E}_i$'s as a many-to-many vertex matching between $G_i$'s.
```

5.2. Complexity Analysis of Algorithm 1

We now proceed to analyze the computational complexity of Algorithm 1. Computing the metric tree $\mathfrak{T}_i$ for a given graph G_i takes $O(|V|^2)$ in Step 1 (see Agarwala et al. 1999 for details). The complexity of Step 2 depends on the complexity of the corresponding embedding algorithm. While this may take $O(|V| \times |E|)$ using graph-dependent dimensionality, it can also be done in linear time through spherical coding. Since computing the EMD (Step 3) is based on the transportation problem, it can be formulated as a linear programming problem and can be solved using a network flow algorithm in $O(|V|^3)$. The FT iteration alternates between finding the optimal transformation for a given flow and the optimum flow for a given transformation. While the exact bound on the complexity of the iterative EMD is not known, in almost all our experiments no more than 6 iterations were necessary for the matching procedure to converge. Finally, Step 4, the mapping of the EMD solution back to the graph solution, is $O(|V|)$. Assuming that the number of FT iterations is constant, the overall complexity of the algorithm is therefore $O(|V|^3)$. Note that the overall complexity can be further improved by using more computationally efficient algorithms for the transportation problem. For example, Atkinson and Vaidy (2004) presented an $O(n^{2.5} \log n \log W)$ algorithm for solving the transportation problem, where W is the magnitude of the largest supply or demand in the EMD formulation and n is the total number of nodes in G_1 and G_2.

6. Experiments

To demonstrate our approach to many-to-many graph matching, we apply it to the problem of view-based 3-D object recognition using two different graph-based shape representations. We first turn to the domain of view-based object recognition using silhouettes. For a given view, an object's silhouette is represented by an undirected shock tree, whose nodes represent *shocks* (Siddiqi et al., 1999) (or, equivalently, skeleton points) and whose edges connect adjacent shock points.[1] To make the tree suitable for embedding and for EMD matching, it is made rooted and node- and edge-weighted. The tree is constructed from the tree-like discrete skeleton as follows.

We will assume that each shock point p on the discrete skeleton is labeled by a 3-dimensional vector $v(p) = (x, y, r)$, where (x, y) are the Euclidean coordinates of the point and r is the radius of the maximal bitangent circle centered at the point.[2] Each shock point becomes a node in the newly constructed graph. Each

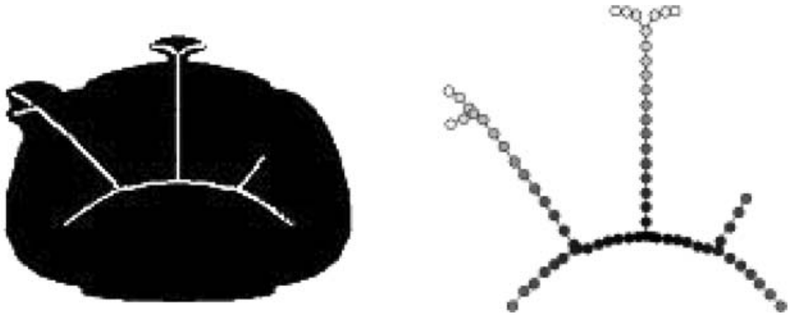

Figure 10. Left: the silhouette and its medial axis. Right: the shock tree constructed from the medial axis. Darker nodes reflect larger radii.

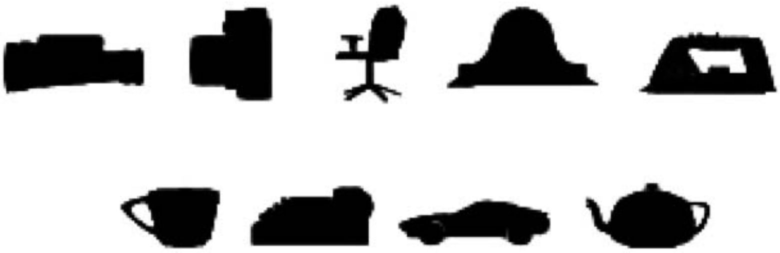

Figure 11. Sample views of the 9 objects.

pair of shock points is connected by an edge whose weight is the Euclidean distance between the points' vector labels. Node and edge weights therefore reflect mass and grouping strength, respectively. The graph is converted into a tree by computing its minimum spanning tree. Thus, tree nodes correspond to shock points, and tree edges connect nearby shock points. We choose the root of the tree to be the node that minimizes the sum of the tree-based shortest path distances to all other nodes.[3] Each node is weighted proportionally to its radius, with the total tree weight being 1. In effect, our conversion of the discrete skeleton to a shock tree replaces step 1 (computation of the metric tree) of the many-to-many matching algorithm.

An illustration of the procedure is given in Fig. 10. The left portion shows the initial silhouette and its shock points (skeleton). The right portion depicts the constructed shock tree. Darker nodes correspond to fragments whose average radii are larger. To reduce the size of the graph, we first subdivided the skeleton into a number of small fragments of approximately 5 shock points each. Each fragment was labeled by the vector average of the vector labels of the fragment's constituent shock points.

Our database consists of 1620 silhouettes of 9 objects, with 180 views per object. A representative view of each object is shown in Fig. 11. For the experiments, we compute the shock tree representation of every silhouette, and use Matoušek's embedding to embed each tree into a normed space with low distortion. This procedure results in a database of weighted point-sets, each representing an embedded graph. As an object is rotated in depth, its views undergo distortion as some parts become more or less foreshortened. This results in skeletal branches of varying length, setting up an ideal many-to-many matching problem.

To test our approach, we randomly selected 19 equidistant views of each object and computed distances between these views and each of the remaining database entries (the distance between a view and itself is always zero). To compute the distance between objects A and B, for each of the 19 views of object A, we find the closest view of object B and average over the resulting distances. These object distances are summarized in Fig. 12, Table 1. Each small square in Table 1 corresponds to a set of 19×19 matching results between the views of the two objects associated with the row and the column. For simplicity of visualization, we computed a normalized matching score for each 19×19 block. The magnitudes of the distances are denoted by shades of gray, with black and white representing the smallest and largest distance, respectively. Due to symmetry of the resulting distances, we only included the upper triangle of results. Intra-object distances, shown along the main diagonal, are very close to zero. The average intra-object distance for all objects was found to be 15.9. According to the table, inter-object distances were near intra-object distances in only 3 out of 36 cases (BINOCULAR and CLOCK, CAMERA and PHONE, and CAR and TEAPOT). The average inter-object distance for all for all objects was found to be 39.3.

To better understand the differences in the recognition rates for different objects, we have selected a subset of the matching results among the 4 views of TEAPOT, taken at 20°, 30°, 60°, and 90°, respectively, as shown in Fig. 12, Table 2. Due to the highly symmetric structure of the object, implying that neighboring views are more likely to be similar, the distance between a view of TEAPOT and its neighboring view is closer than its distance to other objects' views. Conversely, Fig. 12, Table 3 illustrates the fact that due to a low view sampling resolution, certain views of certain objects are more similar to certain views of other objects than they are to neighboring views of the same object. For example, the best (non-identical) match for the third view of CUP is the first view of PHONE. Upon closer inspection of these two degenerate views, it turns out that

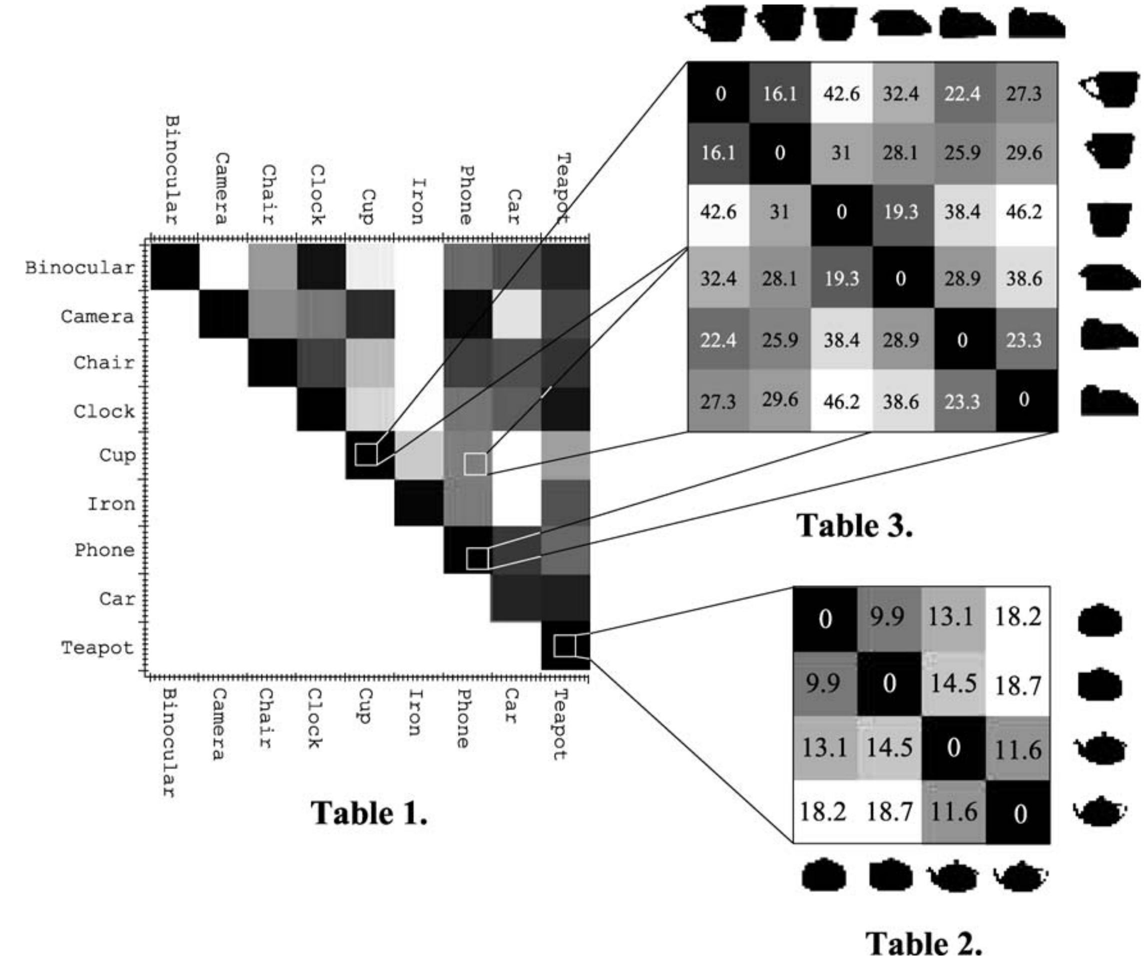

Table 1.

0	16.1	42.6	32.4	22.4	27.3
16.1	0	31	28.1	25.9	29.6
42.6	31	0	19.3	38.4	46.2
32.4	28.1	19.3	0	28.9	38.6
22.4	25.9	38.4	28.9	0	23.3
27.3	29.6	46.2	38.6	23.3	0

Table 3.

0	9.9	13.1	18.2
9.9	0	14.5	18.7
13.1	14.5	0	11.6
18.2	18.7	11.6	0

Table 2.

Figure 12. Summary of many-to-many matchings of object silhouettes. Every entry of Table 1 corresponds to a set of 19×19 matching results between the views of the two objects associated with the row and the column. The shade of gray in each cell denotes average matching distance of each 19×19 block, with black and white representing the smallest and largest distances, respectively. Averaging across all experiments the intra-object distance is 15.9, while the average inter-object is 39.3. Table 2 shows a close up look at the matching results for 4 views of TEAPOT. The distance between an object's view and its neighboring view is closer than the distance between an object's view and other objects' views. Table 3 depicts a subset of results from three separate blocks, highlighting that in some cases, certain views of some objects are more similar to views of other objects than neighboring views of the same object.

there is considerable similarity in their shock tree representations. On the other hand, the first two views of CUP have been optimally matched to each other, along with the last two views of PHONE. The top row of Fig. 13 shows two adjacent views (30° and 40°) of the TEAPOT. The bottom row illustrates the many-to-many feature correspondences that our matching algorithm yields. Corresponding node clusters from each graph (many-to-many mappings) have been shaded with the same color. 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.

Based on the overall matching statistics, we observed that in only 5.74% of the experiments, the closest match selected by our algorithm was not a neighboring view of the correct object. We expect that with increased view sampling resolution, ensuring that for each object view there exists a similar neighboring view, this error rate would decrease significantly. We repeated the experiment using the spherical embedding, resulting in a 4.9% error rate. This is a clear improvement in performance, at a reduced computational cost.

It should be noted that both the embedding and matching procedures can accommodate perturbation,

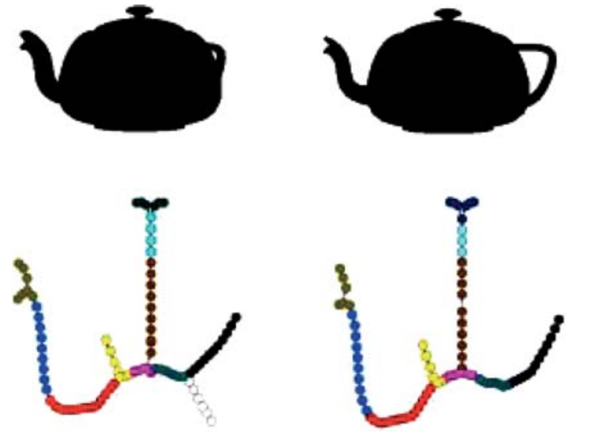

Figure 13. Illustration of the many-to-many correspondences. The top row shows two adjacent views (30° and 40°) of the TEAPOT. The bottom row illustrates the many-to-many feature correspondences that our matching algorithm yields. Corresponding node clusters from each graph (many-to-many mappings) have been shaded with the same color.

such as noise and occlusion. This is due to the fact that the path partitions for unperturbed portions of the graph are unaffected by perturbation. Moreover, the projections of unperturbed nodes will also be unaffected by perturbation. Finally, the matching procedure is an iterative process driven by flow optimization which, in turn, depends only on local features whose attributes can act as matching constraints.

To test the sensitivity of the matching algorithm to perturbation of the query, we performed the following experiment for each of the 9 objects. Each view, in turn, was used as a query (with replacement) and perturbed by deleting a randomly selected connected subset of the skeleton points whose size was chosen randomly to fall between 5% and 25% of the total number of skeleton points. If the closest view to the query was the unperturbed view, matching was scored as correct. For the 9 objects, the average correct score was 89%, reflecting the algorithm's robustness to missing data, a form of occlusion. For the spherical embedding, we observed an average correct score of 91.4%.

We now turn to the domain of blob graphs, a brief overview of which was presented in Section 4. Again, node and edge weights therefore reflect mass and grouping strength, respectively. The results of applying our method to these two images is shown in Fig. 14, in which many-to-many feature correspondences have been colored the same. For example, a set of blobs and ridges describing a finger in the left image is mapped to a set of blobs in ridges on the corresponding finger in the right image.

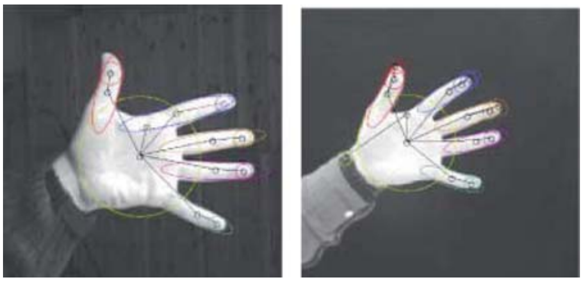

Figure 14. Applying our algorithm to the images in Fig. 1. Many-to-many feature correspondences have been colored the same.

To provide a more comprehensive evaluation, we also tested our framework on the two separate image libraries shown in Fig. 15: the Columbia University COIL-20 (Murase and Nayar, 1995) (20 objects, 72 views per object) and the ETH Zurich ETH-80 (Leibe and Schiele, 2003) (8 categories, 10 exemplars per category, 41 views per exemplar).[4] Figure 15 shows representative views of objects from each database. For each view, we compute a multi-scale blob decomposition, using the algorithm described in Shokoufandeh et al. (2002). Next, we compute the tree metric corresponding to the complete edge-weighted graph defined on the regions of the scale-space decomposition of the view. The edge weights are computed as a function of the distances between the centroids of the regions in the scale-space representation. Each tree is then embedded into a normed space of prescribed dimension. This procedure results in two databases of weighted point sets, each point set representing an embedded graph.

For the COIL-20 database, we begin by removing 36 (of the 72) representative views of each object (every other view), and use these removed views as queries to the remaining view database (the other 36 views for each of the 20 objects). We then compute the distance between each "query" view and each of the remaining database views, using our proposed matching algorithm. Ideally, for any given query view i of object j, $v_{i,j}$, the matching algorithm should return a neighboring view. We will classify this as a correct matching. Based on the overall matching statistics, we observe that for the spherical embedding, in all but 4.8% of the experiments, the closest match selected by our algorithm was a neighboring view. Moreover, among the mismatches, the closest view belonged to the same object in 81.02% of the cases. In comparison, Matoušek's embedding yielded a 10.74% matching error where,

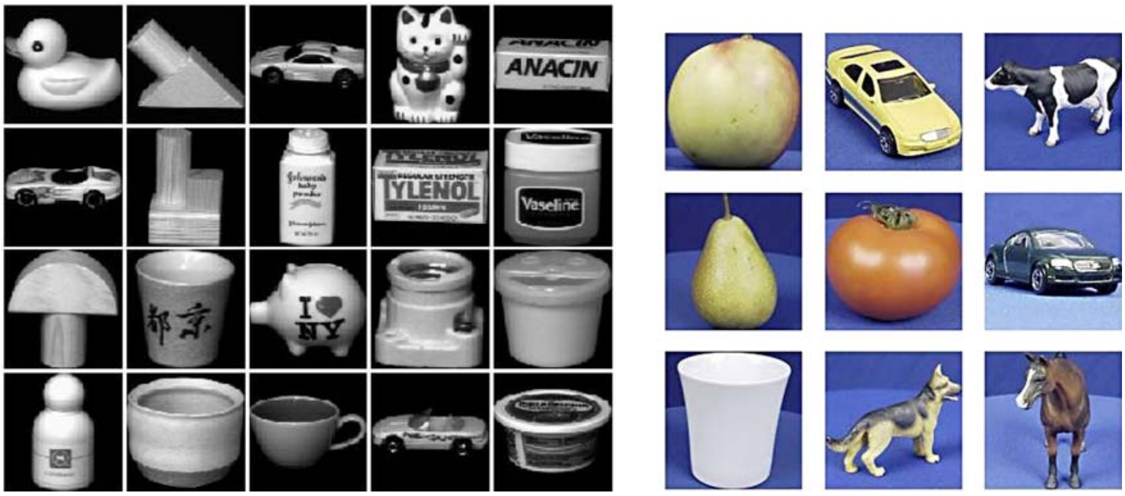

Figure 15. Views of sample objects from the Columbia University Image Library (COIL-20) and the ETH Zurich (ETH-80) Image Set.

among the mismatches, the closest view belonged to the same object in 80.0% of the cases.

For the ETH-80 database, we chose a subset of 32 objects (4 from each of the 8 categories) with full sampling (41 views) per object. For each object, we removed each of its 41 views from the database, one view at a time, and used the removed view as a query to the remaining view database. We then computed the distance between each query view and each of the remaining database views. The criteria for correct classification were similar to the COIL-20 experiment. Our experiments showed that in all but 6.2% of the experiments using the spherical embedding, the closest match selected by our algorithm was a neighboring view. Among the mismatches, the closest view belonged to the same object in 77.19% of the cases, and the same category in 96.27% of the cases. For Matoušek's embedding, in all but 17.5% of the experiments, the closest view was a neighboring view, while among the mismatches, the closest view belonged to the correct object in 67.4% of the cases, and the same category in 81.3% of the cases. The results clearly demonstrate the improved performance offered by the spherical embedding technique.

To demonstrate the framework's robustness, we performed four perturbation studies on the COIL-20 and ETH-80 databases, represented using spherical embedding. The experiments are identical to the COIL-20 and ETH-80 experiments described above, except that the query graph was perturbed by adding/deleting 5%, 10%, 15%, and 20% of its nodes (and their adjoining edges). The choice of spherical embedding was motivated by its better performance over that of Matoušek's embedding. The results are shown in Table 1, and reveal that, like our skeleton tree matching example, the error rates increase gracefully as a function of increased perturbation.

It should be noted that both the skeleton tree and blob graph experiments can be considered worst case for two reasons. First, the sampling resolutions of the viewing sphere were high in each case, meaning that more than the immediate neighbors of a particular view may be similar to it. Given the high similarity among neighboring views, it could be argued that our matching 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 (e.g., collapse similar views into view classes and choose one prototype, or "aspect", per class), neighboring views would be more dissimilar, and our matching results would improve. Second, and perhaps more importantly, many of the objects are symmetric, and if a query neighbor has an identical view elsewhere on the object, that view might be chosen (with equal distance) and scored as an error.

Table 1. Recognition rate as a function of increasing perturbation. Note that the baseline recognition rate (with no perturbation) is 98.0% for COIL-20 and 98.5% for ETH-80.

Perturbation	5%	10%	15%	20%
Recognition rate COIL	91.07%	88.13%	83.68%	77.72%
Recognition rate ETH	93.2%	90.1%	86.3%	82.2%

6.1. Comparison to Other Approaches

In addition to demonstrating the effectiveness of our many-to-many matching algorithm applied to shape retrieval, we compare our matching results to two leading graph matching algorithms: a one-to-one matching algorithm proposed by Pelillo et al. (1999) (using association graphs) and a many-to-many matching algorithm proposed by Sebastian et al. (2004) (using graph-edit distance). For the comparison, we use the Rutgers Tool Database (Siddiqi et al., 1999), which consists of 25 shapes organized into eight classes: brush, hammer, pliers, screwdriver, wrench, hand, profile, and horse. Four of these classes, namely, hammer, pliers, screwdriver, and wrench, can be further grouped into a broader "tools" category. Sample views from each class are shown in Fig. 17. In the experiment, we remove the first shape (the query) from the database and compare it to all remaining database shapes. The shape is then put back in the database, and the procedure is repeated with the second database shape, etc., until all 25 shapes have been used as a query. After computing the similarity values between every database pair, we look at the top matches to see how many of the within-category shapes belong to the same class as the query. Ideally, if an object has n shapes in the database, the top $n - 1$ entries should belong to the same class as the query. Figure 16 shows some examples of the many-to-many feature matching results obtained from our algorithm for some of the objects in the Rutgers Tools Database.

Our results, along with those reported in Pelillo et al. (1999) and Sebastian et al. (2004), are presented in Fig. 18, where correct matches retrieved from the database are colored yellow, while the mismached entries are colored red. Considering only the best matches, we observe that while in Pelillo et al.'s shock tree approach there is a total of 3 mismatched entries, both Sebastian et al.'s graph-edit distance framework and our approach yield only 1 mismatched entry. In addition, considering all within-category matches, both the shock tree and graph-edit distance approaches yield a total of 5 errors, while our approach yields only 3 errors. Moreover, if we further group the hammer, pliers,

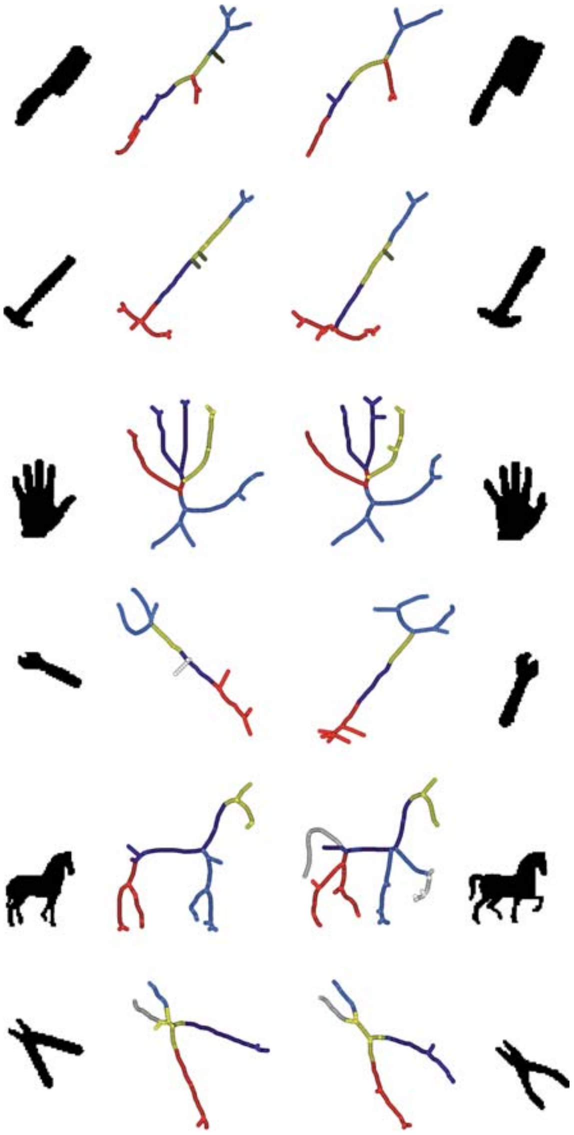

Figure 16. The results of matching skeleton graphs for some pairs of shapes in the Rutgers Tools Database. Same colors indicate corresponding segments. Observe that correspondences are intuitive in all cases.

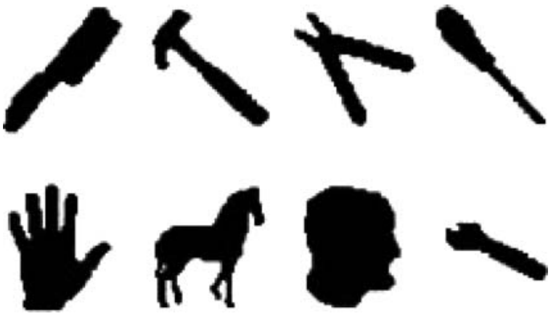

Figure 17. Sample views of objects from the Rutgers Tools Database.

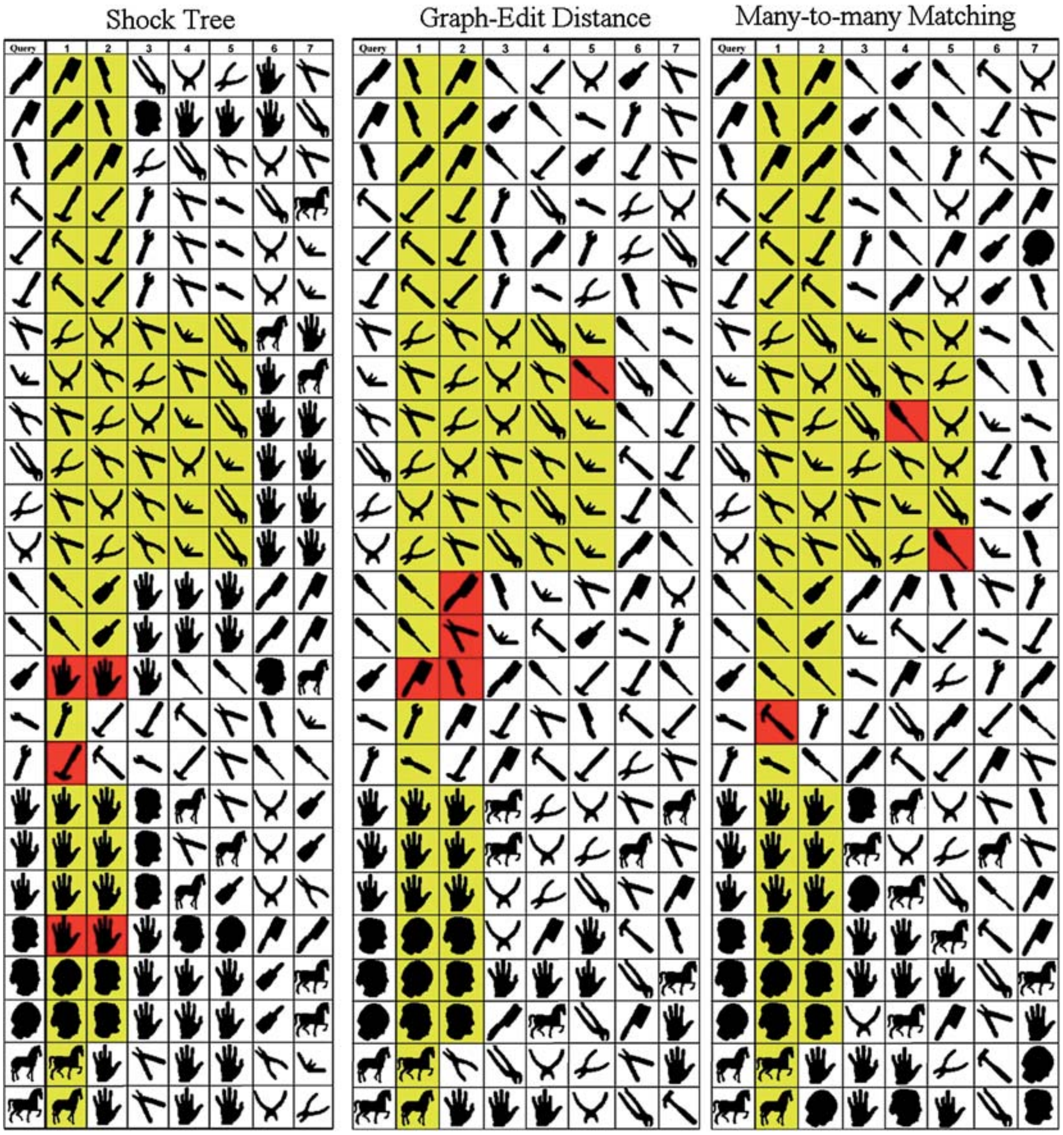
Shock Tree
Graph-Edit Distance
Many-to-many Matching
Query
1
2
3
4
5
6
7

Figure 18. Comparison to two leading graph matching algorithms: Pelillo et al. (1999) (left), Sebastian et al. (2004) (center), and our algorithm (right). In each case, the top seven matched database objects are sorted by their similarity to the query. Correct matches are colored yellow, while mismatched entries are colored red; matches from other categories are colored white.

screwdriver, and wrench shapes into the same "tools" category, our many-to-many matching approach produces a 100% correct matching, while the other two approaches still have mismatched entries. One would expect that as the incidence of many-to-many correspondences increases, both the graph-edit distance algorithm and our algorithm would yield improved scores.

7. Limitations

Our proposed algorithm can be applied to many-to-many matching problems on both undirected and directed graphs. Still, the approach has its limitations. An optimistic reader might think that our embedding approach together with the EMD framework solve the many-to-many graph matching problem. However, its efficacy is contingent on appropriate edge weights in the original graph. Since these edge weights ultimately govern the proximity of the embedded points and hence their propensity to being combined during the EMD step, the edge weights (distances) are effectively a perceptual grouping or abstraction heuristic between features (nodes). If they are chosen or defined poorly, the EMD step will not converge on a meaningful solution.

The framework also assumes that a node's attributes can be mapped into a vector of masses whose splitting and combining during the EMD computation reflects meaningful many-to-many correspondences between the nodes in the original graphs. The EMD is also a global distance that tries to account for all the points. Although our experiments have shown that our framework is robust to perturbation of the graphs in terms of missing and spurious features, the method is still global. If a graph includes a node representing an occluder with large mass, its presence will have an adverse effect on the computed flows, for the algorithm cannot selectively exclude the node. Note, however, that if there are unique attributes shared by nodes to be matched, these attributes can act as constraints on the EMD matching, ensuring that dirt from a pile of a particular "color" can flow only to holes of the same color. We are also currently exploring approaches that use EMD both to compute local correspondences, and then to grow these correspondences.

8. Conclusions

The ability to match image features many-to-many is a critical prerequisite for less constrained object recognition, such as object categorization. Rigid matching schemes that assume one-to-one feature correspondence will fail to match object instances where commonality exists at a higher level of abstraction. Given a set of image features and pairwise relations, the complexity of the resulting many-to-many graph matching problem is intractable. We have presented a novel, computationally efficient approximation algorithm for the many-to-many matching of two graphs. We begin by constructing metric tree representations of the two graphs. Next, we embed them in a geometric space with low distortion using novel encodings of the graph's vertices. Many-to-many graph matching now becomes a many-to-many geometric point matching problem, for which the Earth Mover's Distance algorithm is ideally suited. Moreover, by mapping a node's geometric and structural "context" in the graph to an attribute vector assigned to its corresponding point, we can extend the technique to deal with hierarchical graphs that represent multi-scale structures. We have successfully evaluated the technique on several image databases using two different graph-based multi-scale shape representations that capture coarse shape structure, and include a set of structural perturbation experiments that establish the algorithm's robustness to graph "noise". Moreover, the approach compares favorably to two leading graph matching algorithms on a specific image database.

Acknowledgments

Ali Shokoufandeh acknowledges the partial support provided by grants from the National Science Foundation and the Office of Naval Research. The work of Yakov Keselman is supported, in part, by the NSF grant No. 0125068. Sven Dickinson acknowledges the support of NSERC, CITO, IRIS, PREA, and the NSF. The authors would also like to thank Shree Nayar and Bernt Schiele for their COIL-20 and ETH-80 databases, respectively.

Notes

1. Note that this representation is closely related to Siddiqi et al.'s *shock graph* Siddiqi et al. (1999), except that our nodes (shock points) are neither clustered nor are our edges directed.
2. This vector essentially encodes local shape information of the silhouette.
3. Although we seek a canonical node as root, we note that the shortest-path distance matrix, and hence the structure of the embedding, is invariant to the choice of root.
4. Arguably, the COIL database is not the ideal testbed for an image representation (in our case, a multi-scale blob decomposition) whose goal is to describe the coarse shape of an object. Unlike the PCA-based image characterization for which the COIL database was originally created, the multi-scale blob and ridge decomposition provides invariance to translation, rotation, scale, minor part deformation and articulation, and minor within-class shape deformation. Although a standard database for recognition testing, the COIL database does not exercise these invariants.

References

Agarwala, R., Bafna, V., Farach, M., Paterson, M., and Thorup, M. 1999. On the approximability of numerical taxonomy (fitting distances by tree metrics). *SIAM Journal on Computing*, 28(2):1073–1085.

Athitsos, V., Alon, J., Sclaroff, S., and Kollios, G. 2004. BoostMap: A method for efficient approximate similarity rankings. In

Proceedings, IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC.

Atkinson, D.S. and Vaidya, P.M. 1995. Using geometry to solve the transportation problem in the plane. *Algorithmica*, 13(5):442–461.

Bell, E.T. 1934. Exponential numbers. *Amer. Math. Monthly*, 41:411–419.

Belongie, S., Malik, J., and Puzicha, J. 2002. Shape matching and object recognition using shape contexts. *IEEE PAMI*, 24(4):509–522.

Buneman, P. 1971. The recovery of trees from measures of dissimilarity. In F. Hodson, D. Kendall, and P. Tautu (Eds.), *Mathematics in the Archaeological and Historical Sciences*, Edinburgh University Press, Edinburgh, pp. 387–395.

Carcassoni, M. and Hancock, E.R. 2003. Correspondence matching with modal clusters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1609–1614.

Cohen, S.D. and Guibas, L.J. 1999. The earth mover's distance under transformation sets. In *Proceedings, 7th International Conference on Computer Vision*, Kerkyra, Greece, pp. 1076–1083.

Conway, J.H. and Sloane, N.J.A. 1998. *Sphere Packing, Lattices and Groups*. Springer-Verlag, New York.

Demirci, F., Shokoufandeh, A., Dickinson, S., Keselman, Y., and Bretzner, L. 2004. Many-to-many feature matching using spherical coding of directed graphs. In *Proceedings, 8th European Conference on Computer Vision*, Prague, Czech Republic.

Demirci, F., Shokoufandeh, A., Keselman, Y., Dickinson, S., and Bretzner, L. 2003. Many-to-many matching of scale-space feature hierarchies using metric embedding. In *Scale Space Methods in Computer Vision, 4th International Conference*, Isle of Skye, UK, pp. 17–32.

Gionis, A., Indyk, P., and Motwani, R. 1999. Similarity search in high dimensions via hashing. In *VLDB*, pp. 518–529.

Gold, S. and Rangarajan, A. 1996. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–388.

Gupta, A. 1999. Embedding tree metrics into low dimensional euclidean spaces. In *Proceedings of The Thirty-First Annual ACM Symposium on Theory of Computing*, pp. 694–700.

Gupta, A., Newman, I., Rabinovich, Y., and Sinclair, A. 1999. Cuts, trees and l_1 embeddings. In *Proceedings of Symposium on Foundations of Computer Scince*.

Indyk, P. 2001. Algorithmic aspects of geometric embeddings. In *Proceedings, 42nd Annual Symposium on Foundations of Computer Science*.

Grauman, K.K. and Darrell, T. 2004. Fast contour matching using approximate earth mover's distance. In *Proc. IEEE Conf. on Comp. Vision and Pattern Recognition*, IEEE Computer Society, pp. 220–227.

Keselman, Y. and Dickinson, S. 2005. Generic model abstraction from examples. *IEEE PAMI*, 27(7) .

Keselman, Y., Shokoufandeh, A., Demirci, F. and Dickinson, S. 2003. Many-to-many graph matching via low-distortion embedding. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, Madison, WI.

Kosinov, S. and Caelli, T. 2002. Inexact multisubgraph matching using graph eigenspace and clustering models. In *Proceedings of SSPR/SPR*, Springer, vol. 2396, pp. 133–142.

Leibe, B. and Schiele, B. 2003. Analyzing appearance and contour based methods for object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Madison, WI.

Lindeberg, T. 1998a. Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30:117–154.

Lindeberg, T. 1998b. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:77–116.

Liu, T.-L. and Geiger, D. 1999. Approximate tree matching and shape similarity. In *Proceedings, 7th International Conference on Computer Vision*, Kerkyra, Greece, pp. 456–462.

Matoušek, J. 1999. On embedding trees into uniformly convex Banach spaces. *Israel Journal of Mathematics*, 237:221–237.

Messmer, B. and Bunke, H. 1995. Efficient error-tolerant subgraph isomorphism detection. In D. Dori and A. Bruckstein (Eds.), *Shape, Structure and Pattern Recognition*, World Scientific Publ. Co., pp. 231–240.

Murase, H. and Nayar, S. 1995. Visual learning and recognition of 3-D objects from appearance. *International Journal of Computer Vision*, 14:5–24.

Myers, R., Wilson, R., and Hancock, E. 2000. Bayesian graph edit distance. *IEEE PAMI*, 22(6):628–635.

Pelillo, M., Siddiqi, K. and Zucker, S. 1999. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1105–1120.

Rubner, Y., Tomasi, C., and Guibas, L.J. 2000. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.

Scott, G. and Longuet-Higgins, H. 1991. An algorithm for associating the features of two patterns. In *Proceedings of Royal Society of London*, B244:21–26.

Sebastian, T., Klein, P., and Kimia, B. 2001. Recognition of shapes by editing shock graphs. In *IEEE International Conference on Computer Vision*, pp. 755–762.

Sebastian, T., Klein, P.N., and Kimia, B. 2004. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):550–571.

Shokoufandeh, A., Dickinson, S.J., Jönsson, C., Bretzner, L., and Lindeberg, T. 2002. On the representation and matching of qualitative shape at multiple scales. In *Proceedings, 7th European Conference on Computer Vision*, vol. 3, pp. 759–775.

Shokoufandeh, A., Macrini, D., Dickinson, S., Siddiqi, K., and Zucker, S. 2005. Indexing hierarchical structures using graph spectra. *IEEE PAMI*, 27(7).

Siddiqi, K., Shokoufandeh, A., Dickinson, S., and Zucker, S. 1999. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24.