Matching and Embedding through Edit-Union of Trees

Andrea Torsello and Edwin R. Hancock

Dept. of Computer Science, University of York
Heslington, York, YO10 5DD, UK
atorsell@cs.york.ac.uk

Abstract. This paper investigates a technique to extend the tree edit distance framework to allow the simultaneous matching of multiple tree structures. This approach extends a previous result that showed the edit distance between two trees is completely determined by the maximum tree obtained from both tree with node removal operations only. In our approach we seek the minimum structure from which we can obtain the original trees with removal operations. This structure has the added advantage that it can be extended to more than two trees and it imposes consistency on node matches throughout the matched trees. Furthermore through this structure we can get a "natural" embedding space of tree structures that can be used to analyze how tree representations vary in our problem domain.

1 Introduction

Recently, there has been considerable interest in the structural abstraction of 2D shapes using shock-graphs [11,17,19]. The shock-graph is a characterization of the differential structure of the boundaries of 2D shapes. It is constructed by labeling points on the Blum skeleton of an object. This is done by measuring the rate of change with skeleton length of the radius of the maximal circle inscribed within the object boundary. The shock labels indicate whether the radius is increasing, decreasing, locally maximum or minimum. These cases correspond to situations where the object boundary is tapering, constricting or a local bulge. Recently, several authors have shown how to recognize shapes by matching shock-graphs. For instance Pelillo, Siddiqi and Zucker [14] have used a novel relaxation labeling method inspired by evolutionary game-theory. Sebastian, Kimia, and Klein [12], have performed matching using graph-edit distance.

Although graph-matching allows the pairwise comparison of shock-graphs, it does not allow the shape-space of shock-graphs to be explored in detail. Graph-matching may provide a fine measure of distance between structures, and this in turn may be used to cluster similar graphs. However, it does not result in an ordering of the graphs that has metrical significance under structural variations due to graded shape-changes. This latter approach has been at heart of recent developments in the construction of deformable models and continuous shape-spaces [10]. Here shape variations have been successfully captured

A. Heyden et al. (Eds.): ECCV 2002, LNCS 2352, pp. 822–836, 2002.

by embedding them in a vector-space. The dimensions of this space span the modes of shape-variation. For instance, Cootes and Taylor [7] have shown how such shape-spaces can be constructed using the eigenvectors of a landmark covariance matrix. Sclaroff and Pentland [15], on the other hand, use the elastic modes of boundaries to define the shape-space.

In this paper we take the view that although the comparison of shock-graphs, and other structural descriptions of shape, via graph matching or graph edit distance has proved effective, it is in some ways a brute-force approach which is at odds with the non-structural approaches to recognition which have concentrated on constructing shape-spaces which capture the main modes of variation in object shape. Hence, we aim to address the problem of how to organize shock-graphs into a shape-space in which similar shapes are close to one-another, and dissimilar shapes are far apart. In particularly, we aim to do this in a way such that the space is traversed in a relatively uniform manner as the structures under study are gradually modified. In other words, the aim is to embed the graphs in a vector-space where the dimensions correspond to principal modes in structural variation. There are a number of ways in which this can be achieved. The first is to compute the distance between shock-graphs and to use multidimensional scaling to embed the individual graphs in a low-dimensional space [22]. However, as pointed out above, this approach does not necessarily result in a shape-space where the dimensions reflect the modes of structural variation of the shock-graphs. The second approach is to extract feature vectors from the graphs and to use these as a shape-space representation. A shape-space can be constructed from such vectors by performing modal analysis on their covariance matrix. However, when graphs are of different size, then the problem of how to map the structure of a shock-graph to a vector of fixed length arises. This problem can be circumvented using graph spectral features. This approach is explored in a companion paper.

In this paper we take a different approach to the problem. We aim to embed shock trees in a pattern space by mapping them to vectors of fixed length. We do this as follows. We commence from a set of shock-trees representing different shapes. From this set, we construct a super-tree from which each tree may be obtained by the edit operations of node and edge removal. Hence each shock-tree is a subtree of the super-tree. The super-tree is constructed so that it minimizes the total edit distance to the set of shock-trees. To embed the individual shock-trees in a vector-space we allow each node of the super-tree to represent a dimension of the space. Each shock-tree is represented in this space by a vector which has non-zero components only in the directions corresponding to its constituent nodes. The non-zero components of the vectors are the weights of the nodes. In this space, the edit distance between trees is the L1 norm between their embedded vectors.

Hence the shape-space is arrived at by construction from the individual shock-trees. An important ingredient of this construction is the way in which we impose consistency on the separate trees. By constructing the tree-union, we impose edge-consistency constraints on the individual node correspondences

among the individual constituent trees. The consequence of this is that the L1 norm between embedded vectors provides a more accurate representation of the distribution of trees in shape-space. We exploit this property to locate clusters of trees.

We experiment with our new method on sets of shock-trees. Here we perform multidimensional scaling on the distances between trees. We demonstrate that the clusters delivered by the L1 norms between trees in the vector-space are better than those which result if we use the raw edit distances between pairs of trees.

The proposed method provides a route to graph clustering. This problem has been addressed previously through pairwise clustering [13], but is based on a coarse measure of similarity. The proposed method extracts a more detailed model of the structure present in the graphs. It is worth mentioning similar work by Shokoufandeh et al. where graphs are embedded using spectral encoding [16].

2 Shock Tree

The practical problem tackled in this paper is the clustering of 2D binary shapes based on the similarity of their shock-trees. The idea of characterizing boundary shape using the differential singularities of the reaction equation was first introduced into the computer vision literature by Kimia, Tannenbaum and Zucker [11]. The idea is to evolve the boundary of an object to a canonical skeletal form using the reaction-diffusion equation. The skeleton represents the singularities in the curve evolution, where inward moving boundaries collide. The reaction component of the boundary motion corresponds to morphological erosion of the boundary, while the diffusion component introduces curvature dependent boundary smoothing. Once the skeleton is to hand, the next step is to devise ways of using it to characterize the shape of the original boundary. Here we follow Zucker, Siddiqi, and others, by labeling points on the skeleton using so-called shock-classes [19]. According to this taxonomy of local differential structure, there are different classes associated with behavior of the radius of the maximal circle from the skeleton to the nearest pair of boundary points. The so-called shocks distinguish between the cases where the local maximal circle has maximum radius, minimum radius, constant radius or a radius which is strictly increasing or decreasing. We abstract the skeletons as trees in which the level in the tree is determined by their time of formation [16,19]. The later the time of formation, and hence their proximity to the center of the shape, the higher the shock in the hierarchy.

3 Error-Tolerant Matching of Trees and Edit-Distance

The problem of how to measure the similarity of pictorial information which has been abstracted using graph-structures has been the focus of sustained research activity for over twenty years in the computer vision literature. Moreover, the problem has recently acquired significant topicality with the need to develop

ways of retrieving images from large data-bases. Stated succinctly, the problem is one of inexact or error-tolerant graph-matching. Early work on the topic included Barrow and Burstall's idea [1] of locating matches by searching for maximum common subgraphs using the association graph, and the extension of the concept of string edit distance to graph-matching by Fu and his co-workers [8]. The idea behind edit distance [23] is that it is possible to identify a set of basic edit operations on nodes and edges of a structure, and to associate with these operations a cost. The edit-distance is found by searching for the sequence of edit operations that will make the two graphs isomorphic with one-another and which has minimum cost. The set of edit operations can be problem specific, but a common choice is:

- *node removal:* remove a node and link the children to its parent.
- *node insertion:* the dual of node removal
- *node relabel:* change the label associated to a node.

By making the evaluation of structural modification explicit, edit distance provides a very effective way of measuring the similarity of relational structures. Moreover, the method has considerable potential for error tolerant object recognition and indexing problems.

Unfortunately, the task of calculating edit distance is a computationally hard problem and most early efforts can be regarded as being goal-directed. Zhang and Shasha [27] have investigated a specialization of the tree edit distance problem, which involves adding the constraint that the solution must maintain the order of the children of a node. With this order among siblings, they showed that the tree-matching problem is in P and gave an algorithm to solve it. In subsequent work they showed that the more general unordered case was indeed an NP hard problem [28]. Among other approaches we mention the ones developed by Pelillo et al. [14], and by Bartoli et al. [2], in which the graph theoretic notion of a path string is used to transform the tree isomorphism problem into a single max clique problem.

There is, however, a strong connection between the computation of maximum common subtree and the tree edit distance. In [5] Bunke showed that, under certain constraints applied to the edit-cost function, the maximum common subgraph problem and the graph edit distance problem are computationally equivalent. This is not directly true for trees, because of the added constraint that a tree must be connected. But, extending the concept to the common edited subtree, we can use common substructures to find the minimum cost edited tree isomorphism.

Hierarchical graphs have an order relation induced by paths: given two nodes a and b, (a, b) is in this relation if and only if there is a path from a to b. When the directed graph is acyclical, this relation can be shown to be an order relation. The requirement that matches respect this relation and that the edited trees be connected, prevent us from applying Bunke's result directly to tree matching and search for a common subgraph.

The constraint to the edit-cost function proposed by Bunke in [5] is that the cost of deleting and reinserting the same element with a different label is

not greater than the cost of relabeling it. In this way we can find an optimal edit sequence without the need for a relabel operation. With this constraint on relabel cost, we are left with only node removal and node insertion operations to be performed on the data tree. Since a node insertion on the data tree is dual to a node removal on the model tree, we can further reduce the number of operations to be performed to only node removal, as long as we perform the operations on both trees.

3.1 Inexact Tree Matching as a Common Substructure Problem

Transforming node insertions in one tree into node removals in the other allows us to use only structure reducing operations. This, in turn, means that the optimal matching is completely determined by the subset of nodes left after the minimum removal sequence. Hence, we can pose the edit distance problem as a particular substructure isomorphism problem. Since node removal operations respect the order relation implicit in the hierarchy, we can reduce the substructure isomorphism problem into subproblems in a divide and conquer approach [22]. This approach derives from a number of observations. First, given two trees t_1 and t_2, there are two subtrees t_1' and t_2' rooted at nodes v and v' such that the matching induced by edit distance between those two nodes is equivalent to that obtained with the original trees. Hence the best match between t_1 and t_2 is equivalent to the best match given the association of root nodes (v, v'). Furthermore, this match can be found examining only descendents of v and v'. If we can express the best match given the association of root nodes (v, v') as a function of lower level matches of descendants of v and v', we can build the solution to our matching problem bottom up. In the following sections we will how we can find the best match using the lower level matches.

3.2 Editing the Transitive Closure of a Tree

In this section we show the relations between the graph theoretic concept of transitive closure of a directed acyclic graph and the edit distance. An immediate remark is that for each node removal operation E_v removing node v from the tree t, we can define the corresponding edit operation $\mathcal{E}_v$ on the closure $\mathcal{C}t$ of the tree t. In both cases the edit operation removes the node v, all the incoming edges, and all the outgoing edges. It is important to note that the transitive closure operation and the node removal operation commute, that is we have:

Lemma 1. $\mathcal{E}_v(\mathcal{C}(t)) = \mathcal{C}(E_v(t))$

We call a subtree s of $\mathcal{C}t$ *obtainable* if for each node v of s if there cannot be two children a and b so that (a, b) is in $\mathcal{C}t$. In other words, given two nodes a and b, siblings in s, s is obtainable if and only if there is no path from a to b in t. We can, now, prove the following:

Theorem 1. *A tree $\hat{t}$ can be generated from a tree t with a sequence of node removal operations if and only if $\hat{t}$ is an obtainable subtree of the directed acyclic graph $\mathcal{C}t$.*

Using this result, we can show that the minimum cost edited tree isomorphism between two trees t and t' is, structurally, a common consistent subtree of the two directed acyclic graphs $\mathcal{C}t$ and $\mathcal{C}t'$. The minimum cost edited tree isomorphism is a tree that can be obtained from both model tree t and data tree t' with node removal operations. By virtue of the theorem above, this tree is an obtainable subtree of both $\mathcal{C}t$ and $\mathcal{C}t'$, furthermore, the tree must be generated with minimum combined edit cost.

3.3 Cliques and Common Obtainable Subtrees

In this section we show how to use these results to induce a divide and conquer approach to edited tree matching. Given two trees t and t' to be matched, we calculate the transitive closures $\mathcal{C}t$ and $\mathcal{C}t'$ and look for a common obtainable tree that induces the optimal matches. The maximum such tree corresponds to the maximum common obtainable subtree of $\mathcal{C}t$ and $\mathcal{C}t'$.

For each pair of nodes v and w of the two trees to be matched, we divide the problem into a maximum common obtainable subtree rooted at v and w. That is, we fix the matching of v to w and we search for the maximum edited subtree common to the two subtrees rooted at v and w. We show that, given the cardinality of the subtree rooted at each child of v and w, we can transform the search for the maximum common substructure into the search for a max weighted clique. Solving this problem for each pair of nodes, and looking for the maximum among each node pair, we can find the isomorphism linked to the minimum edit distance.

We commence by transforming the problem from the search of the minimum edit cost linked to the removal off some nodes, to the maximum of a utility function linked to the nodes that are retained. To do this we assume that we have a weight w_i assigned to each node i, that the cost of matching a node i to a node j is $|w_i - w_j|$, and that the cost of removing a node is equivalent to matching it to a node with weight 0. We define the set $M \subset \mathcal{N}^t \times \mathcal{N}^{t'}$ the set of pair of nodes in t and t' that match, the set $L^t = \{i \in \mathcal{N}^t | \forall x, < i, x > \notin M\}$ composed of nodes in the first tree that are not matched to any node in the second, and the set $R^{t'} = \{j \in \mathcal{N}^{t'} | \forall x, < x, j > \notin M\}$, which contains the unmatched nodes of the second set. With these definitions, we can write:

$$d(t,t') = \sum_{i \in L^t} w_i + \sum_{j \in R^{t'}} w_j + \sum_{<i,j> \in M} |w_i - w_j| = \\ = \sum_{i \in \mathcal{N}^t} w_i + \sum_{j \in \mathcal{N}^{t'}} w_j - 2 \sum_{<i,j> \in M} \min(w_i, w_j). \quad (1)$$

We call the quantity

$$\mathcal{U}(M) = \sum_{<i,j> \in M} \min(w_i, w_j).$$

the *utility* of the match M. Clearly the match that maximizes the utility minimizes the edit distance.

Let us assume that we know the utility of the best match rooted at every descendent of v and w. We aim to find the set of siblings with greatest total utility. To do this we make use of a derived structure similar to the association graph introduced by Barrow in [1]. The nodes of this structure are pairs drawn from the Cartesian product of the descendents of v and w and each pair correspond to a particular association between a node in one tree to a node in the other. We connect two such associations if and only if there is no inconsistency between the two associations, that is the corresponding subtree is obtainable. This means that we connect nodes (p, q) and (r, s) if and only if there is no path connecting p and r in t and there is no path connecting q and s in t'. Furthermore, we assign to each association node (a, b) a weight equal to the utility of the best match rooted at a and b. The maximum weight clique of this graph is the set of consistent siblings with maximum total utility, hence the set of children of v and w that guarantee the optimal isomorphism. The utility of the maximum common consistent subtree rooted at v and w will be the weight of the isomorphisms rooted at the children of v and w plus the contribution of v and w, that is the weight of the clique plus the minimum of the weights of v and w. Given a method to obtain a solution for the maximum weight clique problem, we can use it to obtain the solution to our isomorphism problem. We refer to [3,22] for heuristics for the weighted clique problem.

4 Edit-Intersection and Edit-Union

In the previous sections we have seen how the edit distance between two trees is completely determined by the set of nodes that do not get removed by edit operations and, therefore, get matched. That is, in a sense, we are taking the intersection of the sets of nodes of the two structures. With this approach we match the trees by extracting a structure that can be obtained from our original trees by removing some nodes. We have seen how the edit distance between two trees is related to this intersection structure (see Figure 1). We would like to extend the concept to more than two trees so that we can compare a shape tree to a whole set T of trees. Moreover, this allows to determine how a new sample relates to a previous distribution of tree structures. Formally, we would like to find the match that minimizes the sum of the edit distances between the new tree t^* and each tree $t \in T$, with the added constraint that if node a in the new tree t^* is matched to node b in a tree $t_1 \in T$ and to node c in another tree $t_2 \in T$, then b must be matched to c, i.e.

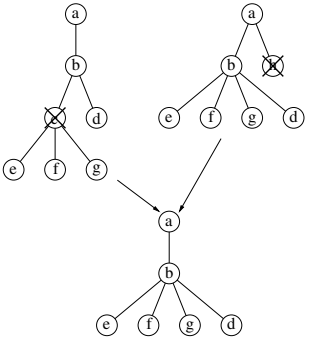

Fig. 1. Edit-Intersection of two trees.

$$< a, b >\in M \wedge < a, c >\in M \Rightarrow < b, c >\in M,$$

were M is the "matches to" relation on nodes. To find this match we could find the maximum substructure that can be obtained from any tree in a set removing appropriate nodes, unfortunately not keeping unmatched nodes we are dropping too much information: the set of common nodes becomes marginal and we lose information about how the nodes distribute in the various structures. To use Bunke's analogy [5], the maximum common substructure give us information about mean of the trees, but it completely drops any information about how sample trees distribute around this mean. To overcome this limitation we can calculate a union of the nodes: a structure from which we can obtain any tree in our set removing appropriate nodes, as opposed to the intersection of nodes, which is a structure that can obtained removing nodes from the original trees.

Any such structure has the added advantage of implicitly creating an embedding space for our trees: we are guaranteed that any node in any tree matches to a node in this structure. Assigning to each node a coordinate in a vector space V, we can associate to tree t the vector $v \in V$ so that $v_i = w_i$, where w_i in the weight of the node of t associate with coordinate i.

4.1 Union of Two Trees

Once more, the edit-union of two trees is completely determined by the set of matched nodes. Start with the two trees and iteratively merge nodes that are matched. At the end you end up with a directed acyclical graph with multiple paths connecting various nodes (see Figure 2). This structure, thus, has more

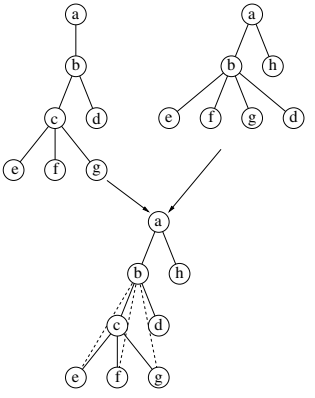

Fig. 2. Edit-Union of two trees.

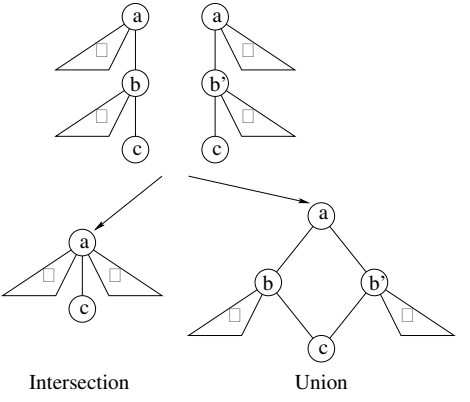

Fig. 3. Edit-Union is not always a tree.

links than necessary and you cannot obtain the first tree by node removal operations alone: you need to remove the superfluous edges. If in the figure we eliminate the edges connecting node b to nodes e,f and g, we obtain a tree. Starting from this tree we can obtain either one of the original trees by node

removal operations alone. Furthermore, the transitive closure of this tree and the closure of the unaltered structure are identical.

One would hope that such a structure always be a tree, so that we can use the matching technique already described to compare a tree to a group of trees. Unfortunately, it is not always possible to find a tree such that we can edit it to obtain the original trees: see, for example, Figure 3. In this figure α and β are subtrees. Because of the constraints posed by matching trees α and trees β, nodes b and b' cannot be matched and neither b can be a child of b' nor b' a child of b. The only alternative is to keep the two alternative paths separate: this way we can obtain the first tree removing the node b' and the second removing b. Actually, removing the nodes is not enough, shortcutting edges need to be removed, but, once again, the transitive closure of the union minus node b' is identical to the closure of the first tree.

4.2 Matching a Tree to a Union

As seen, in general the union of two trees is a directed acyclical graph, and our approach can only match trees, and would fail on structures with multiple paths from one node a to node b, since it would count any match in the subtree rooted at b twice. Hence, given a joined representation we cannot directly use our approach to compare a tree to a tree set or a distribution of trees.

Fortunately, we don't need to perform a generic match between two generic directed acyclic graphs: in an edit-union we have multiple paths between node a and node b, but each tree can have only one, hence multiple paths are mutually exclusive. If we constrain our search to match nodes in only one path and we match the union to a tree, we are guaranteed not to count the same subtree multiple times. Interestingly, this constraint can be merged with the obtainability constraint: we say that a match is *obtainable* if for each node v there cannot be two children a and b and a node c so that there is a path, possibly of length 0, from a to c and one from b to c. This constrain reduces to obtainability for trees when $c = b$, but it also prevents a and b from belonging two to separate paths joining at c. Hence from a node where multiple paths fork, we can extract children matches from one path only.

We want to find the match consistent with the obtainability constraint that minimizes the sum of the edit distance between the new tree and each tree in the set. To this effect we can maximize the sum of the utilities.

$$\mathcal{U}(M) = \sum_{t' \in T} \sum_{<i,j> \in M} \min(w_i^t, w_j^{t'}).$$

Where $M \subset \mathcal{N}^t \times \mathcal{N}^{T^\cup}$ is the set of matches between the nodes $\mathcal{N}^t$ of the tree t and the nodes $\mathcal{N}^{T^\cup}$, $T^\cup$ is the union structure of the set of trees T, and w_i^t is the weight assigned in tree t to node i. When the edit distance is uniform (all weights are 1), the total utility of a single match is equal to the number of trees that have an instance of that node (see Figure 4). Solving the modified weighted clique problems we obtain the correspondence between the nodes in the

new tree and the nodes in each tree in the set, while the edit distance obtained is the sum of the distances from the new tree to each tree in T.

To be able to calculate this quantity we keep, for each node in the union structure, the weights of the matched nodes. A way to do this is to assign to each node in the union a vector of dimension equal to the number of trees in the set. The ith coordinate of this vector will be the weight of the corresponding node in the ith tree, 0 if the ith tree doesn't have a node matching to the current node. This representation also allows us to easily obtain the coordinate of each tree in the set in the embedding space induced by the union: the ith weight of the jth node is the jth coordinate of the ith tree. Such embedding is defined modulo reordering of trees and nodes.

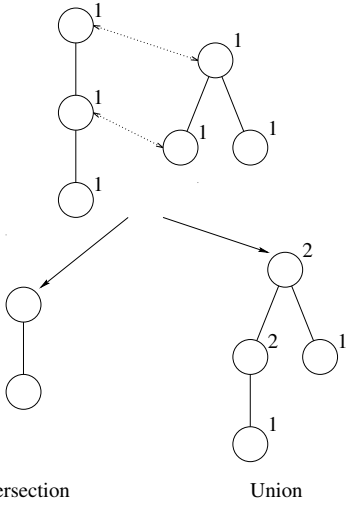

Fig. 4. The weight of a node in the union account for every node mapped to that node.

It is worth noting that this approach can be extended to match two union structures, as long as at most one has multiple paths to a node. To do this we iterate through each pair of weights drawn the two sets, that is we define the utility as:

$$\mathcal{U}(M) = \sum_{t \in T_1, t' \in T_2} \sum_{<i,j> \in M} \min(w_i^t, w_j^{t'}),$$

where $M \subset \mathcal{N}^{(T_1^{\cup})} \times \mathcal{N}^{(T_2^{\cup})}$ is the set of matches between the nodes of the union structures $T_1^{\cup}$ and $T_2^{\cup}$. The requirement that no more than one union has multiple paths to a node is required to avoid double counting.

4.3 Joining Multiple Trees

In the previous paragraph we have seen that we can construct the edit union of a set of trees and how to compare a tree to this superstructure. We now want to show how to construct such structure. Finding the super-structure that minimizes the total distance between trees in the set is computationally infeasible, but we propose a suboptimal iterative approach that at each iteration extends the union adding a new tree to it. This is done by matching the tree to the union and then using the matched nodes to construct the union of the two structures. That is, we pick a new tree t^* and we match it against the union $T^{\cup(t)}$, to obtain the union $T^{\cup(t+1)}$. We proceed until we have joined every tree.

In order to increase the accuracy of the approximation, we want to merge trees with smaller distance first. This is because we can be reasonably confident that, if the distance is small, the extracted correspondences are correct. We

could start with the set of trees, merge them and replace them with the union and reiterate until we end up with only one structure, i.e. at each iteration pick two trees t_1 and t_2 so that the distance $d(t_1, t_2)$ is minimal, merge the two tree and reinsert the union structure $t^{\cup 1,2}$ in our set of trees to be merged. Unfortunately, since we have no guarantees that the edit-union is a tree, we might end up trying to merge two graph with multiple paths to a node, and our matching algorithm cannot cope with that. For this reason, if merging two trees give a union that is not a tree, we discard the union and try with the next-best match. When no trees can be merged without duplicating paths, we merge the remaining structures using the first merging approach described. This way we are guaranteed to merge at each step at most one multi-path structure.

5 Experimental Results

We evaluate the new approach on the problem of shock tree matching. In order to asses the quality of the approach we compare the embedding obtained with those obtained using the graph clustering method described in [22]. In particular, we compare the the first two principal components of the embedding generated joining purely structural skeletal representations, with 2D multi-dimensional scaling of the pairwise distances of the shock-trees weighted with some geometrical information. The addition of matching consistency across shapes allows the embedding to better capture the structural information present in the shapes, yielding embeddings comparable, if not better, than those provided by localized geometrical information.

We run three experiments with 4, 5, and 9 shapes each. In each experiment the shapes belong to two or more distinct visual clusters. In order to avoid scaling effect due to difference in the number of nodes, we normalize the embedding vectors so that they have L1 norm equal to 1, and then we extract the first 2 principal components.

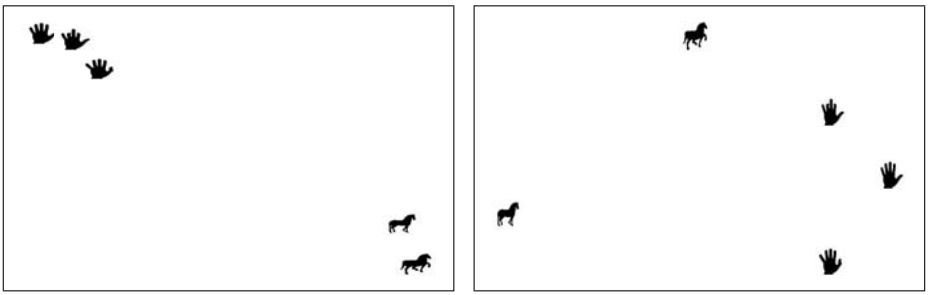

Fig. 5. Top: embedding through union. Bottom: multi-dimensional scaling of pairwise distance

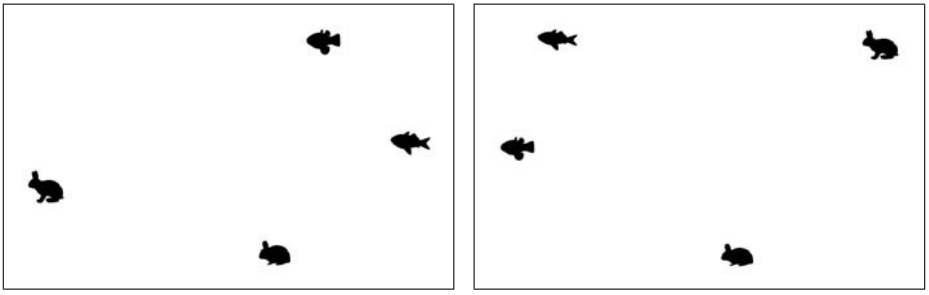

Fig. 6. Top: embedding through union. Bottom: multi-dimensional scaling of pairwise distance

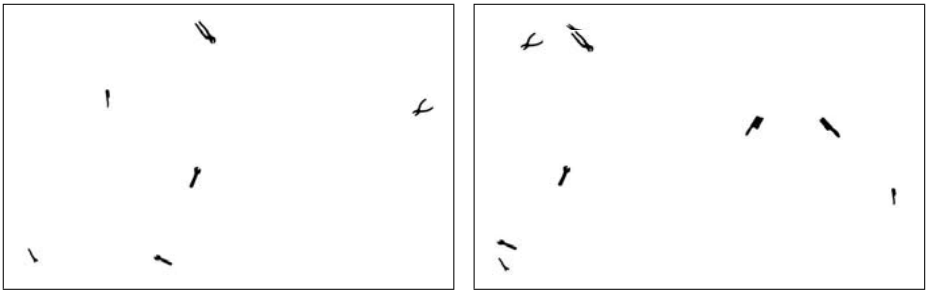

Fig. 7. Top: embedding through union. Bottom: multi-dimensional scaling of pairwise distance

Figure 5 shows a clear example where the embedding obtained through edit-union is better than that obtained through multi-dimensional scaling of the pairwise distances. In this case the pairwise distance algorithm consistently underestimates the distance between shapes belonging to different clusters. This is a general problem of pairwise matching: it works very well when the shapes are close, when the extracted correspondence is reliable, but as the shapes move further apart the advantage the correct correspondence has over alternative ones diminishes, until, eventually, another match is selected, which reports a higher distance. The result of this is a consistent underestimation of the distance the shapes move further apart in shape space. Figures 6 and 7 show examples where the distance in shape space is not big enough to observe the described behavior, yet the embedding obtained through union fares well against the pairwise edit-distance, especially taking into account the fact that it uses only structural information while the edit-distance matches shown weight the structure with geometrical information. In particular, Figure 7 shows a better ordering of shapes, with brushes being so tightly packed that they overlap in the image. It is interesting to note how the union embedding puts the monkey wrench (at the top) somewhere in-between pliers and wrenches: the algorithm is able to consistently match the head to the heads of the wrenches, and the handles to the handles of the pliers.

5.1 Synthetic Data

To augment these real world experiments, we have performed the embedding on synthetic data. The aim of the experiments is to characterize the ability of the approach to generate a shape space. To meet this goal we have randomly generated some prototype trees and, from each tree, we generated five or ten structurally perturbed copies. The procedure for generating the random trees was as follows: we commence with an empty tree (i.e. one with no nodes) and we iteratively add the required number of nodes. At each iteration nodes are added as children of one of the existing nodes. The parents are randomly selected with uniform probability from among the existing nodes. The weight of the newly added nodes are selected at random from an exponential distribution with mean 1. This procedure will tend to generate trees in which the branch ratio is highest closest to the root. This is quite realistic of real-world situations, since shock trees tend to have the same characteristic. To perturb the trees we simply add nodes with using the same approach.

In our experiments the size of the prototype trees varied from 5 to 20 nodes. As we can see from Figure 8, the algorithm was able to clearly separate the clusters of trees generated by the same prototype. Figure 8 shows three experiments with synthetic data. The images at the top are produced embedding 5 structurally perturbed trees per prototype: trees 1 to 5 are perturbed copies of

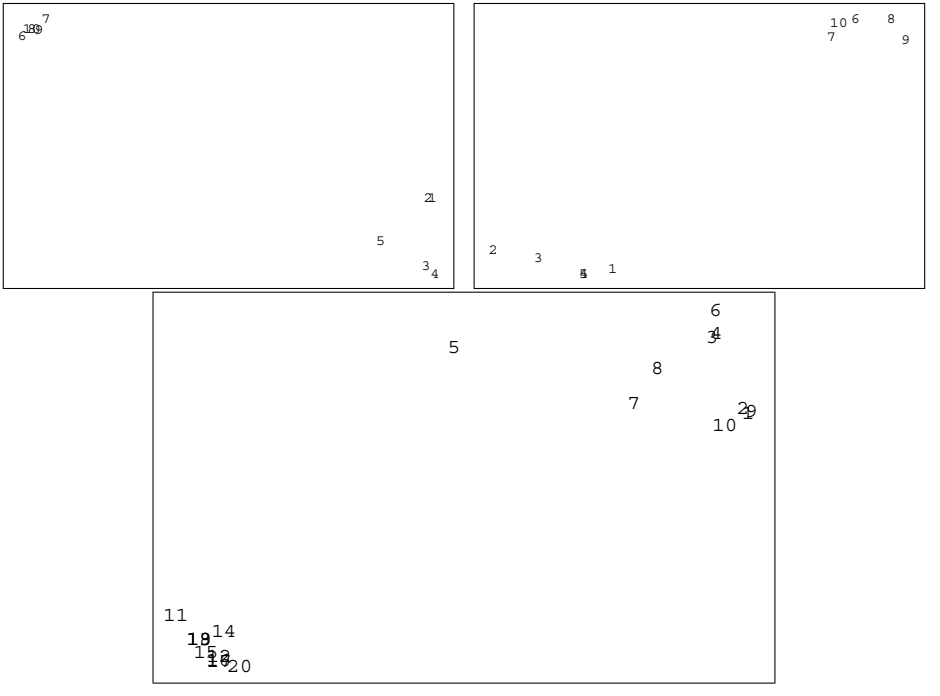

Fig. 8. Synthetic clusters

the first prototype, 6 to 10 of the second. The bottom image shows the result of the experiment with 10 structurally perturbed trees per prototype: 1 to 10 belong to one cluster, 11 to 20 to the other. In each image the clusters are well separated.

6 Conclusions

In this paper we investigated a technique to extend the tree edit distance framework to allow the simultaneous matching of multiple tree structures. With this approach we can impose a consistency of node correspondences between matches, avoiding the underestimation of the distance typical of pairwise edit-distances approaches. Furthermore through this methods we can get a "natural" embedding space of tree structures that can be used to analyze how tree representations vary in our problem domain.

In a set of experiments we apply this algorithm to match shock graphs, a graph representation of the morphological skeleton. The results of these experiments are very encouraging, showing that the algorithm is able to group similar shapes together in the generated embedding space.

References

1. H. G. Barrow and R. M. Burstall, Subgraph isomorphism, matching relational structures and maximal cliques, *Inf. Proc. Letter*, Vol. 4, pp.83, 84, 1976.
2. M. Bartoli et al., Attributed tree homomorphism using association graphs, In *ICPR*, 2000.
3. I. M. Bomze, M. Pelillo, and V. Stix, Approximating the maximum weight clique using replicator dynamics, *IEEE Trans. on Neural Networks*, Vol. 11, 2000.
4. H. Bunke and G. Allermann, Inexact graph matching for structural pattern recognition, *Pattern Recognition Letters*, Vol 1, pp. 245-253, 1983.
5. H. Bunke and A. Kandel, Mean and maximum common subgraph of two graphs, *Pattern Recognition Letters*, Vol. 21, pp. 163-168, 2000.
6. W. J. Christmas and J. Kittler, Structural matching in computer vision using probabilistic relaxation, *PAMI*, Vol. 17, pp. 749-764, 1995.
7. T. F. Cootes, C. J. Taylor, and D. H. Cooper, Active shape models - their training and application, *CVIU*, Vol. 61, pp. 38-59, 1995.
8. M. A. Eshera and K-S Fu, An image understanding system using attributed symbolic representation and inexact graph-matching, *PAMI*, Vol 8, pp. 604-618, 1986.
9. L. E. Gibbons et al., Continuous characterizations of the maximum clique problem, *Math. Oper. Res.*, Vol. 22, pp. 754-768, 1997
10. T. Heap and D. Hogg, Wormholes in shape space: tracking through discontinuous changes in shape, *ICCV*, pp. 344-349, 1998.
11. B. B. Kimia, A. R. Tannenbaum, and S. W. Zucker, Shapes, shocks, and deformations I, *International Journal of Computer Vision*, Vol. 15, pp. 189-224, 1995.
12. T. Sebastian, P. Klein, and B. Kimia, Recognition of shapes by editing shock graphs, in *ICCV*, Vol. I, pp. 755-762, 2001.
13. B. Luo, et al., A probabilistic framework for graph clustering, in *CVPR*, Vol. I, pp. 912-919, 2001.

14. M. Pelillo, K. Siddiqi, and S. W. Zucker, Matching hierarchical structures using association graphs, *PAMI*, Vol. 21, pp. 1105-1120, 1999.
15. S. Sclaroff and A. P. Pentland, Modal matching for correspondence and recognition, *PAMI*, Vol. 17, pp. 545-661, 1995.
16. A. Shokoufandeh, S. J. Dickinson, K. Siddiqi, and S. W. Zucker, Indexing using a spectral encoding of topological structure, in *CVPR*, 1999.
17. K. Siddiqi and B. B. Kimia, A shock grammar for recognition, in *CVPR*, 507-513, 1996.
18. K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker, The hamilton-jacobi skeleton, in *ICCV*, pp. 828-834, 1999.
19. K. Siddiqi et al., Shock graphs and shape matching, *Int. J. of Comp. Vision*, Vol. 35, pp. 13-32, 1999.
20. K-C Tai, The tree-to-tree correction problem, *J. of the ACM*, Vol. 26, pp. 422-433, 1979.
21. A. Torsello and E. R. Hancock, A skeletal measure of 2D shape similarity, *Int. Workshop on Visual Form*, LNCS 2059, 2001.
22. A. Torsello and E. R. Hancock, Efficiently computing weighted tree edit distance using relaxation labeling, in *EMMCVPR*, LNCS 2134, pp. 438-453, 2001
23. W. H. Tsai and K. S. Fu, Error-correcting isomorphism of attributed relational graphs for pattern analysis, *Sys., Man, and Cyber.*, Vol. 9, pp. 757-768, 1979.
24. J. T. L. Wang, K. Zhang, and G. Chirn, The approximate graph matching problem, in *ICPR*, pp. 284-288, 1994.
25. R. C. Wilson and E. R. Hancock, Structural matching by discrete relaxation, *PAMI*, 1997.
26. K. Zhang, A constrained edit distance between unordered labeled trees, *Algorithmica*, Vol. 15, pp. 205-222, 1996.
27. K. Zhang and D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, *SIAM J. of Comp.*, Vol. 18, pp. 1245-1262, 1989.
28. K. Zhang, R. Statman, and D. Shasha, On the editing distance between unorderes labeled trees, *Inf. Proc. Letters*, Vol. 42, pp. 133-139, 1992.