Indexing Hierarchical Structures Using Graph Spectra

Ali Shokoufandeh, *Member*, *IEEE*, Diego Macrini, *Student Member*, *IEEE Computer Society*, Sven Dickinson, *Member*, *IEEE*, Kaleem Siddiqi, and Steven W. Zucker, *Fellow*, *IEEE*

Abstract—Hierarchical image structures are abundant in computer vision and have been used to encode part structure, scale spaces, and a variety of multiresolution features. In this paper, we describe a framework for indexing such representations that embeds the topological structure of a directed acyclic graph (DAG) into a low-dimensional vector space. Based on a novel spectral characterization of a DAG, this topological signature allows us to efficiently retrieve a promising set of candidates from a database of models using a simple nearest-neighbor search. We establish the insensitivity of the signature to minor perturbation of graph structure due to noise, occlusion, or node split/merge. To accommodate large-scale occlusion, the DAG rooted at each nonleaf node of the query "votes" for model objects that share that "part," effectively accumulating local evidence in a model DAG's topological subspaces. We demonstrate the approach with a series of indexing experiments in the domain of view-based 3D object recognition using shock graphs.

Index Terms—Structural indexing, graph spectra, object recognition, shock graphs.

1 Introduction

The recognition of hierarchical (e.g., multiscale or multilevel) image features is a common problem in object recognition.[1] Such structures are often represented as rooted trees or directed acyclic graphs (DAGs), where nodes represent image feature abstractions and arcs represent spatial relations, mappings across resolution levels, parts, etc. [1], [2]. The requirements of matching include computing a correspondence between nodes in an image structure and nodes in a model structure, as well as computing an overall measure of distance (or, alternatively, similarity) between the two structures. Such matching problems can be formulated as *largest isomorphic subgraph (or subtree)* problems, for which a wealth of literature exists in both the graph algorithms and computer vision communities. However, the matching procedure is expensive and must be used sparingly. For large databases of object models, it is simply infeasible to perform an exhaustive search of the database.

1. Such structures are not only common in computer vision, but also appear in linguistics (syntax trees), graphics (CSG trees), computational biology (phylogenetic trees), and a wide range of other domains.

- *A. Shokoufandeh is with the Department of Computer Science, College of Engineering, 3141 Chestnut St., Philadelphia, PA 19104. E-mail: ashokouf@cs.drexel.edu.*
- *D. Macrini and S. Dickinson are with the Department of Computer Science, University of Toronto, 6 King's College Rd., Toronto, Ontario, Canada M5S 3G4. E-mail: {dmac, sven}@cs.toronto.edu.*
- *K. Siddiqi is with the School of Computer Science and the Centre for Intelligent Machines, McGill University, 3480 University St., Montreal, PQ, Canada H3A 2A7. E-mail: siddiqi@cim.mcgill.ca*
- *S.W. Zucker is with the Department of Computer Science, Yale University, 51 Prospect St., PO Box 208285, New Haven, CT 06520-8285. E-mail: steven.zucker@yale.edu.*

Manuscript received 5 Jan. 2004; revised 13 Sept. 2004; accepted 14 Dec. 2004; published online 12 May 2005.
Recommended for acceptance by M. Basu.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMISI-0010-0104.

An indexing mechanism is essential for selecting a small set of candidate models to which the matching procedure is applied. When working with hierarchical image structures, in the form of directed acyclic graphs, indexing is a challenging task and can be formulated as the fast selection of a small set of candidate model graphs that share a subgraph with the query. But, how do we test a candidate without resorting to subgraph isomorphism? If there were a small number of subgraphs shared among many models, representing a vocabulary of *object parts*, one could conceive of a two-stage indexing process, in which image structures were matched to the part vocabulary, with parts "voting" for candidate models [3], [4]. However, we are still faced with the complexity of subgraph isomorphism, albeit for a smaller database (vocabulary of parts). The problem is further compounded by the fact that, due to occlusion and noise, no significant isomorphisms may exist between the query and the model. Yet, at some level of abstraction, the two structures (or their substructures) may be similar. Thus, our indexing problem can be reformulated as finding model (sub)graphs whose structure is *similar* to the query (sub)graph.

In this paper, we begin by outlining the requirements of an effective structural indexing mechanism. Next, we introduce a novel encoding of a directed acyclic graph's topology which satisfies these requirements. Our encoding, or *topological signature*, is derived from an eigenvalue characterization of a DAG's $\{-1, 0, 1\}$ adjacency matrix, and we draw on a number of important results in spectral graph theory to establish its stability under minor perturbation due to noise, occlusion, and node split/merge. Our low-dimensional vector encoding of a graph's structure allows us to retrieve similar structures from a database using a nearest-neighbor search. To deal with large-scale occlusion, we introduce an evidence accumulation framework which accumulates local evidence in a model's topological subspaces that correspond to the model's parts.

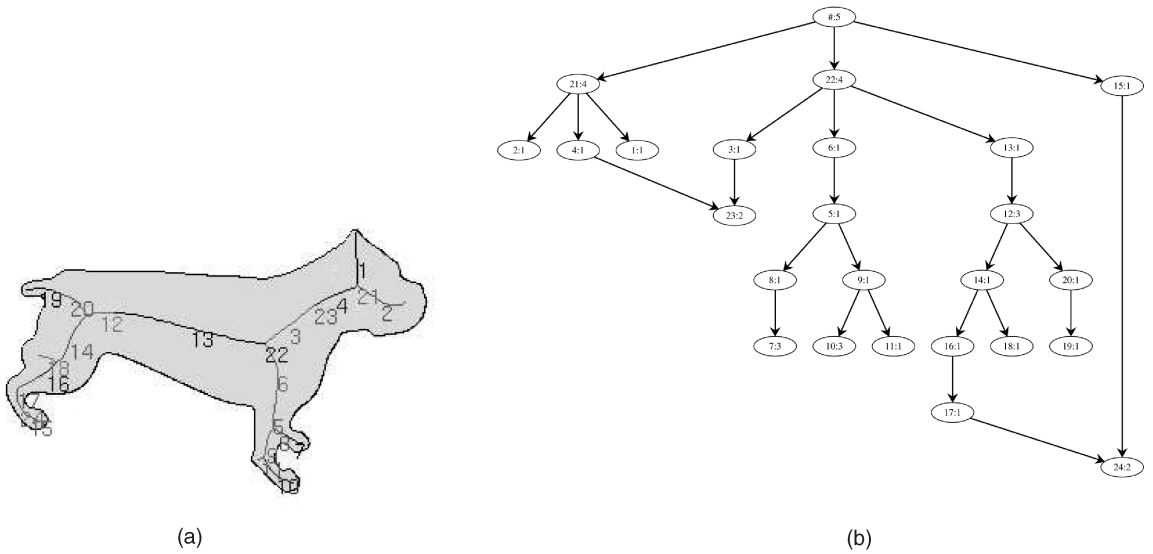


Fig. 1. (a) An illustrative example of a silhouette and (b) its shock graph [6], computed using the Hamilton-Jacobi skeletonization algorithm [7], [8] and the shock graph construction algorithm [9].

Although applicable to any domain in which queries and models can be represented as directed acyclic graphs, such as, e.g., [1], [2], we demonstrate our approach on the domain of view-based 2D silhouette recognition, in which image and model silhouettes are represented as directed acyclic *shock graphs*. 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 (see Kimia et al. [5]). This system of shocks can be abstracted into a *shock graph* where vertices are labeled by their shock types and the shock formation times direct the edges [6] (see Fig. 1).

The many other techniques that are being considered for dimensionality reduction of graph (and tree) structure can be distinguished by the manner in which they represent structure in various nonlinearities. Our approach explicitly attempts to capture summaries of "higher-order" structure at each level, as is explained below.

2 Related Work

Spectral approaches to shape description and indexing are numerous. Turk and Pentland's eigenface approach [10] represented an image as a linear combination of a small set of basis vectors (images) computed from a large database of images. Nayar and Murase extended this work to 3D objects where a dense set of views was acquired for each object [11]. 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 2D region [12] and use the low-order modal coefficients to search a database of 2D shape models. Since the spectral characterizations are global, these methods are not invariant to occlusion. Moreover, such characterizations depend on database contents. In contrast to these approaches, our representation is independent of the contents of the model database by using a uniform basis to represent all objects. Recently, Luo et al. [13], [14] investigated the use of leading eigenvectors of a graph's adjacency matrix to characterize its eigenmodes. They also studied the use of two embedding techniques, component analysis (principle and independent) and multidimensional scaling, and their use for clustering graph data extracted from 2D views.

In recent years, nearest-neighbor (NN) search techniques have become very popular for feature-based retrievals. For instance, Beis and Lowe [15] use a database to store classes of feature vectors that characterize arrangements of lines. A feature vector is formed, for example, by the angles between a set of three coterminating lines. Other perceptual grouping methods, as well as the number of lines considered, define different feature classes. An NN search on the corresponding database is performed for every image feature, allowing each match to vote for a model according to the saliency of the feature. For NN search, Beis and Lowe developed an approximation to the k-d tree search algorithm, called *best bin first*, or *BBF*. In more recent work, Lowe [16] computes a high-dimensional (160 components) feature vector based on the appearance of an interest point and uses BBF search to retrieve objects containing those points.

Indexing into large databases using structural features was popularized by the geometric hashing community. Lamdan et al. [17] used geometric hashing to map affine invariant interest point coordinates, expressed relative to a basis defined by three noncollinear interest points, to models sharing those component interest points. Each interest point voted for one or more models, with the correct model typically emerging with maximal support. In a related viewpoint-invariant indexing scheme, Forsyth et al. [18] used projective invariants, computed over conics and lines, to index into a large database of planar shapes. Clemens and Jacobs [19] presented bounds on the space and speedup that is achievable through any indexing mechanism when the underlying data is based on 2D point features. They proved that there is no quantitative feature that is invariant under all projections of a model into an image.

They also observed that indexing by itself will not produce significant speedup unless it is combined with feature grouping operations.

In a more traditional hashing framework, Flynn and Jain [20] mapped viewpoint-invariant 3D features, computed over 3D surfaces, to locations in a set of interpretation tables whose entries, in turn, pointed to models sharing those features. The major challenge to hash table and voting mechanisms is choosing the bin size or hash function "spread." Fixed-sized bins do not effectively capture the nonuniform distribution of the data. This lack of adaptation has given rise to other data structures designed specifically for similarity searching and for high-dimensional search (see Section 6.2).

The spectral indexing and structural hashing approaches described above attempt to compute features of the query that will prune the database down to a few candidates. Without such a pruning mechanism, the query would have to be exhaustively compared to each model. However, if the models in the database are organized judiciously, an exhaustive search can be avoided. A decision tree [21] is a mechanism for hierarchically partitioning a database. A query shape is matched to the root and, depending on the results of the match, the process is applied recursively to one of its children. At each step, the space of possible models is reduced. Within this framework, a spectral graph decomposition was reported by Sengupta and Boyer for the partitioning of a database of 3D models, where nodes in a graph represent 3D patches [22].

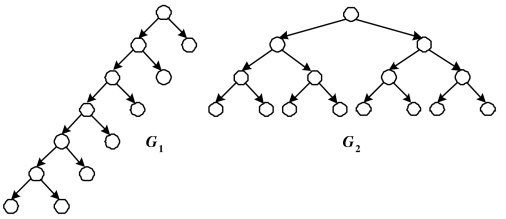
A closely related approach to the partition (decision tree) scheme described above is to organize the database into a set of prototypes (clustering). In this case, the database is organized by grouping similar objects and choosing a representative (prototype) for each group. This idea can be recursively applied, forming a hierarchical representation of the database. Shapiro and Haralick [23] used a simple relational distance metric, followed by either clustering by similar values of the metric or by constructing a binary decision tree, to organize a large database of relational models. Sengupta and Boyer [24] presented one of the earliest frameworks for object recognition through a hierarchically structured database of parametric structural description graphs. The hierarchical structure was constructed through clustering and computing representative members of each cluster. Recently, Sebastian et al. [25] used a similar approach to address the problem of indexing into a database of shock graphs representing the 2D shape silhouettes sampled from a 3D object's viewing sphere and proposed a hierarchical partitioning of the database in which shapes are grouped into categories. A small number of exemplars from each category are chosen to represent the groups, thus forming a database of prototypes used to index into the larger model database. A problem with this approach is that it is not always possible to find shape clusters large enough to significantly reduce the size of the database. Moreover, each match at the prototype level implies an exhaustive search among the shapes belonging to that category, which can be costly if good partitions do not exist.

Recently, Irniger and Bunke [26] presented an approach for filtering graph databases based on a feature vector characterization of graphs in a decision tree model. Their vector characterization encodes such structural information as cardinality of the vertex set, frequency and degree properties of nodes with given labels, as well as frequency of nodes of a particular degree, etc. They showed how a modified decision tree model can be utilized to tackle both graph and subgraph isomorphism problems and also studied the utility of the approach and its performance in filtering random and regular graphs.

The topic of graph matching in computer vision has been studied extensively, with both exact and inexact graph matching algorithms applied to object recognition, including Sanfeliu and Fu [27], Shapiro and Haralick [28], [29], Wong et al. [30], [31], Boyer and Kak [32] (for stereo matching), Kim and Kak [33], Messmer and Bunke [34], Christmas et al. [35], Eshera and Fu [36], Pellilo et al. [37], Gold and Rangarajan [38], Zhu and Yuille [39], Cross and Hancock [40], Huet and Hancock [41], and Siddiqi et al. [6], to name just a few. Although, in some cases, e.g., [6], [41], graph abstractions are sought to improve matching robustness, the above methods still focus on the problem of comparing two graphs and not on the problem of graph indexing (prior to matching). On the topic of graph indexing in computer vision, there has been much less activity. Dickinson et al. [4] processed a graph representing the part structure of an object, yielding a set of connected subgraphs of bounded size. A hash function then mapped the node labels in a subgraph to a hash table location which, in turn, pointed to a second hash table. A second hash function mapped the edge labels in the subgraph to a location in this second hash table containing those models which contain the subgraph as a part. Costa and Shapiro [42] also report a scheme in which small relational subgraphs are used to retrieve model graphs from a large database. Perhaps the closest work to the work reported in this paper is that of Sossa and Horaud, who use a small subset of the coefficients of the d_2-polynomial corresponding to the Laplacian matrix associated with a graph [43]. Although they attempt to encode a graph's structural properties with a low-dimensional descriptor, their method cannot accommodate either noise or occlusion, requiring an exact isomorphism between query and model.

A closely related problem to indexing in the information management community is that of query processing over data that conforms to labeled-tree or labeled-graph data models. A schema for such data, if present, may only partially constrain the data. Such a data model is often referred to as semistructured data model [44], [45]. Most of the work in this community has focused on extracting summaries that are information preserving but not structure preserving [46], [47], [45]. Specifically, the notion of information preserving summaries refers to signatures that are invariant to the node-labels and data types associated with vertices of such abstract representations [44]. Contrary to vision applications in which relational structure among image features is critical and can, in fact, carry more information than the features themselves, preserving such structure in data models and corresponding schemas plays a secondary role in indexing [44], [48].

Finally, there has been growing interest in creating compressed approximate decompositions of data sets modeled as large matrices [49], [50]. Such structures are commonly used in information retrieval and data mining [51], [52]. Specifically, a large collection of n objects, e.g.,



	G_1	G_2
Minimum degree	1	1
Maximum Degree	3	3
Average Degree	1.87	1.87
Average internal degree	2.86	2.86
Degree variance	0.92	0.92

Fig. 2. Naive characterizations of hierarchical structure are compact yet ambiguous. In this case, two trees with the same sorted degree sequence will yield the same simple characterizations.

documents, genomes, or Web pages, is implicitly presented as a set of points in an m-dimensional Euclidean space, where m is the number of features that describes the object. Thus, this collection may be represented by an $m \times n$ matrix A, the columns of which are the object vectors and the rows of which are the feature vectors. The indexing problem can therefore be formulated as the retrieval of matrices with similar compressed decompositions from a large library of such matrices.

3 Criteria for an Effective Index

If the image (or query) DAG 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 structures. Although false positives (e.g., model DAGs 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. As stated in Section 1, we seek a reduced representation for a DAG that will support efficient indexing and matching. An effective topological encoding of a DAG's structure should:

1. map a DAG's topology to a point in some low-dimensional space,
2. capture local topology to support indexing in the presence of occlusion,
3. be invariant to reorderings of the DAG's branches,[2] and
4. be efficiently computed.

Consider the naive, low-dimensional characterizations of hierarchical structure shown in Fig. 2. Although characterization measures such as maximum degree, minimum degree, average degree, degree variance, etc. (whether applied to all nodes or just internal nodes) provide a compact structural encoding and satisfy the aforementioned four criteria, they are obviously too weak. The structure of the two trees is clearly different. However, since they have identical, sorted degree sequences, these particular measures yield identical encodings for the two trees. As a result, any structural characterization should also:

1. be as unique as possible, i.e., different DAGs should have different encodings, and
2. be stable, i.e., small perturbations of a DAG's topology should result in small perturbations of the index.

2. Such reorderings might occur during a nondeterministic construction of a DAG. Since we are not enforcing left-to-right sibling ordering, the same object might be represented by two isomorphic DAGs whose only difference is their branch ordering.

Clearly, we need an intermediate representation that is rich and discriminative, on one hand, yet compact and efficient, on the other.

4 Encoding Graph Structure

To describe the topology of a DAG, we turn to the domain of spectral graph theory, first noting that any directed graph can be represented as an antisymmetric $\{0, 1, -1\}$ adjacency matrix, with 1s (-1s) indicating a forward (backward) edge between adjacent nodes in the graph (and 0s on the diagonal). For a graph G with adjacency matrix A_G, we define the spectrum $\Gamma(A_G)$ as the set of magnitudes of its n eigenvalues, since the eigenvalues of an antisymmetric matrix are complex. In subsequent discussion, for brevity, the term "eigenvalues" will be used in place of "magnitudes of eigenvalues." The spectrum of a graph's adjacency matrix encodes important structural properties of the graph, including the size of the graph and the degree distribution of its nodes. Furthermore, the magnitudes of the eigenvalues of an antisymmetric (or Hermitian) matrix A are invariant to any orthonormal transformation of the form P^tAP. Since a permutation matrix is orthonormal, the magnitudes of eigenvalues of a graph are therefore invariant to any consistent reordering of the graph's branches. However, before we can exploit a graph's spectrum for indexing purposes, we must establish their stability under minor topological perturbation, due to noise, occlusion, node split/merge, or deformation.

We will begin by showing that any structural change to a DAG can be modeled as a two-step transformation of its original adjacency matrix. The first step transforms the DAG's original adjacency matrix to a new matrix having the same spectral properties as the original matrix. The second step adds a noise matrix to this new matrix, representing the structural changes due to noise and/or occlusion. These changes take the form of addition/deletion of nodes/arcs to/from the original DAG. We will then draw on an important result that relates the distortion of the elements of the spectrum of the matrix resulting from the first step to the magnitude of the noise added in the second step. Since the spectrum of the original matrix is the same as that of the transformed matrix, the noise-dependent bounds therefore apply to the original matrix. The result will establish the stability of a DAG's spectrum to minor topological changes.

We begin with some definitions. Let $A_G \in 0, 1, -1^{m \times m}$ denote the adjacency matrix of the graph G on m vertices and assume H is an n-vertex graph obtained by adding $n - m$ new vertices and a set of edges to the graph G. Let $\Psi : \{0, 1, -1\}^{m \times m} \rightarrow \{0, 1, -1\}^{n \times n}$ be a *lifting operator* which

transforms a subspace of $R^{m\times m}$ to a subspace of $R^{n\times n}$ with $n \geq m$. We will call this operator *spectrum preserving* if the eigenvalues of any matrix $\mathcal{A} \in \{0,1,-1\}^{m\times m}$ and its image with respect to the operator ($\Psi(\mathcal{A})$) are the same up to a degeneracy, i.e., the only difference between the spectra of $\mathcal{A}$ and $\Psi(\mathcal{A})$ is the number of zero eigenvalues ($\Psi(\mathcal{A})$ has $n-m$ more zero elements than $\mathcal{A}$). As stated above, our goal is to show that any structural change in graph G can be represented using a spectrum preserving operator and a noise matrix.

Proposition 1. *Let A_H denote the adjacency matrix of the graph H. Then, there exists a spectrum preserving operator $\Psi()$ and a noise matrix $E_H \in \{0,1,-1\}^{n\times n}$ such that $A_H = \Psi(A_G) + E_H$.*

Proof. We define $\Psi()$ as a lifting operator consisting of two steps. First, we will add $n-m$ zero rows and columns to the matrix A_G and denote the resulting matrix by A'_G. Next, A'_G will be pre and postmultiplied by a permutation matrix P and its transpose P^t, respectively, aligning the rows and columns corresponding to the same vertices in A_H and $\Psi(A_G)$. Since the only difference between the spectrum of A'_G and A_G is the number of zero elements and PA'_GP^t has the same spectrum as the matrix A'_G, $\Psi()$ is a spectrum preserving operator. As a result, the noise matrix E_H can be represented as $A_H - \Psi(A_G) \in \{0,1,-1\}^{n\times n}$. □

Armed with a spectrum preserving lifting operator and a noise matrix, we can now proceed to quantify the impact of the noise on the original graph's eigenvalues. Specifically, let $\lambda_i(X)$ for $i \in \{1,\ldots,n\}$ denote the ith largest element of the set of magnitudes of eigenvalues, $\Gamma(X)$, for matrix X. A seminal result of Wilkinson [53] (see also Stewart and Sun [54]) states that:

Theorem 1. *If A and $A+E$ are $n\times n$ antisymmetric matrices, then:*

$$\lambda_i(A) + \lambda_i(E) \leq \lambda_i(A+E) \leq \lambda_i(A) + \lambda_1(E), \quad \text{for all } i \in \{1,\ldots,n\}. \tag{1}$$

For H (perturbed graph) and G (original graph), $A_H = \Psi(A_G) + E_H$, the above theorem yields, for all $i \in \{1,\ldots,n\}$,

$$\begin{array}{ll} \lambda_i(\Psi(A_G)) + \lambda_i(E_H) & \leq \lambda_i(A_H) \leq \lambda_i(\Psi(A_G)) + \lambda_1(E_H) \\ \lambda_i(E_H) & \leq \lambda_i(A_H) - \lambda_i(\Psi(A_G)) \leq \lambda_1(E_H) \\ |\lambda_i(A_H) - \lambda_i(\Psi(A_G))| & \leq |\lambda_1(E_H)|. \end{array} \tag{2}$$

Specifically, we have:

Proposition 2. *(Notation as above), $|\lambda_i(A_H) - \lambda_i(\Psi(A_G))| \leq |\lambda_1(E_H)|$.*

The above chain of inequalities gives a precise bound on the distortion of the elements of spectrum $\Gamma(\Psi(A_G))$ in terms of the largest element of $\Gamma(E_H)$ for noise matrix E_H. Since $\Psi()$ is a spectrum preserving operator, the elements of $\Gamma(A_G)$ follow the same bounds in their distortion.

The above analysis provides a general bound on the distortion of the spectrum as a function of noise. It should be noted that a similar bound can be provided when dealing with more specific classes of noise, such as node merges or splits due to under and oversegmentation. For example, consider the case of a node split due to oversegmentation. Without loss of generality, assume this occurs at the root of a DAG. A generalization of the same argument can be presented for the split of internal nodes.

Specifically, let r denote the root node in DAG G. Let r be replaced by two new nodes, r and r', due to oversegmentation in G', with r being the new root and r' its new child. Considering the worst case scenario, we will also assume that the subgraph $G \setminus \{r\}$ of r in G was broken into two parts, D_1 and D_2, in G' and, while D_1 stays with root r, D_2 becomes a subgraph of r'. First, we observe that the adjacency matrix of G can be represented as (up to a permutation):

$$A_G = \begin{bmatrix} 0 & \mathbf{e}^t & \mathbf{e}^t & \bar{0}^t \\ -\mathbf{e} & A_{D_1} & A_{D_1,D_2} & A_{D_1,G\setminus\{r\cup D_1\cup D_2\}} \\ -\mathbf{e} & -A^t_{D_1,D_2} & A_{D_2} & A_{D_2,G\setminus\{r\cup D_1\cup D_2\}} \\ \bar{0} & -A^t_{D_1,G\setminus\{r\cup D_1\cup D_2\}} & -A^t_{D_2,G\setminus\{r\cup D_1\cup D_2\}} & A_{G\setminus\{r\cup D_1\cup D_2\}} \end{bmatrix},$$

where $\mathbf{e}$ and $\bar{0}$ denote the vectors of all ones and all zeroes, respectively, of appropriate dimension, A_X is the adjacency matrix of subgraph X, $A_{X,Y}$ is the adjacency structure between subgraphs X and Y, and $G \setminus \{r \cup D_1 \cup D_2\}$ is the induced subgraph of G after removing vertex r and subgraphs D_1 and D_2. Observe that matrix A_G has the same nontrivial spectrum as:

$$\Psi(A_G) = \begin{bmatrix} 0 & 0 & \mathbf{e}^t & \mathbf{e}^t & \bar{0}^t \\ 0 & 0 & \bar{0}^t & \bar{0}^t & \bar{0}^t \\ -\mathbf{e} & \bar{0} & A_{D_1} & A_{D_1,D_2} & A_{D_1,G\setminus\{r\cup D_1\cup D_2\}} \\ -\mathbf{e} & \bar{0} & -A^t_{D_1,D_2} & A_{D_2} & A_{D_2,G\setminus\{r\cup D_1\cup D_2\}} \\ \bar{0} & \bar{0} & -A^t_{D_1,G\setminus\{r\cup D_1\cup D_2\}} & -A^t_{D_2,G\setminus\{r\cup D_1\cup D_2\}} & A_{G\setminus\{r\cup D_1\cup D_2\}} \end{bmatrix}.$$

It is not hard to see that the adjacency matrix of graph G' has the following form (again up to a permutation):

$$A_{G'} = \begin{bmatrix} 0 & 1 & \mathbf{e}^t & \bar{0}^t & \bar{0}^t \\ -1 & 0 & \bar{0}^t & \mathbf{e}^t & \bar{0}^t \\ -\mathbf{e} & \bar{0} & A_{D_1} & A_{D_1,D_2} & A_{D_1,G\setminus\{r\cup D_1\cup D_2\}} \\ \bar{0} & -\mathbf{e} & -A^t_{D_1,D_2} & A_{D_2} & A_{D_2,G\setminus\{r\cup D_1\cup D_2\}} \\ \bar{0} & \bar{0} & -A^t_{D_1,G\setminus\{r\cup D_1\cup D_2\}} & -A^t_{D_2,G\setminus\{r\cup D_1\cup D_2\}} & A_{G\setminus\{r\cup D_1\cup D_2\}} \end{bmatrix}.$$

Let n_1, n_2, and n_3 denote the sizes of subgraphs D_1, D_2, and $G \setminus \{r \cup D_1 \cup D_2\}$, respectively, and define the matrices:

$$E_1 = \begin{bmatrix} 0 & 0 & \bar{0}^t & \mathbf{e}^t & \bar{0}^t \\ 0 & 0 & \bar{0}^t & \bar{0}^t & \bar{0}^t \\ \bar{0} & \bar{0} & \tilde{0}_{n_1\times n_1} & \tilde{0}_{n_1\times n_2} & \tilde{0}_{n_1\times n_3} \\ -\mathbf{e} & \bar{0} & \tilde{0}_{n_2\times n_1} & \tilde{0}_{n_2\times n_2} & \tilde{0}_{n_2\times n_3} \\ \bar{0} & \bar{0} & \tilde{0}_{n_3\times n_1} & \tilde{0}_{n_3\times n_2} & \tilde{0}_{n_3\times n_3} \end{bmatrix},$$

$$E_2 = \begin{bmatrix} 0 & 0 & \bar{0}^t & \bar{0}^t & \bar{0}^t \\ 0 & 0 & \bar{0}^t & \mathbf{e}^t & \bar{0}^t \\ \bar{0} & \bar{0} & \tilde{0}_{n_1\times n_1} & \tilde{0}_{n_1\times n_2} & \tilde{0}_{n_1\times n_3} \\ \bar{0} & -\mathbf{e} & \tilde{0}_{n_2\times n_1} & \tilde{0}_{n_2\times n_2} & \tilde{0}_{n_2\times n_3} \\ \bar{0} & \bar{0} & \tilde{0}_{n_3\times n_1} & \tilde{0}_{n_3\times n_2} & \tilde{0}_{n_3\times n_3} \end{bmatrix},$$

and

$$\mathcal{I} = \begin{bmatrix} 0 & 1 & \bar{0}^t \\ -1 & 0 & \bar{0}^t \\ \bar{0} & \bar{0} & \tilde{0}_{(n_1+n_2+n_3)\times(n_1+n_2+n_3)} \end{bmatrix},$$

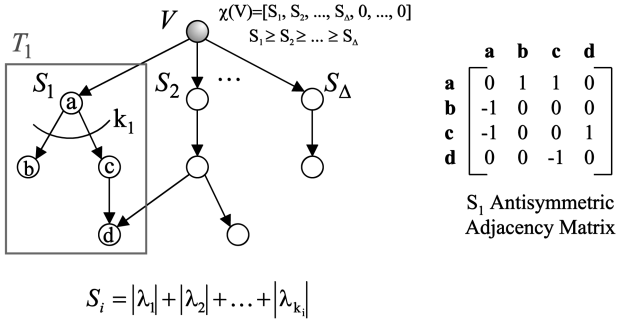


Fig. 3. Forming a topological signature vector, or TSV. The subtrees rooted at the children of the root, V, each yield a sum, the collection of which are sorted to become the components of a vector (the TSV) assigned to V. Each sum, e.g., the sum corresponding to node a, is computed as the sum of the k largest magnitudes of the eigenvalues of the adjacency submatrix corresponding to the directed acyclic subgraph rooted at a, where k is the outdegree of a.

where $\tilde{0}_{x\times y}$ represents the all-zeroes matrix of size $x \times y$. Using this notation, it turns out that $A_{G'} = \Psi(A_G) + (-E_1) + E_2 + \mathcal{I}$. Observe that matrices E_1 and E_2 are isospectral (congruent up to a permutation). Using Proposition 2, this implies that the distortion for the spectrum $\Gamma(G)$ of graph G is bounded by at most $\lambda_1(E_1) + \epsilon$, for a small $\epsilon > 0$, under node split. As mentioned earlier, a similar argument can be also presented for the case of node merge.

The above result has several important consequences for our application of a graph's spectrum to graph indexing. Namely, if the perturbation E_H is small in terms of its complexity, i.e., small spectral radius $\lambda_1(E_H)$, then the spectrum of the new graph H (e.g., the query graph) will remain close to the spectrum of the original graph G (e.g., the model graph), *independent* of where the perturbation is applied to G. The magnitude of the spectral distortion is a function of the number of edges/vertices added/deleted to/from the graph due to noise or occlusion. Specifically, if the noise matrix E_H introduces ℓ new vertices to G, then the distortion of every element of the spectrum can be bounded by $\sqrt{\ell - 1}$ (Neumaier [55]). This bound can be further tightened if the noise matrix has a simple structure. For example, if E_H represents a simple path on ℓ vertices, then its norm can be bounded by $(2\cos\pi/(\ell+1))$ (Lovász and Pelikán [56]). In short, large distortions are due to the introduction/deletion of large, complex subgraphs to/from G, while small structural changes will have little impact on the higher order elements of $\Gamma(A_G)$. The spectrum of a graph is therefore stable under minor perturbations in graph structure.

5 Formulating an Index

Having established the stability of a DAG's spectrum under minor perturbation of the graph, we can now proceed to define an index based on the magnitudes of the eigenvalues. We could, for example, define a vector to be the sorted elements of the spectrum of a DAG, with the resulting index used to retrieve nearest neighbors in a model DAG database having similar topology. However, for large DAGs, the dimensionality of the index (and model DAG database) would be prohibitively large. Moreover, since the graph's spectrum reflects global structure, it cannot accommodate significant occlusion. A more common solution is simply to truncate the spectrum at some (arbitrary) number, but this has the effect of concentrating indexing too heavily on (weak) global aspects of the structure. Our solution to the problem of dimensionality is based on eigenvalue sums rather than on the eigenvalues themselves, while our solution to the problem of occlusion is based on computing both local and global indices.

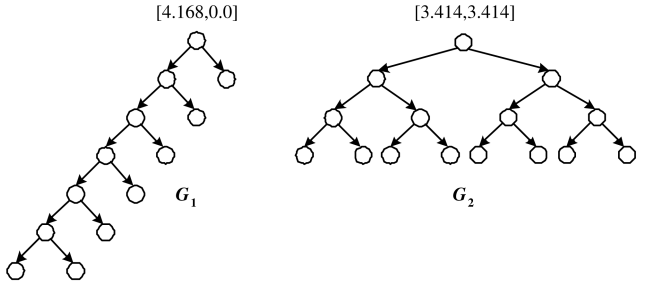


Fig. 4. The TSV's computed for the two trees in Fig. 2. Unlike the naive characterizations, the TSV's provide diverging signatures, reflecting the different structure.

Specifically, let T be a DAG whose maximum branching factor is $\Delta(T)$ and let the subgraphs of its root be $T_1, T_2, \ldots, T_\Delta$, as shown in Fig. 3. For each subgraph, T_i, whose root degree is $\delta(T_i)$, we compute the spectrum of T_i's submatrix, sort the sequence 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_is become the components of a $\Delta(T)$-dimensional vector assigned to the DAG's root. If the number of S_is is less than $\Delta(T)$, then the vector is padded with zeroes. We can recursively repeat this procedure, assigning a vector to each nonterminal node (i.e., node with nonzero out-degree) in the DAG, computed over the subgraph rooted at that node. We call each such vector a *topological signature vector*, or TSV. Returning to our example of Fig. 2, we now have diverging signatures, as shown in Fig. 4.

Although the sum of eigenvalue magnitudes is invariant to any consistent reordering of the DAG'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 magnitude from each subgraph (nonunique but less ambiguous), but this would be less representative of the subgraph's structure. We choose the $\delta(T_i) - 1$ largest eigenvalues for two reasons. First, the largest eigenvalues are more informative of graph structure. Second, by summing $\delta(T_i) - 1$ elements, we effectively normalize the sum according to the local complexity of the subgraph's root. Since we assume that the branching structure is more salient at higher levels in a hierarchical structure, this gives us a mechanism for at least distinguishing increased branching structure at the root. As shown in [6], these eigenvalue sums can be computed in polynomial time.

The above definition of the TSV ignores information related to terminal nodes (leaves). The TSV of a DAG, G_v, rooted at node v, that has terminal nodes $\{l_i\}_{i=1}^{n>0}$ as its children, will be equal to that of the graph with these leaves

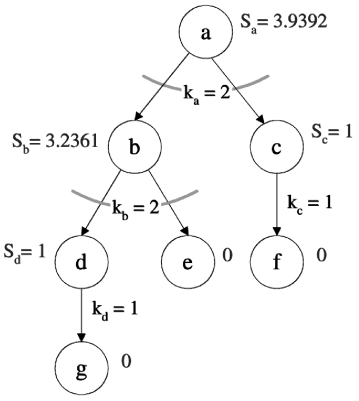


Fig. 5. An augmented signature at every nonterminal node. Next to each node is the result of summing the $k_v = \delta(G)$ largest eigenvalues corresponding to the subgraph rooted at the node v. The TSV indices are $\chi(a) = [3.9392, 3.2361, 1]$, $\chi(b) = [3.2361, 1, 0]$, and $\chi(c) = \chi(d) = [1, 0, 0]$ (padded with zeros according to the maximum branching factor).

removed. The reason for this is that the zero eigenvalue magnitude sum of a leaf node is indistinguishable from the padded zeros of a TSV. To solve this, we simply add, as an extra dimension, the eigenvalue magnitude sum S_v corresponding to G_v. Since this sum will be greater than the sum of the eigenvalue magnitude sums of the subgraphs rooted at the children of v, it will always be the first dimension of the TSV (see Fig. 5).

Our topological index satisfies the six criteria outlined in Section 3. The eigen-decomposition yields a low-dimensional (criterion 1) vector assigned to each node in the DAG, which captures the local topology of the subgraph rooted at that node (criterion 2—this will allow us to handle occlusion and will be addressed in Section 6). Furthermore, a node's vector is invariant to any consistent reordering of the node's subgraph (criterion 3). The components of a node's vector are based on summing the largest eigenvalues of its subgraph's adjacency submatrix. Although our dimensionality-reducing summing operation has cost us some uniqueness, since the elements of each sum are positive values and are monotonic with respect to structural complexity, the partial sums still have relatively low ambiguity [54] (criterion 4). From the sensitivity analysis in Section 4, we have shown our index to be stable to minor perturbations of the DAG's topology (criterion 5). As shown in [6], these sums can be computed even more efficiently (criterion 6) than the eigenvalues themselves. Observe that two directed graphs T and T', with adjacency matrices A_T and $A_{T'}$, are isomorphic if and only if there exists a permutation matrix P such that $A_{T'} = P^t A_T P$ [57]. In particular, A_T and $A_{T'}$ are similar and therefore have the same characteristic polynomial, spectrum, and determinant. As a result, the vectors obtained from the aforementioned labeling procedure for all DAGs isomorphic to T will be the same in $R^{\Delta(T)-1}$. Moreover, this extends to any DAG T' which has a subgraph isomorphic to a subgraph of T, i.e., the TSV labeling of T and the corresponding isomorphic subgraph of T' will be identical. But there is more to the fundamental trade-off between uniqueness and compactness described earlier, which we next address in the process of candidate selection (see Section 6).

6 Candidate Selection

Given a query DAG corresponding to an image, our task is to search the model DAG database for one or more model DAGs which are similar to the image DAG. If the number of model DAGs is large, an exhaustive 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 DAG (or one of its subgraphs, if it is occluded or poorly segmented). Hence, we begin by mapping the topology of the image DAG 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.

6.1 A Database for Model Graphs

Our eigenvalue characterization of a DAG's topology suggests that a model DAG's topological structure can be represented as a vector in $\delta + 1$-dimensional space, where δ is an upper bound on the degree of any vertex of any image or model DAG. If we could assume that an image DAG represents a properly segmented, unoccluded object, then the TSV computed at the image DAG's root could be compared with those topological signature vectors representing the roots of the model DAGs. It follows from Proposition 2 that the vector distance between the image DAG's root TSV and a model DAG's root TSV will decrease as the similarity of their respective DAGs increases, as finding two subgraphs with "close" eigenvalue sums represents an approximation to finding their largest isomorphic subgraph.

Unfortunately, this simple framework cannot support large structural perturbations (caused by, for example, scene clutter or large occlusion), which could result in the addition or removal of significant structure. In either case, altering the structure of the DAG will affect the TSVs computed at its nodes. The signatures corresponding to the roots of those subgraphs (DAGs) that survive the occlusion will not change. However, the signature of a root of a subgraph that has undergone significant perturbation will change significantly which, in turn, will affect the signatures of any of its ancestor nodes, including the root of the entire DAG. We therefore cannot rely on indexing solely with the root's signature. Instead, we will exploit the local subgraphs 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., [58], [59], [60], [61]. As shown in Fig. 6, rather than storing a model DAG's root signature, we will store the signatures of *each* nonterminal node in the model DAG, along with a pointer to the object model containing that node as well as a pointer to the corresponding node in the model DAG (allowing access to node label information, e.g., their shock types in the case of shock graphs). Since a given model subgraph can be shared by other model DAGs, a given signature (or location in $\delta + 1$-dimensional space) will point to a list of (model object, model node) ordered pairs. At runtime, the signature at

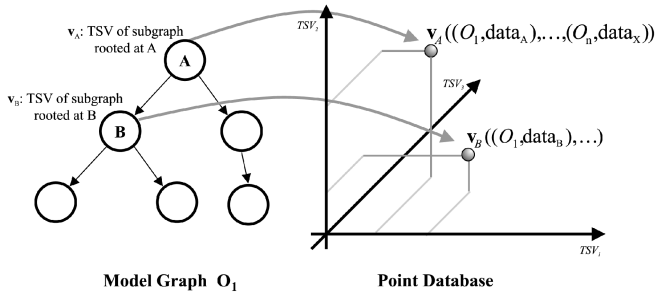


Fig. 6. Populating the Model Database. The TSV at each nonterminal node in a model graph defines a position in the database at which a pointer to both the model and the node is stored. The curved arrows indicate this *insertion* process.

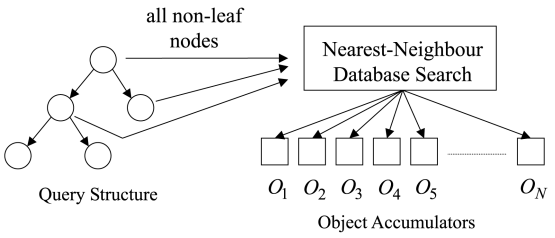


Fig. 7. Voting for Model Graphs. Each nonterminal node in the query defines a TSV, which is used to vote for nearby model TSVs which, in turn, represent subgraphs of model graphs.

each nonterminal node in the image DAG becomes a separate index, with each nearby candidate in the database "voting" for one or more (model object, model node) pairs.

The above indexing framework provides the locality of representation required to support large perturbations, such as occlusion. However, there is another important benefit with regard to isospectral graphs, i.e., graphs that are not isomorphic but have the same spectrum (see Cvetković et al. [57]). Let p denote the probability of such an event, i.e., the probability of two nonisomorphic graphs being isospectral. Under a uniform accumulation voting model, the TSV of every nonterminal node will generate a vote (or set of votes, depending on the number of models that share such a substructure) and the model with the maximum votes will be selected. An erroneous model selection (model DAG D_1 and query DAG D_2 are iso-spectral) under our scheme implies that not only are the graphs isospectral, but the majority of their (nonleaf) matching subgraphs are also isospectral. Since a node's votes are independent, the probability of this event is at most $p^m < p^{n/2}$, which is an asymptotically vanishing sequence for large values of n.

6.2 Searching the Model Database

At runtime, we compute the indices of all the nonterminal nodes in the query and find their nearest neighbors in the index database. Nearest-neighbor (NN) search in high-dimensional spaces is a well-studied subject, with a recent survey found in [62]. When performing a NN search, we can retrieve either the k nearest-neighbors (k-NN) of a query point or all the points within a fixed-size hypervolume centered at a query point. The k-NN algorithm has the advantage of fixing the number of retrieved points regardless of their distance to the query, with the risk of retrieving points far from the query; however, a small k may exclude good candidate points. Range queries, on the other hand, avoid the problem of excluding good candidates at the cost of a potentially large number of matches. For our experiments, we have adopted the SR-tree technique proposed by Katayama and Satoh [63], providing an effective range query mechanism.

The retrieved points vote for their associated model graphs and accumulate the votes as evidence, as shown in Fig. 7. Each such retrieved point generates 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, e.g., the shock type, is not compatible with the label of its corresponding query 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 subgraph, allowing larger and more complex subgraphs (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 [64], [65]. Since the magnitudes of the eigenvalues, and, hence, their sum, are proportional to both the branching factor as well as graph size, the magnitude of the signature is also used to weight the vote. If we let $\chi(q)$ be the TSV of a query graph node q and $\chi(m)$ be the TSV of a model graph node m that is sufficiently close, the weight of the resulting vote, i.e., the local evidence for the model, is computed as:

$$W(q,m) = \frac{||\chi(q)||}{1 + ||\chi(m) - \chi(q)||}, \tag{3}$$

where $||\cdot||$ denotes the L_2-norm.

The weighting function (3) fails to account for how much of the query is explained by the model or how much of the model is explained by the query. We can incorporate an Occam's Razor-like principle into our ranking algorithm —we want the simplest model that best accounts for the query. This can be easily accomplished by redefining the vote weighting function, such that, for a query point q and neighboring model point m, we would like to increase the weight of the vote for an object model M, if m represents a larger proportion of M. Similarly, we would like to increase the weight of the vote for M if q represents a larger proportion of the query. The relative merit of these two goals is a function of the task domain and depends, for example, on whether queries are cluttered or have missing data and the extent to which model objects share structure. Our weight function therefore becomes:

$$\begin{aligned} W(q,m) = &\frac{(1-\omega)||\chi(q)||}{\Sigma_{\mathcal{Q}}(1 + ||\chi(m) - \chi(q)||)} \\ &+ \frac{\omega||\chi(m)||}{\Sigma_{\mathcal{M}}(1 + ||\chi(m) - \chi(q)||)}, \end{aligned} \tag{4}$$

where q and m are nodes in the query graph $\mathcal{Q}$ and model graph $\mathcal{M}$, respectively. The weight is normalized by the sums of the TSV norms of the entire query and model graphs, i.e., $\Sigma_{\mathcal{G}(V,E)} = \sum_{v \in V} ||\chi(v)||$. The convexity parameter

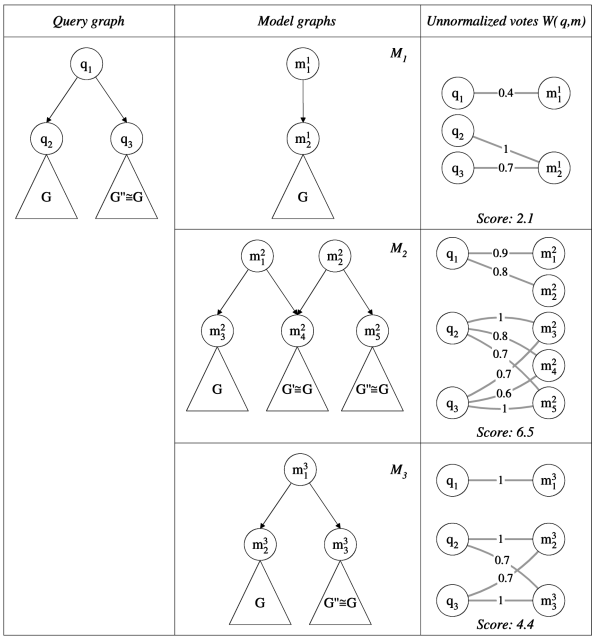


Fig. 8. Limitations of a Simple Vote Accumulation Strategy. Consider a query graph and a small database with only three models. For simplicity, votes are not normalized and, consequently, do not add up to 1. Here, $W(q,m)$ represents index similarity, where perfect similarity equals 1 and total dissimilarity equals 0. Although the query is isomorphic to M_3, model M_2 emerges with more votes, due to the fact that query structure is replicated in M_2 and can, therefore, unfairly vote multiple times.

ω weights the two goals above: The first term favors models that cover a larger proportion of the query, while the second term favors models with more nodes accounted for. By normalizing by $\Sigma_{\mathcal{Q}}$ and $\Sigma_{\mathcal{M}}$, we obtain vote weights whose sum is in the interval $[0,1]$ for a one-to-one vote correspondence between query nodes and model nodes. When $w = 1/2$, for example, the sum of all votes for a given model will be equal to one when there is a one-to-one mapping between all query nodes and all model nodes, and their corresponding TSVs are equal.

6.3 Evidence Accumulation

Counting the votes is not necessarily a trivial task. A simple algorithm is to let every nonterminal query node vote for several points, each of which, in turn, votes for a set of models. Then, select the models with the most votes to be candidates, an approach followed, for example, in our previous work [66]. However, as shown in Fig. 8, this many-to-many voting scheme is problematic. Even though model M_3 is isomorphic to the query, model M_2 receives the most votes because query structure is replicated in M_2. A fair voting process must ensure that no query node votes for more than one node in the same model and that no model node receives more than one vote from the query nodes. In other words, given a *large number of models*, we want to find the best one-to-one assignment of votes from query nodes to model nodes for each candidate model.

A simple way of ensuring a one-to-one assignment of votes is to maintain, for each candidate model M, a list of all (q_i, m_j) pairs of votes, for $q_i \in Q$ and $m_j \in M$. Next, compute a maximum weight bipartite matching (MWBM) per model graph, yielding a one-to-one assignment of votes per model and, ultimately, the overall vote for each model.

Unfortunately, this approach is inefficient. Namely, regardless of whether or not there are many-to-many assignments of votes, we will have to compute k maximum matchings, where k is the number of models that receive at least one vote, resulting in a matching complexity of order proportional to the number of votes and the number of nodes *in the models*, which is potentially large. Clearly, the gain in performance in finding the optimal one-to-one assignment of votes is offset by its increased computational complexity.

We instead propose an algorithm that finds the optimal solution by focusing only on those queries that will benefit from a careful counting of votes. The idea is to compute the distances between all query nodes' TSV's and use these distances to determine which query nodes might vote for multiple nodes in the same model. Given the range for the NN search (the diameter of the search hypervolume centered at the query point), we know that two query points whose distance from each other is within this range are potential competitors, i.e., there may be points belonging to a model that fall within the range of both query points, yielding a potential many-to-many mapping. Query points further apart, however, cannot possibly map to the same model points. Only for those sets of "nearby" query points do we need to compute maximum matchings to ensure a one-to-one vote correspondence.

A query point in isolation may vote for many different models and multiple nodes per model. We must ensure that, for each model, the query node votes only for a single (closest) model node. If there are $|Q|$ "nearby" query nodes voting for model nodes belonging to models in M, we will compute $|M|$ bipartite matchings, each mapping the set of $|Q|$ query nodes to the nodes belonging to a particular model and receiving votes from a query node in Q. In fact, for each member of Q, we need to hypothesize a correspondence (i.e., edge in the bipartite graph) only to the closest $|Q|$ model nodes receiving votes from that member. This bounds the complexity of the bipartite matching problem for a given set Q.

To provide a precise specification of the algorithm, we first need to define a number of data structures. Let $P_1, \ldots, P_n$ be a partition of the query points' topological indices $\chi(q) \neq \vec{0},\ \forall q \in \mathcal{Q}$, such that two indices that *overlap* belong to the same partition. Given the range queries $\vec{r}_1$ and $\vec{r}_2$ for two subgraphs rooted at $q_1, q_2 \in \mathcal{Q}$, the indices $\chi(q_1)$ and $\chi(q_2)$ are said to overlap if the d-dimensional hypervolume centered at $\chi(q_1)$ and $\chi(q_2)$ with dimensions defined by $\vec{r}_1$ and $\vec{r}_2$ intersect one another (see Fig. 9).[3] Let $S_{j,1}, \ldots, S_{j,p}$ be the sets of nearest-neighbor points in the database that are within range $\{\vec{r}_l\}_{l=1}^{p}$ of the query node indices $\{\chi(q_l)\}_{l=1}^{p}$ in the same partition P_j, for $p = |P_j|$ and $1 \leq j \leq n$. Finally, let $V_{i,j}$ be a $p \times p$ matrix containing in each column q the best p votes for the object model graph $\mathcal{M}_i$ received from every index $\chi(q) \in P_j$.[4] See Fig. 10 for an

3. A range query is usually defined by a weighted L_s distance. Given two n-dimensional vectors $\vec{x}$ and $\vec{y}$, the L_s distance between them is defined as $L_s(\vec{x},\vec{y}) = \left(\sum_{i=1}^{n} w_i |\vec{x}_i - \vec{y}_i|^s\right)^{1/s}$. In our experiments, we use the weighted maximum distance (L_∞) for all the range queries, i.e., $L_\infty(\vec{x},\vec{y}) = \max\{w_i|\vec{x}_i - \vec{y}_i|\}$. We set the weights so that the range search returns all the TSV's within a radius of 40 percent of each dimension of the query node's TSV.

4. This can by implemented by a list of p priority queues bounded by length p. In the worst case, this data structure would hold $p \times p$ elements.

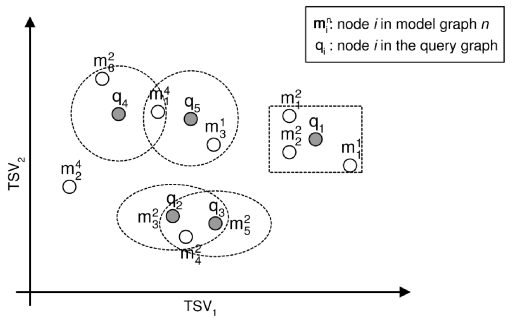


Fig. 9. An example of overlapping index ranges for a TSV space of dimension $d = \max(\Delta(\mathcal{Q}), \max\{\Delta(\mathcal{M}_i)\}) = 2$. Each point v represents the 2D TSV $\chi(v)$ of node v. In this example, the range query of nodes q_2 and q_3 is determined by a weighted L_2-norm, while that of q_1 is given by a weighted L_∞-norm. In general, a weighted L_s-norm is used to allow more variation along particular dimensions. The range queries for nodes q_4 and q_5, on the contrary, are specified by an unweighted L_2-norm that treats every dimension equally. The TSV's of nodes q_2 and q_3, along with those of q_4 and q_5, are said to overlap for their given range queries.

example of these matrices, which we will use to buffer the votes for each model and for each set of overlapping indices. We can now define the multiple one-to-one vote correspondence algorithm (MOOVC), as shown in Algorithm 1; its complexity and optimality analysis are given in [9].

Algorithm 1. The Multiple One-to-One Vote Correspondence (MOOVC) Algorithm.

1. compute the sets $\{P_j\}_{j=1}^{M}$ of overlapping nodes in the query graph.
2. **for** every partition set P_j, $1 \le j \le M$ **do**
3. let $\mathcal{S}$ be an empty set of $|P_j| \times |P_j|$ matrices $V_{i,j}$.
4. **for** each node $q_n \in P_j$, $1 \le n \le |P_j|$ **do**
5. perform a range search for the NN of q_n, forming a list $\mathcal{L}$ of model nodes m_j^i, such that m_j^i is the node j in model graph i within the specified range from q_n.
6. **for** each node $m_j^i \in \mathcal{L}$ **do**
7. add the tuple $(W(q_n, m_j^i), m_j^i)$ to a matrix $V_{i,j} \in \mathcal{S}$ if $W(q_n, m_j^i)$ is greater than the smallest vote in column n of $V_{i,j}$. (V can be implemented by $|P_j|$ priority queues, each bounded to length $|P_j|$.)
8. **end for**
9. **end for**
10. compute a maximum weight bipartite matching (MWBM) for each matrix $V_{i,j} \in \mathcal{S}$ so as to obtain the best one-to-one assignment of votes $\{(q, m)\}_{v=1}^{N \le |P_j|}$ from query nodes in P_j to model nodes.
11. Finally, for each $V_{i,j} \in \mathcal{S}$, sum over the resulting N-mapping of votes,

$$T_{i,j} = \sum_{v=1}^{N} W(\mathrm{MWBM}(V_{i,j})_v),$$

and tally a vote $T_{i,j}$ for each object model $\mathcal{M}_i$ from the partition P_j. This vote is the largest sum that can be obtained by a one-to-one assignment of votes from query nodes in the partition j to the nodes in the model graph i.

12. **end for**

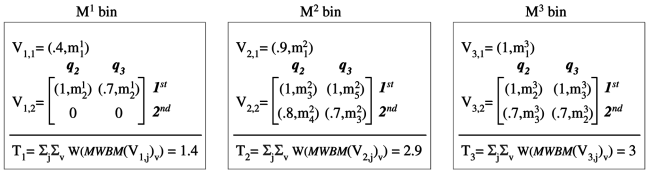


Fig. 10. Intermediate steps of the MOOVC algorithm for the example in Fig. 8. In the example, the range queries induce the partitions $P_1 = \{\chi(q_1)\}$ and $P_2 = \{\chi(q_2), \chi(q_3)\}$. The results, S_i, of the range queries for each query node i are $S_1 = \{\chi(m_1^1), \chi(m_1^2), \chi(m_2^2), \chi(m_1^3)\}$, $S_2 = S_3 = \{\chi(m_2^1), \chi(m_3^2), \chi(m_4^2), \chi(m_5^2), \chi(m_2^3), \chi(m_3^3)\}$.

After the evidence accumulation is complete, those models whose support is sufficiently high are selected as candidates for verification. The bins containing the overall votes for every model $\mathcal{M}_i$ and the vote buffer matrices $V_{i,j}$ can, in effect, be organized in a heap, requiring a maximum of $O(\log \ell)$ operations to maintain the heap when evidence is added, where ℓ is the number of nonzero object accumulators.

7 Experiments

We evaluate our indexing framework on the task of view-based 3D object recognition, including both object identification (correct object) and pose estimation (correct position on the object's viewing sphere). In any given trial, a query view is presented to the system and the top $\mathcal{K}$ candidate views from the entire database (less the query) receiving the most votes are returned. For a recognition task, these $\mathcal{K}$ candidates would then be verified by computing the distance between each candidate and the query, with the closest candidate taken as the response to the query. However, in this paper, we explore only the problem of indexing and would like to know how highly ranked the correct response is *prior* to verification. If its rank is high, that would imply that we would need to verify fewer candidates, resulting in a recognition system with better performance.

For object identification, we consider the identity of a candidate view to be correct if its corresponding 3D object is the same as that of the query view. For pose estimation, we consider the pose of a candidate view to be correct if it represents one of the query view's nine closest neighbors on the viewing sphere, as shown in Fig. 11. For both identification and pose estimation, we measure indexing performance by the position of the highest ranking candidate view whose identity or pose, depending on the

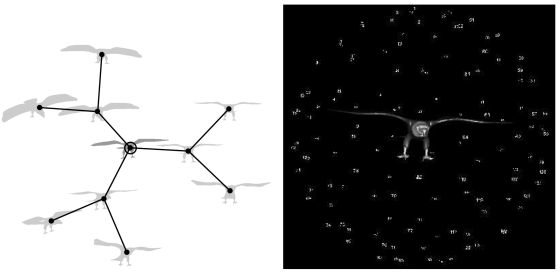


Fig. 11. The structure of the viewing sphere. Left: configuration of the nine closest neighbors of the query view (center) on the viewing sphere. Right: one of the query's neighbors, as seen on the 3D viewing sphere.

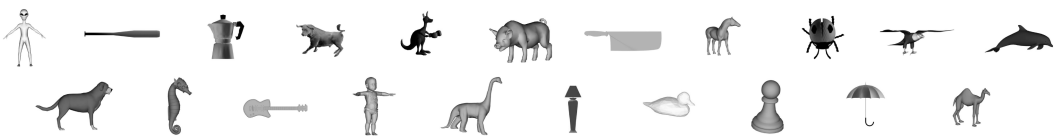


Fig. 12. The 21 3D models used to generate a database of 2,688 views. The objects are chosen so as to present a large variation in terms of shape complexity. Note that simple objects (e.g., 1, 6, 16, and 19) are represented by small graphs that are also subgraphs of larger graphs in the database.

task, is correct. Averaging over 2,688 trials (i.e., every model view becomes a query), this gives us a sense for how large $\mathcal{K}$ should be in order to ensure that the correct response is among the candidates.

It is important to note that our criterion for correct pose estimation is somewhat harsh for a number of reasons. Since we sample the entire viewing sphere, views on one side of the viewing sphere are similar to views on the opposite side. This similarity is more pronounced as the viewing distance, i.e., the radius of the viewing sphere, grows with respect to the object's depth, i.e., the camera geometry approaches orthographic projection. Moreover, additional object planes of reflective symmetry or axes of rotational symmetry may introduce additional ambiguous views with respect to the query. These sources of ambiguity, which should really be removed from the database, effectively reduce the ranking of the best ranked of the query's nine neighbors. In the following sections, we evaluate the performance of the indexer on both tasks while varying the database size, the viewing sphere sampling resolution, and the degree of missing data. First, however, we describe the particular hierarchical, view-based 3D shape representation that we use in the experiments.

7.1 A Directed Acyclic Graph Representation of 2D Shape

Our indexing framework will support any hierarchical (DAG-based) image or shape description, such as a scale space [1], [2]. In this paper, we demonstrate our framework using a hierarchical shape representation of a 2D silhouette based on a coloring of the shocks (singularities) of a curve evolution process acting on simple closed curves in the plane [5]. We have recently abstracted this system of shocks into a *shock graph* where vertices are labeled by their shock types and the shock formation times direct the edges (see Fig. 1). 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 [6] and demonstrated it on a small database of shock trees computed using curve evolution techniques. The far more extensive experimentation carried out in the current article is based on shock graphs computed using the more efficient and robust skeletonization algorithm developed in [7], [8] and the shock graph construction algorithm reported in [9].

The experiments reported in this paper are applied to a database of 2,688 shock graphs, representing 21 objects, shown in Fig. 12, with 128 views per object. For each view, we obtain the shock graph pertaining to the outermost contour of the silhouette. With any indexing or recognition application, success is a function of the contents of the database. Specifically, one expects results to be better as the distance between database members increases, i.e., the ambiguity decreases. The fact that nearby views of the same object can have similar shock graphs only increases database ambiguity. Since the identification task compares the *parent* (3D object) of the query to the *parent* of a candidate view, the effects of this particular type (nearby view) of ambiguity are minimized. However, for the pose estimation task, this ambiguity represents a major challenge and better results would be expected on a database of 2,688 distinct objects with one canonical view chosen per object.

7.2 Varying the Number of Objects

In this experiment, we examine the scalability of our indexing framework. We plot the average position of the correct response in the indexing ranking as a function of the database size, fixing the number of views per object at 128. Starting with a database of seven objects (896 views), the seven objects shown in the first row of Fig. 12, we consecutively add two objects at a time, following the row-major order in Fig. 12, until we have 21 objects (2,688 views) in total. The results of the experiment are shown in Fig. 13. In Fig. 13a, we show the average of the ranking position of the correct response as a function of database size. The curves, from bottom to top, correspond to the increasing database sizes (i.e., 7, 9, 11, 13, 15, 17, 19, and 21). In Fig. 13b, we plot the number of candidates that need to be examined to ensure that, in a given percentage of the trials, the correct response is among the candidates. Figs. 13c and 13d give the corresponding plots for pose estimation.

Fig. 13 shows that the indexing strategy is effective in reducing the number of models that need to be verified. The average rank of the correct response is low both in object recognition[5] and in pose estimation. For example, in the largest database of 2,688 views, many of which are ambiguous, the highest rank of one of the query's neighbors (Fig. 13c) is still in the top 40, on average. We would expect this rank to decrease with effective view clustering and the removal of ambiguous views due to symmetries. Figs. 13b and 13d also gives us some insight into appropriate values of $\mathcal{K}$ in our indexing strategy. For example, in the database with 2,688 views, we only need to verify the first approximately 50 (for object recognition) and 250 (for pose estimation) candidates to ensure that in 98 percent of the trials, the correct response is included.

5. Here, the average rank varies mostly as a function of of the ambiguity introduced by the class of graphs added to the database.

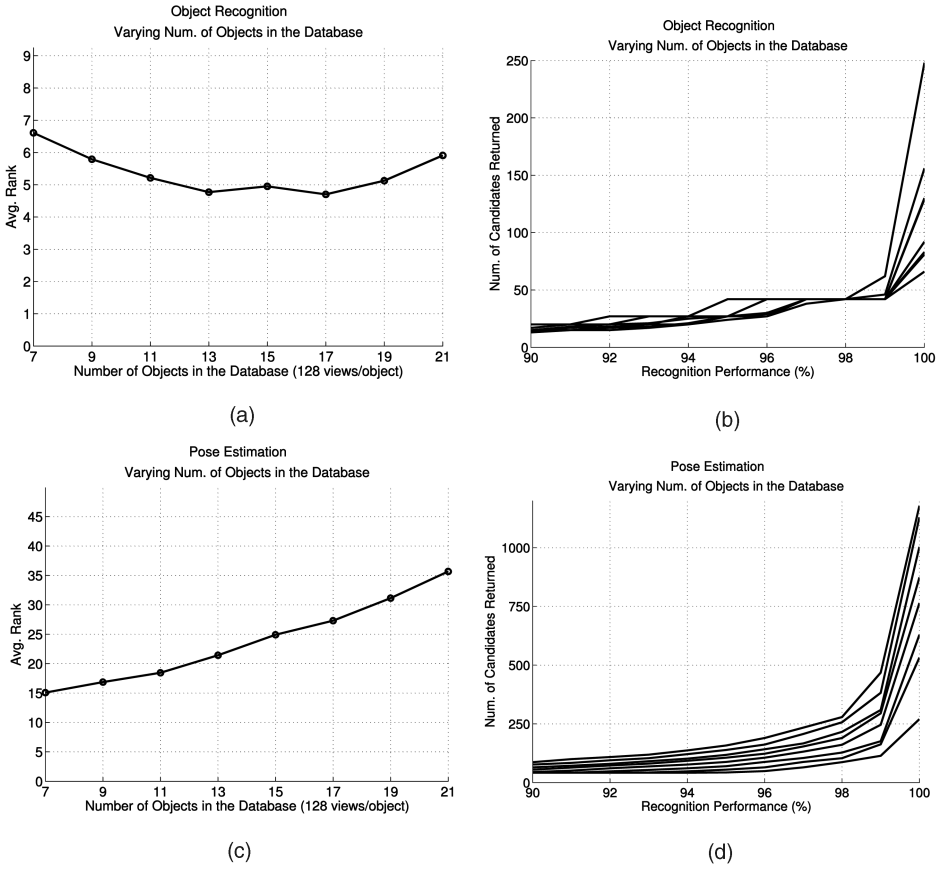


Fig. 13. Indexing performance as a function of increasing database size. (a) Rank of correct response for object identification; (b) number of candidates required to ensure that, in a given percentage of the 2,688 trials, the correct response is among the candidates with eight curves reflecting increasing database sizes from bottom to top; (c) and (d) are the corresponding plots for pose estimation.

7.3 Varying the Sampling Resolution

In this experiment, we examine the performance of our indexing framework as a function of viewing sphere sampling resolution. Having fewer views per object reduces the size of the database, thereby reducing search complexity. Fewer views per object also reduces the number of ambiguous views per object, which may boost the ranking of the correct response. However, the cost of subsampling is the increased distance between a query and its neighbors on the viewing sphere. In the worst case, as the sampling resolution falls, all of a view's immediate neighbors may belong to entirely different view classes, making it very unlikely that any of the query's neighbors will rank highly. This is a limitation of our evaluation strategy and could be alleviated if we were to ensure that at least two samples from each view class remain (since our queries are taken from the database without replacement). Ultimately, only one prototypical view per view class need be stored in the database for effective recognition (and qualitative pose estimation).

The subsampling is performed using a simple strategy. For each object, we generate subsampled (from the original 128 views per object) view sets of size 96, 64, and 32 by randomly eliminating views on the object's viewing sphere. Next, we use every view in the original database with 128 views per object as a query against each subsampled database (the query view is also removed from the subsampled model database if it is still present). We conducted 2,688 trials for each of three randomly sampled databases for each of the three sampling resolutions (a total of 24,192 trials) and computed the average performance with the results shown in Fig. 14. It should be noted that our random strategy for subsampling the viewing sphere is clearly inefficient since it does not focus on eliminating redundant views from the database. As sampling resolution is lowered, redundant views may remain while other views may lose all their similar neighbors, effectively reducing indexing performance.

Figs. 14a and 14b plot identification and pose estimation performance, respectively, as a function of sampling resolution. We can see that the performance on identification and pose estimation is still strong when sampling resolution drops to 64 views per object (over the entire viewing sphere). Furthermore, when the sampling resolution drops to 32 views per object, the decrease in performance is more pronounced but still degrades gracefully, reflecting the stability of the indexing and the viewpoint invariance of the shock graph representation.

7.4 Varying the Degree of Missing Data

In the final experiment, shown in Fig. 15, we plot recognition performance (for the entire database) as a function of degree of missing data. This experiment is designed to reflect the ability of the indexing framework to

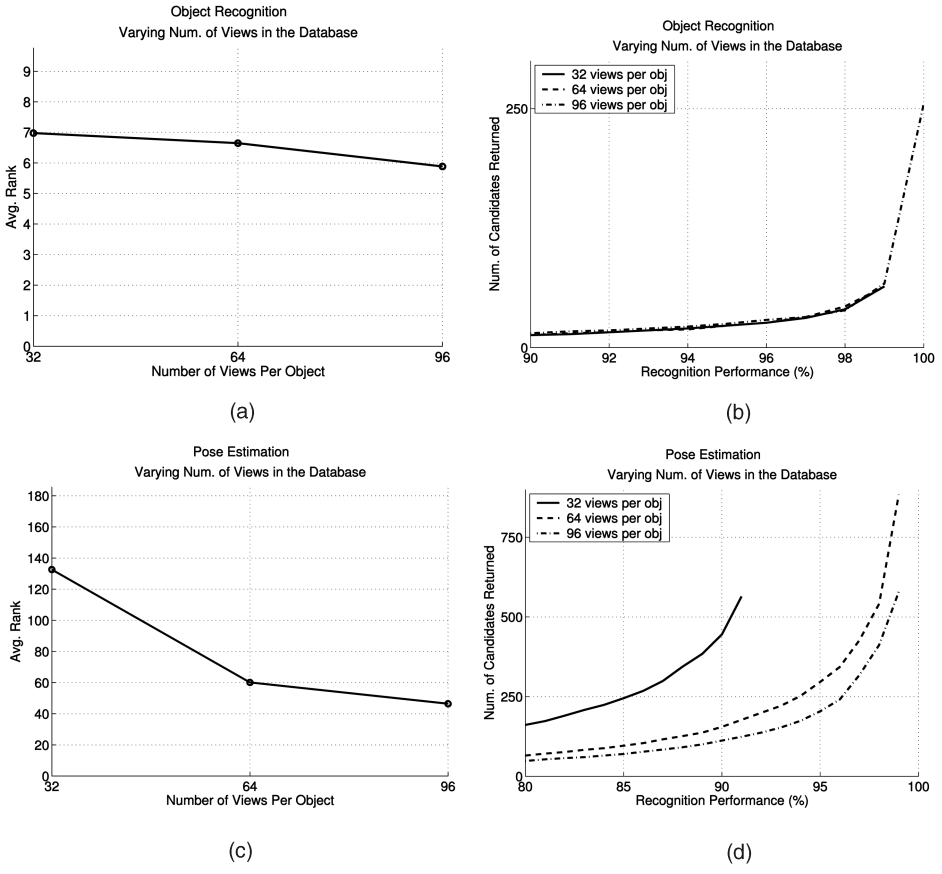


Fig. 14. Varying the viewing sphere sampling resolution: (a) object identification as a function of increased sampling resolution; (b) number of candidates required to ensure that, in a given percentage of the trials, the correct response is among the candidates; (c) and (d) are the corresponding plots for pose estimation. Note that, when the number of views per object is low, there are some queries for which none of its neighboring views are in the database.

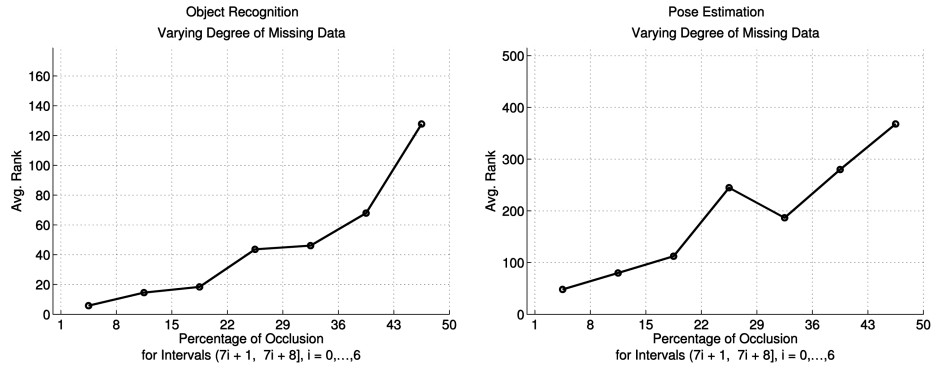


Fig. 15. Recognition performance as a function of amount (percentage) of missing data in the query.

accommodate large-scale perturbation. To generate a partial query, we randomly choose a node in the query DAG and delete the subgraph rooted at that node, provided that the order (size) of the original graph does not drop by more than 50 percent. We repeated the experiment 10 times for a total of 26,880 trials.

As shown in Fig. 15, the performance of the indexer decreases gradually as a function of the amount of missing data, reflecting the framework's ability to effectively index with partially visible objects. Since an increasing amount of missing data does not have a dramatic impact on the performance, we can conclude that the framework satisfies the important property of stability.

These results, representing the first systematic large-scale perturbation study in the shock graph community, can be considered an approximation to a form of "graph" occlusion (with an invisible occluder) or a form of region oversegmentation. Granted, this is not equivalent to placing randomly shaped occluders with random sizes over random parts of the query object, but it does represent a significant test of the framework's ability to successfully index on the basis of local information. It should also be

$\mathcal{K} = 200$	Unormalized Many-to-Many Votes	Normalized Many-to-Many Votes	Unormalized One-to-One Votes	Normalized One-to-One Votes
Obj. Rec.	75.7	95.4	**99.7**	**99.7**
Pose Est.	60.1	81.7	94.4	**96.2**

Fig. 16. Comparing the indexing framework to a preliminary version [66] that did not consider how well the model accounted for the query (and vice versa) nor computed a one-to-one vote correspondence. The boldface entries represent the best performance for each task. See text for a detailed comparison.

pointed out that, for any of our pose estimation experiments, the fact that the query is removed from the database means that it must match a "nearby" view whose structure, when similar, can be thought of as a noisy version of the query. Together, we believe these effects make a strong case for the robustness of the framework.

7.5 A Comparison with Our Previous Approach

We can compare the above results to a preliminary version of our indexing framework, reported in [66], in which we adopted both the simple vote weighting function, given in (3), and the simple, many-to-many voting accumulation strategy which led to our MOOVC algorithm for one-to-one vote correspondence. Fig. 16 compares the improved framework to its predecessor. In the second column, the indexing votes are weighted according to (3) instead of (4), while allowing a many-to-many correspondence for the nodes in each model graph. The third column shows the performance when a one-to-one vote correspondence is enforced and the votes are not weighted according to the improved function, (4). The fourth column shows the performance of the indexing algorithm when a one-to-one assignment of votes is allowed and each vote is weighted by (3), as was originally proposed in [66]. The final column is the normalized, one-to-one voting strategy. These results reveal that a one-to-one vote correspondence produces much better indexing performance and that (4) is particularly important if a one-to-one vote correspondence is not ensured.

8 Conclusions

We have presented a low-dimensional description, or signature, of a directed acyclic graph's topology that provides an effective mechanism for indexing into large databases of DAGs. The signature is based on a novel combination of the spectral properties of the graph's underlying adjacency matrix. Our sensitivity analysis of these properties under minor perturbation due to noise, occlusion, and node split/merge establishes the stability of these properties and, hence, the signature. Each nonterminal node in a query graph yields an independent index which can be used to efficiently accumulate evidence for a small set of models sharing the query's topological structure in the presence of large perturbation, such as occlusion. In a series of experiments in the domain of 2D object recognition using shock graphs, 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 our indexing framework scales well with increasing database size, with decreasing sampling resolution, and with increasing perturbation, in the form of missing data. Still, 2,688 graphs is not a large database and we plan to explore the framework's scalability on databases containing tens of thousands of graphs. This may require the integration of both node and edge attributes into the index to reduce ambiguity. Although we demonstrate the framework on a shape representation (computer vision) application, the framework is indeed general and could be applied to any indexing domain including, for example, any hierarchical or multiscale structure in computer vision, part-whole hierarchies in knowledge representation, syntax or parse trees in computational linguistics, CSG trees in computer graphics, and phylogenetic trees in computational biology.

Acknowledgments

The authors would like to acknowledge the programming support of Maxim Trokhimtchouk, Carlos Phillips, and Pavel Dimitrov. The authors would also like to express their thanks to Allan Jepson for his comments on earlier drafts of this work and to the three reviewers for their thoughtful comments and suggestions; all have helped to improve the quality of the paper. They are also grateful to Norio Katayama for the use of his SR-tree implementation. Finally, the authors would like to acknowledge the generous support of NSERC, CITO, PREA, US National Science Foundation, AFOSR, ONR, DARPA, CFI, and FQRNT.

References

[1] A. Witkin, "Scale Space Filtering," *From Pixels to Predicates,* A. Pentland, ed. Norwood, NJ: Ablex, 1986.

[2] T. Lindeberg, "Detecting Salient Blob-Like Image Structures and Their Scales with a Scale-Space Primal Sketch—A Method for Focus-of-Attention," *Int'l J. Computer Vision,* vol. 11, no. 3, pp. 283-318, 1993.

[3] S. Dickinson, A. Pentland, and A. Rosenfeld, "3-D Shape Recovery Using Distributed Aspect Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 174-198, Feb. 1992.

[4] S.J. Dickinson, A.P. Pentland, and A. Rosenfeld, "From Volumes to Views: An Approach to 3-D Object Recognition," *CVGIP: Image Understanding,* vol. 55, no. 2, pp. 130-154, 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," *Int'l J. Computer Vision,* vol. 15, pp. 189-224, 1995.

[6] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker, "Shock Graphs and Shape Matching," *Int'l J. Computer Vision,* vol. 30, pp. 1-24, 1999.

[7] P. Dimitrov, C. Phillips, and K. Siddiqi, "Robust and Efficient Skeletal Graphs," *Proc. Int'l Conf. Computer Vision and Pattern Recognition,* 2000.

[8] K. Siddiqi, S. Bouix, A.R. Tannenbaum, and S.W. Zucker, "Hamilton-Jacobi Skeletons," *Int'l J. Computer Vision,* vol. 48, no. 3, pp. 215-231, 2002.

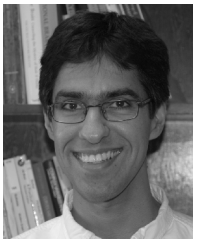
[9] D. Macrini, "Indexing and Matching for View-Based 3-D Object Recognition Using Shock Graphs," master's thesis, Dept. of Computer Science, University of Toronto, 2003.
[10] M. Turk and A. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience,* vol. 3, no. 1, pp. 71-86, 1991.
[11] H. Murase and S. Nayar, "Visual Learning and Recognition of 3-D Objects from Appearance," *Int'l J. Computer Vision,* vol. 14, pp. 5-24, 1995.
[12] S. Sclaroff and A. Pentland, "Modal Matching for Correspondence and Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 6, pp. 545-561, June 1995.
[13] B. Luo, R.C. Wilson, and E. Hancock, "Graph Spectral Approach for Learning View Structure," *Proc. 16th Int'l Conf. Pattern Recognition,* vol. 3, pp. 785-788, Aug. 2002.
[14] B. Luo, R.C. Wilson, and E.R. Hancock, "Spectral Embedding of Graphs," *Pattern Recognition,* vol. 36, pp. 2213-2223, 2003.
[15] J. Beis and D. Lowe, "Shape Indexing Using Approximate Nearest-Neighbor Search in Highdimensional Spaces," *Proc. Int'l Conf. Computer Vision and Pattern Recognition,* pp. 1000-1006, 1997.
[16] D. Lowe, "Object Recognition from Local Scale-Invariant Features," *Proc. Int'l Conf. Computer Vision,* pp. 1150-1157, 1999.
[17] Y. Lamdan, J. Schwartz, and H. Wolfson, "On Recognition of 3-D Objects from 2-D Images," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 1407-1413, April 1988.
[18] D. Forsyth, J. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell, "Invariant Descriptors for 3D Object Recognition and Pose," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 10, pp. 971-992, Oct. 1991.
[19] D. Clemens and D. Jacobs, "Space and Time Bounds on Indexing 3-D Models from 2-D Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 10, pp. 1007-1017, Oct. 1991.
[20] P.J. Flynn and A.K. Jain, "3D Object Recognition Using Invariant Feature Indexing of Interpretation Tables," *CVGIP: Image Understanding,* vol. 55, no. 2, pp. 119-129, 1992.
[21] B. Messmer and H. Bunke, "Subgraph Isomorphism in Polynomial Time," Technical Report IAM- 95-003, Univ. of Bern, 1995.
[22] K. Sengupta and K. Boyer, "Modelbase Partitioning Using Property Matrix Spectra," *Computer Vision and Image Understanding,* vol. 70, no. 2, pp. 177-196, 1998.
[23] L. Shapiro and R. Haralick, "Organization of Relational Models for Scene Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 4, no. 6, pp. 595-602, Nov. 1982.
[24] K. Sengupta and K. Boyer, "Organizing Large Structural Modelbases," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 4, pp. 321-332, 1995.
[25] T.B. Sebastian, P.N. Klein, and B.B. Kimia, "Shock-Based Indexing into Large Shape Databases," *Proc. European Conf. Computer Vision,* pp. 731-746, 2002.
[26] C. Irniger and H. Bunke, "Graph Database Filtering Using Decision Trees," *Proc. 17th Int'l Conf. Pattern Recognition,* vol. 3, pp. 383-388, Aug. 2004.
[27] A. Sanfeliu and K.S. Fu, "A Distance Measure Between Attributed Relational Graphs for Pattern Recognition," *IEEE Trans. Systems, Man, and Cybernetics,* vol. 13, pp. 353-362, May 1983.
[28] L.G. Shapiro and R.M. Haralick, "A Metric for Comparing Relational Descriptions," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 7, no. 1, pp. 90-94, Jan. 1985.
[29] L.G. Shapiro and R.M. Haralick, "Structural Descriptions and Inexact Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 3, pp. 504-519, 1981.
[30] A.K.C. Wong and M. You, "Entropy and Distance of Random Graphs with Application to Structural Pattern Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 7, no. 9, pp. 599-609, Sept. 1985.
[31] A. Wong, S. Lu, and M. Rioux, "Recognition and Shape Synthesis of 3D Objects Based on Attributed Hypergraphs," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 11, pp. 279-290, 1989.
[32] K. Boyer and A. Kak, "Structural Stereopsis for 3-D Vision," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, no. 2, pp. 144-166, Mar. 1988.
[33] W. Kim and A.C. Kak, "3D Object Recognition Using Bipartite Matching Embedded in Discrete Relaxation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 3, pp. 224-251, Mar. 1991.
[34] B.T. Messmer and H. Bunke, "A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, pp. 493-504, May 1998.
[35] W.J. Christmas, J. Kittler, and M. Petrou, "Structural Matching in Computer Vision Using Probabilistic Relaxation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 17, no. 8, pp. 749-764, Aug. 1995.
[36] M.A. Eshera and K.S. Fu, "A Graph Distance Measure for Image Analysis," *IEEE Trans. Systems, Man, & Cybernatics,* vol. 14, pp. 398-408, May 1984.
[37] M. Pelillo, K. Siddiqi, and S. Zucker, "Matching Hierarchical Structures Using Association Graphs," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 11, pp. 1105-1120, Nov. 1999.
[38] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for Graph Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 4, pp. 377-388, Apr. 1996.
[39] S. Zhu and A.L. Yuille, "Forms: A Flexible Object Recognition and Modelling System," *Int'l J. Computer Vision,* vol. 20, no. 3, pp. 187-212, 1996.
[40] A. Cross and E. Hancock, "Graph Matching with a Dual-Step EM Algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 11, pp. 1236-1253, Nov. 1998.
[41] B. Huet and E. Hancock, "Relational Object Recognition from Large Structural Libraries," *Pattern Recognition,* vol. 35, pp. 1895-1915, 2002.
[42] M. Costa and L. Shapiro, "Relational Indexing," *Proc. Int'l Workshop Structural and Syntactic Pattern Recognition,* pp. 130-139, 1996.
[43] H. Sossa and R. Horaud, "Model Indexing: The Graph-Hashing Approach," *Proc. Int'l Conf. Computer Vision and Pattern Recognition,* pp. 811-814, 1992.
[44] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes, "Exploiting Local Similarity for Efficient Indexing of Paths in Graph Structured Data," *Proc. Int'l Conf. Data Eng.,* 2002.
[45] R. Goldman and J. Widom, "Dataguides: Enabling Query Formulation and Optimization in Semistructured Databases," *Proc. 23rd Int'l Conf. Very Large Data Bases,* pp. 436-445, 1997.
[46] M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim, "Xtract: A System for Extracting Document Type Descriptors from XML Documents," *Proc. ACM SIGMOD Conf.,* pp. 165-176, 2000.
[47] D. Lee and M. Yannakakis, "Online Minimization of Transition Systems," *Proc. 24th ACM Symp. Theory of Computing,* pp. 264-274, 1992.
[48] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu, "A Query Language and Optimization Techniques for Unstructured Data," *Proc. ACM SIGMOD Conf. Management of Data,* pp. 505-516, 1996.
[49] P. Drineas and R. Kannan, "Pass Efficient Algorithms for Approximating Large Matrices," *Proc. ACM-SIAM Symp. Discrete Algorithms,* pp. 223-232, 2003.
[50] P. Drineas, R. Kannan, and M.W. Mahoney, "Fast Monte Carlo Algorithms for Matrices II: Computing Low-Rank Approximations to a Matrix," http://cs-www.cs.yale.edu/homes/mmahoney/matrix/, May 2003.
[51] S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman, "Indexing by Latent Semantic Analysis," *J. Am. Soc. of Information Science,* vol. 41, no. 6, pp. 391-407, 1990.
[52] C.H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala, "Latent Semantic Indexing: A Probabilistic Analysis," *Proc. ACM-SIAM Symp. Discrete Algorithms,* pp. 159-168, 1998.
[53] J. Wilkinson, *The Algebraic Eigenvalue Problem.* Oxford, England: Clarenson Press, 1965.
[54] G. Stewart and J. Sun, *Matrix Perturbation Theory.* San Diego: Academic Press, 1990.
[55] A. Neumaier, "Second Largest Eigenvalue of a Tree," *Linear Algebra and Its Applications,* vol. 46, pp. 9-25, 1982.
[56] L. Lovász and J. Pelicán, "On the Eigenvalues of a Tree," *Periodica Math. Hungarica,* vol. 3, pp. 1082-1096, 1970.
[57] D. Cvetković, P. Rowlinson, and S. Simić, *Eigenspaces of Graphs.* Cambridge Univ. Press, 1997.
[58] Y. Lamdan, J. Schwartz, and H. Wolfson, "Affine Invariant Model-Based Object Recognition," *IEEE Trans. Robotics and Automation,* vol. 6, no. 5, pp. 578-589, Oct. 1990.
[59] P. Flynn and A. Jain, "3D Object Recognition Using Invariant Feature Indexing of Interpretation Tables," *CVGIP: Image Understanding,* vol. 55, no. 2, pp. 119-129, Mar., 1992.
[60] K. Sengupta and K. Boyer, "Using Geometric Hashing with Information Theoretic Clustering for Fast Recognition from a Large CAD Modelbase," *Proc. IEEE Int'l Symp. Computer Vision,* pp. 151-156, Nov. 1995.

[61] K. Boyer and K. Sengupta, "Object Recognition Using Large Structural Modelbases," *Proc Third Int'l Workshop on Visual Form,* May 1997.
[62] C. Böhm, S. Berchtold, and D.A. Keim, "Searching in High-Dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases," *ACM Computing Surveys (CSUR),* vol. 33, no. 3, pp. 322-373, 2001.
[63] N. Katayama and S. Satoh, "The SR-Tree: An Index Structure for High-Dimensional Nearest Neighbor Queries," *Proc. ACM SIGMOD Int'l Conf. Multimedia and Design,* pp. 369-380, May 1997.
[64] D. Cvetković and P. Rowlinson, "The Largest Eigenvalue of a Graph: A Survey," *Linear and Multilinear Algebra,* vol. 28, pp. 3-33, 1990.
[65] N. Biggs, *Algebraic Graph Theory.* Cambridge Univ. Press, 1993.
[66] A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker, "Indexing Using a Spectral Encoding of Topological Structure," *Proc. Int'l Conf. Computer Vision and Pattern Recognition,* pp. 491-497, June 1999.

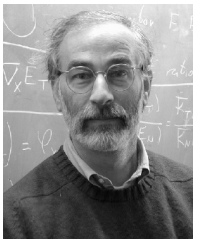
Ali Shokoufandeh received the BSc degree in computer science from the University of Tehran and the MSc and PhD degrees in computer science from Rutgers in 1996 and 1999, respectively. He is an assistant professor of computer science at Drexel University. His research focuses on computer vision, pattern recognition, extremal graph theory, and combinatorial optimization. He was the recipient of the Center for Discrete Mathematics and Theoretical Computer Science (a National Science Foundation and Technology Center) Graduate Awards in 1998 and 1999. He is a member of the IEEE and SIAM.

Diego Macrini received the engineering degree in software engineering from the University of Belgrano in 1998 and the MS degree in computer science from the University of Toronto in 2003, where he is currently pursuing the PhD degree in the same area. His major field of interest is computer vision with an emphasis on shape representation, object recognition, and visual motion analysis. He is a student member of the IEEE Computer Society.

Sven Dickinson received the BASc. degree in systems design engineering from the University of Waterloo in 1983 and the MS and PhD degrees in computer science from the University of Maryland in 1988 and 1991, respectively. He is currently an associate professor of computer science at the University of Toronto. From 1995-2000, he was an assistant professor of computer science at Rutgers University, where he also held a joint appointment in the Rutgers Center for Cognitive Science (RuCCS). From 1994-1995, he was a research assistant professor in the Rutgers Center for Cognitive Science and, from 1991-1994, he was a research associate at the Artificial Intelligence Laboratory, University of Toronto. He has held affiliations with the MIT Media Laboratory (visiting scientist, 1992-1994), the University of Toronto (visiting assistant professor, 1994-1997), and the Computer Vision Laboratory of the Center for Automation Research at the University of Maryland (assistant research scientist, 1993-1994, and visiting assistant professor, 1994-1997). His major field of interest is computer vision with an emphasis on shape representation, object recognition, and mobile robot navigation. In 1996, he received the US National Science Foundation CAREER award for his work in generic object recognition and, in 2002, he received the Government of Ontario Premiere's Research Excellence Award (PREA), also for his work in generic object recognition. From 1998-2002, he served as associate editor of the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, during which he also coedited a special issue on graph algorithms and computer vision, which appeared in 2001. He currently serves as associate editor for the journal *Pattern Recognition Letters*. He is a member of the IEEE and the IEEE Computer Society.



Kaleem Siddiqi received the BS degree from Lafayette College in 1988 and the MS and PhD degrees from Brown University in 1990 and 1995, all in the field of electrical engineering. He is an associate professor and William Dawson Scholar in the School of Computer Science at McGill University. He is also a member of McGill's Center for Intelligent Machines. Before moving to McGill in 1998, he was a postdoctoral associate in the Department of Computer Science at Yale University (96-98) and held a visiting position in the Department of Electrical Engineering at McGill University (95-96). His research interests are in the areas of computer vision, image analysis, and human psychophysics. He is a member of Phi Beta Kappa, Tau Beta Pi, and Eta Kappa Nu.



Steven W. Zucker obtained his education at Carnegie-Mellon University in Pittsburgh and at Drexel University in Philadelphia. He is the David and Lucile Packard Professor at Yale University. He is a member of the Departments of Computer Science and Biomedical Engineering and is the director of the program in applied mathematics. Before moving to Yale in 1996, he was a professor of electrical engineering at McGill University and the director of the program in artificial intelligence and robotics at the Canadian Institute for Advanced Research. He was elected a fellow of the Canadian Institute for Advanced Research, a fellow of the IEEE, and (by)fellow of Churchill College, Cambridge. He was a postdoctoral research fellow in computer science at the University of Maryland, College Park, a professor inviteé at the Institut National de Recherche en Informatique et en Automatique, Sophia-Antipolis, France, a visiting professor of computer science at Tel Aviv University, and an SERC fellow of the Isaac Newton Institute for Mathematical Sciences, University of Cambridge. He is also an external faculty member of the Sloan-Swartz Center for Theoretical Neurobiology at the University of California, San Francisco.