Shape-Space from Tree-Union

Andrea Torsello and Edwin R Hancock
Dept. of Computer Science, University of York
Heslington, York, YO10 5DD, UK

Abstract

In this paper we investigate how to construct a shape space for sets of shock trees. To do this we construct a super-tree to span the union of the set of shock trees. This super-tree is constructed so that it both minimizes the total tree edit distance and preserves edge consistency constraints. Each node of the super-tree corresponds to a dimension of the pattern space. Individual such trees are mapped to vectors in this pattern space.

1. Introduction

Recently, there has been considerable interest in the structural abstraction of 2D shapes using shock-graphs.The shock-graph is a characterization of the differential structure of the boundaries of 2D shapes. 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. In this paper we address the problem of how to organize shock-graphs into a shape-space in which similar shapes are close to one-another, and such that the space is traversed in a relatively uniform manner as the shapes 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 edit-distance between shock-graphs and to use multidimensional scaling to embed the individual graphs in a low-dimensional space [6]. However, this approach does not necessarily result in a shape-space where the dimensions reflect the modes of structural variation of the shock-graphs. Furthermore, pairwise distance algorithms consistently underestimate the distance between shapes belonging to different clusters. When two shapes are similar, the node-correspondences can be estimated reliably, but as shapes move farther apart in shape space the estimation becomes less reliable. This is due to the fact that correspondences are chosen minimize the distance between trees: as the shock-trees move further apart the advantage the "correct" correspondence has over alternative ones diminishes, until, eventually, a match which yields a lower distance is selected.

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, this approach becomes problematic as soon as graphs of different size are used.

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.

2. Tree Edit-Distance

The idea behind edit distance 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. 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. 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 edit distance between two trees is completely determined by the subset of nodes left after the optimal removal sequence. In this section we show how to find the set correspondences that minimizes the edit distance between two trees. To find he edit distance we make use of two results presented in [8]. We call $\mathcal{C}(t)$ the closure of tree t, $E_v(t)$ the edit operation that removes node v from t and $\mathcal{E}_v(\mathcal{C}(t))$ the equivalent edit operation that removes v from the closure. The first result is that edit and closure operations commute: $\mathcal{E}_v(\mathcal{C}(t)) = \mathcal{C}(E_v(t))$. For the second result we need some more definitions: 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, for s to be obtainable, there cannot be a path in t connecting two nodes that are siblings in s. We can, now, introduce 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 $\mathcal{C}t$.*

By virtue of the theorem above, the node correspondences yielding the minimum edit distance between trees t and t' form an obtainable subtree of both $\mathcal{C}t$ and $\mathcal{C}t'$, hence we reduce the problem to the search for a common substructure: the maximum common obtainable subtree (MCOS).

We commence by transforming the problem from the search of the minimum edit cost linked to the removal of 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 the edit distance becomes:

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

We call he *utility* of the match M. the quantity

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

Clearly the match that maximizes the utility minimizes the edit distance. That is, Let $O \subset \mathcal{P}(\mathcal{N}t \times \mathcal{N}t')$ be the set of matches that satisfy the obtainability constraint, the node correspondence $M^* = (N_t^*, N_{t'}^*)$ is

$$M^* = \underset{M \in O}{\operatorname{argmax}} \mathcal{U}(M),$$

and the closure of the MCOS is the restriction to N_t^* of $\mathcal{C}t$.

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. 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. Given a method to obtain a maximum weight clique, we can use it to obtain the solution to our isomorphism problem. We refer to [8] for heuristics for the weighted clique problem.

3. Edit-intersection and edit-union

The edit distance between two trees is completely determined by the set of nodes that do not get removed by edit operations, that is, in a sense, the *intersection* of the sets of nodes. 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 us 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 tree $t_2 \in T$, then b must be matched to c, i.e. $< a, b > \in M \wedge < a, c > \in M \Rightarrow < b, c > \in M$, were M is the "matches to" relation.

To find the match we 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: assigning to each node a coordinate in a vector space V, we can associate each tree t to a vector $v \in V$ so that $v_i = w_i^t$, where w_i^t is the weight of the node of t associated with node i of the union, $w_i^t = 0$ if no node in t is associated with i.

3.1. 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. The result will be a directed acyclical graph with multiple paths connecting various nodes (see Figure 1). This structure, thus, has more links than necessary and cannot be obtained from the first tree by node removal operations alone. Removing the superfluous edges, we obtain a tree starting from which we can obtain either one of the original trees by node removal operations alone. Furthermore, this reduced structure maintains the same transitive closure. If this structure could always be reduced to a tree, we could use the matching technique already described to compare a tree to a group of trees. Unfortunately, this is not always possible: see, for example, Figure 2. 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 nor one can be a child of the other. 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.

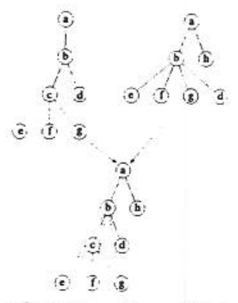
Figure 1. Edit-Union of two trees.

3.2. 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, we cannot directly use our approach to compare a tree to a tree set.

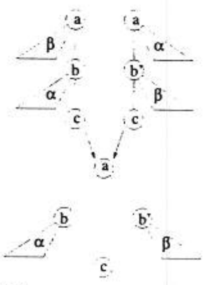
Figure 2. Edit-Union not a tree.

Fortunately, different paths are present in separate trees and so we can consider them mutually exclusive, hence we don't need to perform a generic match between two generic directed acyclic graphs. 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 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.

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 from the two sets, defining 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 at most one union has multiple paths to a node is required to avoid double counting. The solution the modified weighted clique problems yield the correspondences between the nodes of the trees in the two sets.

3.3. Joining multiple trees

In this section want to show how to construct the edit-union structure. Finding the super-structure that minimizes the total distance between trees in the set is computationally infeasible, but we propose a suboptimal method that iteratively extends the union adding a new tree to it.

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 the closest two and replace them with the union and reiterate until we end up with only one structure. Unfortunately, since we have no guarantees that the edit-union is a tree, we might end up trying to merge two graphs with multiple paths to a node. 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 always merging the new nodes to the same structure. This way we are guaranteed to merge at each step at most one multi-path graph.

4. 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 described in [8]. In particular, we compare 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

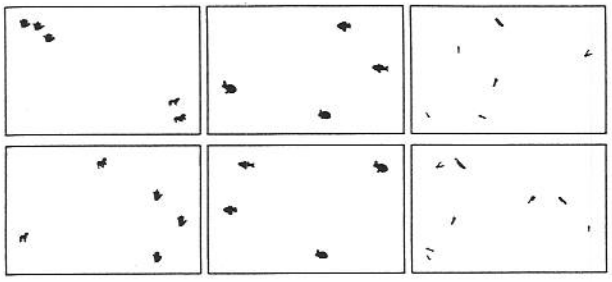
Figure 3. Top: embedding through union. Bottom: 2D MDS of pairwise distance.

in the shapes, yielding embeddings comparable to 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.

Figure 3 shows a comparison between embedding obtained through edit-union of shock trees and through multidimensional scaling of the pairwise distances. The first column shows a clear example where the pairwise edit-distances approach underestimate the distance while edit-union keep the clusters well separated. The second and third column 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 approach benefits from the added geometrical information. In particular, the third column shows a better ordering of shapes, with brushes being so tightly packed that they overlap. It is interesting to note how the union embedding puts the monkey wrench (top-center) somewhere in-between pliers and wrenches: the union is able to consistently match the head to the heads of the wrenches, and the handles to the handles of the pliers.

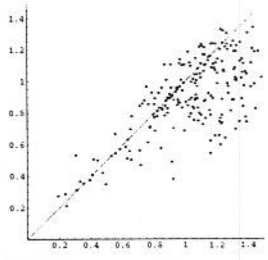
Figure 4. Edit-union vs pairwise distances.

Figure 4 plots the distances obtained through edit union of weighted shock trees (x axis) versus the corresponding pairwise edit distances (y axis). The plot clearly highlights that the pairwise distance approach tends to underestimate the distances between shapes.

4.1. 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. The trees are perturbed by adding nodes using the same approach.

Figure 5. Synthetic clusters.

In our experiments the size of the prototype trees varied from 5 to 20 nodes. As we can see from Figure 5, the algorithm was able to clearly separate the clusters of trees generated by the same prototype. Figure 5 shows three experiments with synthetic data. The first and second images are produced embedding 5 structurally perturbed trees per prototype: trees 1 to 5 are perturbed copies of the first prototype, 6 to 10 of the second. The last 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.

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] H. Bunke and A. Kandel, Mean and maximum common subgraph of two graphs, *Pattern Recognition Letters*, Vol. 21, pp. 163-168, 2000.

[3] T. F. Cootes, C. J. Taylor, and D. H. Cooper, Active shape models - their training and application, *CVIU*, Vol. 61, pp. 38-59, 1995.

[4] T. Heap and D. Hogg, Wormholes in shape space: tracking through discontinuous changes in shape, in *ICCV*, pp. 344-349, 1998.

[5] T. Sebastian, P. Klein, and B. Kimia, Recognition of shapes by editing shock graphs, in *ICCV*, Vol. I, pp. 755-762, 2001.

[6] B. Luo, et al., Clustering shock trees, in *CVPR*, pp. 912-919, 2001.

[7] M. Pelillo, K. Siddiqi, and S. W. Zucker, Matching hierarchical structures using association graphs, *PAMI*, Vol. 21, pp. 1105-1120, 1999.

[8] A. Torsello and E. R. Hancock, Efficiently computing weighted tree edit distance using relaxation labeling, in *EMMCVPR*, LNCS 2134, pp. 438-453, 2001