Indexing using a Spectral Encoding of Topological Structure

Ali Shokoufandeh, Rutgers University
Sven J. Dickinson*, Rutgers University
Kaleem Siddiqi†, McGill University
Steven W. Zucker‡, Yale University

Abstract

In an object recognition system, if the extracted image features are multilevel or multiscale, the indexing structure may take the form of a tree. Such structures are not only common in computer vision, but also appear in linguistics, graphics, computational biology, and a wide range of other domains. In this paper, we develop an indexing mechanism that maps the topological structure of a tree into a low-dimensional vector space. Based on a novel eigenvalue characterization of a tree, this topological signature allows us to efficiently retrieve a small set of candidates from a database of models. To accommodate occlusion and local deformation, local evidence is accumulated in each of the tree's topological subspaces. We demonstrate the approach with a series of indexing experiments in the domain of 2-D object recognition.

1 Introduction

In an object recognition system, *indexing* is the process by which a collection of one or more extracted image features belonging to an object is used to select, from a large database of object models, a small set of candidates likely to contain the object. From this relatively small set of candidates, a verification procedure is applied to select the most promising candidate. If the extracted image features are multilevel or multiscale, the indexing structure may take the form of a tree. Such structures are not only common in computer vision, e.g., [22, 7], but also appear in linguistics (syntax trees), graphics (CSG trees), computational biology (phylogenetic trees), and a wide range of other domains.

If the image (or query) tree has rich structure in terms of depth and/or branching factor, its topology alone may serve as a discriminating index into a database of model trees. Although false positives (e.g., model trees that have the same structure, but whose node labels are different) may arise, they may be few in number and can be pruned during verification. To be an effective index, a topological encoding of a tree's structure should: **1)** map a tree's topology to a point in some low-dimensional space; **2)** capture local topology to support matching/indexing in the presence of occlusion; **3)** be invariant to re-orderings of the tree's branches, i.e., re-orderings which do not affect the parent-child relationships in the tree; **4)** be as unique as possible, i.e., different trees should have different signatures; **5)** be stable, i.e., small perturbations of a tree's topology should result in small perturbations of the index; and **6)** should be efficiently computed.

In this paper, we present a novel encoding of a tree's topology which satisfies the above criteria. Our encoding, or *topological signature*, is derived from an eigenvalue characterization of a tree's $\{0, 1\}$ adjacency matrix. We draw on a number of important theorems in the domain of eigenspaces of graphs to show how our index meets the above criteria. Although applicable to a wide range of tree indexing problems, we demonstrate our approach in the domain of 2-D silhouette recognition, in which image and model silhouettes are represented as trees. In current work, we are exploring the application of our indexing mechanism to problems in computational linguistics, computer graphics, and bioinformatics.

2 Related Work

Eigenspace approaches to shape description and indexing are numerous. Due to space constraints, we cite only a few examples. Turk and Pentland's eigenface approach [21] represented an image as a linear combination of a small number of basis vectors (images) computed from a large database of images. Nayar and Murase extended this work to general 3-D objects where a dense set of views was acquired for each object [9]. Other eigenspace methods have been applied to higher-level features, offering more potential for generic shape description and matching. For example, Sclaroff and Pentland compute the eigenmodes of vibration of a 2-D region [15], while Shapiro and

*Sven Dickinson gratefully acknowledges the support of the National Science Foundation (IRI-9623913 and ISI-9818332).

†Kaleem Siddiqi gratefully aclnowledges the support of the National Sciences and Engineering Research Council of Canada (NSERC).

‡Steven Zucker gratefully aclnowledges the support of the Air Force Office of Scientific Research (AFOSR).

Brady looked at how the modes of vibration of a set of 2-D points could be used to solve the point correspondence problem under translation, rotation, scale, and small skew [17]. In an attempt to index into a database of graphs, Sossa and Horaud use a small subset of the coefficients of the d_2-polynomial corresponding to the Laplacian matrix associated with a graph [20], while a spectral graph decomposition was reported by Sengupta and Boyer for the partitioning of a database of 3-D models, where nodes in a graph represent 3-D surface patches [16]. Sarkar [14] and Shi and Malik [18] have formulated the perceptual grouping and region segmentation problems, respectively, as graph partitioning problems and have used a generalized eigensystem approach to provide an efficient approximation. We note that in contrast to many of the above approaches to indexing, the representation that we present in this paper is independent of the contents of the model database and uses a uniform basis to represent all objects.

3 A Novel Topological Description

3.1 Eigenspaces of Graphs

To describe the topology of a tree, we turn to the domain of eigenspaces of graphs, first noting that any graph can be represented as a symmetric $\{0,1\}$ adjacency matrix, with 1's indicating adjacent nodes in the graph (and 0's on the diagonal). The eigenvalues of a graph's (or tree's) adjacency matrix encode important structural properties of the graph (or tree). Furthermore, the eigenvalues of a symmetric matrix A are invariant to any orthonormal transformation of the form P^tAP. Since a permutation matrix is orthonormal, the eigenvalues of a tree are invariant to any consistent re-ordering of the tree's branches. However, before we can exploit a tree's eigenvalues for indexing purposes, we must establish their stability under minor topological perturbation, due to noise, occlusion, or deformation.

We begin with the case in which the image tree is formed by either adding a new root to the model tree, adding one or more subtrees at leaf nodes of the model tree, or deleting one or more entire model subtrees. In this case, the model tree is a subtree of the query tree, or vice versa. The following theorem relates the eigenvalues of two such trees:

Theorem 1 (see Cvetković et al. [3]) *Let A be a symmetric[1] matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ and let B be one of its principal[2] submatrices. If the eigenvalues of B are $\nu_1 \geq \nu_2 \geq \ldots \geq \nu_m$, then $\lambda_{n-m+i} \leq \nu_i \leq \lambda_i (i = 1, \ldots, m)$.*

This important theorem, called the *Interlacing Theorem*, implies that as A and B become less similar (in the sense that one is a smaller subtree of the other), their eigenvalues become proportionately less similar (in the sense that the intervals that contain them increase in size, allowing corresponding eigenvalues to drift apart).

The other case we need to consider consists of a query tree formed by adding to or removing from the model tree, a small subset of internal (i.e., non-leaf) nodes. The upper bounds on the two largest eigenvalues ($\lambda_1(T)$ and $\lambda_2(T)$) of any tree, T, with n nodes and maximum degree $\Delta(T)$ are $\lambda_1(T) \leq \sqrt{n-1}$ and $\lambda_2(T) \leq \sqrt{(n-3)/2}$, respectively (Neumaier, 1982 [10]). The lower bounds on these two eigenvalues are $\lambda_1(T) \geq \sqrt{\Delta(T)}$ (Nosal, 1970 [11]) and $\lambda_1(T)\lambda_2(T) \geq \frac{2n-2}{n-2}$ (Cvetković, 1971 [2]). Therefore, the addition or removal of a small subset of internal nodes will result in a small change in the upper and lower bounds on these two eigenvalues. As we shall see in the following subsection, our topological description exploits the largest eigenvalues of a tree's adjacency matrix. Since these largest eigenvalues are stable under minor perturbation of the tree's internal node structure, so too is our topological description.

3.2 Formulating an Index

We now seek a compact representation of the tree's topology based on the eigenvalues of its adjacency matrix. We could, for example, define a vector to be the sorted eigenvalues of a tree. The resulting index could be used to retrieve nearest neighbors in a model tree database having similar topology. There are two problems with this approach. First, the eigenvalues don't encode the ordering of nodes in the tree; if the tree was inverted with a leaf becoming the root, the eigenvalues would remain invariant. Second, for large trees, the dimensionality of the index (and model tree database) would be prohibitively large. Our solution to the first problem will be to compute an eigenvalue-based description at each node in terms of the eigenvalues of its subtrees, while to solve our second problem, this description will be based on eigenvalue sums rather than on the eigenvalues themselves.

Specifically, let T be a tree whose maximum branching factor is $\Delta(T)$, and let the subtrees of its root be $T_1, T_2, \ldots, T_S$. For each subtree, T_i, whose root degree is $\delta(T_i)$, compute the eigenvalues of T_i's

[1] The original theorem is stated for Hermitian matrices, of which symmetric matrices are a subclass.

[2] A principal submatrix of a graph's adjacency matrix is formed by selecting the rows and columns that correspond to a subset of the graph's nodes.

submatrix, sort the eigenvalues in decreasing order by absolute value, and let S_i be the sum of the $\delta(T_i) - 1$ largest absolute values. The sorted S_i's become the components of a $\Delta(T)$-dimensional vector assigned to the tree's root. If the number of S_i's is less than $\Delta(T)$, then the vector is padded with zeroes. We can recursively repreat this procedure, assigning a vector to the root of each subtree in the tree for reasons that will become clear in the next section.

Although the eigenvalue sums are invariant to any consistent re-ordering of the tree's branches, we have given up some uniqueness (due to the summing operation) in order to reduce dimensionality. We could have elevated only the largest eigenvalue from each subtree (non-unique but less ambiguous), but this would be less representative of the subtree's structure. We choose the $\delta(T_i) - 1$-largest eigenvalues for two reasons: 1) the largest eigenvalues are more informative of subtree structure, 2) by summing $\delta(T_i)-1$ elements, we effectively normalize the sum according to the local complexity of the subtree root.

To efficiently compute the submatrix eigenvalue sums, we turn to the domain of semidefinite programming. A symmetric $n \times n$ matrix A with real entries is said to be positive semidefinite, denoted as $A \succeq 0$, if for all vectors $x \in R^n$, $x^t A x \geq 0$, or equivalently, all its eigenvalues are non-negative. We say that $U \succeq V$ if the matrix $U - V$ is positive semidefinite. For any two matrices U and V having the same dimensions, we define $U \bullet V$ as their inner product, i.e., $U \bullet V = \sum_i \sum_j U_{i,j} V_{i,j}$. For any square matrix U, we define $\text{trace}(U) = \sum_i U_{i,i}$. Let I denote the identity matrix having suitable dimensions. The following result, due to Overton and Womersley [12], characterizes the sum of the first k largest eigenvalues of a symmetric matrix in the form of a semidefinite convex programming problem:

Theorem 2 (Overton and Womersley [12]) *For the sum of the first k eigenvalues of a symmetric matrix A, the following semidefinite programming characterization holds:*

$$\begin{aligned} \lambda_1(A) + \ldots + \lambda_k(A) = \quad & \max \quad A \bullet U \\ & \text{s.t.} \quad \text{trace}(U) = k \\ & \qquad 0 \preceq U \preceq I, \end{aligned}$$

The elegance of Theorem (2) lies in the fact that the equivalent semidefinite programming problem can be solved, for any desired accuracy ϵ, in time polynomial in $O(n\sqrt{n}L)$ and $\log \frac{1}{\epsilon}$, where L is an upper bound on the size of the optimal solution, using a variant of the Interior Point method proposed by Alizadeh [1]. In effect, the complexity of directly computing the eigenvalue sums is a significant improvement over the $O(n^3)$ time required to compute the individual eigenvalues, sort them, and sum them.

3.3 Properties of the Index

Our topological index satisfies the six criteria outlined in Section 1. The eigendecomposition yields a low-dimensional (criterion 1) vector assigned to each node in the tree, which captures the local topology of the subtree rooted at that node (criterion 2). Furthermore, a node's vector is invariant to any consistent re-ordering of the node's subtrees (criterion 3). The components of a node's vector are based on summing the largest eigenvalues of its subtree's adjacency submatrix. Although our dimensionality-reducing summing operation has cost us some uniqueness, our partial sums still have very low ambiguity (criterion 4). From Theorem 1, along with our analysis of eigenvalue bounds, we have shown our index to be stable to minor perturbations of the tree's topology (criterion 5). As shown in Theorem 2, these sums can be computed even more efficiently (criterion 6) than the eigenvalues themselves. The vector labeling of all rooted trees isomorphic to T not only has the same vector labeling but spans the same subspace in $R^{\Delta(T)-1}$. Moreover, this extends to any rooted tree which has a subtree isomorphic to a subtree of T.

4 Candidate Selection

Given a query tree corresponding to an image, our task is to search the model tree database for one or more model trees which are similar to the image tree. If the number of model trees is large, a linear search of the database is intractable. Therefore, the goal of our indexing mechanism is to quickly select a small number of model candidates for verification. Those candidates will share coarse topological structure with the image tree (or one of its subtrees, if it is occluded or poorly segmented). Hence, we begin by mapping the topology of the image tree to a set of indices that capture its structure, discounting any information associated with its nodes. We then describe the structure of our model database, along with our mechanism for indexing into it to yield a small set of model candidates. Finally, we present a local evidence accumulation procedure that will allow us to index in the presence of occlusion.

4.1 A Database for Model Trees

Our eigenvalue characterization of a tree's topology suggests that a model tree's topological structure can be represented as a vector in δ-dimensional space, where δ is an upper bound on the degree of any vertex of any image or model tree. If we could assume that

an image tree represents a properly segmented, unoccluded object, then the vector of eigenvalue sums, call it the *topological signature vector (or TSV)*, computed at the image tree's root could be compared with those topological signature vectors representing the roots of the model trees. The vector distance between the image tree's root TSV and a model tree's root TSV would be inversely proportional to the topological similarity of their respective trees: recall from Section 3 that finding two subtrees with "close" eigenvalue sums represents an approximation to finding the largest isomorphic subtree.

Unfortunately, this simple framework cannot support either cluttered scenes or segmentation errors, both of which result in the addition or removal of tree structure. In either case, altering the structure of the tree will affect the TSV's computed at its nodes. The signatures corresponding to those subtrees that survive the occlusion will not change. However, the signature of a node having one or more subtrees which have undergone any perturbation will change which, in turn, will affect the signatures of any of its ancestor nodes, including the root. We therefore cannot rely on indexing solely with the root's signature. Instead, we will take advantage of the local subtrees that survive the occlusion.

We can accommodate such perturbations through a local indexing framework analogous to that used in a number of geometric hashing methods, e.g., [6, 4]. Rather than storing a model tree's root signature, we will store the signatures of *each* node in the model tree, along with a pointer to the object model containing that node as well as a pointer to the corresponding node in the model tree (allowing access to node label information). Since a given model subtree can be shared by other model trees, a given signature (or location in δ-dimensional space) will point to a list of (model object, model node) ordered pairs. At runtime, the signature at each node in the image tree becomes a separate index, with each nearby candidate in the database "voting" for one or more (model object, model node) pairs. To quickly retrieve these nearby candidates, we will precompute the sorted pairwise distances between every signature in the database and every other signature in the database.

4.2 An Efficient Indexing Mechanism

We achieve efficient indexing through a δ-dimensional Voronoi decomposition $P(B)$ of the model space $V(B)$. For a given image TSV, the Voronoi decomposition will allow us to find the nearest model TSV in expected $O(\log^{\delta}(kn))$ time for fixed δ [13]. From the ranked list of neighbors L computed for each model TSV, either the ℓ nearest neighbors or all neighbors within a radius r can be computed in constant time (assuming fixed ℓ or r). To construct the Voronoi database, we partition R^{δ} into regions, so that for each vector $v \in V(B)$, the region $P(v)$ will denote the set of points in R^{δ} which are closer to v than any other vector in $V(B)$ with respect to a metric norm, such as $d(.,.)$. Such a partition is well-defined. The complexity of the Voronoi decomposition is $O((kn)^{\lfloor(\delta+1)/2\rfloor+1}) + O((kn)^{\lfloor(\delta+1)/2\rfloor}\log(kn))$ ([13]), although this is a cost incurred at preprocessing time.

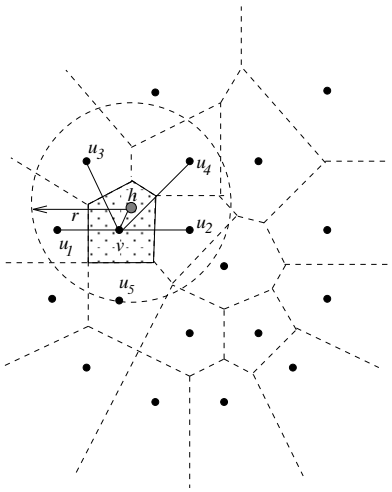

Figure 1: Selecting the Candidate Model Objects

The process of selecting candidates at runtime is shown in Figure 1. Let H be an image tree, and let F_H and $V(F_H)$ be defined as before, and let $d(u,v) = ||v-u||_2$. For each TSV $h \in V(F_H)$, we will find the region $P(v)$ (and corresponding vector v) in $P(B)$, in which h resides. Using the list $L(v)$, we will find the set of model TSV's $\{u_1,..,u_\ell\}$ such that $d(h,v) + d(v,u_i) \leq r$. Clearly, since the metric norm $d(.,.)$ satisfies the triangle inequality, the set $\{v\} \cup \{u_1,..,u_\ell\}$ is a subset of the TSV's whose distance from h is less than r. Each node in the image tree therefore leads to a number of (model object, model node) candidate votes. In the next section, we discuss the weighting of these votes, along with the combination of the evidence over all nodes in the image tree.

4.3 Accumulating Local Evidence

Each node in the image tree will generate a set of (model object, model node) votes. To collect these votes, we set up an accumulator with one bin per model object. Furthermore, we can weight the votes that we add to the accumulator. For example, if the

label of the model node is not compatible with the label of its corresponding image node, then the vote is discarded, i.e., it receives a zero weight. If the nodes are label-compatible, then we can weight the vote according to the distance between their respective TSV's – the closer the signatures, the more weight the vote gets.

We can also weight the vote according to the complexity of its corresponding subtree, allowing larger and more complex subtrees (or "parts") to have higher weight. This can be easily accommodated within our eigenvalue framework, for the richer the structure, the larger its maximum eigenvalue:

Theorem 3 (Lovász and Pelikán [8]) *Among the trees with n vertices, the star graph ($K_{1,n-1}$), has the largest eigenvalue ($\sqrt{n-1}$), while the path on n nodes (P_n) has the smallest eigenvalue ($2\cos\pi/(n+1)$).*

Since the size of the eigenvalues, and hence their sum, is proportional to both the branching factor as well as the number of nodes, the magnitude of the signature is also used to weight the vote. If we let u be the TSV of an image tree node and v the TSV of a model tree node that is sufficiently close, the weight of the resulting vote, i.e., the local evidence for the model, is computed as (we use $p = 2$):

$$W = \frac{||u||_p}{1 + ||v - u||_p} \tag{1}$$

Once the evidence accumulation is complete, those models whose support is sufficiently high are selected as candidates for verification. The bins can, in effect, be organized in a heap, requiring a maximum of $O(\log k)$ operations to maintain the heap when evidence is added, where k is the number of non-zero object accumulators. Once the top-scoring models have been selected, they must be individually verified according to some matching algorithm.

5 Experiments

To demonstrate our approach to indexing, we turn to the domain of 2-D object recognition. Although we could choose any tree-based image description, e.g., [22, 7], our representation for 2-D shape is based on a coloring of the shocks (singularities) of a curve evolution process acting on simple closed curves in the plane [5]. Intuitively, the taxonomy of shocks consists of four distinct types: the radius function along the medial axis varies monotonically at a 1, achieves a strict local minimum at a 2, is constant at a 3 and achieves a strict local maximum at a 4. We have recently abstracted this system of shocks into a *shock graph* where vertices are labelled by their shock types, and the shock formation times direct the edges (see Figure 2). The space of such shock graphs is completely characterized by a small number of rules, which in turn permits the reduction of each graph to a *unique rooted tree*. In recent work, we developed an algorithm for matching two shock trees based on both topological structure and geometric structure [19]. Still, the problem of indexing was not addressed. How, from a shock-based encoding of an occluded scene, can we select from a large database of 2-D objects, a small set of candidates to which we can apply our matching algorithm in order to recognize the embedded object(s)?

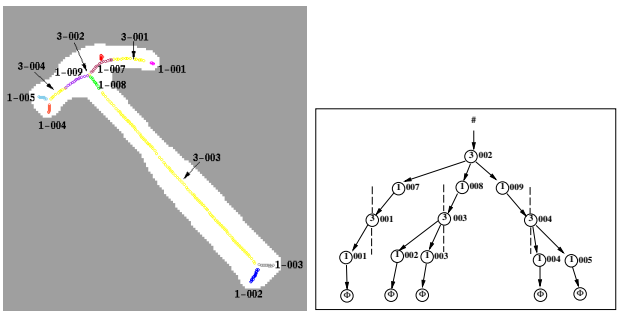

Figure 2: An illustrative example taken from [19]. The labels on the shocks of the hammer (left) correspond to vertices in the derived shock graph (right).

We test our indexing algorithm on a database of 60 object silhouettes, some representative examples of which are shown in Figure 3. In the first experiment, we select 20 shapes from the database, compute their shock trees, compute the topological signature vectors for each of their nodes, and populate the resulting vectors in a model database. Each element, in turn, will be removed from the database and used as a query tree for the remaining database of 19 model trees. For each of the 20 trials, the 19 object candidates will be ranked in decreasing order of accumulator contents. To evaluate the quality of the indexing results, we will compute the distance between the query tree and each of the candidates, using the matcher developed in [19], and note which model tree is the closest match. If indexing is to be effective, the closest matching model should be among the best (highest-weight) candidates returned by the indexing strategy.

In the second and third experiments, we apply the same procedure to databases of size 40 and 60 model trees, respectively, in order to evaluate the scaling properties of our indexing algorithm. Thus, in the second experiment, we have 40 indexing trials, while

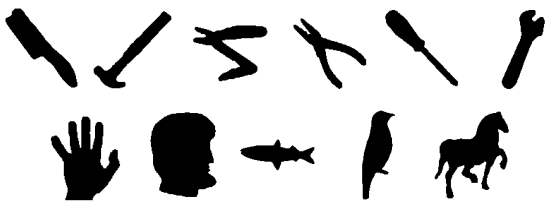

Figure 3: Samples from a Database of 60 Object Silhouettes

in the third experiment, we have 60 indexing trials. Finding the position of the closest model shape among the sorted candidates for any given query shape requires that we first compute the 60×60 distance matrix.

The results of the first experiment are shown in Figure 4(a), where the horizontal axis indicates the rank of the target object (or closest matching object) in the sorted candidates, and the vertical axis represents the number of times that rank is achieved. For this experiment, the average rank is 1.6, which implies that on average, 8.4% of the sorted candidates need to be verified before the closest matching model is found. The results of the second and third experiments are shown in Figures 4(b) and (c), respectively. The results are very encouraging and show that as database size increases, the indexing algorithm continues to prune over 90% of the database (avg. rank of 7.9% in expt. 2, 8.8% in expt. 3).

In a final experiment, we generate some occluded scenes from shapes in our database. In Table 1, we show three examples of occluded query images (left column) and the top ten (sorted left to right) model candidates from a database of 40 model shapes. Twice, the rank of the target is 4th, while once it is 3rd, indicating that for these three examples, at most 10% of the model indexing candidates need to be verified. We are currently conducting a more comprehensive set of occlusion experiments.

It should be noted that the indexing mechanism reflects primarily the topological structure of the query. Thus, in row 1 of Table 1, for example, the topological structure of the query (brush occluding the hammer) is more similar to the pliers-like objects (two of the top three candidates) than to the hammer itself. Topological similarity of shock trees is a necessary but not sufficient condition for shape similarity, as it ignores the geometries of the object's parts (nodes in its shock tree). Therefore, the fact that objects with different shape can rank high in the candidate list is not surprising.

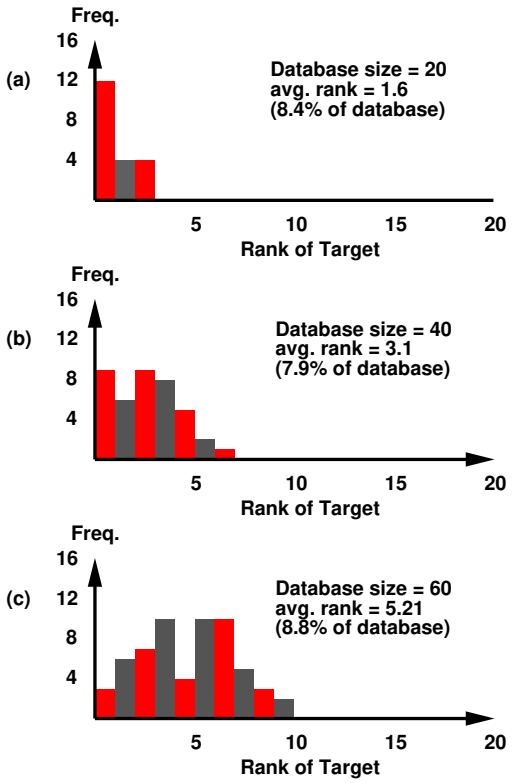

Figure 4: Indexing Results for Three Databases of Increasing Size. In each case, the horizontal axis indicates the rank of the target object (or closest matching object) in the sorted candidates, and the vertical axis represents the number of times that rank is achieved. (See text for discussion.)

6 Conclusions

We have presented a computationally efficient eigendecomposition of a tree that captures the coarse topological structure of the tree. Furthemore, this eigendecomposition is stable under minor perturbation of tree structure. Each of the tree's subtrees yields an independent topological index which can be used to efficiently accumulate evidence for a small set of models sharing the query's topological structure, even in the presence of noise and occlusion. In a series of experiments in the domain of 2-D object recognition (shock trees), we demonstrate that our topological index is very effective in selecting a small number of model candidates likely to contain the target object. Furthermore, our experiments show that its performance scales well with increasing database size. Our indexing framework is general and can be applied to any indexing domain, including, for example, hierarchical or multiscale structures in computer

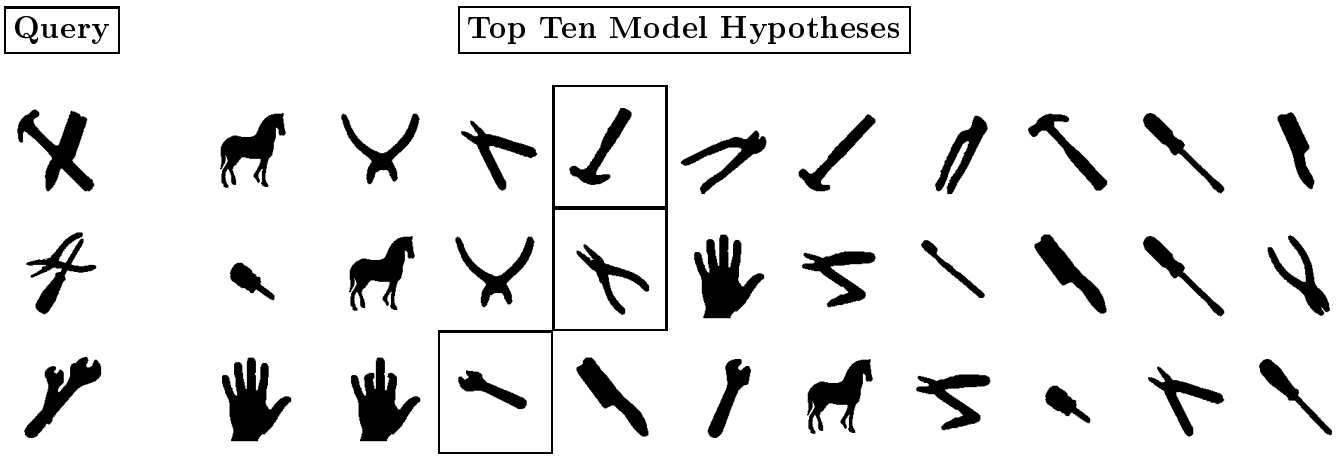

Table 1: Indexing using an occluded query shape. For each row, the query is shown to the left, while the top ten candidate models (from a database of 40 models) are shown on the right, in decreasing order of weight. The database shape whose distance to the query is smallest, as computed by the matching algorithm described in [19], is enclosed in a box. The likelihood that the closest shape is among the top-ranked candidates is high (see text for discussion).

vision, part-whole hierarchies in knowledge representation, parse trees in computational linguistics, CSG trees in computer graphics, and phylogenetic trees in computational biology.

References

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.

[2] D. Cvetković. *Graphs and their spectra*. PhD thesis, University of Beograd, 1971.

[3] D. Cvetković, P. Rowlinson, and S. Simić. *Eigenspaces of Graphs*. Cambridge University Press, Cambridge, United Kingdom, 1997.

[4] P. Flynn and A. Jain. 3D object recognition using invariant feature indexing of interpretation tables. *CVGIP:Image Understanding*, 55(2):119–129, March 1992.

[5] B. B. Kimia, A. Tannenbaum, and S. W. Zucker. Shape, shocks, and deformations I: The components of two-dimensional shape and the reaction-diffusion space. *International Journal of Computer Vision*, 15:189–224, 1995.

[6] Y. Lamdan, J. Schwartz, and H. Wolfson. Affine invariant model-based object recognition. *IEEE Transactions on Robotics and Automation*, 6(5):578–589, October 1990.

[7] T. Lindeberg. Detecting Salient Blob–Like Image Structures and Their Scales With a Scale–Space Primal Sketch—A Method for Focus–of–Attention. *IJCV*, 11(3):283–318, December 1993.

[8] L. Lovász and J. Pelicán. On the eigenvalues of a tree. *Periodica Math. Hung.*, 3:1082–1096, 1970.

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

[10] A. Neumaier. Second largest eigenvalue of a tree. *Linear Algebra and its Applications*, 46:9–25, 1982.

[11] E. Nosal. Eigenvalues of graphs. Master's thesis, University of Calgary, 1970.

[12] M. L. Overton and R. S. Womersley. Optimality conditions and duality theory for minimizing sums of the largest eigenvalues of symmetric matrices. *Math. Programming*, 62(2):321–357, 1993.

[13] F. Preparata and M. Shamos. *Computational Geometry*. Springer-Verlag, New York, NY, 1985.

[14] S. Sarkar. Learning to form large groups of salient image features. In *IEEE CVPR*, Santa Barbara, CA, June 1998.

[15] S. Sclaroff and A. Pentland. Modal matching for correspondence and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):545–561, June 1995.

[16] K. Sengupta and K. Boyer. Using spectral features for modelbase partitioning. In *Proceedings, International Conference on Pattern Recognition*, Vienna, Austria, August 1996.

[17] L. Shapiro and M. Brady. Feature-based correspondence: an eigenvector approach. *Image and Vision Computing*, 10(5):283–288, June 1992.

[18] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, June 1997.

[19] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, to appear.

[20] H. Sossa and R. Horaud. Model indexing: The graph-hashing approach. In *Proceedings, IEEE CVPR*, pages 811–814, 1992.

[21] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[22] A. Witkin. Scale space filtering. In A. Pentland, editor, *From Pixels to Predicates*. Ablex, Norwood, NJ, 1986.