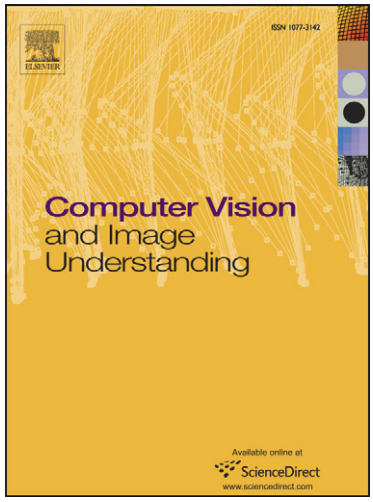


Object Categorization using Bone Graphs

Diego Macrini, Sven Dickinson, David Fleet, Kaleem Siddiqi

PII: S1077-3142(11)00090-7
DOI: 10.1016/j.cviu.2011.03.002
Reference: YCVIU 1760

To appear in: *Computer Vision and Image Understanding*

Received Date: 1 November 2010
Revised Date: 1 March 2011
Accepted Date: 15 March 2011

Please cite this article as: D. Macrini, S. Dickinson, D. Fleet, K. Siddiqi, Object Categorization using Bone Graphs, *Computer Vision and Image Understanding* (2011), doi: 10.1016/j.cviu.2011.03.002

Object Categorization using Bone Graphs

Diego Macrini, Sven Dickinson, David Fleet, Kaleem Siddiqi

Abstract

The bone graph [23, 25] is a graph-based medial shape abstraction that offers improved stability over shock graphs and other skeleton-based descriptions that retain unstable ligature structure. Unlike the shock graph, the bone graph's edges are attributed, allowing a richer specification of relational information, including how and where two medial parts meet. In this paper, we propose a novel shape matching algorithm that exploits this relational information. Formulating the problem as an inexact directed acyclic graph matching problem, we extend a leading bipartite graph-based algorithm for matching shock graphs [41]. In addition to accommodating the relational information, our new algorithm is better able to enforce hierarchical and sibling constraints between nodes, resulting in a more general and more powerful matching algorithm. We evaluate our algorithm with respect to a competing shock graph-based matching algorithm, and show that for the task of view-based object categorization, our algorithm applied to bone graphs outperforms the competing algorithm. Moreover, our algorithm applied to shock graphs also outperforms the competing shock graph matching algorithm, demonstrating the generality and improved performance of our matching algorithm.

Key words: Medial Shape Representation, Graph-Based Shape Representation, Inexact Graph Matching, Object Categorization

1. Introduction

The recognition of 3-D objects from their silhouettes demands a shape representation which is invariant to minor changes in viewpoint and articulation. This invariance can be achieved by parsing a silhouette into parts and relationships that are stable across similar object views. Medial descriptions, such as skeletons and shock graphs, attempt to decompose a shape into

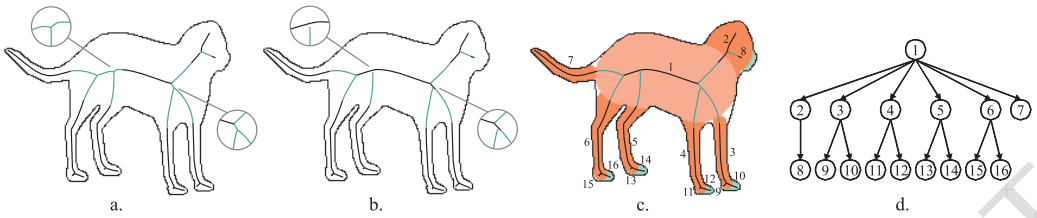


Figure 1: Computing a bone graph from an unstable skeleton is a two-pass procedure that's based on identifying and rectifying the ligature-induced instability in a shape's medial axis: (a) skeletal instability arises from part oversegmentation and undersegmentation. For example, the medial axis of the dog's body is given by two skeletal branches (instead of one) due to the junction point that represents the connection between these branches and the skeletal segment extending from the shorter rear leg. A similar situation occurs near the front legs. The vicinity of the part oversegmentation is enlarged in each case, showing the resulting perturbation of the skeleton. Those skeletal segments shown in green are ligature regions, and they contribute little to the shape of the object. A purely local analysis of ligature is problematic in the presence of such oversegmentation, as illustrated by the non-intuitive labeling of the body part in the vicinity of the oversegmentation as ligature. (b) Detecting and removing ligature-induced skeletal instability uses a local ligature analysis to first identify and rectify the medial representations of part protrusions. (c) A second ligature analysis then yields a set of salient parts, called *bones* (shown in black). The bones capture the coarse part structure of the object, as indicated by the colored parts reconstructed from the bones. (d) The bones give rise to a *bone graph*, an intuitive and stable representation whose nodes represent the salient parts and whose edges, derived from the final ligature analysis, capture part attachment.

parts, but suffer from instabilities that lead to similar shapes being represented by dissimilar part sets. In response to this ligature-induced instability, we recently introduced the *bone graph* [23, 25], a medial parts-based shape decomposition based on identifying and regularizing the ligature structure of a given medial axis, and capturing a more intuitive notion of an object's parts than a skeleton or a shock graph. An example of the bone graph computed from an input silhouette is shown in Figure 1.

As shown in Figure 1, the bone graph forms a hierarchy of parts with edges spanning adjacent parts. Like the shock graph [34, 41], a bone graph node encodes the geometry of a part. However, unlike the shock graph, a bone graph edge is attributed, specifying the attachment position (on each part) and the orientation of the attachment. Therefore, the view-based recognition of 3-D objects using bone graphs requires a matching algorithm that can compare the attributes of nodes and edges as well as the structures of two

graphs in the presence of spurious and missing nodes. Figure 2 illustrates an example of similar views of two horses, which are represented by similar, but not isomorphic, bone graphs. One important difference between these two graphs is that in the graph on the right, the front legs are connected directly to the body of the horse, while in the graph on the left, there is a small shape part between the legs and the body. This small part is not noise, as it represents a true part between the legs and the torso. However, there is no natural corresponding part on the other shape, and the algorithm must be able to leave it unassigned, yet still establish correspondences between the front legs. Figure 3 shows the node correspondences found by the algorithm introduced in this paper.

The comparison of bone graphs defines an inexact graph matching between DAGs with high-dimensional node and edge attributes. There is a large body of work in the area of inexact graph matching, but most of it is focused on graphs in which either the nodes or the edges are attributed. Our experience with the inexact graph matching algorithm proposed by Shokoufandeh et al. [24, 37, 38, 41, 44] for DAGs with attributed nodes motivates our extension of this approach in order to incorporate edge information into the matching problem. We propose a generalization of this algorithm that expands the range of constraints that can be accounted for at matching time, leading to a general framework for representing domain knowledge about the relevance of structural differences between the graphs. Because the matching problem is intractable, the addition of domain knowledge is especially important to guide the search for approximate solutions that are relevant to the task at hand.

We evaluate our graph matching framework for the problem of comparing bone graphs. This requires the definition of similarity functions for nodes and edges, and the specification of the domain assumptions about the relative importance of the structural differences between two graphs. Our proposed node similarity function is a robust measure that can gracefully account for the deformations produced by perspective transformations, part articulation, and within-class deformation. Our proposed edge similarity function is able to measure the position and orientation of each part attachment in terms of its local context. The result is a recognition pipeline based on a novel 2-D medial axis-based shape representation. We evaluate each component of this pipeline extensively, and compare it to a competing shock graph recognition framework.

The contributions of the paper are twofold: 1) We introduce a novel

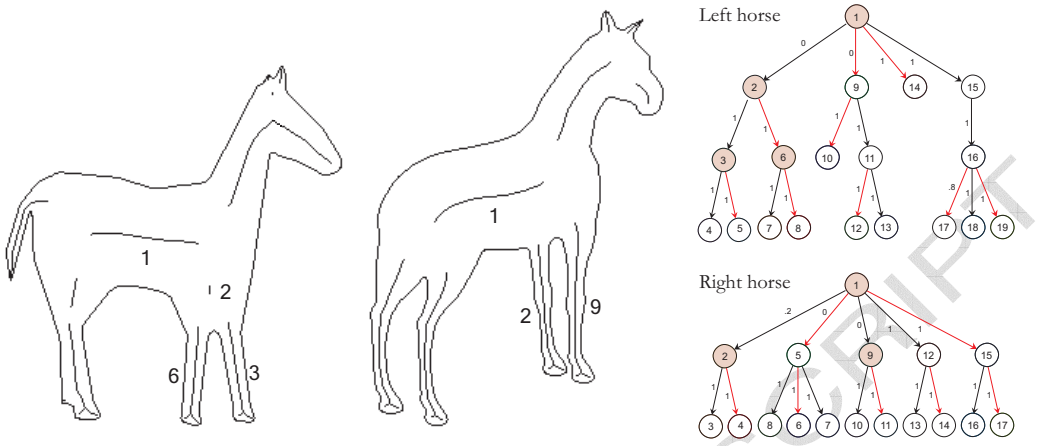


Figure 2: Example of missing non-terminal nodes and natural violations to node adjacency relations in the bone graph. We show the bone graph representations of two different horse exemplars as seen from similar viewpoints (only the non-ligature points of the medial axes are drawn). In this example, the left horse has *internal* shape parts between its torso and its front/back legs and between its neck and ears that have no natural correspondence on the right horse. The representation of these parts in the graph leads to differences in the parent-child relations of some nodes. For example, the torso (node 1) is the *grandparent* of the front legs (nodes 3 and 6) on the top graph, whereas on the bottom graph, the torso is the parent of the front legs (nodes 2 and 9). In order to find the desired correspondences between the legs and ears, the graph similarity function that drives the matching process must favor solutions in which the missing parts are left unassigned. Natural violations to node adjacency relations pose another challenge to bone graph matching. In this example, the enforcement of node adjacencies would exclude a node correspondence solution in which the torsos and front legs of the horses are assigned to one another. This is because the torso of the left horse (node 1) *is not* adjacent to the front legs (nodes 6 and 3) due to the presence of a shape part between them (node 2), but the torso of the right horse *is* adjacent to the horse's front legs.

framework for inexact graph matching, extending our body of previous work on matching node-attributed graphs [24, 37, 38, 41] to accommodate node- and edge-attributed graphs, and expanding the range of domain-dependent topological constraints that can be enforced during matching; and 2) We apply the matching framework to a powerful new structured shape representation, the bone graph [23], yielding the first end-to-end object categorization system based on bone graphs. The improved stability of the node- and edge-attributed bone graph, combined with our matching algorithm's ability to exploit both its edge attributes and its domain-dependent topological constraints, yield an object categorization system which outperforms a

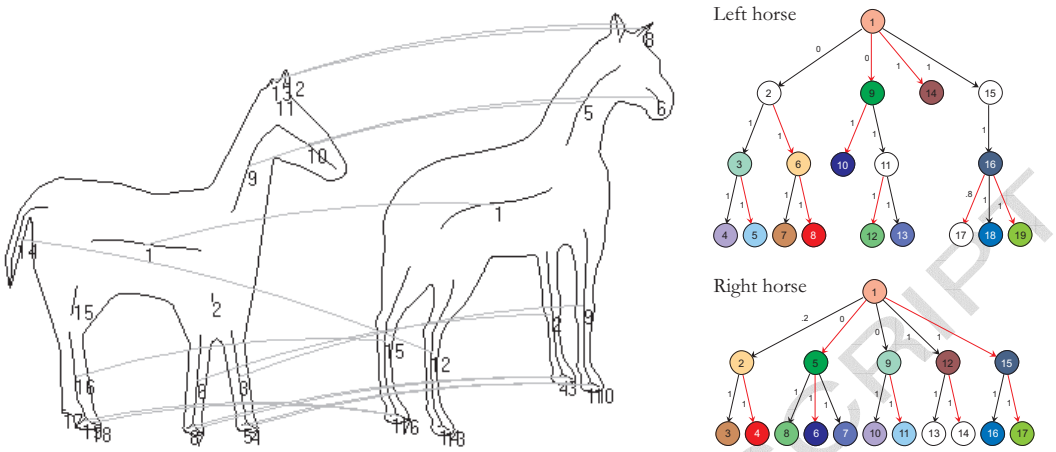


Figure 3: Matching result for the horse example. The correspondences found by the algorithm are shown as arcs connecting shape parts, and as matching node colors in the graphs. In this case, the four internal shape parts (labeled 2, 11, 15, and 17) in between the left horse's torso and legs, and neck and ears are left unassigned (white nodes) at matching time.

state-of-the-art shape categorization system based on shock graphs.

The remainder of the paper is organized as follows. In Section 2, we review other approaches to graph-based shape matching. In Section 3, we present our approach to the inexact matching of hierarchical graph structures, incorporating both node and edge constraints. We evaluate our matching framework in Section 4), by applying it to the task of view-based object categorization using bone graphs. Finally, we discuss limitations and conclusions in Section 5.

2. Related Work

The problem of matching graph representations of shape has received considerable attention in the vision research community. The graph matching task demands an algorithm that combines differences in graph topology and node and edge attributes into a continuous measure of graph similarity. Since this problem (subgraph isomorphism) is NP-complete (maximum common isomorphic subgraph is NP-hard), the choice is either to seek an approximate solution or to consider graphs that can be matched in polynomial time (e.g., rooted trees). In addition, domain-dependent constraints may be assumed to simplify the matching problem. For example, Umeyama [43] proposes a

matching algorithm for graphs of *equal size* using an eigen-decomposition method to find the permutation matrix that minimizes the difference of the edge weights. Another example is the work of Gold and Rangarajan [16], who use a graduated assignment heuristic to find the *largest isomorphic subgraph* between two attributed graphs. Unfortunately, these specific constraints are too limiting for the task of matching shape representations, since the graphs representing similar shapes might not have the same number of nodes or large isomorphic subgraphs.

Conte et al. survey the field of exact and inexact graph matching in pattern recognition in [6] and [7]. In this section, we review the body of work in the area that is most relevant to the problem of matching graph-based shape representations. The most popular classes of graphs in this area are given by trees, directed acyclic graphs (DAG), and attributed relational graphs (ARG) [12]. The attributed tree- and DAG-based representations encode hierarchical relations between nodes (i.e., shape parts), which, in turn, provide structural constraints that simplify the matching problem. On the other hand, an ARG is a more general structure than a tree or a DAG (it is an attributed undirected graph that might contain cycles), but provides weaker constraints to exploit during matching.

Pelillo et al. [29, 30] match node-attributed trees by constructing a maximum weight clique problem. This algorithm looks for the set of nodes in the query and in the model graphs that preserve the hierarchical constraints imposed by the representation while maximizing the pairwise node similarities among the nodes. Another matching approach for node-attributed trees is the work of Sebastian et al. [35], which measures the edit distance between two graphs (this is also known in the literature as error-correcting or error-tolerant algorithms [6]). The graph edit distance is defined as the minimum cost of the deformation path that makes two graphs isomorphic. Four edit operations are defined to deform a graph representation of one shape into another. Three of these operations allow for different types of merges and deletions of nodes, while the fourth operation allows for altering node attributes. The merge operation allows for assigning many-to-many correspondences between nodes (or one-to-many, if merges are applied only to the query graph) and can be used to find similarities at higher levels of abstraction. However, this flexibility comes at high computational cost, as finding a common graph between largely dissimilar graphs can define an extremely large space of possible edit operations. This is particularly problematic for cluttered or occluded scenes, where much of the edit cost may be due to the

removal of extraneous clutter.

Pelillo et al. [31] also propose a solution to the many-to-many matching of node-attributed trees by reducing tree isomorphism to the problem of solving a maximum weight clique in an association graph. Their solution to the matching problem uses replicator dynamical systems from evolutionary game theory. In more recent work, Demirci et al. [10] present a framework for many-to-many matching, where features and their relations are represented using directed edge-weighted graphs. The method begins by transforming the graph into a metric tree. Next, using graph embedding techniques, the tree is embedded into a normed vector space. This two-step transformation reduces the problem of many-to-many graph matching to a simpler problem of matching weighted distributions of points in a normed vector space. The distance between two weighted point distributions is computed using the Earth Mover's distance [5, 32].

For the case of node-attributed DAGs, Shokoufandeh et al. [37, 38] propose a recursive bipartite matching algorithm in which a node similarity function measures both the attribute similarity of nodes and the topological similarity of the subgraphs rooted at each node. The result is a one-to-one assignment of node correspondences that yields a similarity value between arbitrary DAGs. This approach has been used to match shock graphs [38] as well as multiscale blob decompositions of shape [37]. In this latter approach, edge attributes are not accounted for explicitly, but can be embedded in the node similarity function. For example, in [37] the relative orientation of parts (i.e., edge attributes) is seen as a parameter of the node similarity function. However, since edge attributes are not compared explicitly, the ability of the matcher to independently measure structural and geometrical similarity is reduced.

All the above matching approaches suffer from one important limitation; they either ignore edge attributes or allow only for scalar edge weights. A popular method for matching fully-attributed graphs is to treat the problem as a tree search with backtracking [18]. The basic mechanism in this family of approaches is to iteratively grow a solution set (initially empty) by adding node correspondences that are compatible with the mappings already in the set. The search is usually guided by the cost (or similarity weight) of the partial mapping spanned by the solution set and a heuristic that estimates the cost of matching the remaining nodes. The heuristic allows the matching algorithm to prune unfruitful search paths (and backtrack the population of the solution set), and/or to determine the order in which the search tree is

traversed. The search is usually performed using a depth-first or best-first strategy.

Our algorithm for matching fully-attributed DAGs is a tree search approach, and in that respect it is similar to the previous work on ARGs. However, it differs from such work in the way it treats noisy nodes and edge attributes. In the case of ARG-based methods, noisy nodes are matched (with a cost) to a special null node that represents the node deletion operation in a graph edit distance. The addition of null nodes (one for each graph) increases considerably the number of possible solutions that must be investigated. In our approach, the hierarchical structure of DAGs is used to constrain the node correspondences at each iteration of the algorithm, which allows for solutions with unmatched noisy nodes without adding null nodes. Another important difference is that the ARG matching approaches seek a mapping that preserves node adjacency by editing the graphs to create an isomorphism, while this is not the case in our approach (unless node adjacency preservation is specified by the domain constraints). This results in a different treatment of edge similarity, since in our case there is no strict one-to-one correspondence between edges if node adjacency is not preserved.

A different family of methods is based on formulating the problem as a continuous nonlinear optimization. Fischler and Elschlager [13] propose a relaxation labeling approach that assigns a label to each node in the model ARG that determines its correspondence to a node in the query ARG. The algorithm begins by computing the probability of each possible label assignment as a function of the node attributes and any other local information, such as the attributes of the edges incident on the model and query nodes. In subsequent iterations, the probabilities are updated by taking into account the assignment probabilities of neighboring nodes until either convergence or a maximum number of iterations are reached. A similar approach is proposed by Kittler and Hancock [20]. Christmas et al. [4] extend the relaxation labeling approach to consider edge attribute information at each iteration (i.e., not just in the initialization step). In the same spirit, Wilson and Hancock [45] propose an error-correcting method that exploits structural constraints by defining a probabilistic dictionary of consistent node mappings for neighborhoods of nodes (augmented with null nodes to represent deletions). The matching solution in this formulation is given by the maximum a posteriori (MAP) estimate of a Bayesian formulation of the node correspondence probabilities. Huet and Hancock [19] extend this approach to consider edge attributes in the probability of node neighborhood transformations encoded

by the consistency dictionary in [45]. Similar probabilistic error-correcting approaches based on relaxation labeling are proposed by Myers et al. [28] and by Torsello and Hancock [42] (for trees). Finally, Luo and Hancock [22] use the EM algorithm [11] (instead of relaxation labeling) to find the MAP solution to a probabilistic graph matching problem in which the query graph is treated as observed data and the model graph acts as hidden random variables.

The probabilistic methods described above do not enforce a one-to-one mapping between nodes (they allow for one-to-many correspondences), and yield matches that are not symmetric (the results depend on whether each graph takes the role of the model or the query). These properties are appropriate for some domains but are not desirable in others. In contrast, the matching algorithm that we present in Section 3 yields a one-to-one node mapping that does not depend on the role of each graph. Our approach does bear some resemblance to the probabilistic methods in that the similarity of node attributes may be updated at each iteration of the algorithm as a function of the current matching information, whereas previous tree search approaches assume that the node attribute similarities are constant.

It should also be noted that there are graph matching approaches specifically tailored to the problem of matching skeletal-based representations of object silhouettes. Zhu and Yuille [46] propose an error-correcting graph matching algorithm in which the relative sizes of shape parts are used to define the penalty for missing parts. Baseski et al. [2] propose an error-correcting algorithm for trees based on a coarse skeleton representation of each shape, and integrate knowledge about object categories into their shape similarity measure. In contrast to these approaches, Bai and Latecki [1] propose a framework that abstracts out the structure of the skeleton graph entirely, and instead computes the similarity of skeletal paths connecting the terminal points of the skeleton. The similarity of the shortest paths between each pair of terminal points is used to establish the correspondences between skeleton graphs.

3. Inexact Graph Matching for Bone Graphs

In this section, we explore the problem of matching graph-based representations in which both the nodes and edges of the graphs have arbitrary sets of attributes. In a recent paper [23], we performed graph matching experiments with bone graphs and shock graphs using the matching algorithm

for node-attributed (NA) DAGs proposed by Shokoufandeh et al. [37]. Since this algorithm shows promising results when applied to bone graphs, we propose a generalization of it for the problem of matching fully-attributed (FA) DAGs. In addition, our generalization of the algorithm expands the type of domain knowledge that can be incorporated into the matching process, and allows us to improve on the results obtained with the previous version of the algorithm.

The problem of finding node correspondences between graphs that might not be isomorphic is known as inexact graph matching. In our case, we are also interested in computing a measurement of graph similarity that can be used to rank order model graphs with respect to a query graph. Given two graphs $\mathcal{G}(V, E)$ and $\mathcal{G}'(V', E')$, our problem is to find the values of node assignment variables $a_{vv'} \in \{0, 1\}$, for $v \in V$ and $v' \in V'$, that maximize some function of graph similarity, $F(\mathcal{G}, \mathcal{G}')$. In the case of attributed graphs, the graph similarity measure is a function of the attribute similarity of nodes and/or edges and the structural similarity of the underlying graphs. For example, in [21, 27, 40] the problem is stated as

$$F(\mathcal{G}, \mathcal{G}') = \max_{M \in \mathcal{M}} \frac{1}{2} \sum_{v \in V} \sum_{v' \in V'} M(v, v') N_a(v, v'), \tag{1}$$

where $N_a(v, v')$ are constant weights representing measures of node attribute similarity, and $\mathcal{M} = \{M\}$ is the set of all $|V| \times |V'|$ binary matrices $M(v, v') = a_{vv'}$ whose nonzero entries represent node assignments that satisfy some set of constraints. Frequently, the constraints enforce one-to-one node mappings that preserve node adjacency, and therefore define a maximum common isomorphic subgraph problem, which, for general graphs, is NP-hard [15].

In some domains, requiring the preservation of node adjacency may exclude natural solutions to the node correspondence problem. For example, Figure 2 shows that a minor shape difference can make a mapping between salient bone graph nodes invalid if node adjacency constraints are enforced. As a workaround, inexact graph matching can be defined in terms of edit operations that add or eliminate nodes with the objective of finding a graph isomorphism with minimum edit cost [3]. In this case, the similarity weight between nodes v and v' is not simply a function of their attributes (like $N_a(v, v')$ in Eq. 1), but also accounts for the costs of the edit operations needed to make v and v' respect their adjacency to any previously matched nodes. The minimization of edit costs is usually posed as tree search with

backtracking, in which a solution set (initially empty) is iteratively grown by adding node correspondences that are compatible with the mappings already in the set. We discuss the related work in this area in Section 2.

The tree search approach to graph matching provides a simple mechanism for measuring the similarity of attributed graphs, but is limited to exploiting only node adjacency constraints. Our algorithm for graph matching is similar to the tree search approach in that we: (a) iteratively grow a solution set of node correspondences, (b) use heuristics to find an approximate solution in polynomial time, and (c) define the similarity between two nodes as a function of both their attributes and the set of previously selected correspondences. However, unlike tree search, we consider constraints beyond adjacency in the evaluation of node similarity. This allows us to account for edge attribute similarities and for the hierarchical dependencies between the nodes of a DAG.

Our matching algorithm seeks to approximate the result of evaluating all possible ways of populating a solution set of node assignments, and selecting the set with the maximum sum of assignment weights. One simple way of expressing this function is to assume that every node in V is mapped to every node in V' with some weight, which is nonzero only if the nodes are assigned to each other. Then, if we order the set of $|V| \times |V'|$ correspondences $\{(v, v')\}_{v \in V, v' \in V'}$ as a list, and let $\mathcal{L}$ be the set of all possible permutations of that list, the graph similarity function can be expressed as

$$F(\mathcal{G}, \mathcal{G}') = \max_{A \in \mathcal{L}} \frac{1}{N} \sum_{v \in V} \sum_{v' \in V'} w(v, v', A_{vv'}), \tag{2}$$

where $A_{vv'}$ is the set of node correspondences that appear before (v, v') in the totally ordered set $A \in \mathcal{L}$, and $w(v, v', A_{vv'}) \in [0, 1]$ are node similarity weights, which are nonzero if and only if v is assigned to v'. N is a normalization constant that penalizes for unmatched nodes (i.e., nodes of a graph whose similarity weights are zero for every node of the other graph), and is needed for rank ordering matching results. We let $N = \max(|V|, |V'|)$ for bone graph and shock graph matching, which is an appropriate normalization constant when the nonzero solutions are constrained to be one-to-one node mappings.

The dependency of node correspondence weights on $A_{vv'}$ allows for measuring node similarity with respect to structural and attribute constrains given by the node assignments already in the solution. Generally, these constraints render some permutations redundant (e.g., they yield node mappings

with zero weights), which can be exploited by the matching algorithm. In Section 3.2, we propose a novel definition of similarity weights that leads to a general approach for graph matching, which includes the approach of Shokoufandeh et al. [37] as a special case. Our approach allows us to naturally incorporate domain knowledge that cannot be expressed when the node correspondence weights are treated as constants. We present an efficient algorithm that exploits the structure of the problem to recompute only a small set of similarity weights at each iteration of the algorithm. Furthermore, in Section 3.2.3, we describe a straightforward extension to our algorithm in order to explore a bounded number of matching solutions. Later, in Section 4, we evaluate empirically the benefit of considering multiple solutions for the case of bone graphs and shock graphs.

3.1. Notation

We assume that we are given a pair of fully-attributed DAGs of the form $G(V, E, \lambda, \gamma)$, where V is a set of nodes, E is a set of edges $\mathbf{e} = (u, v)$ directed from node $u \in V$ to node $v \in V$, $\lambda(v)$ is a function that maps each node $v \in V$ to a domain-dependent set of node attributes, and $\gamma(\mathbf{e})$ is a function that maps each edge $\mathbf{e} \in E$ to a domain-dependent set of edge attributes. The given graphs are constant throughout the matching process, and so need not be passed as arguments to the functions that require access to their node adjacency matrices, or to the attributes of their node and edges.

We also assume that, in addition to the graphs, we are given a domain-dependent function of node attribute similarity $N_a(v, v')$, for $v \in V, v' \in V'$, and a domain-dependent function of edge attribute similarity $E_a(\mathbf{e}, \mathbf{e}')$, for $\mathbf{e} \in E, \mathbf{e}' \in E'$. All measures of similarity referred to throughout this section, whether they involve graphs, nodes, or edges, are assumed to be values in the interval $[0, 1]$. A similarity value equal to zero represents total dissimilarity, while a value of one represents equality.

The algorithms discussed below make frequent references to the problem of maximum-weight bipartite matching (MWBM) [14]. Then, it is convenient to define special notation for this problem. To this end, let $M = \text{MWBM}(A, B, \mathbf{W})$ be the solution to the MWBM problem, where A and B are disjoint sets of elements, and $\mathbf{W}$ is an $|A| \times |B|$ matrix of similarity weights between each pair of elements spanning A and B. The solution to this problem is the set $M = \{(a, b)\}$ of one-to-one correspondences *with nonzero weights* between the elements of A and B that yields the maximum

sum of similarity weights. In the case of ties, we assume that M is any set with maximum cardinality among all possible sets with maximum weight.

In some cases we are only interested in the value of the weight sum rather than the set of correspondences, and so we define an appropriate function:

$$\bar{M}(A, B, \mathbf{W}) = \sum_{(a,b)\in \mathrm{MWBM}(A,B,\mathbf{W})} \mathbf{W}(a, b),$$

where $\mathbf{W}(a, b)$ is the weight associated with matching element $a \in A$ and $b \in B$. Finally, it should be assumed that whenever we evaluate $\bar{M}$, the set of correspondences M can also be recovered from the solution if needed.

3.2. The FA-DAG Matcher

We begin by reviewing the matching algorithm of Shokoufandeh et al. [37] for node-attributed DAGs (NA-DAG), and then motivate our generalization of this work to the problem of matching fully-attributed DAGs (FA-DAG) and the exploitation of a wider range of domain knowledge. The NA-DAG matcher measures the similarity between two DAGs by searching for the node correspondences that maximize the sum of pairwise node similarities, while attempting to respect the hierarchy imposed by the edge directions in each graph. In each iteration of the algorithm, one node correspondence is added to the solution set in a greedy fashion. The selection of each correspondence is given by first solving the $M = MWBM(V, V', \mathbf{W})$ problem, where the weights for each possible correspondence (v, v') are a function of the attribute similarities of the nodes and a measure of the structural similarity of the subgraph rooted at each node. Then, the node correspondence $(v, v') \in M$ with the largest weight is added to the solution set (initially the empty set). Next, the selected node correspondence is used to split the graphs into two subgraphs formed by the descendants of v and v', and two subgraphs formed by their respective non-descendants. Finally, the algorithm proceeds recursively by matching the pairs of descendant and non-descendant subgraphs independently until there are no more possible correspondences left to select. This algorithm is depicted in Figure 4.

The measure of structural similarity encoded in the correspondence weights is a low-dimensional spectral signature computed from the adjacency matrix of the subgraph rooted at each node. This measure is an attempt to let the matcher account for the structure underneath nodes before committing to a node correspondence. In practice, the signature vectors can be seen as part of

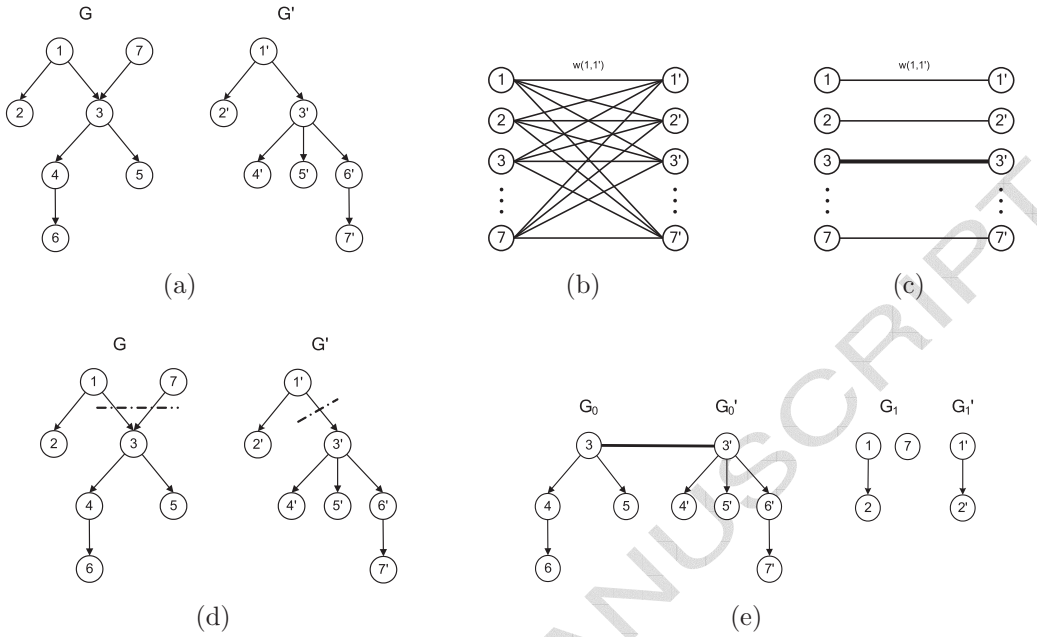


Figure 4: The NA-DAG matcher. (a) Given a pair of directed acyclic graphs G and G', (b) form a bipartite graph in which the edge weights are the pairwise node similarities. Then, (c) compute a maximum weight matching and add the best edge to the solution set. Finally, (d) split the graphs at the matched nodes and (e) recursively descend.

the node attributes, since they are constant and can be precomputed before the matching process takes place. We take this view, and assume that the same technique can be used in our generalization of the algorithm, if desired.

The pairwise node correspondence weights in each iteration of the algorithm remain constant and correspond to a possibly suboptimal solution to Equation 1 [38]. In [37], variable weights are considered to overcome the problem that whenever the graphs are split and each pair of subgraphs is matched, the subgraphs of non-descendants can lead to correspondences that violate the hierarchical constraints imposed by the DAGs. For example, a pair of sibling nodes in one graph can be assigned to a pair of parent-child nodes in the other graph. To prevent this, some of the weights in $\mathbf{W}$ are penalized in each iteration of the algorithm. Unfortunately, the update of weights can lead to a solution that is not consistent with Equation 1, which assumes that the correspondence weights are constant.

We generalize the NA-DAG matching algorithm to fully-attributed DAGs

by considering the objective function given by Equation 2, and defining correspondence weights $w(v, v', A_t)$ that are a function of the node and edge attributes of nodes v and v', and the solution set, A_t, at each iteration, t, of the algorithm. This definition of correspondence weights also allows us to encode the graph split operation of the NA-DAG matcher as a structural constraint of the problem. In fact, by encoding structural constraints as weights, we provide a more general approach for incorporating domain knowledge into the matching problem. The weights can be used to penalize correspondences that violate different types of node relations, whereas the graph split operation of the NA-DAG matcher is restricted to a single type of node relation (i.e., descendant), and to a binary decision on whether or not the node correspondences satisfy the relation.

We decompose $w(v, v', A_t)$ into four factors, which measure similarity as a function of

graph structure, such that the violation of a hierarchical relation with respect to nodes in A_t is penalized;

edge attributes, where the attribute similarity of all edges incident on nodes v and v' is evaluated. This similarity is computed while accounting for the paths that connect nodes in A_t to both nodes v and v';

assignment uniqueness, such that correspondences involving nodes already in A_t are assigned a zero weight;

node attributes, where the similarity of node attributes is measured using a domain-dependent function.

Our objective is to let any of these factors veto a node assignment by yielding a similarity value equal to zero. This can be expressed as a product over all the factors. Then, we let the similarity weights be

$$w(v, v', A_t) = S(v, v', A_t)E(v, v', A_t)U(v, v', A_t)N_a(v, v', A_t), \tag{3}$$

where S, E, and U are the structural, edge, and uniqueness similarity functions, and N_a is a domain-dependent similarity function for node attributes. The structural and edge similarity functions deserve special consideration and are defined in Sections 3.2.1 and 3.2.2, respectively. The uniqueness

similarity is simple, and can be defined as

$$U(v, v', A_t) = \begin{cases} 1 & \text{if } \forall u' \in V' \ (v, u') \notin A_t \text{ and } \forall u \in V \ (u, v') \notin A_t, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The matching algorithm is simply given by the successive iteration of two main steps. In the first step, we create/update a complete bipartite graph with node sets V and V', and edge weights $\mathbf{W}_t$ computed as a function of the current solution set A_t (initially the empty set). In the second step, we solve the MWBM problem, $M = \text{MWBM}(V, V', \mathbf{W}_t)$, and add the correspondence $(v, v')^* \in M$ with largest weight to the solution set, such that $A_{t+1} = A_t \cup \{(v, v')^*\}$. The algorithm terminates when there are no more node correspondences to consider, i.e., $\mathbf{W}_t = \mathbf{0}$. In Section 3.2.3, we present this algorithm and also present a less greedy variation of it.

An example of the first three iterations of the algorithm is shown in Figure 5. In this example, we assume that the structural similarity $S(v, v', A_t)$ is equal to zero if the union of $\{(v, v')\}$ and A_t is a set in which an ancestor-descendant assignment is inverted. i.e., a set in which a pair of ancestor-descendant nodes in one graph is assigned to a pair of ancestor-descendant nodes in the other graph with the ancestors assigned to the descendants.

3.2.1. Measuring Structural Similarity

The structural similarity of a node assignment (v, v') given a set of previous assignments A_t can be thought of as a measure of how much the node relationships in each graph would differ if node v is assigned to node v'. For example, Figure 6 shows an example in which an assignment between an ancestor of node v and a descendant of node v' has been previously established. In this case, if node v is assigned to node v', the resulting set of node assignments would not respect the hierarchical ordering imposed by the DAGs.

There are three types of hierarchical relations in which we are interested for bone graphs: ancestor, descendant, and sibling. However, we can describe our approach more generally by assuming a set of node relations $R = \{r_k\}_{k=1}^K$ that we are interested in maintaining, and defining predicates $r_k(a, b)$ that are true iff node a and b satisfy relation r_k. Briefly, our goal is simply to ensure that both nodes v and v' have similar relations with the previously matched nodes $(u, u') \in A_t$. Whenever this is not the case, i.e., $r_k(u, v) \neq r_k(u', v')$ for any $k = 1, \ldots, K$, we penalize the assignment (v, v') according to a domain-dependent penalty $p_k \in [0, 1]$ for the relation k. This can be expressed as a

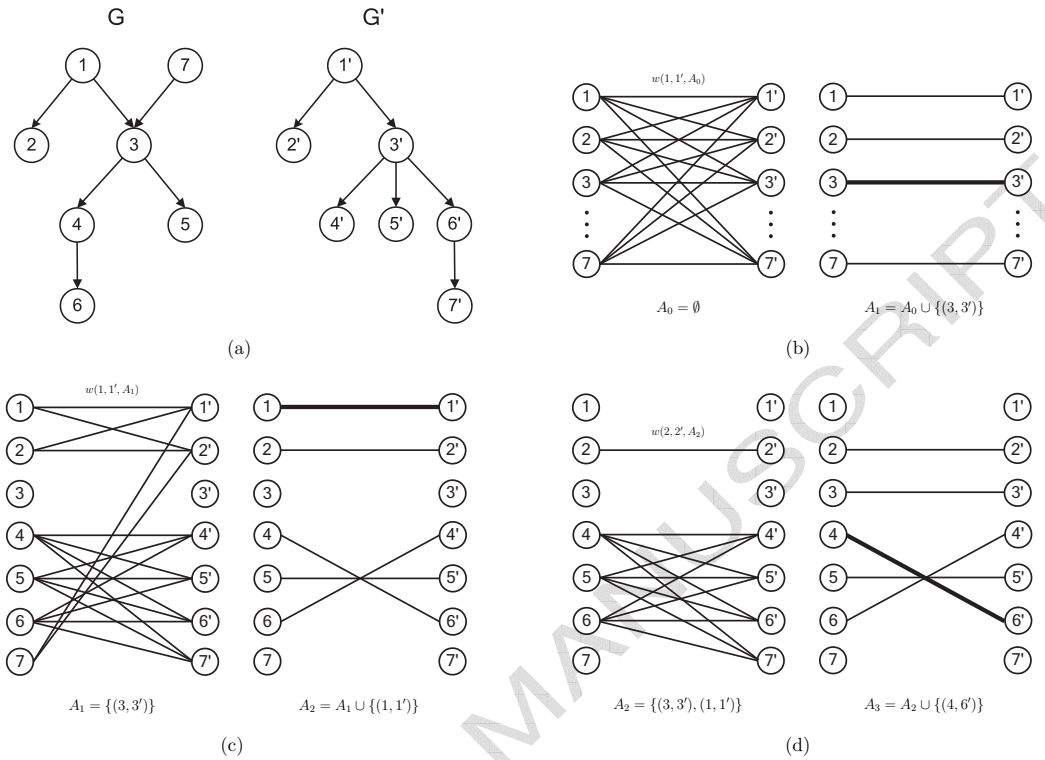


Figure 5: The generalized DAG matching algorithm. (a) The given pair of DAGs. (b) In the first iteration of the algorithm, we form a bipartite graph with similarity weights computed as a function of the empty solution set A_0. Then, we solve the MWBM problem and add the correspondence with largest weight, in this case $(3, 3')$, to the solution set. (c) In the next iteration, we recompute the similarity weights as a function of the augmented solution set A_1. Here, all correspondences of the form $(3, v')$ and $(v, 3)$ have a zero weight due to the $U(v, v', A_t)$ term in Equation 3. Similarly, the $S(v, v', A_t)$ term evaluates to zero for the correspondences that violate structural constraints. For example, the correspondence $(1, 4')$, if added to A_1, would have an ancestor of node 3 assigned to a descendant of node $3'$, and so $S(1, 4', A_1) = 0$. Only nonzero correspondences need be considered when solving the MWBM problem. (d) A possible third iteration of the algorithm. The algorithm terminates when all the possible correspondences between unassigned nodes have zero weights.

product over all relations of interest between nodes v and v' and the nodes in the current solution set. Then, we define the structural similarity function as

$$S(v, v', A_t) = \prod_{(u,u') \in A_t} \prod_{r_k \in R} p_k^{[r_k(u,v) \neq r_k(u',v')]}, \tag{5}$$

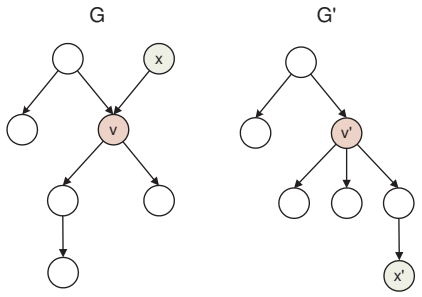


Figure 6: Example of hierarchical similarity. Here we assume that the node correspondence (x, x') is in the current solution set A_t, and want to compute the structural similarity for the node correspondence (v, v'). Since x is an ancestor of v, but x' is a descendant of v', the addition of (v, v') to the solution would create a node mapping that does not respect the hierarchical ordering between the nodes of each DAG. A structural similarity equal to zero would prevent this assignment from being added to the solution.

where $[\mathcal{P}]$ is the square bracket notation, which is equal to one if the predicate $\mathcal{P}$ is true, and zero otherwise.

In our experiments, we specify the penalties for the ancestor, descendant, and sibling relations as follows. For the ancestor and descendant relations, we let their penalties be $p_1 = 0, p_2 = 0$, so that violations of such relations are not allowed in the solution. This differs from the NA-DAG matching approach of Shokoufandeh et al. [37], which does not penalize for violations to the ancestor relation (see Figure 7). For the sibling relation (see [37] for a discussion on this relation), we let the penalty be $p_3 = 0.8$, which encodes the fact that shape parts frequently switch parents when the shapes are represented by bone graphs or shock graphs.

In summary, our structural similarity function is a general approach to incorporating assumptions about the importance of preserving certain node relations when matching two graphs. Our formulation of the problem allows us to specify arbitrary types of node relations, and to penalize mismatches according to the importance of each relation.

3.2.2. Measuring Edge Similarity

The edge similarity of a node assignment (v, v') given a set of previous assignments A_t is the normalized sum of pairwise edge attribute similarities for the inward and outward edges incident on nodes v and v'. We compute this value by assuming that there is a one-to-one correspondence between inward edges and between outward edges, and find the set of edge correspon-

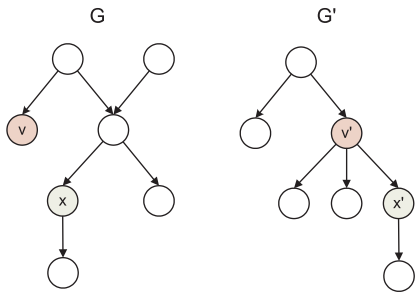


Figure 7: Example of the ancestor relation. Here we assume that the node correspondence (x, x') is in the current solution, and want to compute the structural similarity for the candidate correspondence (v, v'). Since v' is an ancestor of x', but v is not an ancestor of x, the addition of (v, v') to the solution would create a node mapping that does not respect the ancestor relation. In this case, we apply a penalty to the weight of the candidate correspondence (v, v') to make it less likely to be added to the solution. In contrast, the NA-DAG matcher does not penalize for the violation of this node relation.

dences that maximizes the sum of pairwise similarities. We account for the previous node matches $(u, u') \in A_t$ by ensuring that if both u and u' are connected to v and v' by a directed path, then the edges linking these paths to nodes v and v' are assigned to one another when we sum the pairwise edge similarities. An example of this constraint is illustrated in Figure 8.

We define the problem of measuring edge similarity as that of solving a MWBM problem for the set of edges incident on nodes v and v'. For the special case in which both v and v' have empty sets of incident edges, we assume that their edge similarity is one. Otherwise, we represent the pairwise edge attribute similarity between edges, and the constraints on path information from previously matched nodes as the weights of the MWBM problem. Thus, the edge similarity function becomes

$$E(v, v', A_t) = \frac{\bar{M}(E_v, E_{v'}, \mathbf{W})}{\max(|E_v|, |E_{v'}|)}, \tag{6}$$

where $E_v = \{\mathbf{e}_k\}$ and $E_{v'} = \{\mathbf{e}'_l\}$ are the sets of incident edges on nodes v and v', respectively, $\bar{M}$ is the solution to the MWBM problem (Sec. 3.1), and $\mathbf{W}$ is the $|E_v| \times |E_{v'}|$ matrix of edge correspondence weights defined below. The denominator of this equation normalizes the sum of edge correspondence weights to unity, yielding a similarity measure that penalizes for unmatched edges.

In order to define the edge correspondence weights, let $P(a, b)$ be the

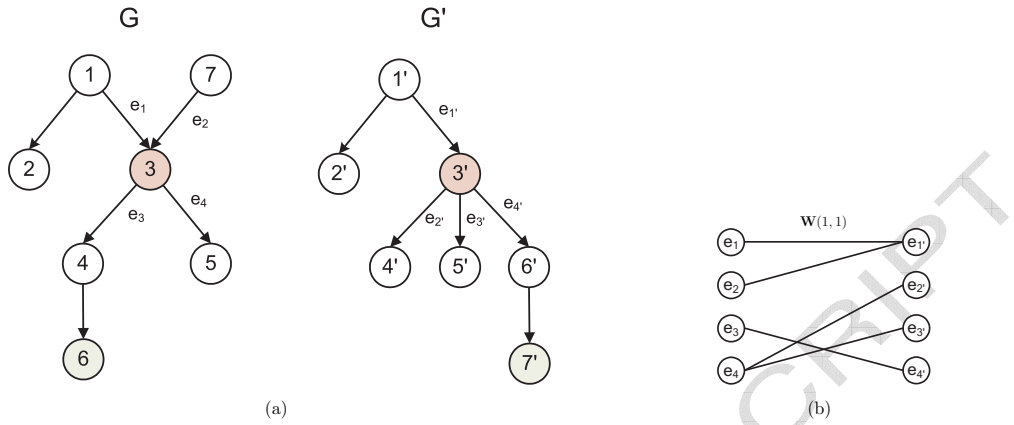


Figure 8: The edge similarity function. (a) We evaluate $E(3, 3', A_t)$ by summing the pairwise similarities of edges incident on nodes 3 and $3'$ as a function of the solution set A_t. In this example, we assume that the node correspondence $(6, 7')$ is the only element in A_t. $E(3, 3', A_t)$ is equal to the maximal sum of nonzero one-to-one correspondences between edges. A correspondence $(\mathbf{e}_i, \mathbf{e}_j)$ may have a nonzero weight $\mathbf{W}(i, j)$ iff: (1) both edges have equal direction, and (2) they are consistent with the contents of A_t. For example, here the only possibly nonzero correspondence for either edge $\mathbf{e}_3$ or $\mathbf{e}_{4'}$ is $(\mathbf{e}_3, \mathbf{e}_{4'})$, since there is a directed path from nodes 3 to 6 and from nodes $3'$ to $7'$. In contrast, edges $\mathbf{e}_4, \mathbf{e}_{2'}$, and $\mathbf{e}_{3'}$ have no ancestors or descendants in A_t, and have more than one possible correspondence. (b) $E(3, 3', A_t)$ is equal to the solution of the MWBM problem shown here, where the weights of the correspondences that meet conditions (1) and (2) are given by a domain-dependent measure of edge attribute similarity, $E_a(\mathbf{e}, \mathbf{e}')$.

(possibly empty) set of edges along the *directed* path that connects nodes a and b (i.e., a non-empty set means that a is either an ancestor or a descendant of b). Then, we let the edge correspondence weights $\mathbf{W}(k, l)$ be equal to zero if

- the edges $\mathbf{e}_k$ and $\mathbf{e}'_l$ have opposite direction with respect to nodes i and j (this condition is optional, as it may not be appropriate in some domains); or
- there exists a node correspondence $(u, u') \in A_t$ such that the path $P(u, v)$ includes $\mathbf{e}_k$ but the path $P(u', v')$ does not include $\mathbf{e}'_l$, or conversely, that $P(u', v')$ includes $\mathbf{e}'_l$ but $P(u, v)$ does not include $\mathbf{e}_k$.

Otherwise, we let $\mathbf{W}(k, l)$ be equal to a domain-dependent function $E_a(v, v')$ that measures the similarity of the edge attributes.

In conclusion, our edge similarity function incorporates edge attribute information in a nontrivial way by combining it with the information provided by the node correspondences in the solution. This measure of similarity exploits the structural dependencies between nodes to obtain an assignment of edge correspondences that is consistent with the node assignments that are in the solution set. As this set grows in each iteration of the algorithm, so too does the information available for assigning the edge correspondences. This, in turn, makes the measure of edge attribute similarity less ambiguous, and strengthens the overall measure of node similarity weights.

3.2.3. Searching the Solution Space

The matching algorithm of Shokoufandeh et al. [37] takes a greedy approach to establish node correspondence. This is motivated by the assumption that the measures of node attribute similarity should provide a strong indication of the best correspondences, and make a broad search of the solution space less necessary. In our generalization of this algorithm, we propose to relax the greediness of the approach by incorporating a bounded queue to evaluate more that one possible solution to the problem. We form a queue of solution sets of size bounded by the maximum number of solutions that we want to evaluate. In each iteration of the algorithm, we add one node correspondence to a solution set in the queue. The algorithm terminates when the queue is empty.

We present the algorithm that considers a single solution first, and later introduce a queue to consider multiple solutions. The two main steps of the algorithm are the computation of node similarity and the selection of node correspondences. In the first iteration of the algorithm, $t = 0$, the weights of all possible pairwise node correspondences, $\mathbf{W}_t$, are computed as a function of the empty solution set, $A_t = \emptyset$, according to Equation 3. Next, the MWBM problem, $M = \text{MWBM}(V, V', \mathbf{W}_t)$, is solved and correspondence $(v, v')^* \in M$ with largest weight is added to the solution set, such that $A_{t+1} = A_t \cup \{(v, v')^*\}$. In the subsequent iterations, $t \geq 1$, the correspondence weights are updated to reflect the current contents of the solution set, and a new correspondence is added to the solution as previously described. The algorithm terminates when all possible new node correspondences have zero weight (i.e., they are incompatible).

When considering multiple solutions, we queue the alternate solution sets that can be obtained in the first few iterations of the algorithm. The motivation for this is that the first elements added to a solution set condition

all subsequent correspondences, and so selecting the correct ones early on is important. We bound the maximum number of solution sets that can be considered by a given constant $K \geq 1$. We also let the maximum number of solution sets added to the queue in each iteration be a given constant $1 \leq \tau \leq K$. Then, we modify the matching algorithm such that at each iteration, we pop one solution set A from the queue of active solution sets Q, and compute the correspondence weights as a function of it. If all correspondences have zero weight, we add A to the set of completed solutions S. Otherwise, we solve a MWBM problem for the nodes of the graph using the current weights, and then select the top $n = \min(\tau, K - |Q| - |S| + 1)$ correspondences $(v, v')^*_k \in M$, for $k = 1, \ldots, n$. Each of these correspondences is used to create new solution sets $A^k = A \cup \{(v, v')^*_k\}$, which are added to the back to the queue. The algorithm terminates when the queue of active solution sets is empty. This algorithm is shown on the next page.

3.2.4. Algorithm Complexity

We focus our complexity analysis on the case in which the maximum number of solution sets is one, since a greater constant limit does not affect the algorithm's complexity. We also assume that the algorithm is efficiently implemented such that only the weights that may change from one iteration to the next are updated. For the case of bone graphs, we update only the weights that involve ancestors, descendants, and siblings of the nodes v and v' that are added to the solution set in the previous iteration of the algorithm. In this case, we use appropriate data structures in order to efficiently visit the required weights, and to evaluate the structural and uniqueness similarity factor of the weight function. By recursively following the parents and children associated with each node correspondence added to the solution set, we can visit the weights that must be updated in time $O(N)$, for $N = \max(|V|, |V'|)$. The ancestor, descendant, and sibling relations can be evaluated in constant time by precomputing appropriate data structures for each graph. An efficient representation of the ancestor and descendant relations can be computed in linear time (they are given by the transitive closure of a graph [17]), while that of the sibling relation can be computed in quadratic time (it requires a combination of the transitive closure and the adjacency matrix of a graph [37]). We also use an appropriate data structure to evaluate the membership of a node to the solution set in order to compute the uniqueness similarity factor of the weight function in constant time.

The computation of the edge similarity factor demands more work, as it

Algorithm 1 The FA-DAG matcher

Require: $G(V, E, \gamma, \lambda), G'(V', E', \gamma', \lambda'), K \geq 1, \tau \leq K$ {for K, τ defined in Sec. 3.2.3}

$Q \leftarrow \{\emptyset\}$ {let Q be a queue with an empty solution set in it}
$S \leftarrow \emptyset$ {let S be the empty set of completed solution sets}
while $|Q| \neq 0$ **do**
  $A \leftarrow Q.Pop()$ {retrieve a solution set from the queue}
  $\mathbf{W} \leftarrow f(G, G', A)$ {set node correspondence weights according to Eq. 3}
  if $\mathbf{W} \neq \mathbf{0}$ **then**
    $M \leftarrow MWBM(V, V', \mathbf{W})$ {solve the MWBM problem (see Sec. 3.1)}
    $L \leftarrow Sort(M)$ {sort correspondences by decreasing weight}
    $n \leftarrow Min(\tau, K - |Q| - |S| + 1)$ {set max number of new solutions}
    $L \leftarrow Prune(L, n)$ {reduce the list to top n candidates}
    for all (v, v') in L **do**
      $A' \leftarrow A \cup \{(v, v')\}$ {create a new solution set with A and (v, v') in it}
      $Q.Push(A')$ {add the new solution set to the queue}
    end for
  else
    $S \leftarrow S \cup \{A\}$ {add A to the set of completed solution sets}
  end if
end while
return $\underset{A \in S}{\operatorname{argmax}} \sum_{v \in V} \sum_{v' \in V'} w(v, v', A)$ {select the set with maximum weight sum}

requires solving a MWBM problem. The complexity of this step is $O(e^2 \log e)$, where e is the maximum number of incident edges for any node in either graph. Then, the complexity of updating the weights in each iteration is $O(Ne^2 \log e)$. In addition to updating weights, each iteration of the algorithm must solve a MWBM problem for the nodes, which has complexity $O(N^2 \log N)$. The number of iterations of the algorithm can be bounded by N if only one-to-one correspondences are allowed. Thus, the complexity of the algorithm is $O(N^3 \log N + N^2 e^2 \log e)$. In the case of hierarchical structures, such as bone graphs and shock graphs, the number of nodes grows much faster than the maximum number of incident edges on a node, and e^2 is generally smaller than N for DAGs of significant height, which results in $O(N^3 \log N)$ complexity. Similarly, the complexity of the NA-DAG matcher algorithm is $O(N^3 \log N)$ [37], since the edge similarity term need not be computed.

3.3. Similarity Functions for Bone Graphs

We present measures of similarity for the attributes of nodes and edges in the bone graph. These similarity measures are the domain-dependent components of the matching algorithms described in the previous sections. Our goal is to compare silhouettes obtained by the perspective projection of 3-D objects onto a plane. Moreover, since we aim to find similarities between the silhouettes of different exemplars of the same object class, we propose *coarse* measures of node and edge similarity that attempt to not overpenalize the within-class deformations of both shape parts and part relationships.

3.3.1. Node Attribute Similarity

We define the similarity $N_a(v, v')$ between the attributes of nodes v and v' as a function of the length and width of the shape part represented by each node. Our measure of similarity is inspired by that of shock graphs [26], which compares the radius functions of the skeletal segments encoded as the attributes of nodes. The radius function of a bone captures the variation of the width along a shape part and the length of the part. By comparing radius functions, we obtain a similarity measure that is invariant to relative rotation, translation, and bending. This measure is not scale invariant, and so we assume that the global scale of each shape is normalized. We obtain a continuous representation of the radius function by approximating its discrete values with a piecewise linear function. This approximation is the result of minimizing the number of line segments without surpassing a maximum

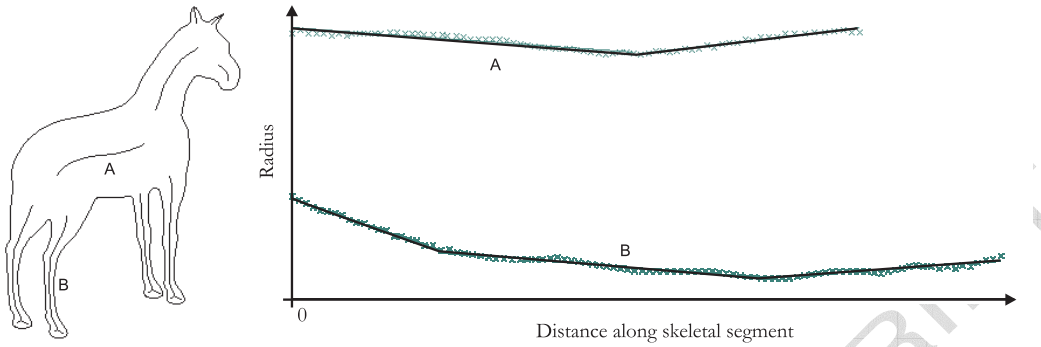


Figure 9: Examples of piecewise linear approximations of radius functions. The discrete points of the radius functions for a back leg and the torso parts are plotted, with their corresponding piecewise linear approximations overlaid.

fitting error (see [26] for details). Figure 9 illustrates examples of the radius functions of bone graph nodes, and their approximation by piecewise linear functions.

In the shock graph, the piecewise linear approximation of the radius function is used to create the shock graph nodes, and to compare their attributes at matching time. During the construction of a shock graph, this approximation is used to decompose the skeletal branches into segments with constant or monotonically varying radii, which then become nodes in the graph. At matching time, the approximation is used to compare the radius functions encoded by the nodes. For the bone graph, we only use the approximation of the radius function at matching time, since we do not partition skeletal segments according to radius. Thus, our problem is to compare radius functions that vary arbitrarily.

We propose a measure of similarity that accounts for the absolute radius differences and for the variations of the radii along the medial axes. We begin by subdividing the radius functions into an equal number of segments for the part of the domain on which they are both defined. Given two piecewise linear functions, $r(x)$ and $r'(x)$, defined on the respective intervals $[0, L]$ and $[0, L']$, we subdivide the line segments that fall in the interval $[0, \min(L, L')]$ into N subsegments. The subdivision is performed such that the x-coordinates $\{x_i\}_{i=0}^{N}$, for $x_i < x_{i+1}$, correspond to the endpoints, intersection points, and knots between the segments of both functions (see Figure 10). Then, we compute the areas and slopes of the pair of linear functions defined on each interval $[x_i, x_{i+1}]$, and combine them into a piecewise measure of similarity.

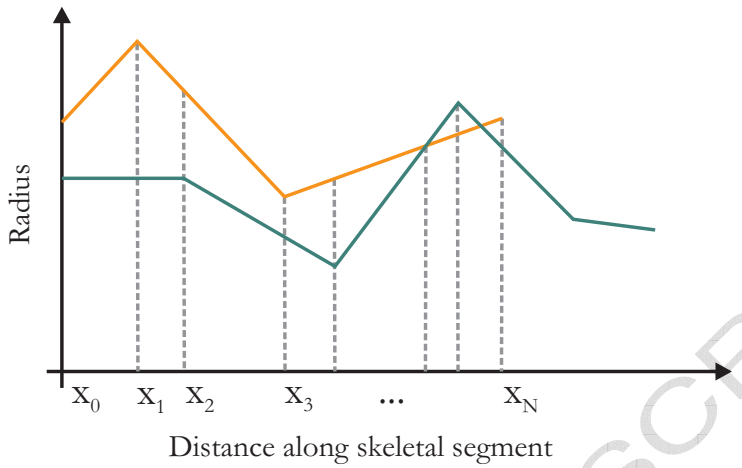


Figure 10: Example of the comparison of two radius functions. The piecewise linear functions are subdivided into N segments for the part of the domain on which they are both defined. The segments are partitioned such that each x-coordinate x_i, for $i = 0, \ldots, N$, corresponds to an endpoint, intersection point, or segment knot on either function.

First, we define the area-based similarity as a normalized measure of area difference. Let a_i and a'_i be the respective nonzero areas under $r(x)$ and $r'(x)$ on the interval $[x_i, x_{i+1}]$, and

$$\alpha_i = 1 - \frac{|a_i - a'_i|}{a_i + a'_i} \tag{7}$$

be their area-based similarity. Next, we define the slope-based similarity as a Gaussian function of slope difference. Let m_i and m'_i be the respective slopes of $r(x)$ and $r'(x)$ along the interval $[x_i, x_{i+1}]$, and

$$\beta_i = e^{-\frac{(m_i - m'_i)^2}{2\sigma^2}} \tag{8}$$

be their slope-based similarity (we let $\sigma = 0.15$ in our experiments). Finally, we combine the area- and slope-based similarities and the relative length of each interval into a measure of node attribute similarity

$$N_a(v, v') = \sum_{i=0}^{N-1} \alpha_i \beta_i \frac{(x_{i+1} - x_i)}{\max(L, L')}. \tag{9}$$

Our measure of node attribute similarity penalizes the differences in the lengths of the medial axes, and in the relative widths and width variations

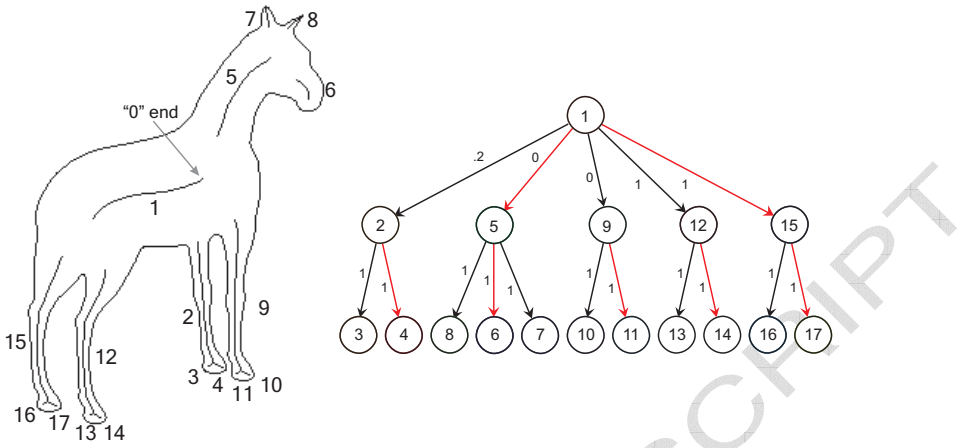


Figure 11: Example of the edge attributes of a bone graph. The attribute of an edge corresponds to the attachment position between a child bone and its parent bone. For clarity, we show the absolute value of each position and let the color of the edge represent its sign (i.e., the side of the attachment). The parameterization of edge attributes assumes that the lengths of all bones are normalized to the unit length. In the case of the root node 1, the "0" end is chosen arbitrarily. The "0" end of all other nodes is specified as the skeletal point closer to the attachment with their parent bone.

along the medial axes. In our experiments, we consider a value of σ that is constant for all shapes and shape parts, but this could be replaced by a values obtained as a function of the model shapes or model parts being matched. We leave the problem of learning shape deformation parameters conditioned on object classes as the subject of future research.

3.3.2. Edge Attribute Similarity

We define the similarity $E_a(\mathbf{e}, \mathbf{e}')$ between the attributes of edges $\mathbf{e}$ and $\mathbf{e}'$ as a function of the relative side and position of the attachment represented by each edge. The attribute $\gamma(\mathbf{e}) = p_{u,v}$ of edge $\mathbf{e} = (u, v)$ is the normalized signed position of the attachment of node v onto node u. The sign of $p_{u,v}$ encodes the side of the attachment, and the absolute value of $p_{u,v}$ encodes the attachment position between endpoints 0 and 1 along the bone represented by node u [23]. Figure 11 illustrates an example of the edge attributes of a bone graph.

The "0" end of the position along the medial axis encoded by a node is chosen arbitrarily for root nodes and for nodes with multiple parents. In the case of nodes with a single parent, the "0" end is the point closer to the

attachment point with the parent node. The sign of the position is specified by considering that the bone rotates around its "0" end. For example, in Figure 11, a black edge represents an attachment on the side corresponding to a clockwise rotation of the parent bone, while a red edge is associated with a counterclockwise rotation. We take this into consideration, and for the cases in which the parameterization is ambiguous, i.e., nodes whose in-degree is not one, we evaluate the node and edge attributes with respect to both possible choices for the "0" end, and keep the one that maximizes similarity.

We compute the edge attribute similarity as the product of sign similarity and position similarity. We define a constant penalty $\omega \in [0, 1]$ for attachments on opposite sides, and let the sign similarity be

$$\delta(p_{u,v}, p_{u',v'}) = \begin{cases} 1 & \text{if } \text{sign}(p_{u,v}) = \text{sign}(p_{u',v'}) \\ \omega & \text{otherwise.} \end{cases} \tag{10}$$

In our experiments, we found that $\omega = 0$ was too strict a penalty, and that any value in $[0.1, 0.8]$ yielded similarly good results (we let $\omega = 0.6$ in our experiments). Next, we let the position similarity be the absolute difference in the positions of the attachments

$$\psi(p_{u,v}, p_{u',v'}) = 1 - |p_{u,v} - p_{u',v'}|. \tag{11}$$

The product of these two measures of similarity becomes our edge attribute function

$$E_a(\mathbf{e}, \mathbf{e}') = \delta(\gamma(\mathbf{e}), \gamma'(\mathbf{e}'))\ \psi(\gamma(\mathbf{e}), \gamma'(\mathbf{e})'). \tag{12}$$

Our measure of edge attribute similarity combines a linear penalty for the differences in the positions of the attachments with a constant penalty for the differences in the sides of the attachments. This represents a coarse measure of the similarity of part relations, and does not account for the relative ordering of child nodes that are attached to the same point of a parent node. Similarly to the case of node attributes, our measure of edge similarity could be improved by learning, at training time, the relative importance of the differences in the position and side of part attachments as a function of each object's class. Then, this information could be exploited at matching time, when a novel exemplar is compared against a model of a known object class. We leave this problem as future research.

4. Experiments

In this section, we evaluate the bone graph representation together with the graph matching algorithm presented in Section 3 for the task of object categorization. The goal in this task is to recognize 2-D views of novel 3-D exemplars of known object categories. This task differs significantly from the exemplar-based object recognition and pose estimation experiments considered in [23]. In particular, in our previous experiments, the task of estimating pose from unknown views of known exemplars allowed us to evaluate the benefits of addressing the over- and under-segmentation of skeletal parts, but provided little information about how the representation might cope with views of novel exemplars. The categorization problem, on the other hand, addresses this question directly. It also provides us with a more stringent task to evaluate the representation and the matching algorithms as part of a complete recognition framework.

Like the experiments in [23], we compare the bone graph representation to the shock graph. In review, both bone graphs and shock graphs are derived from the medial axis of a closed contour (silhouette). In the case of a shock graph [41], the points making up the medial axis are labeled according to their radius function: monotonically increasing, constant, monotonically decreasing from both directions toward a local minimum, and monotonically decreasing in both directions away from a local maximum. Contiguous points sharing the same label are clustered and become the nodes in the shock graph, in which nodes are labeled medial point clusters (forming branches) and directed edges represent branch adjacency, directed from larger to smaller (in terms of radius) parts. Unlike the shock graph, every medial axis point is not mapped to a node in a bone graph. As illustrated in Figure 1, those medial axis points that are classified as ligature, i.e., non-salient points that contribute little to the boundary shape, are abstracted out and are used to define the connections between the remaining salient shape parts. Like the shock graph, the edges of a bone graph are directed from larger to smaller parts. However, unlike the shock graph, the edges of a bone graph are attributed, and encode attachment positions.

This comparison between shock graphs and bone graphs is meaningful because both representations take a skeleton as input and yield a graph of skeletal parts as output. This means that for the bone graph to outperform the shock graph, its structure and edge attribute properties must be more beneficial for matching than those of the shock graph. In our experiments, we

measure the individual contribution of these properties by matching graphs with and without node and edge attributes. In contrast to the experiments in [23], where for the purpose of comparison we sub-partitioned the nodes of the bone graph according to radius, here we are concerned with evaluating the bone graph as defined in [23]. This definition of bone graph leads to shapes that are represented in terms of fewer nodes than required by the shock graph, and has the advantage of significantly improving matching time, since the time complexity of the matching algorithms are functions that grow with the size of the input graphs. For example, in our dataset the average size of bone graphs is 10.8 nodes, while that of shock graphs is 20.5.

4.1. The Dataset

The dataset in our experiments is a subset of the Princeton Shape Benchmark (PSB) [36]. This dataset is publicly available and widely used for evaluating learning and recognition approaches using 2-D and 3-D shape representations. We select 8 classes and 5 3-D exemplars per class out of the 1,814 exemplars and 90 classes available in the PSB. Our selection of exemplars, shown in Figure 12, includes models whose part structure varies from simple (e.g., the hot air balloon class) to fairly complex (e.g., the walking human class). This subset provides enough data to compare the bone graph and shock graph representations and to discuss the limitations of our approach while performing a comprehensive set of experiments. In future research, we expect to consider larger datasets by combining our representation and matching approach with learning and indexing components.

We populate a model database of 2-D shapes by taking 25 uniformly-sampled views per exemplar in our dataset, which yields a total of 1000 shapes. We collect the sets of 2-D views of each 3-D exemplar by sampling a portion of its viewing sphere. We select the views that fall within the azimuth and elevation angles in the ranges $[-180, -90]$ and $[-30, 30]$ degrees, respectively. We take 5 evenly spaced views along each coordinate (i.e., 25 views per object). We found that this selection of views provides a sufficiently dense sampling of a 3-D object while capturing most of its non-degenerate views. As an example, the set of views of one of the horse exemplars is shown in Figure 13.

4.2. Experimental Set Up

We evaluate two graph matching algorithms in our experiments, and for simplicity, we refer to them as algorithms A and B. Algorithm A is the

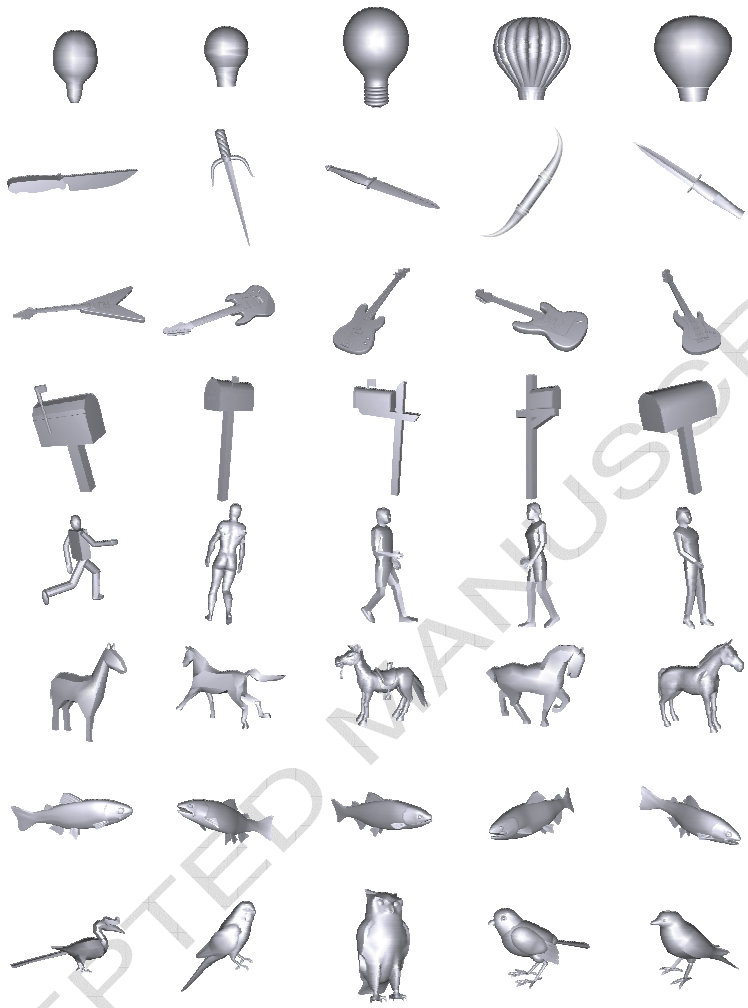


Figure 12: The Dataset of 3-D Models. It is formed by 8 classes with 5 exemplars per class. Half of the dataset is formed by "inorganic" objects, while the other half contains "organic" objects. This set of objects captures a wide range of part structure complexity, which ranges from very simple, in the case of the hot air balloon class, to fairly complex, in the case of the walking human and bird classes.

matching algorithm of Shokoufandeh et al. [37], and does not exploit edge attributes. Algorithm B is our generalization of algorithm A (Section 3.2), and accounts for edge attributes, as well as structural graph constraints that differ from those of algorithm A (Section 3.2.1). We consider the most greedy

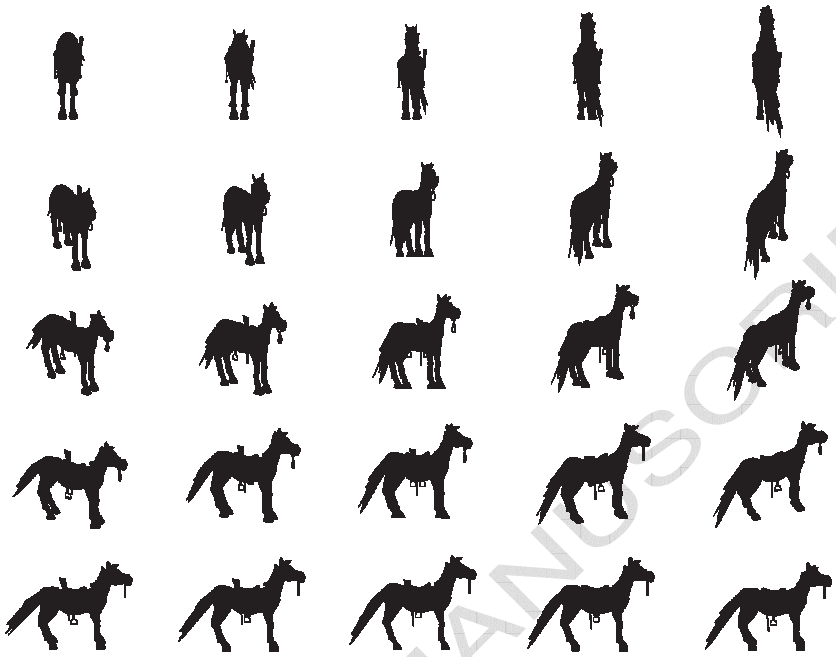


Figure 13: Subset of the Model Database of Object Views. Here we show the complete collection of 25 views used to represent one of the exemplars belonging to the horse class.

version of algorithm B, in which the size of the solution queue is bounded to one. Later, we evaluate the performance of the algorithm as a function of the size of the solution queue. Both matching algorithms require a domain-dependent node similarity function, while algorithm B also requires a domain-dependent edge similarity function. We use the node similarity function presented in [26] for shock graphs, and, when needed, let the edge similarity function be the identity function. For the bone graph, we use the node similarity function described in Section 3.3.1 and the edge similarity function described in Section 3.3.2.

The node similarity functions used for the bone graph and the shock graph are both based on comparing only the skeletal radius function represented by each node. That is, these functions do not account for skeletal curvature or other skeletal properties encoded in the node attributes. The two functions differ mainly in that with the shock graph, the radius function of a

node is assumed to be constant or vary monotonically, while with the bone graph, this need not be the case. Thus, the minor differences between the node similarity functions should not provide an unfair advantage for either representation.

In order to understand the influence of edge attributes of the bone graph, we carry out experiments with and without edge attributes (i.e., we let the edge similarity of the bone graph be the identity function) using algorithm B. Similarly, we evaluate the influence of graph structure in the matching process of both bone graphs and shock graphs by performing experiments with both node and edge similarity functions set to the identity function. In each figure plotting experimental results, we specify the attributes considered at matching time by labeling each representation as fully attributed (FA), node attributed (NA), edge attributed (EA), and unattributed (UA). For example, when matching the bone graph (BG) using both its edge and node attributes, we label the results as FA-BG.

We perform two sets of experiments in which all 1000 shapes are considered as queries while the contents of the model database vary as a function of the query class. In the first set of experiments, we evaluate recognition performance as a function of number of model exemplars for the query class. Here we expect the recognition performance to decrease as the shape variation of the exemplars in the query class becomes underrepresented by the reduction of its model exemplars. In the second set of experiments, we evaluate performance as a function of number of model views per exemplar of the query class. In this case, we also expect the performance to decrease as the query class is represented using fewer views per exemplar than the other classes. These experiments allow us to measure the performance of bone graphs and shock graphs relative to the performance associated with the numbers of exemplars and views of the query class.

In each experiment trial, we take an exemplar's view and compare it against all shapes in the model database that belong to exemplars different from the query's. That is, for each query view, the model database contains 4 exemplars of the *query* class and 5 exemplars of every other class (each exemplar is represented by 25 views). We say that class recognition is correct if the view that receives the highest similarity score belongs to the same object class as the query view. In the case of ties between a view of the query class and a view of a non-query class, we consider the outcome to be unsuccessful, since we seek a unique answer to the categorization question.

4.3. Object Categorization

We measure the influence of within-class shape deformation by evaluating recognition performance as a function of decreasing number of exemplars for the query class. In the first set of trials, each given query view is removed from the model database, along with all other views of the same query exemplar. In the next set of trials, we remove all the views of another exemplar of the query class. We repeat this procedure until there is only one exemplar of the query class remaining in the model database. All ways of removing one, two, and three exemplars out of four are considered, and the average and standard error of their recognition rate is plotted. That is, the values 3, 2 and 1 along the x-axis in Figure 14(a) correspond to the average performance obtained from 4, 6, and 4 sets of 1000 trials, respectively. Each of such trials represents a different way of removing exemplars of the query category prior to matching. The number of exemplars of the classes that are not that of the query view remains constant in each trial.

Figure 14(a) plots the recognition performance as the number of exemplars for each query class decreases from 4 to 1 (all the non-query classes are represented by 5 exemplars). The results exhibit the superiority of bone graphs and algorithm B over shock graphs and algorithm A. For the case of 4 exemplars, the score of the bone graph framework is 80.4% while that of the shock graph framework is 73.6%. This performance difference, 6.8 percentage points, is significant, as it is comparable to the decrease in performance due to matching bone graphs using 3 exemplars instead of 4, which has a score of 73.9%.

We determine whether the performance improvement of the bone graph is due to the representation, the matching algorithm, or both, by evaluating the four combinations of matching algorithms and shape representations. Figure 14(b) plots these results and shows that algorithm B is the most effective algorithm for matching both bone graphs and shock graphs. The results also show that the superiority of bone graphs over shock graphs is not due to the matching algorithm alone, since, for example, in the case of 4 exemplars the algorithm used only accounts for less than half the performance difference between the two representations. The standard error in the results of the exemplar removal trials suggests that both representations are sensitive to the choice of model exemplars that are left in the database. Moreover, the decrease in performance as the number of exemplars is reduced is slightly more pronounced for bone graphs than for shock graphs, especially when algorithm A is considered. We attribute this to the larger number of nodes

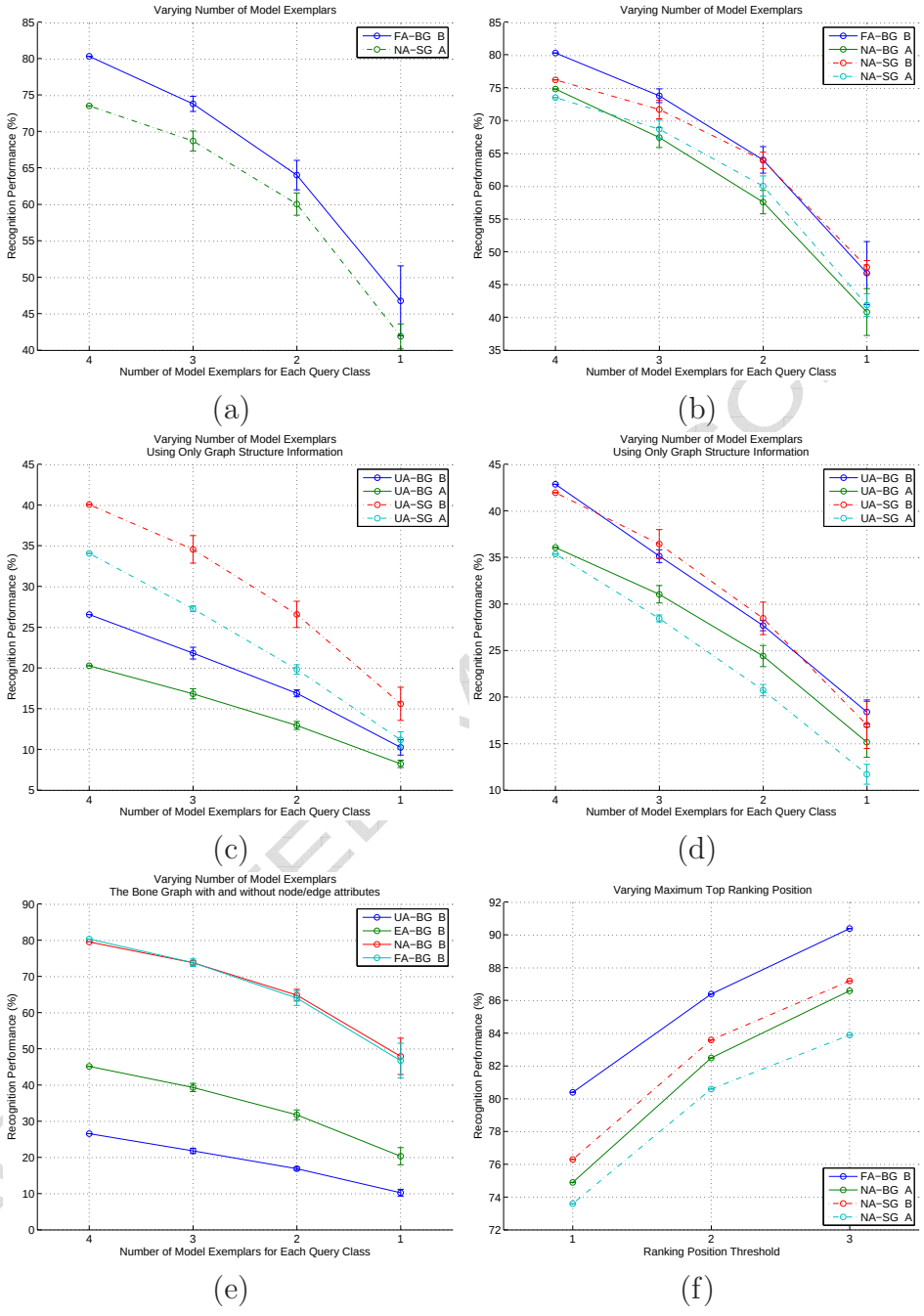


Figure 14: System Evaluation (see text for discussion)

in the shock graphs, which, in some cases, helps reduce the penalties incurred by missing parts. Note that the number of missing/different parts between query and model shapes increases proportionally to the dissimilarity between the most similar model and the query.

We evaluate the contribution of graph structure stability to the matching process by considering the problem of matching bone graphs and shock graphs without their node and edge attributes. Figure 14(c) plots recognition performance for unattributed (UA) bone graphs and shock graphs using algorithms A and B. Here the structure of the bone graph seems to contribute little information. In the case of 4 exemplars, the recognition score is just 26.6% with algorithm B, which is slightly more than twice the chance level of 10.26% (i.e., 4 correct exemplars over 39 exemplars). On the other hand, the score for shock graphs is 34.1% with algorithm A, and 40.1% with algorithm B. One reason for this large performance disparity between bone graphs and shock graphs is that the partitioning of skeletal branches according to radius performed by the shock graph effectively encodes geometrical attributes of the shape parts into the structure of the graph. In contrast, the structure of the bone graph is influenced less by the geometrical properties of the shape parts, which leads to more dissimilar shapes having equal structure. This effect can be seen in Figure 14(d), where we plot the same results but count ties for first ranking position as successful trials. Now, the recognition score for bone graphs and algorithm B jumps from 26.6% to 42.9%, while that for shock graphs and algorithm B has a more modest increase from 40.1% to 42%. This confirms that bone graphs have a much larger number of ties in the similarity ranking than shock graphs.

We analyze the contribution of the edge attributes to the matching process by comparing the results of matching fully-attributed, node-attributed, edge-attributed, and unattributed bone graphs. Figure 14(e) plots these results and shows that the presence of edge information does not improve recognition performance when combined with node attributes, but it does lead to a significantly better score than using graph structure alone. This suggests that the graph structure of bone graphs can become significantly more informative when combined with edge attributes. This result has important implications for the problem of indexing graphs based on their unattributed structure, such as the spectral approach proposed by Shokoufandeh et al. [39]. In the case of bone graphs, the results suggest that indexing features based on graph structure would indeed benefit from incorporating edge attribute information.

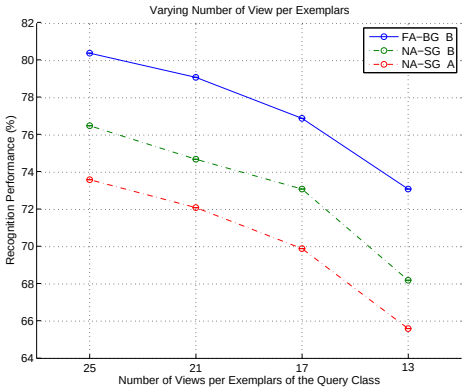


Figure 15: System Evaluation (see text for discussion)

In the experiments above we treat all unsuccessful cases equally, i.e., regardless of whether the correct answer is close to the top or at the bottom of the ranking. However, it is also important to evaluate the actual ranking position of the correct answer that is best ranked. This is useful when the goal of the matching process is that of pruning the model classes down to a small set of candidates. Figure 14(f) plots recognition performance when the correct answer is ranked in the top three positions. These results show that with bone graphs, the correct answer is ranked within the top three positions for 90.3% of the queries, while with shock graphs and algorithm A this happens only with 83.9% of the queries. In addition, the performance of the shock graph improves with algorithm B to 87.2%, which provides further evidence of the superior performance obtained by our matching algorithm regardless of the shape representation used.

Finally, we evaluate the performance of algorithm B when the size of the solution queue grows up to 12. In particular, we conduct trials in which the constants (K, τ) discussed in Section 3.2.3 are $(1, 1)$, $(3, 1)$, $(6, 2)$, $(9, 3)$, and $(12, 4)$. We test the case of bone graphs when the query class is represented by four exemplars and 25 views per exemplar. In these trials, the performance remains constant at 80.4% up to a solution queue of size 6, and increases monotonically to only 80.6% for a queue of size 12. Since this represents a small performance increase at a high computational cost, it justifies the approach of considering only one greedy solution. Naturally, a more exhaustive search of the solution space could lead to better performance, but the computational cost would render the approach impractical.

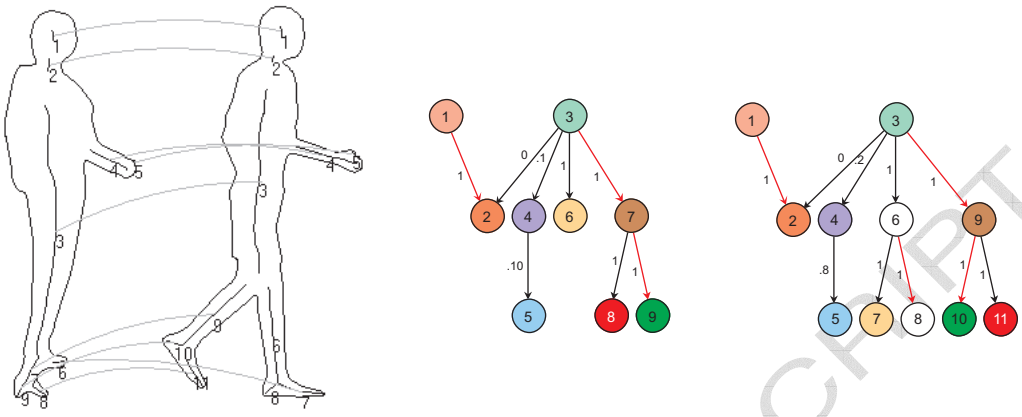
4.4. Example Matches

Figures 16, 17, and 18 illustrate a number of successful matches drawn from the experiments. The set of node correspondences found for each shape is shown by arcs connecting each pair of shapes, and by matching node colors between the corresponding bone graph representations of each shape. The examples demonstrate cases in which the matcher finds natural correspondences between parts with largely dissimilar geometries, such as the torso and the legs. We have not measured the correctness of part correspondence among successful matches, but we have found that in practice, the correct matches generally yield mostly correct part correspondences.

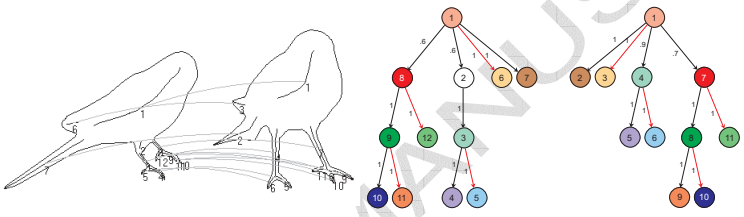
It is interesting to analyze some examples of unsuccessful matches in order to determine the limitations of our approach. We show four such cases in Figures 19 and 20. Case (a) in Figure 19 illustrates the limitation of edge attributes for representing ordering and orientation of multiple end-to-end attachments. Here, the neck and tips next to it of the electric guitar have similar attachment (edge) attributes with the guitar's body to the posts of the mailbox and its box. However, it can be seen that there is a strong visual difference in how these parts are attached to each other. Case (b) in 19 provides an example of two simple but very different objects with isomorphic graph structures. Here, in spite of the geometrical differences between the node attributes, the fact that the knife view is able to explain all nodes of the hot air balloon allows it to rank ahead of views of the correct class with more similar nodes. This illustrates that small details in a simple shape, such as convex corners, may have a disproportionate influence in the matching process. Finally, the examples in Figure 20 show that strong structural similarities can lead to matches between exemplars of object classes that may seem to have nothing in common, such as a fish and a knife, or a guitar and a bird.

It should be noted that the inherent ligature-induced instability of a shock graph suggests a matching framework that's many-to-many, rather than one-to-one. While a number of many-to-many graph matching frameworks do exist, e.g., [3, 8–10, 33], they often make restrictive assumptions (lack of occlusion or clutter, lack of edge attributes, etc.). We have therefore limited our study to one-to-one matching frameworks, although we expect that a many-to-many matching framework would benefit the matching of both shock graphs and bone graphs.

Finally, while our matching framework was developed to match bone graphs, it is applicable to other classes of directed acyclic graphs. The match-



(a) Correct match for the class walking human



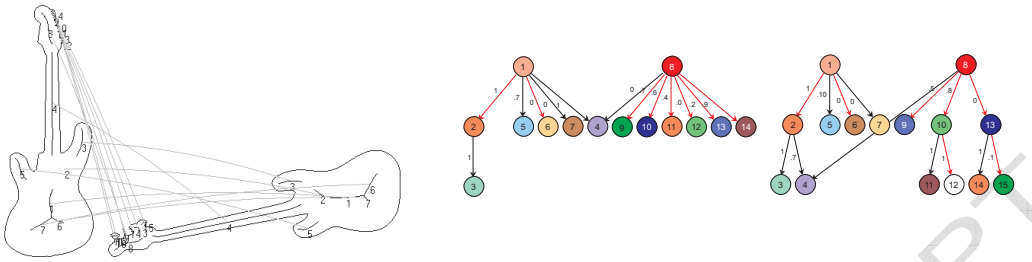
(b) Correct match for the class standing bird

Figure 16: Successful Matching Examples for Bone Graphs and Matching Algorithm B. In each pair of shapes and graphs, the one on the left is the query view and the one on the right is the best ranked model view. The part correspondences found by the matcher are illustrated as arcs between each pair of shapes, and as matching node colors between each pair of graphs. The number of each bone graph node corresponds to the shape part represented by it.

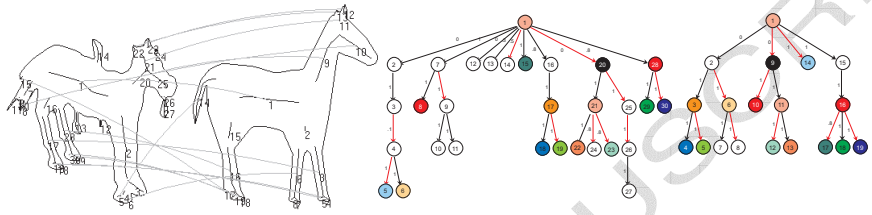
ing algorithm only demands node and edge similarity functions in order to compare the domain-dependent attributes of nodes and edges, and a set of penalty weights for the possible violation of hierarchical relations in the assignment of node correspondences. The penalty weights are also domain-dependent, and depend on the amount of noise in the graphs and the expected structural graph stability in the representation of similar objects.

5. Conclusions

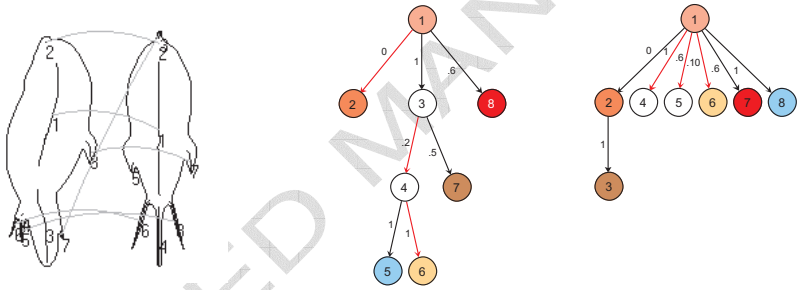
In order to exploit all the information encoded by a bone graph when comparing shapes, we explore the problem of matching directed acyclic graphs



(a) Correct match for the class electrical guitar



(b) Correct match for the class horse

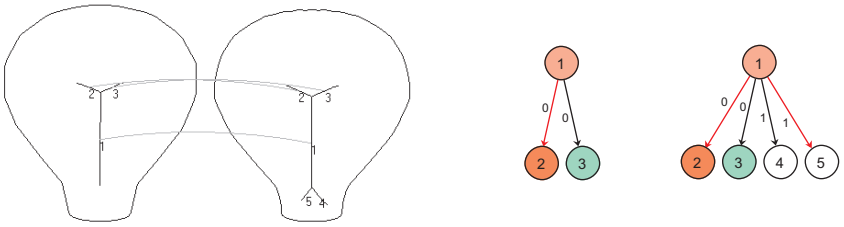


(c) Correct match for the class fish

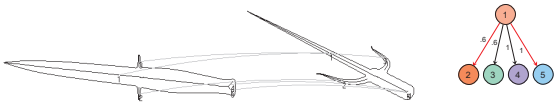
Figure 17: More Successful Matching Examples for Bone Graphs. See Figure 16 for details.

(DAG) with arbitrary sets of attributes for both nodes and edges. We build on our previous work on matching hierarchical structures in the presence of noise, occlusion and clutter. We propose a novel generalization of the graph matching approach of Shokoufandeh et al. [37] that incorporates edge information, expands the range of domain constraints considered, and ensures (if desired) that node correspondences do not violate the ancestor/descendant relations between nodes. This framework is a contribution to the problem of matching *generic* DAGs, and therefore can be applied to a multitude of problems in pattern recognition.

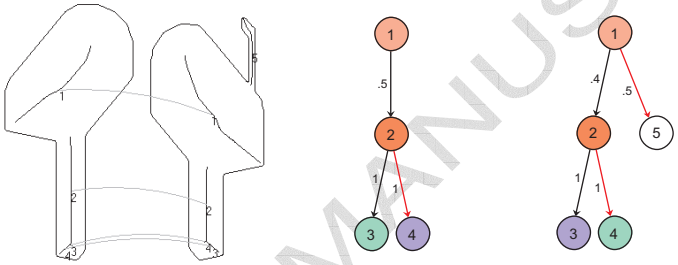
Our new matching algorithm applied to the domain of bone graphs forms



(a) Correct match for the class hot air balloon



(b) Correct match for the class knife



(c) Correct match for the class mailbox

Figure 18: More Successful Matching Examples for Bone Graphs. See Figure 16 for details.

a coherent framework for view-based object recognition, which we evaluate for the task of object categorization. This task is a central problem in computer vision and requires an approach that can accommodate the within-category shape variation of object views. We compare the bone graph against the shock graph by matching them using both the algorithm of Shokoufandeh et al. [37] and our proposed generalization of this algorithm. This provides a relevant comparison since both representations take the same shape skeletons as input and yield graph-based encodings of their skeletal parts as output. This means that for the bone graph to outperform the shock graph, its structure and attributes must be more beneficial for matching than those of the shock graph. Our experimental results show that the bone graph significantly outperforms the shock graph for the object categorization task, and that the additional structural constraints exploited by our graph matching

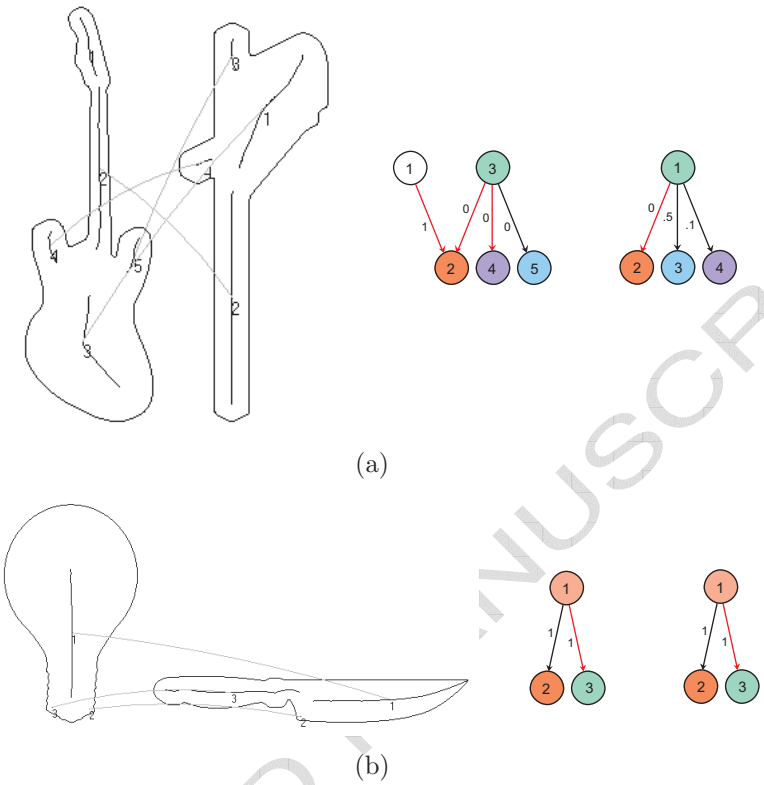


Figure 19: Incorrect Matching Examples for Bone Graphs and Matching Algorithm B. Here we show two cases in which the graph structures and edge attributes are very similar, even though the shapes are not. The strong structural and edge attribute similarities in these cases help the wrong model exemplars rank well by compensating for the dissimilarity between the node attributes.

algorithm improve the recognition performance of both representations. The bone graph also leads to better computational performance as it represents the object silhouettes using, on average, half the number of nodes than the shock graph.

The experiments in Section 4 also evaluate whether the information provided by the edge attributes of the bone graph contribute to improving recognition performance. The results suggest that edge attributes improve performance in the absence of node attributes, but that when both node and edge attributes are considered, the edge attributes do not offer a significant benefit. A possible explanation for this is that the parameters that determine the

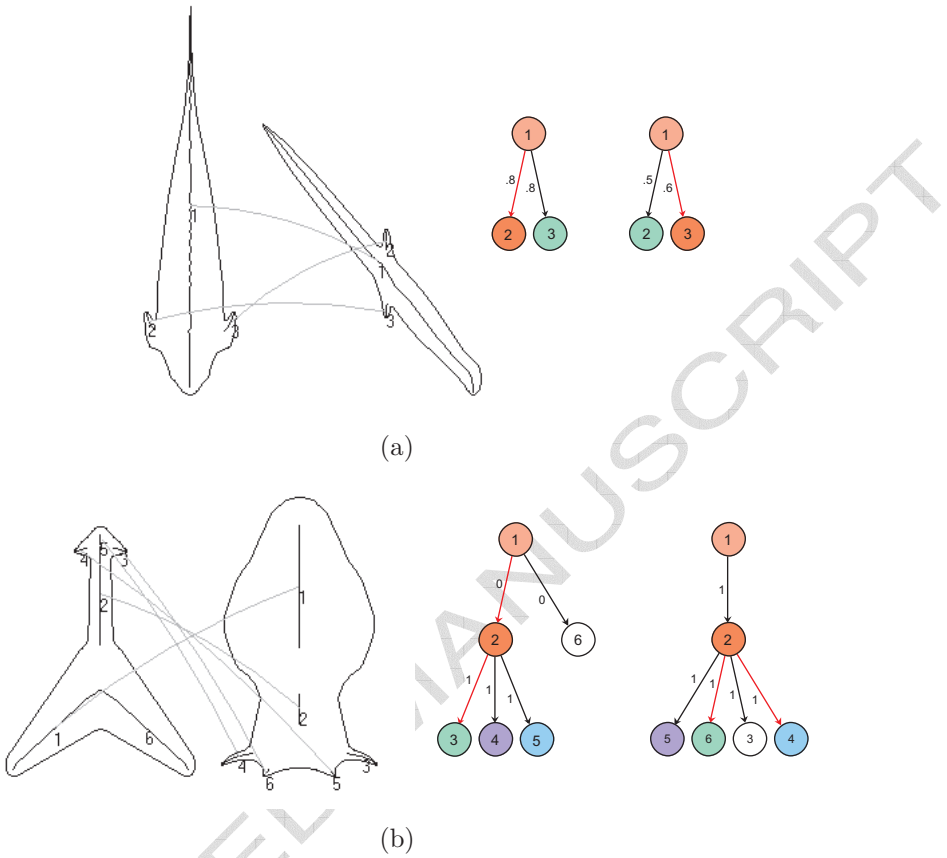


Figure 20: Incorrect Matching Examples for Bone Graphs and Matching Algorithm B. Here we show more cases in which two views of conceptually dissimilar classes have similar part structures.

penalty for edge attribute dissimilarities are sensitive to the category of the model being matched. That is, the positions and sides of part attachments (i.e., the edge attributes) may provide discriminating features for some object classes but not for others, which can lead to an over- or under-penalization of dissimilarities. We have found evidence of this phenomenon when analyzing a significant number of matching cases in our experiments.

A limitation of our view-based object recognition framework is that the rules to detect protrusions, discussed in [23], may fail to capture the part variability of some objects. When this problem affects the decomposition of

large shape parts (of similar silhouettes), it can lead to bone graphs with significantly different structures. Because the structures of the graphs are used to constrain the assignment of node correspondences, the matching algorithm cannot correct this type of parsing errors. A possible solution to this problem is to perform a denser sampling of the viewing sphere of each model object in order to ensure that all the representational variations of its views have a representative in the model database. Since this can lead to a large number of redundant model views, it must be integrated with a clustering procedure to find and eliminate views that are too similar. Another solution is to represent each shape using multiple bone graphs computed by varying the parameters that control the detection of protrusions. The disadvantage of this solution is its high computational cost, since a pair of query and model views would be represented by a multitude of graphs, which must be matched against each other.

6. Acknowledgements

The authors would like to thank Allan Jepson and Ali Shokoufandeh for their thoughtful feedback and support over the course of this work. The authors would also like to gratefully acknowledge the support of OCE, NSERC, CIFAR, and DARPA.

References

[1] X. Bai and L. J. Latecki. Path similarity skeleton graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1282–1292, 2008.

[2] E. Baseski, A. Erdem, and S. Tari. Dissimilarity between two skeletal trees in a context. *Pattern Recognition*, 42(3):370–385, 2009.

[3] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.

[4] W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:749–764, 1995.

[5] S. D. Cohen and L. J. Guibas. The earth mover's distance under transformation sets. In *International Conference on Computer Vision*, pages 1076–1083, Kerkyra, Greece, 1999.

[6] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[7] D. Conte, P. Foggia, C. Sansone, and M. Vento. How and why pattern recognition and computer vision applications use graphs. *Applied Graph Theory in Computer Vision and Pattern Recognition*, 52:85–135, April 2007.

[8] F. Demirci, B. Platel, A. Shokoufandeh, L. Florack, and S. Dickinson. The representation and matching of images using top points. *Journal of Mathematical Imaging and Vision*, 35(2):103–116, 2009.

[9] F. Demirci, A. Shokoufandeh, and S. Dickinson. Skeletal shape abstraction from examples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):944–952, May 2009.

[10] F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, and S. Dickinson. Object recognition as many-to-many feature matching. *International Journal of Computer Vision*, 69(2):203–222, August 2006.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum-likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 39:1–38, 1977.

[12] M.A. Eshera and K.S. Fu. A similarity measure between attributed relational graphs for image analysis. In *International Conference on Pattern Recognition*, pages 75–77, 1984.

[13] M. Fischler and R. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22:67–92, 1973.

[14] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph matching problems. *Journal of the ACM*, 38:815–853, 1991.

[15] M. Garey and D. Johnson. *Computer and Intractability: A Guide to the Theory of NP-Completeness*. Freeman: San Francisco, 1979.

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

[17] A. Goralcikova and V. Konbek. A reduct and closure algorithm for graphs. *Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science 74:301–307, 1979.

[18] W.E.L. Grimson and T. Lozano-Pérez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(9):469–482, 1987.

[19] B. Huet and E. R. Hancock. Shape recognition from large image libraries by inexact graph matching. *Pattern Recognition Letters*, 20:1259–1269, 1999.

[20] J. Kittler and E. R. Hancock. Combining evidence in probabilistic relaxation. *International Journal of Pattern Recognition and Artificial Intelligence*, 3:29–51, 1989.

[21] J. Kobler. *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser: Boston, 1993.

[22] B. Luo and E. R. Hancock. Structural graph matching using the em algorithm and singular value decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:1120–1136, 2001.

[23] D. Macrini, S. Dickinson, D. Fleet, and K. Siddiqi. Bone graphs: Medial shape parsing and abstraction. *Computer Vision and Image Understanding, Special Issue on Graph-Based Representations*, to appear, 2011.

[24] D. Macrini, A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker. View-based 3-D object recognition using shock graphs. In *International Conference on Pattern Recognition*, pages 24–28, Quebec City, August 2002.

[25] D. Macrini, K. Siddiqi, and S. Dickinson. From skeletons to bone graphs: Medial abstraction for object recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, Anchorage, Alaska, June 2008.

[26] Diego Macrini. Indexing and matching for view-based 3-d object recognition using shock graphs. Master's thesis, University of Toronto, 2003.

[27] E. Mjolsness, G. Gindi, and P. Anandan. Optimization in model matching and perceptual organization. *Neural Computation*, 1:218–229, 1989.

[28] R. Myers, R. C. Wilson, and E. R. Hancock. Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:628–635, 2000.

[29] M. Pelillo, K. Siddiqi, and S. Zucker. Attributed tree matching and maximum weight cliques. *International Conference on Image Analysis and Processing*, September 1999.

[30] M. Pelillo, K. Siddiqi, and S. Zucker. Continuous-based heuristics for graph and tree isomorphisms. *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*, 1999.

[31] M. Pelillo, K. Siddiqi, and S.W. Zucker. Many-to-many matching of attributed trees using association graphs and game dynamics. In *International Workshop on Visual Form*, pages 583–593, 2001.

[32] 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.

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

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

[35] Thomas Sebastian, Philip Klein, and Benjamin Kimia. Recognition of shapes by editing shock graphs. In *International Conference on Computer Vision*, pages 755–762, 2001.

[36] Philip Shilane, Patrick Min, Michael Kazhdan, and Thomas Funkhouser. The princeton shape benchmark. In *Shape Modeling International*, Genova, Italy, June 2004.

[37] A. Shokoufandeh, L. Bretzner, D. Macrini, M.F. Demirci, C. Jonsson, and S. Dickinson. The representation and matching of categorical shape. *Computer Vision and Image Understanding*, 103:139–154, 2006.

[38] A. Shokoufandeh and S. Dickinson. A unified framework for indexing and matching hierarchical shape structures. In *International Workshop on Visual Form*, pages 28–46, Capri, Italy, May 2001.

[39] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S. Zucker. Indexing hierarchical structures using graph spectra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1125–1140, July 2005.

[40] K. Siddiqi, A. Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker. Shock graphs and shape matching. In *International Conference on Computer Vision*, pages 222–229, 1998.

[41] K. Siddiqi, A. Shokoufandeh, Sven J. Dickinson, and Steven W. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 35(1):13–32, 1999.

[42] Andrea Torsello and Edwin R. Hancock. Efficiently computing weighted tree edit distance using relaxation labeling. In *EMMCVPR*, pages 438–453, 2001.

[43] S. Umeyama. An eigen decomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703, September 1988.

[44] M. van Eede, D. Macrini, A. Telea, C. Sminchisescu, and S. Dickinson. Canonical skeletons for shape matching. In *International Conference on Pattern Recognition*, Hong Kong, August 2006.

[45] R. C. Wilson and E. R. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:634–648, 1997.

[46] Song Chun Zhu and A. L. Yuille. Forms: A flexible object recognition and modeling system. *International Journal of Computer Vision*, 20(3):187–212, 1996.