Many-to-Many Feature Matching Using Spherical Coding of Directed Graphs

M. Fatih Demirci[1], Ali Shokoufandeh[1], Sven Dickinson[2], Yakov Keselman[3], and Lars Bretzner[4]

[1] Department of Computer Science, Drexel University, Philadelphia, PA 19104, USA
{mdemirci,ashokouf}@mcs.drexel.edu

[2] Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M5S 3G4
sven@cs.toronto.edu

[3] School of Computer Science, Telecommunications and Information Systems, DePaul University, Chicago, IL 60604, USA
ykeselman@cs.depaul.edu

[4] Computational Vision and Active Perception Laboatory, Department Of Numerical Analysis and Computer Science, KTH, Stockholm, Sweden
bretzner@nada.kth.se

Abstract. In recent work, we presented a framework for many-to-many matching of multi-scale feature hierarchies, in which features and their relations were captured in a vertex-labeled, edge-weighted directed graph. The algorithm was based on a metric-tree representation of labeled graphs and their metric embedding into normed vector spaces, using the embedding algorithm of Matoušek [13]. However, the method was limited by the fact that two graphs to be matched were typically embedded into vector spaces with different dimensionality. Before the embeddings could be matched, a dimensionality reduction technique (PCA) was required, which was both costly and prone to error. In this paper, we introduce a more efficient embedding procedure based on a spherical coding of directed graphs. The advantage of this novel embedding technique is that it prescribes a single vector space into which both graphs are embedded. This reduces the problem of directed graph matching to the problem of geometric point matching, for which efficient many-to-many matching algorithms exist, such as the Earth Mover's Distance. We apply the approach to the problem of multi-scale, view-based object recognition, in which an image is decomposed into a set of blobs and ridges with automatic scale selection.

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 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 Figure 1, in which an object is decomposed into a set of ridges and blobs extracted at appropriate scales [19]. The ridges and blobs map to nodes in a directed graph, with parent/child edges directed from coarser scale nodes to overlapping finer scale nodes, and sibling edges between nodes that share a parent. 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 oversegmented 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.

In recent work [10, 7], we presented a framework for many-to-many matching of undirected graphs and directed graphs, respectively, where features and their relations were represented using edge-weighted graphs. The method began with transforming a graph into a metric tree. Next, using the graph embedding technique of Matoušek [13], 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.

The previous procedure suffered from a significant limitation. Namely, each graph was embedded into a vector space of arbitrary dimensions, and before the embeddings could be matched, a dimensionality reduction step was required, which was both costly and prone to error. Specifically, we used an inefficient Principal Components Analysis (PCA)-based method to project the two distributions into the same normed space. In this paper, we present an entirely different embedding method based on a spherical coding algorithm. This efficient

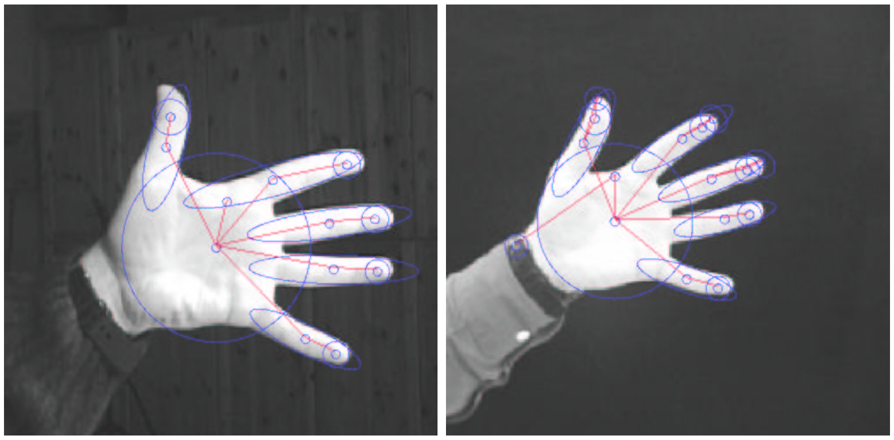

Fig. 1. The Need for Many-to-Many Matching. In the two images, the two objects are similar, but the extracted features are not necessarily one-to-one. Specifically, the ends of the fingers in the left hand have been over-segmented in the right hand.

(linear-time) method embeds metric trees into vector spaces of prescribed dimensionality, precluding the need for a dimensionality reduction step. We demonstrate the framework on the problem of multi-scale shape matching, in which an image is decomposed into a set of blobs and ridges with automatic scale selection.

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., [14, 12, 15, 18]). 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 (p-time algorithms are available only for trees); 2) the method, in its current form, does not accommodate edge weights; 3) the method does not deal well with occlusion and scene clutter, resulting in much effort spent in “editing out” extraneous graph structure; and 4) 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).

In the context of line and segment matching, Beveridge and Riseman [3] addressed this problem via exhaustive local search. Although their method found good matches reliably and efficiently (due to their choice of the objective function and a small neighborhood size), it is unclear how the approach can be generalized to other types of feature graphs and objective functions.

In a novel generalization of Scott and Longuet-Higgins [17], Kosinov and Caelli [11] showed how inexact graph matching could be solved using the renormalization of projections of vertices into the eigenspaces of graphs combined with a form of relational clustering. Our framework differs from their approach in that: (1) it can handle information encoded in a graph's nodes, which is desirable in many vision applications; (2) it does not require an explicit clustering step; (3) it provides a well-bounded, low-distortion metric representation of graph structure; (4) it encodes both local and global structure, allowing it to deal with noise and occlusion; and 5) it can accommodate multi-scale representations.

Low-distortion embedding techniques haven proven to be useful in a number of graph algotithms, including clustering and, most recently, on-line algorithms. Indyk [9] provides a comprehensive survey of recent advances and applications of low-distortion graph embedding. Gupta [8] proposes a randomized procedure for embedding metric trees into a vector space of prescribed dimensions. Our spherical coding is a deterministic variation of this procedure. For recent results related to the properties of low-distortion tree embedding, see [1, 13].

3 Notation and Definitions

Before describing our many-to-many matching framework, some definitions are in order. To begin, a *graph* G is a pair $(\mathcal{A}, E)$, where $\mathcal{A}$ is a finite set of vertices and E is a set of connections (edges) between the vertices. An edge $e = (u, v)$ consists of two vertices such that $u, v \in \mathcal{A}$. A graph $G = (\mathcal{A}, E)$ is *edge-weighted*, if each edge $e \in E$ has a weight, $\mathcal{W}(e) \in \mathbb{R}$. Let $G = (\mathcal{A}, E)$ denote an edge-weighted graph with real edge weights $\mathcal{W}(e)$, $e \in E$. We will say that $\mathcal{D}$ is a *metric* for G if, for any three vertices $u, v, w \in \mathcal{A}$, $\mathcal{D}(u, v) = \mathcal{D}(v, u) \geq 0$, with $\mathcal{D}(u, v) = 0$ if and only if $u = v$, and $\mathcal{D}(u, v) \leq \mathcal{D}(u, w) + \mathcal{D}(w, v)$.

One way of defining metric distances on a weighted graph is to use the *shortest-path* metric $\delta(., .)$ on the graph or its subgraphs, i.e., $\mathcal{D}(u, v) = \delta(u, v)$, the shortest path distance between u and v for all $u, v \in \mathcal{A}$. We will say that the edge weighted tree $\mathfrak{T} = \mathfrak{T}_G(\mathcal{A}', E')$ is a *tree metric* for G, with respect to distance function $\mathcal{D}$, if for any pair of vertices u, v in G, the length of the unique path between them in $\mathfrak{T}$ is equal to $\mathcal{D}(u, v)$.

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 tree metric distance. To create a general tree metric from an ultra-metric, we need to satisfy the *4-point* condition (see [4]): $\mathcal{D}[x, y] + \mathcal{D}[z, w] \leq \max\{\mathcal{D}[x, z] + \mathcal{D}[y, w], \mathcal{D}[x, w] + \mathcal{D}[y, z]\}$, for all x, y, z, w. A metric that satisfies the 4-point condition is called an *additive metric*.

A *metric embedding* is a mapping $f : \mathcal{A} \rightarrow \mathcal{B}$, where $\mathcal{A}$ 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 $||.||_k$, and for any pair $p, q \in \mathcal{A}$ we have

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

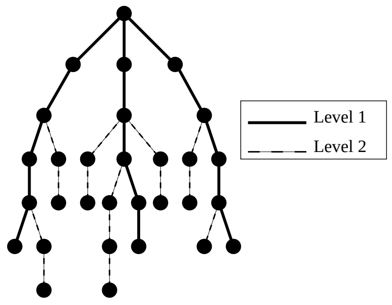

Fig. 2. Path partition of an example tree. The three level 1 paths are bold, while the seven level 2 paths are dashed (there are no other levels in this particular case - see text).

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

To capture the topological structure of a tree, we use the concept of *caterpillar decomposition* and caterpillar dimension. We illustrate the caterpillar decomposition of a rooted tree with no edge weights in Figure 2. 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 m.

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 $v \in \mathcal{A}$. The first segment of $P(v)$ of weight l_1 follows some path P^1 of level 1 in $\mathfrak{P}$, the second segment of weight l_2 follows a path P^2 of level 2, and the last segment of weight l_α follows a path P^α of level $\alpha \leq m$. The sequences $\langle P^1, \ldots, P^\alpha \rangle$ and $\langle l_1, \ldots, l_\alpha \rangle$ will be referred to as the *decomposition sequence* and the *weight sequence* of $P(v)$, respectively.

Finally, we introduce the notion of *spherical codes* in our embedding procedure. A spherical code is the distribution of a finite set of n points on the surface of a unit sphere such that the minimum distance between any pair of points is maximized [6]. Equivalently, one can try to minimize the radius r of a d-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.

4 Metric Embedding of Graphs via Spherical Coding

4.1 Problem Formulation

Our interest in low-distortion embedding is motivated by its ability to transform the problem of many-to-many matching in finite graphs to the problem of geometric point matching in low-dimensional vector spaces. For graphs, the problem of low-distortion embedding is a challenging one. Let $G_1 = (\mathcal{A}_1, E_1, \mathcal{D}_1)$, $G_2 = (\mathcal{A}_2, E_2, \mathcal{D}_2)$ denote two graphs on vertex sets $\mathcal{A}_1$ and $\mathcal{A}_2$, edge sets E_1 and E_2, under distance metrics $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively ($\mathcal{D}_i$ represents the distances between all pairs of nodes in G_i). Ideally, we seek a single embedding mechanism that can map each graph to the same vector space, in which the two embeddings can be directly compared.

We will tackle the problem in two steps. Given a d-dimension target space $\mathbb{R}^d$, we will seek low-distortion embeddings f_i that map sets $\mathcal{A}_i$ to sets $\mathcal{B}_i$ under distance function $||.||_k$, $i \in \{1, 2\}$. The fixed-dimension embedding is based on a novel spherical coding of the shortest-path metric on a tree. To apply this embedding to our directed acyclic graphs therefore requires that we map them to trees with low distortion. It is here that we introduce the concept of relative scale to the points, allowing us to match hierarchical graphs. Using these mappings, the problem of many-to-many hierarchical vertex matching between G_1 and G_2 is reduced to that of computing a mapping $\mathcal{M}$ between subsets of $\mathcal{B}_1$ and $\mathcal{B}_2$.

It is known that a minimum-distortion embedding of a metric tree into the d-dimensional Euclidean space will have distortion of $O(L^{\frac{1}{d-1}}\sqrt{\min(\log L, d)})$, where L is the number of leaves in the tree [8]. Observe that as the dimension d of the target space decreases, the distortion of the embedding increases. We would therefore like to strike a good balance between distortion and dimension.

4.2 Construction of a Tree Metric for a Distance Function

Let $G = (\mathcal{A}, E)$ denote an edge-weighted graph and $\mathcal{D}$ denote a shortest-path metric for G, i.e., $\mathcal{D}(u, v) = \delta(u, v)$, for all $u, v \in \mathcal{A}$. 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. To create a general tree metric from an ultra-metric, we need to satisfy the *4-point* condition. Observe that a metric $\mathcal{D}$ is additive if and only if it is a tree metric (see [4]). Therefore, our construction of a tree metric will consist of: 1) constructing an ultra-metric from $\mathcal{D}$, and 2) modifying the ultra-metric to satisfy the 4-point condition. For details of one such approximation framework, see Agarwala et al. [1]. The construction of a tree metric in their algorithm is achieved by transforming the general tree metric problem to that of ultra-metrics. Their algorithm, which follows the two-step procedure outlined above, generates an approximation (tree

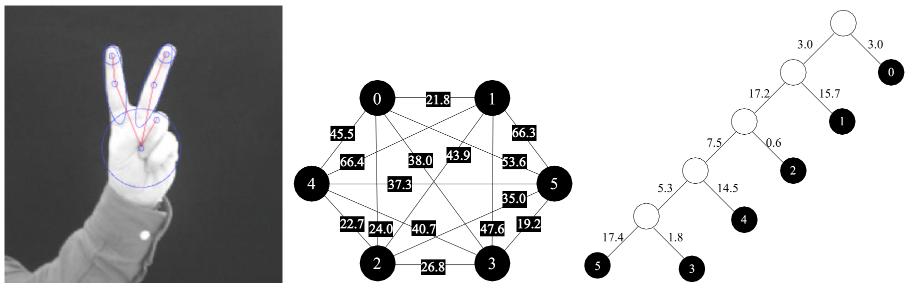

Fig. 3. Metric tree representation of the Euclidean distances between nodes in a graph. The gesture image (a) consists of 6 regions (the region representing the entire hand is not shown). The complete graph in (b) captures the Euclidean distances between the centroids of the regions, while (c) is the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

metric $\mathfrak{T}$) to an optimal additive metric in time $O(n^2)$. It should be noted that this construction does not necessarily maintain the vertex set of G invariant. We will have to make sure that in the embedding process (see Section 4), the extra vertices generated during the metric tree construction are eliminated. An example of constructing a metric tree from a graph is shown Figure 3.

4.3 Construction of Spherical Codes

To embed our metric trees into Euclidean spaces of fixed dimension, we introduce the concept of *spherical codes*. Such codes, in turn, will allow us to directly compare two embeddings. The embedding framework is best illustrated through an example, in which a weighted tree is embedded into $\mathbb{R}^2$, as shown in Figure 4. 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 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 subcones C_b, C_e, and C_f will correspond to the 4 directions defining the embedding. Finally, to complete the embedding, we translate the subcones away from the origin along their directional rays to positions defined by the path lengths in the tree. For

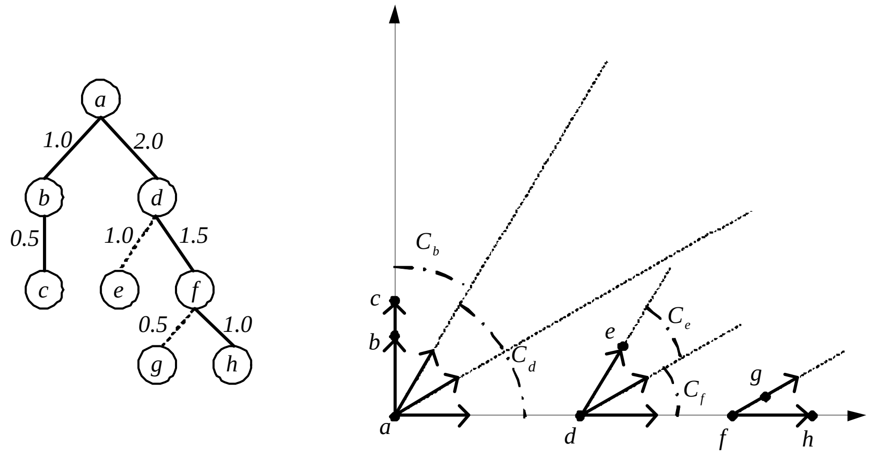

Fig. 4. An edge weighted tree and its spherical code in 2D. The Cartesian coordinates of the vertices are: $a = (0, 0)$, $b = (0, 1.0)$, $c = (0, 1.5)$, $d = (2.0, 0)$, $e = (2.5, 0.87)$, $f = (3.5, 0)$, $g = (3.93, 0.25)$, and $h = (4.5, 0)$.

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 $f : \mathcal{A} \rightarrow \mathcal{B}$ under $||.||_2$ 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 [6]. In Figure 4, 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.

4.4 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 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 [20], directed geometric structure in [19], and shape context in [2]. 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.

Specifically, let $G = (\mathcal{A}, E)$ be a graph to be embedded. 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 R between u, v. For a vertex $u \in \mathcal{A}$, we let $N(u)$ denote the set of vertices $v \in \mathcal{A}$ 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 $v \in N(u)$. Feature 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 [19], 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 [16] 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 [10].

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 [5] 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 [10, 7].

5.1 The Final Algorithm

Our algorithm for many-to-many 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 4 (see [1] for details).
2: Construct low-distortion embeddings $f_i(\mathfrak{T}_i)$ of $\mathfrak{T}_i$ into $(\mathcal{B}_i, ||.||_2)$ according to Section 4.
3: Compute the EMD between $\mathcal{E}_i$'s by applying the FT iteration, computing the optimal transformation $T$ according to Section 5 (see [10] for details).
4: Interpret the resulting optimal flow between $\mathcal{E}_i$'s as a many-to-many vertex matching between $G_i$'s.
```

6 Experiments

As an illustration of our approach, let's first return to the example shown in Figure 1, where we observed the need for many-to-many matching. The results of applying our method to the these two images is shown in Figure 5, 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.

To provide a more comprehensive evaluation, we tested our framework on two separate image libraries, the Columbia University COIL-20 (20 objects, 72 views per object) and the ETH Zurich ETH-80 (8 categories, 10 exemplars per

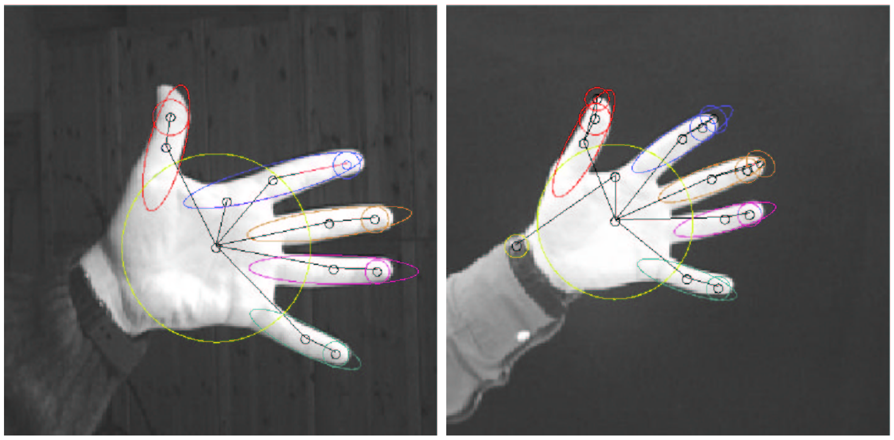

Fig. 5. Applying our algorithm to the images in Figure 1. Many-to-many feature correspondences have been colored the same.

category, 41 views per exemplar)[5]. For each view, we compute a multi-scale blob decomposition, using the algorithm described in [19]. 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. Finally, each tree is 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 either $v_{i+1,j}$ or $v_{i-1,j}$ as the closest view. We will classify this as a correct matching. Based on the overall matching statistics, we observe that 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 to the many-to-many

[5] Arguably, the COIL database is not the ideal testbed for an image representation (in our case, a multi-scale blob and ridge 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.

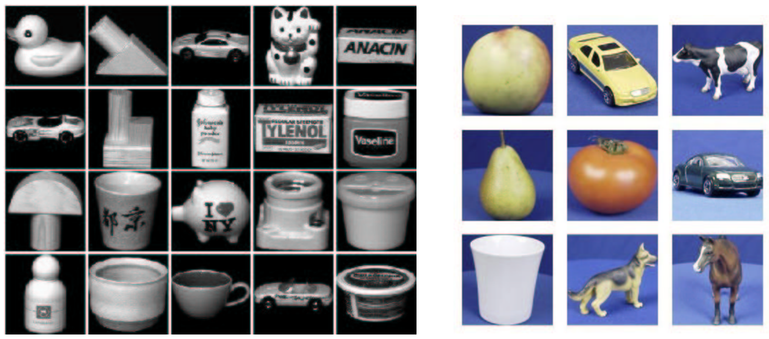

Fig. 6. Views of sample objects from the Columbia University Image Library (COIL-20) and the ETH Zurich (ETH-80) Image Set.

matching algorithm based on PCA embedding [7] for a similar setup, the new procedure showed an improvement of 5.5%.

It should be pointed out that these results can be considered worst case for two reasons. First, the original 72 views per object sampling resolution was tuned for an eigenimage approach. 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, 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. Many of the objects in the database are rotationally symmetric, yielding identical views from each viewpoint.

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 was similar to the COIL-20 experiment. Our experiments showed that in all but 6.2% of the experiments, 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. Again, these results can be considered worst case for the same reasons discussed above for the COIL-20 experiment.

Both the embedding and matching procedures can accommodate local perturbation, due to noise and occlusion, because path partitions provide locality. If a portion of the graph is corrupted, the projections of unperturbed nodes will not be affected. Moreover, the matching procedure is an iterative process driven by flow optimization which, in turn, depends only on local features, and is thereby

Perturbation	5%	10%	15%	20%
Recognition rate	91.07%	88.13%	83.68%	77.72%

Table 1. Recognition rate as a function of increasing perturbation. Note that the baseline recognition rate (with no perturbation) is 95.2%

unaffected by local perturbation. To demonstrate the framework's robustness, we performed four perturbation experiments on the COIL-20 database. The experiments are identical to the COIL-20 experiment above, except that the query graph was perturbed by adding/deleting 5%, 10%, 15%, and 20% of its nodes (and their adjoining edges). The results are shown in Table 1, and reveal that the error rate increases gracefully as a function of increased perturbation.

7 Conclusions

We have presented a novel, computationally efficient approach to many-to-many matching of directed graphs. To match two graphs, we begin by constructing metric tree representations of the graphs. Next, we embed them in a geometric space with low distortion using a novel encoding of the graph's vertices, called spherical codes. 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 structure. We evaluate the technique on two major image databases, using a multi-scale image representation that captures coarse image structure, and include a set of structural perturbation experiments to show the algorithm's robustness to graph "noise".

8 Acknowledgment

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.

References

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

2. S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE PAMI*, 24(4):509–522, April 2002.
3. R. Beveridge and E. M. Riseman. How easy is matching 2D line models using local search? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):564–579, June 1997.
4. P. Buneman. The recovery of trees from measures of dissimilarity. In F. Hodson, D. Kendall, and P. Tautu, editors, *Mathematics in the Archaeological and Historical Sciences*, pages 387–395. Edinburgh University Press, Edinburgh, 1971.
5. S. D. Cohen and L. J. Guibas. The earth mover's distance under transformation sets. In *Proceedings, 7th International Conference on Computer Vision*, pages 1076–1083, Kerkyra, Greece, 1999.
6. J. H. Conway and N. J. A. Sloane. *Sphere Packing, Lattices and Groups*. Springer-Verlag, New York, 1998.
7. F. Demirci, A. Shokoufandeh, Y. Keselman, S. Dickinson, and L. Bretzner. Many-to-many matching of scale-space feature hierarchies using metric embedding. In *Scale Space Methods in Computer Vision, 4th International Conference*, pages 17–32, Isle of Skye, UK, June, 10–12 2003.
8. A. Gupta. Embedding tree metrics into low dimensional euclidean spaces. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 694–700, 1999.
9. P. Indyk. Algorithmic aspects of geometric embeddings. In *Proceedings, 42nd Annual Symposium on Foundations of Computer Science*, 2001.
10. Y. Keselman, A. Shokoufandeh, F. Demirci, and S. Dickinson. Many-to-many graph matching via low-distortion embedding. In *Proceedings, IEEE Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003.
11. S. Kosinov and T. Caelli. Inexact multisubgraph matching using graph eigenspace and clustering models. In *Proceedings of SSPR/SPR*, volume 2396, pages 133–142. Springer, 2002.
12. T.-L. Liu and D. Geiger. Approximate tree matching and shape similarity. In *Proceedings, 7th International Conference on Computer Vision*, pages 456–462, Kerkyra, Greece, 1999.
13. J. Matoušek. On embedding trees into uniformly convex Banach spaces. *Israel Journal of Mathematics*, 237:221–237, 1999.
14. B. Messmer and H. Bunke. Efficient error-tolerant subgraph isomorphism detection. In D. Dori and A. Bruckstein, editors, *Shape, Structure and Pattern Recognition*, pages 231–240. World Scientific Publ. Co., 1995.
15. R. Myers, R. Wilson, and E. Hancock. Bayesian graph edit distance. *IEEE PAMI*, 22(6):628–635, 2000.
16. Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.
17. G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two patterns. *Proceedings of Royal Society of London*, B244:21–26, 1991.
18. T. Sebastian, P. Klein, and B. Kimia. Recognition of shapes by editing shock graphs. In *IEEE International Conference on Computer Vision*, pages 755–762, 2001.
19. A. Shokoufandeh, S.J. Dickinson, C. Jönsson, L. Bretzner, and T. Lindeberg. On the representation and matching of qualitative shape at multiple scales. In *Proceedings, 7th European Conference on Computer Vision*, volume 3, pages 759–775, 2002.
20. K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.