Indexing through laplacian spectra

M. Fatih Demirci [a,b,*], Reinier H. van Leuken [a], Remco C. Veltkamp [a]

[a] Institute for Information and Computing Sciences, Utrecht University, 3584CH Utrecht, The Netherlands
[b] Department of Computer Engineering, TOBB University of Economics and Technology, Ankara, Turkey

Abstract

With ever growing databases containing multimedia data, indexing has become a necessity to avoid a linear search. We propose a novel technique for indexing multimedia databases in which entries can be represented as graph structures. In our method, the topological structure of a graph as well as that of its subgraphs are represented as vectors whose components correspond to the sorted laplacian eigenvalues of the graph or subgraphs. Given the laplacian spectrum of graph G, we draw from recently developed techniques in the field of spectral integral variation to generate the laplacian spectrum of graph $G + e$ without computing its eigendecomposition, where $G + e$ is a graph obtained by adding edge e to graph G. This process improves the performance of the system for generating the subgraph signatures for 1.8% and 6.5% in datasets of size 420 and 1400, respectively. By doing a nearest neighbor search around the query spectra, similar but not necessarily isomorphic graphs are retrieved. Given a query graph, a voting schema ranks database graphs into an indexing hypothesis to which a final matching process can be applied. The novelties of the proposed method come from the powerful representation of the graph topology and successfully adopting the concept of spectral integral variation in an indexing algorithm. To examine the fitness of the new indexing framework, we have performed a number of experiments using an extensive set of recognition trials in the domain of 2D and 3D object recognition. The experiments, including a comparison with a competing indexing method using two different graph-based object representations, demonstrate both the robustness and efficacy of the overall approach.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Spectral graph theory; Indexing; Laplacian spectrum; Spectral integral variation; Information retrieval

1. Introduction

Shape matching is one of the fundamental problems in computer vision. In a typical matching problem the objective is to compute an overall similarity value between an unknown shape (query) and a model, and to find the correspondences between their feature sets. The similarity value between two shapes can be used for shape recognition by using stored exemplars for different shape classes as models. A linear search of a database, i.e., computing the similarity between the query and each database entry and selecting the closest one, is inefficient for large database systems. Therefore, an effective and efficient indexing mechanism is essential to select a small collection of candidates to which the actual matching process is applied. Criminology, medicine, trademark retrieval, and content-based image retrieval on the web are only a few examples of applications which are likely to contain large collections.

For recognition purposes, it is very common to represent object views by graphs whose nodes correspond to image features and whose edges indicate relations between these features, e.g., [46,28,51,26,40]. Both nodes and edges may be labeled by attributes. These graph representations express many significant object properties such as geometric or hierarchical structures. Such representations, however, have drawbacks: matching two graphs is a difficult problem.

Graph matching problems are often formulated as largest isomorphic subgraph problems, for which a rich body of research exists in the literature, such as pattern recognition [30,29], chemical structures [37], or computer vision

* Corresponding author. Address: Department of Computer Engineering, TOBB University of Economics and Technology, Ankara, Turkey. Fax: +90 312 292 4180.
   E-mail addresses: mfdemirci@etu.edu.tr (M.F. Demirci), renier@cs.uu.nl (R.H. van Leuken), remco.veltkamp@cs.uu.nl (R.C. Veltkamp).

[52,23]. This problem has been studied for both theoretical and practical interests. While it is an open question whether the detection of graph isomorphism can be solved in polynomial-time, the problem of subgraph isomorphism is known to be NP-complete [15]. Although (sub)graph isomorphism detection is computationally expensive, some graph isomorphism detection algorithms with only polynomial-time complexity have been developed for specific graph classes, e.g., planar graphs [22]. It is also possible to derive such polynomial-time complexity algorithms for graphs with certain restrictions [21].

When working with graph structures, indexing is formulated as the problem of efficiently selecting a small set of database graphs, which share a subgraph with the query. Several frameworks have been proposed to use (sub)graph isomorphism algorithms with indexing methods. Shapiro and Haralick [42] proposed a method to organize similar graphs in clusters where each cluster is indexed by a representative graph. Sossa and Horaud [48] used the coefficients of the d_2-polynomial of the laplacian matrix of a graph to index into graph datasets. These coefficients, however, are only unique for small graphs with less than 12 vertices.

One important indexing method is a decision tree approach. Here, the goal is to hierarchically partition the database so that the query is first matched to the root. Depending on the result of this match, the query is then matched to either the right or the left child of the root. This process is repeated recursively until a match is found at an internal node (or leaf), or it exits with a failure indicating no database graphs are isomorphic to the query. Messmer and Bunke [32] use this approach to organize the set of all permutations of the adjacency matrix of database graphs in a decision tree. At run time, the (sub)graph isomorphisms from the query to the database graphs are found by a decision tree traversal. A significant drawback of this method is its space requirement. All permutations of the adjacency matrix have to be encoded in decision trees, whose sizes grow exponentially with the size of the database graph. A set of pruning techniques is discussed to cut down the space complexity.

So far, we have only considered the problem of (sub)-graph isomorphism. However, due to noise, occlusion, or segmentation errors, (sub)graph isomorphism may not exist between the query and the database. Furthermore, only a certain degree of similarity between two graphs may be present. The indexing problem, therefore, is reformulated as efficiently retrieving database graphs whose (sub)structure is similar to the query. Although considerable research has been devoted to the problem of inexact (or error-tolerant) graph matching, rather less attention has been paid to this type of indexing based on graph structures.

Costa and Shapiro [10] present a graph-based indexing method, where small relational subgraphs are used to efficiently retrieve similar graphs from a large database. An integrated framework related to the approach reported in this paper is that of Shokoufandeh et al. [44]. This framework is designed especially for tree structures in which the sum of the largest eigenvalues of the adjacency matrix for each subtree of the root form the components of its δ-dimensional vector, where δ is the root degree. To account for occlusion and local deformation, these vectors are also computed for the root of each subtree. At indexing time, each non-leaf node of the query is represented as such a vector, and a nearest neighbor search is performed for each vector. Although effective, by summing up the largest eigenvalues one loses uniqueness, resulting in less representative graphs in the vector space. In addition, it is not clear how this approach can be extended to general graph structures.

One of the primary aspects of graph theory is to derive the principal properties and structure of graphs from their graph spectra. In the computer vision and pattern recognition communities, eigenvalue-based frameworks have been applied to various problems including shape description and indexing. Sengupta and Boyer [41] used eigenvalue-based feature representation of CAD models to capture their gross characteristics. This representation is used to partition the database into structurally homogeneous groups. Shapiro and Brady [43] used eigenvectors of proximity graphs to compute the feature correspondences. Turk and Pentland [50] proposed an eigenface approach in which images were represented as linear combinations of a small set of images computed from a large database. The algorithm was applied to face recognition. Sclaroff and Pentland [39] computed the eigenmodels of 2D regions and used the model coefficients in a linear search of 2D shapes. However, since the characterizations in this approach are global, it is not clear how this method performs for retrieving models with local similarities. Some other eigenvalue-based methods consist of applications such as edge detection [49], motion estimation [17], and 3D object representation as 2D images [8].

1.1. Our contributions

In this paper, we propose a novel approach to the graph-based indexing problem. Instead of using the adjacency matrix for graph characterization as done by some earlier work, we characterize our graphs based on the laplacian spectrum, which is more natural, more important, and more informative about the input graphs [33]. The definition of a laplacian matrix along with other graph-theoretical concepts used in this paper is given in the next section. Given a graph $G = (V, E)$, the sorted eigenvalues of its laplacian matrix become the components of its signature, an O($|V|$)-dimensional vector. Since the laplacian spectrum is used as a graph signature without an approximation such as considering only largest eigenvalues, a high level of uniqueness is maintained. We will discuss techniques to reduce the cost for computing such a signature in the framework.

Having established the signatures, the indexing now amounts to a nearest neighbor search in a model database. For a query graph and a large graph dataset, we can, therefore, formulate the indexing problem as that of fast selection of candidate graphs whose signatures are close to

the query signature. This formulation alone cannot support occlusion or segmentation errors as two graphs may share similar structures up to only some level. To perform indexing locally and thus to encode the topology of subgraphs in the framework, we adopt a technique analogous to that used in the decision tree approach [32]. Given the laplacian spectrum of a principal submatrix B of matrix A, we draw on an important theorem from spectral graph theory to show that our graph characterization can be used to retrieve similar graphs or subgraphs from large database systems through a nearest neighbor search.

The local indexing method used in the framework is effective, but the signature of each subgraph of a graph is computed individually from its eigendecomposition. Given the laplacian spectrum of graph G, we use recently developed techniques in the domain of spectral integral variation to generate the laplacian spectrum of graph $G + e$ without the need for computing its eigendecomposition. To our knowledge, the proposed framework is the first framework that uses spectral integral variation in an indexing algorithm.

Our approach is of particular interest to applications where the size of the database is large, but the size of each graph is relatively small (less than around 24 vertices). Although our method has a similar start-up to [32], it differs by a number of important factors. First, we use the laplacian rather than the adjacency matrix for graph characterization. Second, since the permutation-similar matrices result in the same set of sorted eigenvalues, we consider such laplacian matrices once, avoiding the need for a high-load compilation process described for this type of adjacency matrices in [32]. Third, probably the most important difference is that our method is intended for retrieving similar database graphs, requiring no significant graph isomorphism, although the framework can easily be modified to isomorphism detection.

The rest of the paper is organized as follows. Following a review of graph-theoretical concepts and preliminaries in Section 2, we describe our indexing mechanism and its complexity in Section 3. To avoid computing the signature of each subgraph individually from its eigendecomposition, we adopt the concept of spectral integral variation in Section 4. After evaluating the framework on two different recognition domains and performing a comparison of our approach with a competing indexing algorithm in Section 5, we end the paper with conclusions and our future work in Section 6.

2. Preliminaries

Before describing our framework, some definitions are in order. A graph G is a pair (V, E), where V is a finite set of vertices and E is a set of connections (edges) between the vertices. The size of a graph is defined as the number of vertices. An edge $e = (u, v)$ connects two vertices where $u, v \in V$. A graph $G = (V, E)$ is called edge-weighted if each edge $e \in E$ has a weight $w(e) \in \mathbb{R}$. Unweighted graphs are a special case of weighted graphs, where each of the edges has weight 1. A graph is *simple* if it does not contain self-loops or multiple edges and thus its edge set consists of distinct pairs. All graphs considered in this paper are simple. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are *isomorphic*, if there is a bijection $f : V_1 \rightarrow V_2$ such that for any vertex pair u and $v \in V_1$, $(u, v) \in E_1$ iff $(f(u), f(v)) \in E_2$.

The *adjacency matrix* A of a graph $G = (V, E)$ is a $|V| \times |V|$ matrix whose element with row index u and column index v is

$$A(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 0 & \text{otherwise.} \end{cases}$$

Let $D(G)$ be the diagonal matrix of vertex degrees with elements of $D(u, u) = \sum_{v \in V} A(u, v)$. The matrix $L(G) = D(G) - A(G)$ is called the *laplacian matrix* of G.[1] The laplacian matrix is a positive semidefinite and symmetric matrix with at least one zero eigenvalue. The multiplicity of zero as an eigenvalue of $L(G)$ is equal to the number of connected components in the graph. This implies that the second smallest eigenvalue known as algebraic connectivity is positive if and only if G is connected. There exist many important theorems about laplacian matrices and in many problems in physics and chemistry they play a central role. The reader is referred to [34,35,31,33] for surveys on this topic.

The spectrum of a graph's laplacian matrix is obtained from its eigendecomposition. Specifically, the eigendecomposition of a laplacian matrix is $L(G) = P\Lambda P^T$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{|V|})$ is the diagonal matrix with the eigenvalues in increasing order and $P = (p_1 | p_2 | \ldots | p_{|V|})$ is the matrix with the ordered eigenvectors as columns. The laplacian spectrum is the set of eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_{|V|}\}$. The spectrum is permutation invariant, i.e., two isomorphic graphs have the same set of sorted eigenvalues. However, the converse is not true, as two graphs that have the same spectra are not necessarily isomorphic.

Two graphs are called *cospectral* (or, isospectral) if they have the same eigenvalues. Previously, Godsil and McKay [16] and more recently Haemers and Spence [19] have shown that the laplacian matrix has more representational power than the adjacency matrix, in terms of resulting in fewer cospectral graphs. According to the results given in [19], of more than a billion graphs with 11 vertices characterized by the adjacency matrix, approximately 21% is cospectral, while this fraction is only 9% for the laplacian matrix. As specific graph classes, trees were also investigated for cospectrality by Zhu and Wilson [55]. The authors report that out of more than two million trees with 21 vertices, 21.3% of them do not have a unique adjacency spectrum. With the laplacian spectrum, this ratio decreases to 0.05% only.

We have two reasons for constructing the graph characterizations using the laplacian spectrum. The first one

---

[1] The laplacian matrix is also called Kirchhoff matrix or the matrix of admittance in the literature.

comes from the cospectrality studies given above. Overall, these studies show that the laplacian spectrum is more representative and more informative than the adjacency spectrum. The second, probably the most important reason stems from the fact that the laplacian spectrum is closely related to important graph invariants. Thus, it encodes important information about graph structures such as the diameter, mean distance, algebraic connectivity, degree distribution of graph nodes, graph size, independence number, isoperimetric number, genus, and expanding properties [34,35,14,36,31,33]. Using these relations, we compute the structural similarity between two graphs based on the closeness of their laplacian spectra as described in the next section.

3. Encoding the graph structure for indexing

3.1. Indexing formulation

Given a query and a large database, our objective in an indexing mechanism is to efficiently retrieve a small set of candidates, which share topological similarity with the query or one of its subgraphs. We assume that the database graphs are known in advance and the query graph is given at run time only. If the graph has a rich structure in terms of diameter and branching factor, a novel encoding of its topology can be used as an index into a large graph database. In our framework, we encode the topology of a graph through the laplacian spectrum. Specifically, sorted eigenvalues of the laplacian matrix are assigned to the graph as its signature. To compute the similarity between two graphs, we compute the Euclidean distance between their signatures, which is inversely proportional to the structural similarity of the graphs. For a given query, retrieving similar graphs can be reduced to a nearest neighbor search among a set of points.

Unfortunately, the above formulation cannot support occlusion or segmentation errors: two graphs may share similar structures up to only some level. Although adding or removing graph structure changes the laplacian spectrum, the spectrum of the subgraphs that survive such alteration will not be affected. Therefore, our indexing mechanism cannot depend on the signature of the whole graph only. Instead, we will combine the signatures of the subgraphs with our indexing mechanism.

3.2. Local indexing

Let $G = (V, E)$ be a graph and let G' be a graph obtained from G by adding a new edge e' such that $e' \notin E$. Then the following theorem, known as the interlacing theorem, relates the laplacian spectrum of both graphs.[2]

Theorem 1. *The eigenvalues of G and G' interlace*:

$$0 = \lambda_1(G) = \lambda_1(G') \leqslant \lambda_2(G) \leqslant \lambda_2(G') \leqslant \ldots \leqslant \lambda_n(G) \leqslant \lambda_n(G').$$

In addition, it is known that $\sum_{i=1}^{n}(\lambda_i(G') - \lambda_i(G)) = 2$ [1]. Therefore, at least one inequality is strict. Overall this theorem implies the following. Assume that we are given a pair of isomorphic graphs g_1 and g_2. If we construct G_1 and G_2 out of g_1 and g_2 by adding different edges to each of them, one at a time, the laplacian spectra of G_1 and G_2 become proportionally less similar. As a result, the similarity between the signatures of G_1 and G_2 may not reflect the similarity between the signatures of their subgraphs g_1 and g_2. Therefore, constructing the indexing mechanism based on graph signatures is too weak. An ideal indexing framework should, in fact, select candidate database elements based on both local and global similarities. To account for local as well as global information in our framework, we will adopt the following method analogous to that used in the decision tree approach [32].

For a given database graph $G = (V, E)$, rather than storing its signature in the system only, we compute the signatures of each subgraph of G in our algorithm. In this process, we gradually increase the size of the subgraphs. Since the sorted eigenvalues are invariant under consistent reorderings of the graph's vertices, it is sufficient to represent the spectrum of permutation-similar matrices once. This property avoids the need for a high-load compilation process described for adjacency matrices in the decision tree approach.

Associated with each signature in the system is a pointer to the corresponding graph or subgraph in the database. In case a signature represents more than one graph, one pointer is created from the signature to each graph. At runtime, we first generate the signature of each subgraph of the query. Given a query signature s_q, we then retrieve its nearest neighbors of the same size from the database through a nearest neighbor search (see Fig. 1). Each neighbor of s_q retrieved from the database gets a vote whose value is inversely proportional to the distance from s_q. Thus, as a result, each signature of the query generates a set of votes. Moreover, we weigh the votes according to the size of the subgraphs corresponding to the signatures, i.e., the bigger the size, the more weight the vote receives. To collect these votes, we will use the following strategy.

Let $S_q = \{s_{q1}, \ldots, s_{qm}\}$ be the set of query signatures. For a particular signature $s_{qi} \in S_q$, let $N_{s_{qi}} = \{n_1, \ldots, n_k\}$ be the set of elements returned by the nearest neighbor search and let $|s_{qi}|$ denote the size of its corresponding subgraph. ($|s_{qi}| = |n_j|$ for $j = 1, \ldots, k$). We compute the weight of the vote between s_{qi} and a signature s_{di} corresponding to a database (sub)graph as follows:

$$W_{s_{qi}s_{di}} = \begin{cases} \frac{|s_{qi}|}{1+||s_{qi}-s_{di}||_2} & \text{if } s_{di} \in N_{s_{qi}}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

---

[2] This theorem is obtained by Courant-Weyl (see Theorem 2.1 in [11]). The reader may also refer to [18].
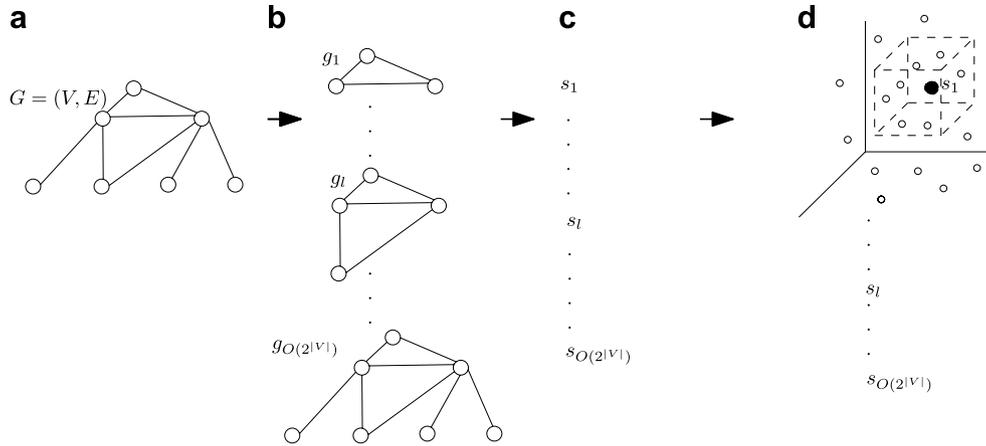
Fig. 1. Retrieving similar graphs. For graphs given in (a), its subgraphs are constructed in (b). A signature is computed for each subgraph in (c). Given a signature, retrieving its similar graphs from a large database is formulated as a nearest neighbor search as shown in (d).

Given a query $G_q = (V_q, E_q)$ and a database graph $G_d = (V_d, E_d)$ of size $|V_q|$ and $|V_d|$, respectively, let S_q^k denote the set of signatures for subgraphs of size k of the query graph, and let S_d^k be this set for the database graph. For certain size k, we first determine the similarity between the sets S_q^k and S_d^k using the Hausdorff distance, which has been successfully used in several problems, e.g., [38,24,25]. One advantage of using this distance measure comes from the fact that the Hausdorff distance does not allow for comparing portions of the sets S_q^k and S_d^k, i.e., the similarity between each signature pair (s_{qi}, s_{di}), where $s_{qi} \in S_q^k$ and $s_{di} \in S_d^k$ is taken into account. More specifically, the weight of the votes between signature sets S_q^k and S_d^k is computed as follows:

$$h(S_q^k, S_d^k) = \max_{s_{qi} \in S_q^k} \{ \min_{s_{di} \in S_d^k} \{ W_{s_{qi} s_{di}} \} \}. \tag{2}$$

However, since $h(S_q^k, S_d^k)$ is not symmetric, the average of $h(S_q^k, S_d^k)$ and $h(S_d^k, S_q^k)$ is taken:

$$H(S_q^k, S_d^k) = (h(S_q^k, S_d^k) + h(S_d^k, S_q^k))/2. \tag{3}$$

The total weight of the votes accounting for both local and global similarities is then computed as:

$$W_{S_q S_d} = \sum_{j=1}^{\min(|V_q|, |V_d|)} H(S_q^j, S_d^j). \tag{4}$$

After performing a nearest neighbor search around the query signatures, we compute the weights of the votes between the query and the database graphs having at least one signature as the nearest neighbor of the query. We then sort the database graphs based on these weights. In this process, we only add the sufficiently high-support database graphs to the indexing hypothesis. Since a small number of structurally different graphs may also share the same laplacian spectrum, each graph in the hypothesis should still be verified by some matching algorithm. Despite the fact that such graphs may exist in the indexing hypothesis, the number of them is very small. In addition, based on Theorem 1,

not only do isomorphic graphs share the same signature, non-isomorphic but similar graphs or subgraphs have close signatures in the vector space. The database, therefore, can be pruned without losing structurally similar graphs. The complexity of the algorithm is presented in the next section.

3.3. Complexity analysis

Let us first analyze the computational complexity of the signature generation for the database graphs, which is a preprocessing step performed offline. Let n_d denote the maximum number of vertices in a database graph. Given a single graph, the total number of its subgraphs of size t is $O\left(\binom{n_d}{t} \right)$. Assume that there are m graphs in the database, the total number of signatures generated by the framework is bounded by

$$m \times \sum_{t=0}^{n_d} \binom{n_d}{t} = O(m \times 2^{n_d}).$$

Notice, however that the actual number of subgraphs whose signatures are represented in the system is strictly less than $m \times \sum_{t=0}^{n_d} \binom{n_d}{t}$, since our signature is permutation invariant. In addition, subgraphs of size 2 and 3 are considered too small to represent a significant part of the original image for our experiments presented in Section 5. We generate signatures of subgraphs starting from size 4 in the framework.

On retrieval, we perform an approximate nearest neighbor search for each query signature, using a *balanced-box decomposition tree* (BBD-tree) as introduced by Arya et al. [2]. Given any positive real ϵ, a signature is an $(1 + \epsilon)$-approximate nearest neighbor of the query signature s_q if its distance from s_q is within a relative error bound of ϵ from the true nearest neighbor. More generally, for $1 \leqslant k \leqslant n$, a kth $(1 + \epsilon)$-approximate nearest neighbor of s_q is a signature whose relative error from the true kth

nearest neighbor is ϵ, where n is the database size. Given an integer $k \geqslant 1$, $(1+\epsilon)$-approximations to the k nearest neighbors of s_q can be found using a BBD-tree in $O(kd \log n)$ time, where d is the dimension of the search space, and n the number of signatures in this search space.

Thus, for a query subgraph of size t, the running time T_t of a k-nearest neighbor search using the BBD-tree is

$$T_t = O\left(kt \log \left(m \times \binom{n_d}{t} \right) \right).$$

For each complete query of size of n_q, the database is queried with its $O\left(\sum_{t=0}^{n_q} \binom{n_q}{t} \right)$ subgraph signatures. Therefore, the total running time for a complete query q is

$$T_{n_q} = O\left(\sum_{t=0}^{n_q} \binom{n_q}{t} T_t \right).$$

4. More efficient indexing through spectral integral variation

The local indexing procedure described above requires individual computation of the laplacian spectrum for each subgraph from its eigendecomposition. Although for database graphs known a priori this process is performed off-line, in applications where new database entries are being inserted frequently, this step plays an important role in the efficiency of the whole system. In this section, we draw on recent-developed techniques from the domain of spectral integral variation to avoid the individual computation of the laplacian spectrum for each subgraph from its eigendecomposition. Specifically, we will study the effect on the laplacian spectrum when an edge is added into graph $G = (V, E)$. Let $G + e$ be a graph obtained by adding an edge $e = (u, v)$ into G such that $\{u, v\} \in V$ and $e \notin E$. Our interest in this topic is motivated by its ability to identify the changed eigenvalues of graph G, and therefore to generate the laplacian spectrum of graph $G + e$ without the need for computing its eigendecomposition. Before we focus on this topic, let us first reconsider Theorem 1, which shows that when an edge is added into the graph, none of its laplacian eigenvalues can decrease, while the trace of the laplacian matrix increases by 2. This important observation implies that given the laplacian spectrum of G, one can estimate the ranges of eigenvalues for $G + e$. The concept of spectral integral variation, however, provides more information.

It is shown in [47] that if an edge is added to a graph and the laplacian spectrum changes by integer quantities, there can only be two possibilities: either one eigenvalue increases by 2 (and $n - 1$ eigenvalues remain fixed) or two eigenvalues increase by 1 (and $n - 2$ eigenvalues remain fixed). These two cases are called spectral integral variation in one place and spectral integral variation in two places, respectively. The following lemma characterizes these two possible situations.

Lemma 1. *Let $G = (V, E)$ be a graph with $|V| = n$ vertices and $\Gamma_{(G)} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ be its laplacian spectrum. The spectral integral variation of G by adding an edge $e \notin E$ occurs only in the following two ways:*

(1) *The spectral integral variation occurs in one place, and thus*

$$\Gamma_{(G+e)} = (\Gamma_{(G)} \setminus \lambda_k) \cup \{\lambda_k + 2\}, \quad \text{where } k \in \{1, 2, \ldots, n\}.$$

(2) *The spectral integral variation occurs in two places, and thus*

$$\Gamma_{(G+e)} = (\Gamma_{(G)} \setminus \{\lambda_k, \lambda_l\}) \cup \{\lambda_k + 1, \lambda_l + 1\},$$
$$\text{where } k, l \in \{1, 2, \ldots, n\} \text{ and } k \neq l.$$

The proof for this lemma is given by Yizheng [53]. Parts (a) and (b) of Fig. 2 show two graphs where adding an edge results in spectral integral variation in one and two places, respectively.

In our framework, we detect the changed eigenvalue(s) when spectral integral variation occurs in both one and two places using the theorems given below. This allows us to generate the laplacian spectrum of $G + e$ given that of G in these cases. In a somewhat related direction, there exists some work on characterizing graphs stating that when an edge is added, (one of) the changed eigenvalue is the algebraic connectivity [27,5]. Recall that the algebraic connectivity of a graph is defined as the second smallest laplacian eigenvalue.

Let $G = (V, E)$ be a graph with $|V| = n$ vertices. For $u \in V$, define $N(u) = \{v \in V : (u, v) \in E\}$. Assume that $e = (u, v)$ is added to $G = (V, E)$ such that $e \notin E$. The following theorems characterize and identify the changed
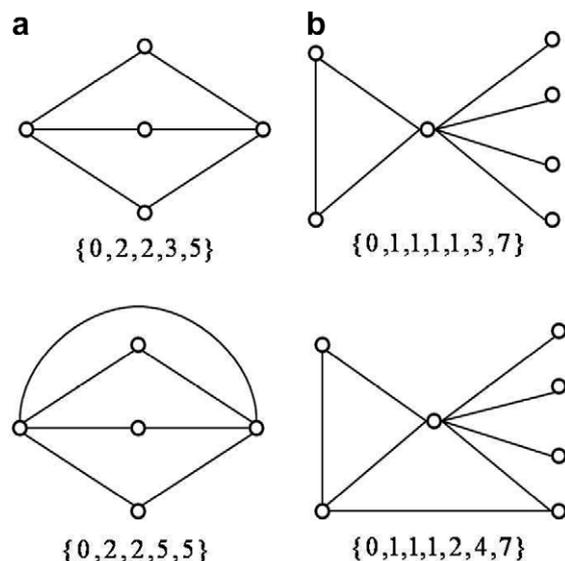

Fig. 2. Spectral integral variation in one and two places are shown in (a) and (b), respectively. Bottom graphs are formed by adding one edge to the graphs shown at the top. The laplacian spectrum is written below each graph. Observe that while only one eigenvalue increases by 2 in (a), two eigenvalues increase by 1 in (b).

laplacian eigenvalue(s) when spectral integral variation occurs in one and two places, respectively. Theorem 2 appears in [47], while Theorem 3 appears in [27].

Theorem 2. *$N(u) = N(v)$ if and only if the spectrum of $L(G)$ overlaps the spectrum of $L(G + e)$ in $n - 1$ places. Moreover, the laplacian eigenvalue of G that increases by 2 is given by the degree of vertex u (or, that of vertex v) in this case.*

For the following theorem, suppose that the degrees of vertices u and v are shown by d_u and d_v, respectively, and let t denote the number of vertices that are adjacent to both vertex u and vertex v. Without loss of generality, suppose also that $d_u \geqslant d_v$. Furthermore, let $1_x, 0_x$ denote the $x \times 1$ matrices whose entries are all $1, 0$, respectively, and let $1_x^t, 0_x^t$ denote their transposes.

Theorem 3. *Let laplacian matrix L of graph G be given by*

$$L = \begin{bmatrix} d_u & 0 & -1_x^t & 0_x^t & -1_x^t & 0_x^t \\ 0 & d_v & 0_x^t & -1_x^t & -1_x^t & 0_x^t \\ -1_x & 0_x & L_{11} & L_{12} & L_{13} & L_{14} \\ 0_x & -1_x & L_{21} & L_{22} & L_{23} & L_{24} \\ -1_x & -1_x & L_{31} & L_{32} & L_{33} & L_{34} \\ 0_x & 0_x & L_{41} & L_{42} & L_{43} & L_{44} \end{bmatrix},$$

where the blocks $l_{11}, l_{33}, l_{33}, L_{44}$ are of sizes $d_u - t, d_v - 1, t$, and $n - 2 - d_u - d_v + t$, respectively. Spectral integral variation occurs in two places if and only if the following conditions hold:

$$L_{11}1_x - L_{12}1_x = (d_v + 1)1_x, \quad (5)$$

$$L_{21}1_x - L_{22}1_x = -(d_u + 1)1_x, \quad (6)$$

$$L_{31}1_x - L_{32}1_x = -(d_u - d_v)1_x, \quad (7)$$

$$L_{41}1_x - L_{42}1_x = 0. \quad (8)$$

In the case that conditions (5)–(8) are satisfied, then the two eigenvalues of L that increase by 1 are

$$\lambda_1 = \frac{d_u + d_v + 1 - \sqrt{(d_u + d_v + 1)2 - 4(d_u d_v + t)}}{2} \quad (9)$$

and

$$\lambda_2 = \frac{d_u + d_v + 1 + \sqrt{(d_u + d_v + 1)2 - 4(d_u d_v + t)}}{2}. \quad (10)$$

Given the laplacian spectrum for graph $G = (V, E)$, when an edge e is added to construct $G + e$, we first check if the laplacian spectrum of $G + e$ can be generated using the theorems given above. In case Theorem 2 or Theorem 3 holds, the laplacian eigenvalues of $G + e$ overlap those of G in $|V| - 1$ and $|V| - 2$ places, respectively. These theorems also enable us to find the changed eigenvalue(s). After presenting the concept of spectral integral variation, we now describe the signature generation process performed offline for each given database graph.

Suppose that the minimum number of edges in a subgraph for which we compute the signature is k. Given a database graph $G = (V, E)$, we first create its subgraph $\widehat{G}$

with $|V|$ vertices and k edges and compute its laplacian eigenvalues from its eigendecomposition. Since the second smallest laplacian eigenvalue is positive if and only if the graph is connected and the multiplicity of zero as a laplacian eigenvalue reflects the number of connected components, only the positive eigenvalues of $\widehat{G}$ are used as the signature. Next, when we add an edge to $\widehat{G}$, we check whether spectral integral variation occurs using Theorems 2 or 3 and if so, we generate the eigenvalues without computing the eigendecomposition. We then repeat this process and consider all subgraphs until the whole graph is constructed. At run time, we also apply the same procedure to construct the signatures for query graphs. The algorithm is shown in Fig. 3. Although the above formulation enables us to identify the changed laplacian eigenvalues when they are increased by integer quantities, our empirical results show that it speeds up the signature generation step for database with 1440 graphs known a priori by 6.5%. We will present our report on this part in Section 5.

Revisiting the key new features of our approach, the encoding of a graph's structure captures its local topology, thus allowing for its use in the case of occlusion and segmentation errors. Furthermore, the signature of a graph is invariant under the reorderings of its vertices. This, in turn, allows us to compare the signatures of a large number of graphs without solving the computationally expensive correspondence problem between their vertices. Since we generate signatures of each subgraph to account for the local topology, there is a high computational complexity associated with the generation of the signatures for the database graphs, as shown in the previous section. This complexity, however, can be reduced by considering subgraphs starting from some predefined size and in practice it is further lowered by employing the concept of spectral integral variation.

5. Experiments

To examine the fitness of the new indexing framework, we have performed a number of experiments using an extensive set of recognition trials in the domain of 2D

Input: $G = (V, E)$, $\hat{G} = (V, \hat{E})$, and $\Gamma_{(\hat{G})}$
Output: The laplacian spectrum $\Gamma_{(\hat{G})}$ for every subgraph $\hat{G}$ of G

Algorithm Generate Spectra $(G = (V, E), \hat{G} = (V, \hat{E}), \Gamma_{(\hat{G})})$
  if $\hat{E} == E$ **then** terminate
  for every subgraph $\hat{G} = (V, \hat{E})$ of $G = (V, E)$, where
    $\hat{E} = \hat{E} \cup \{e : e \in E, e \notin E'\}$
    if spectral integral variation occurs in one place (Theorem 2) **then**
      generate $\Gamma_{(\hat{G})}$ according to Theorem 2
    else if spectral integral variation occurs in two places (Theorem 3) **then**
      generate $\Gamma_{(\hat{G})}$ according to Theorem 3
    else compute $\Gamma_{(\hat{G})}$ from its eigendecomposition
  endfor
  call Generate Spectra $(G = (V, E), \hat{G} = (V, \hat{E}), \Gamma_{(\hat{G})})$
end

Fig. 3. Generating laplacian spectra through spectral integral variation.

and 3D object recognition. The experiments, including a comparison with a competing indexing method using two different graph-based object representations and the results for a set of occluded queries, are presented below. We first perform our experiments using silhouettes. For a given shape, its silhouette is represented by an undirected shock graph [46]. The graph is constructed from the discrete skeleton using the method described in [13]. To summarize this process, a shock point p on the discrete skeleton is labeled by a 3-dimensional vector $v(p) = (x, y, r)$, where (x, y) are the Euclidean coordinates and r is the radius of the maximal bitangent circle centered at the point. Each shock point becomes a node in the graph and edges connect nearby shock points. An illustration of this procedure is given in Fig. 4, where the left portion shows an input image taken from the database, while the right portion presents the constructed shock graph superimposed on top of the image.

We used the MPEG-7 dataset CE-Shape-1 part B for this representation type. The MPEG-7 database consists of 70 classes and 20 shapes per class. The top of Fig. 5 shows a few sample classes, while the bottom of the figure presents different shapes taken from a particular class.

We also conduct our experiments in the domain of 3D object recognition using Reeb graphs. These graph representations allow for topological properties to be represented in a coarse sense. Let $f : S \rightarrow \mathbf{R}$ be a real-valued function on surface S. The Reeb quotient space is defined by the equivalence relation $\sim$ given by: $(\alpha, f(\alpha)) \sim (\gamma, f(\gamma))$ for $\alpha, \gamma \in S$ iff $f(\alpha) = f(\gamma)$ and α, γ are in the same connected component of $f^{-1}(f(\alpha))$. This means two points $(\alpha, f(\alpha))$ and $(\gamma, f(\gamma))$ are represented as the same node in the Reeb graph if values of f are the same and they belong to the same connected component of the inverse image of $f(\alpha)$ (or, equivalently $f(\gamma)$). The Reeb quotient space is coded in a Reeb graph such that the vertices represent critical points of function f, while the edges show the connections between them. See [3,20,9,6,7] for details. The right of Fig. 6 shows a Reeb graph constructed for the image shown in the left.

The second database used in the experiments consists of Reeb graphs constructed for the McGill 3D Shape Benchmark [54]. The database consists of 420 objects classified in 19 classes. Fig. 7 shows representative views of objects from the database.
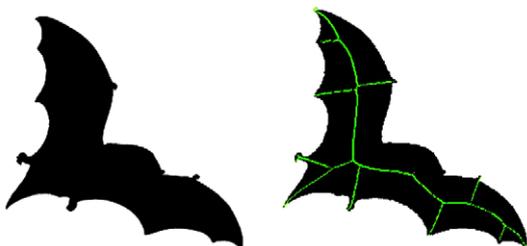

Fig. 5. Sample images of the MPEG-7 database are shown in the top row. The bottom row represents a set of different shapes of a particular class from the database.
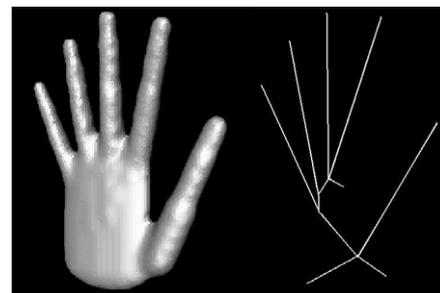

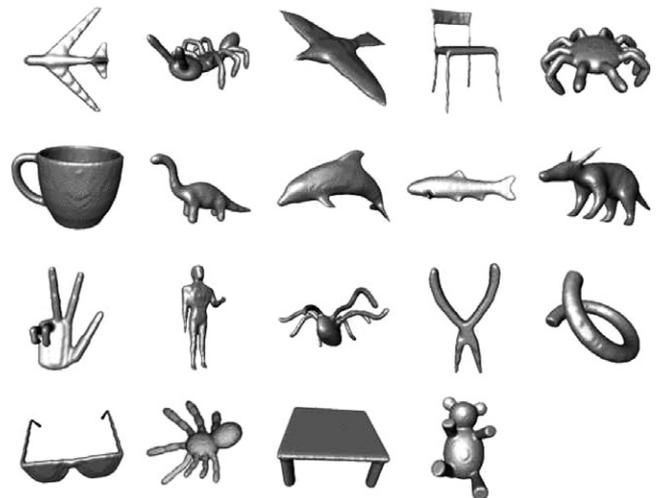Fig. 6. The Reeb graph constructed for the object on the left is shown on the right.

Fig. 7. Views of sample objects from the McGill 3D shape Benchmark.

We first represent each object in each database as a graph. Given a graph, we compute the signatures for each of its subgraphs and populate the resulting signatures in the vector space. In our experimental setup, we applied the following leave-one-out procedure to the datasets to evaluate the framework. We initially remove the first graph from the database and use it as a query for the remaining database graphs. The graph is then put back in the database, and the procedure is repeated with the second graph from the

Fig. 4. Left: a view of a bat. Right: the shock graph constructed from the medial axis and superimposed on the left image.

database, etc., until all database graphs have been used as a query.

There exist several performance measures to assess the quality of a retrieval system or indexing mechanism. Precision and recall are two well-known examples. In some applications high precision is necessary, meaning that the relevant items that are returned must be at the top of the ranking. In some other applications, however, high recall is preferred, meaning that false negatives are to be avoided (the returned result must contain all or the most relevant objects). A good indexing system should, in fact, perform well according to both of these two measures. We conducted two sets of experiments to cover both scenarios. In the first experiment, the class of the query should be determined quickly (best match must appear high in the ranking). In the second experiment, all the objects belonging to the query class should be returned in a small candidate set.

For each query, the database graphs are ranked in decreasing order of vote weights in these experiments. Here, we consider the top highest-weight candidates. We say that our indexing system is effective, if at least one graph belonging to the same class as the query is among such candidates. A qualitative measure, therefore, should be based on the smallest size of the candidate list containing one image from the query class. According to the results of this experiment, in 37.3% of the cases, the highest-weight database graph belongs to the correct shape class for shock graphs and this ratio is 29.6% for Reeb graphs. Moreover, the average position of the closest matching graph among the highest-weight candidates is 7.4 and 4.3 for shock and Reeb graphs, respectively. In addition, the worst position of the closest matching graph is 12 for shock graphs, while this number is 9 for Reeb graphs. These results present that to determine the correct class of the query, more than 99% and 97% of shock and Reeb graph datasets can be pruned by our indexing mechanism.

While the previous evaluation method is suitable for classification tasks, in some retrieval applications, however, it is a prerequisite to retrieve all images from the database that belong to the query class. In the second experiment, the system's performance is evaluated by computing the total number of retrieved images that is necessary to retrieve the entire query class (maximum minimal scope). Our results show that the first 134 of the candidate return set always contain all the graphs belonging to the query class for shock graphs; this number is 54 for Reeb graphs. This indicates that for this task our framework prunes more than 90% and 87% of the shock and Reeb graph datasets, respectively. In other words, the recall in each dataset is 100% if the scope is set to the first 10% and 13% of the sorted candidate models for shock and Reeb graphs, respectively. In Fig. 8, we show percentage recall values for various scopes for shock and Reeb graph datasets.

The experimental results presented above clearly show the efficiency of the proposed indexing framework using different graph-based object representations. We now compare the performance of our method to other indexing frameworks on the same datasets. For this purpose, we first compare our results to that of an indexing system in which the graph signatures are generated using the eigenvalues for adjacency matrices. The experiments are identical to the ones described above. The results along with our scores are shown in Table 1, and reveal that our indexing framework outperforms the indexing system with adjacency eigenvalues in all criteria mentioned above. These results confirm that graph characterizations through laplacian matrices are more powerful and more informative than that of adjacency matrices. Representing graphs by laplacian eigenvalues, therefore, is more effective than that by adjacency eigenvalues. Additionally, we also compare our indexing framework to the one presented in [44]. This indexing algorithm was used in [54] on a subset of the McGill dataset with the object parts represented by directed acyclic graphs (DAG) through medial surfaces [45]. The subset of the McGill dataset used in this experiment includes a total of 320 exemplars taken from several object classes (hands, humans, teddy bears, glasses, pliers, tables, chairs, cups, airplanes, birds, dolphins, dinosaurs, four-leg-
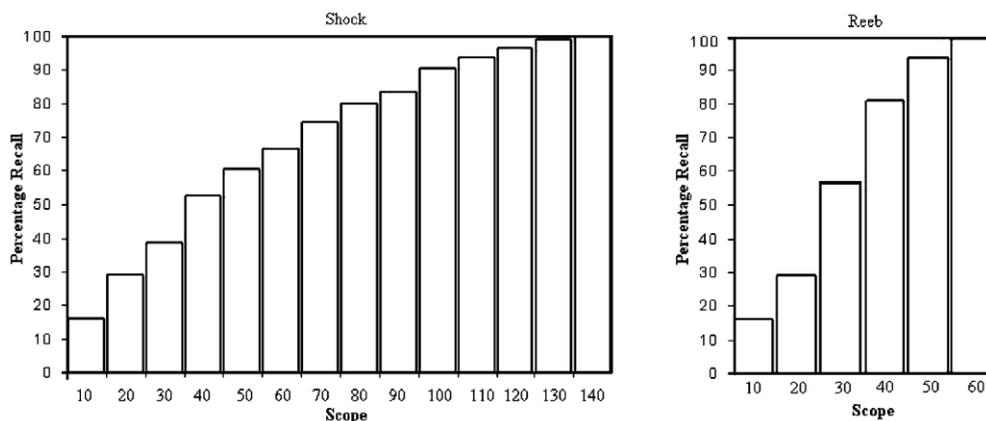

Fig. 8. Percentage recall values for various top ranked highest-weight candidate graphs for shock graphs of MPEG-7 and Reeb graphs of McGill datasets.

Table 1
Results for indexing system constructed using eigenvalues for laplacian and adjacency matrices

Criteria	HW (%)	AP	WP	MMS	PC (%)	PI (%)
Shock graphs–laplacian	37.3	7.4	12	137	99	90
Reeb graphs–laplacian	29.6	4.3	9	54	97	87
Shock graphs–adjacency	18	19.1	23	257	97	82
Reeb graphs–adjacency	17.3	10.2	22	92	94	78

HW, percentage of highest-weighted graph belonging to the same class as the query; AP, average position of the closest matching graph from the query class; WP, worst position of the closest matching graph from the query class; MSS, maximum minimal scope; PC, percentage of database that can be pruned to determine the right query class; PI, percentage of database that can be pruned to retrieve all instances of the query.

ged animals, and fish). To be consistent with the test in [54], we also merge the categories "four-legged" and "dinosaurs" into a broader class, "four-limbs". The results reported in [54] indicate that on average 70% of the models that are in the same class as the query are in the top 80 (25% of 320). Moreover, for 9 out of these 13 object classes, all instances of the query are in the top 80. Our results, on the other hand, show that 100% of the query classes are always in the top 48 (15% of 320). The improvement clearly demonstrates the better efficacy obtained by the proposed indexing framework. We believe that the improvement is due to (1) the more powerful representation of laplacian matrices, (2) the more effective signature construction by our algorithm, i.e., low loss of uniqueness in the signature, and (3) the better way of encoding local topology.

Our next set of experiments deals with measuring the time efficiency of the indexing framework when spectral integral variation is used during the signature generation for database graphs. Although these signatures are computed offline, for dynamic datasets where a new graph is likely to be added later, the time we spent in this process plays an important role. For each of the datasets, we generate graph signatures with and without spectral integral variation and measure the time taken in each case. We should note here that involving spectral integral variation in the framework does not change the effectiveness of the indexing system. According to the results, we observe that 47.3 and 115.8 min were spent to generate all subgraphs to be represented in the vector space for Reeb and shock graph datasets on an Intel(R) Pentium(R) 4 CPU 3.00 GHz computer. After involving spectral integral variation in the framework, the time we spent in this process decreases to 45.5 and 108.3 min, respectively. These results indicate that 1.8% and 6.5% time improvements are gained with spectral integral variation for these two datasets. In addition, our experiments show that when a new edge is added to a graph, spectral integral variation occurs in 6.1% and 21.4% of the cases for Reeb and Shock graph datasets. Therefore, the laplacian spectrum of the new graph is generated using the theorems given in Section 4. Although the experiments present that the larger dataset results in better time improvement, the overall time improvement of the framework depends on the percentage of the case in which spectral integral variation occurs. Balińska et al. [4] present an important survey on integral graphs (graphs whose adjacency spectrum consists entirely

of integers) and laplacian integral graphs (graphs whose laplacian spectrum consists entirely of integers). The authors observe that such graphs can be found in all classes of graphs and among graphs of all orders.

Finally, to evaluate the fitness of our approach for dealing with occlusion in images, we generated two sets of occluded scenes, each with two images. In the first set, there are 14 occluded scenes in which both images were selected from different classes in the MPEG-7 dataset, whereas in the second set there are 8 scenes where only one image was taken from the MPEG-7. In each scene, one shape occludes the other to a certain extent. The percentage of the occlusion in the first set varies from 2% to 16%, with on average 6.0%. In the second set, the MPEG-7 image is occluded for 12% to 31%, with on average 21.8%. Each of these new occluded scenes was used as a query against the complete database. We define our indexing schema to be effective if one of the query shapes appears in the highest vote-weight candidates. The results of these first and second sets of the occluded scenes are presented in Figs. 9 and 10, respectively. In each figure, the left column shows the occluded query images and the top ten candidates sorted by weight from left to right appear in each row. The average position of the closest shape belonging to the class of either of the query shapes was recorded as 1.7 for the first set. This number was 2.0 for the second set. We should point out that our signatures represent topological structures. Thus, images from different classes but with similar topologies may be assigned high weights. To give an example, consider the top row of Fig. 9, where the query consists of occluded shapes of a spoon and pencil. While neither a spoon nor a pencil is similar to a butterfly, the current combination of them in the query becomes similar to the butterfly retrieved as the highest rank. Shapes belonging to different classes, therefore, may be ranked high in the candidate list.

It should be noted that both shock and Reeb graph experiments can be considered as worst-case for two reasons. First, one may argue that the MPEG-7 and McGill datasets are not the ideal testbeds for an indexing algorithm. Some classes (especially in MPEG-7) are very close to each other (such as ten different device classes) and they can, in fact, be grouped into a broader class. As a result, performing such a grouping operation will improve the overall quality of the framework. Second, since our signatures are constructed based on graph topology, evaluating
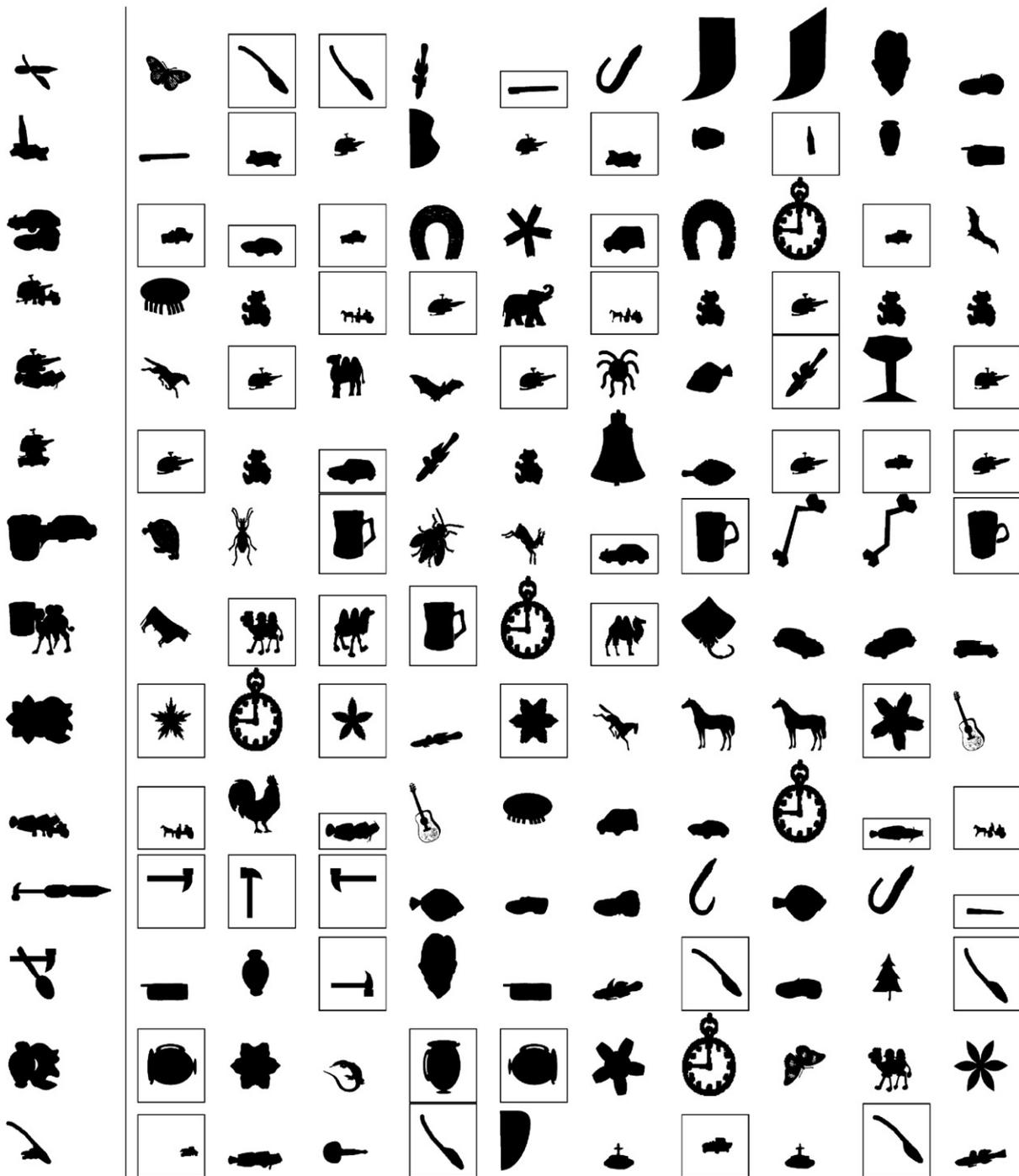
Fig. 9. Results of occlusion experiments where two images in each occluded scene are selected from different classes of the MPEG-7 dataset. The leftmost column shows the occluded query scenes for each row, while the top ten ranked candidate models are shown on the right, in the decreasing order of weight. An image inside a box indicates that it belongs to the class of one of the query shapes. Images belonging to different classes but are topologically similar to the query are ranked high in the candidate list.

these results such that the graph topology is taken into consideration would yield even better performance. Thus, in addition to checking if the shapes corresponding to the query and the highest-weight candidate graphs belong to the same object class, we might also consider the possibility that two graphs from two different object classes may, in fact, be close in terms of their topologies. We expect that the evaluation of our results using a distance matrix created by a matching algorithm, which takes into account the topology, will produce even better scores.

6. Conclusions and future work

In this paper, we have proposed a novel, graph-based indexing method using the eigenvalue characterization of laplacian matrices. The sorted eigenvalues of the laplacian
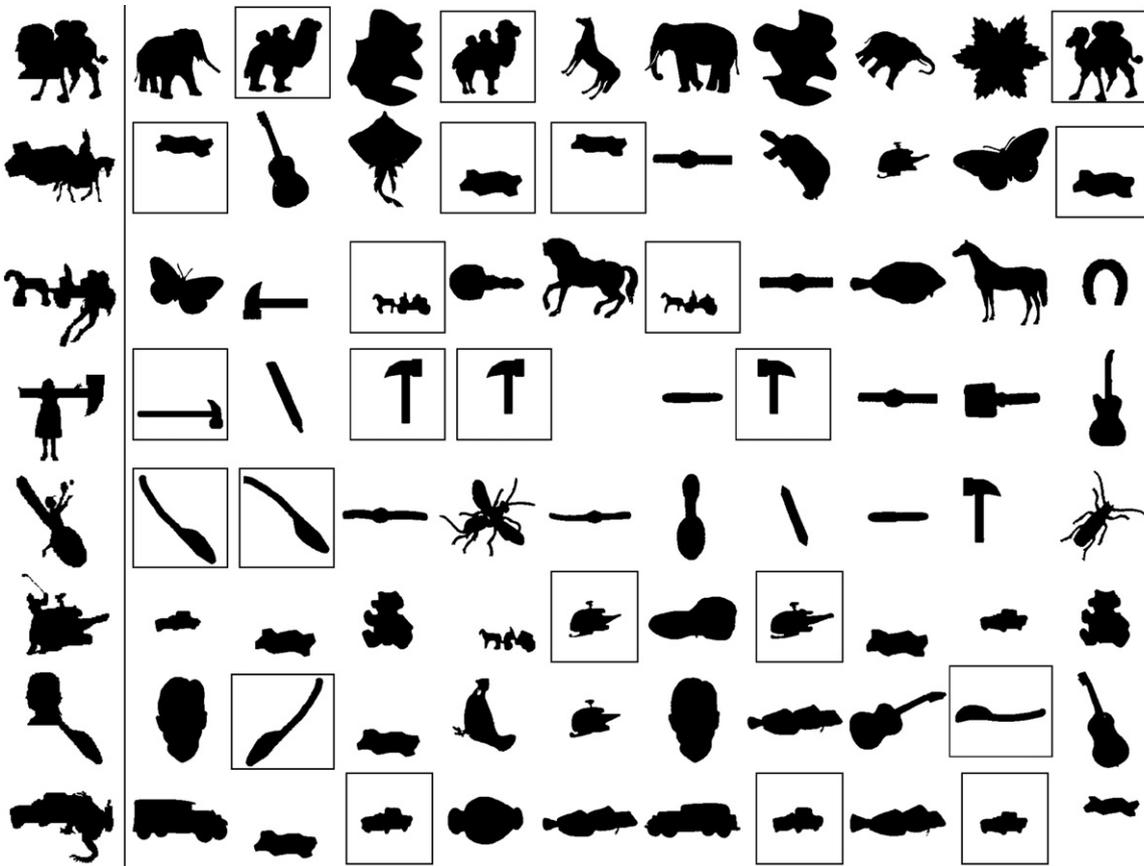
Fig. 10. Results of occlusion experiments where only one of the two images in each occluded scene is selected from the MPEG-7 dataset. The leftmost column shows the occluded query scenes for each row, while the top ten ranked candidate models are shown on the right, in the decreasing order of weight. An image inside a box indicates that it belongs to the class of the query shape taken from the MPEG-7. Topologically similar images to the query are also ranked high in the candidate list.

matrix of a graph $G = (V, E)$ become the components of an $O(|V|)$-dimensional vector. A nearest neighbor search around this vector returns graphs that are similar to G. This implies that no graph isomorphism is required; our method retrieves those graphs that are similar in terms of their topologies. To account for partial similarities, we create signatures for subgraphs of G. We draw from recently developed techniques in the field of spectral integral variations to overcome the problem of computing the laplacian spectrum for every subgraph individually from its eigendecomposition.

By using the laplacian spectrum as a signature, we capture the graph topology to a large extent. The signature of a graph is invariant under the reorderings of its vertices. This, in turn, allows us to compare the signatures of a large number of graphs without solving the computationally expensive correspondence problem between their vertices. Although determining graphs that can uniquely be defined by their graph spectra is a difficult problem [12], we showed in this paper that representing graphs by their laplacian spectra is more discriminating than that by adjacency spectra.

Our framework can index any multimedia database where objects can be represented as graphs. We have suc-

cessfully evaluated the approach on several databases using two different graph-based object representations. Moreover, the approach compares favorably to a leading indexing algorithm. We also demonstrated the robustness of the proposed framework on a set of occlusion experiments.

An increasing number of applications use indexing systems for fast selection of candidate elements. Although we applied our method to shape indexing in this paper, we will test the framework to some other applications with a particular interest to layout indexing. In a typical layout indexing problem, a query and a set of database images, each with multiple shapes, are given. For a given query image, one would like to efficiently retrieve images, which not only contain similar shapes to those in the query, but similar layouts as well. Extension of our method to both shape and layout indexing in a unified framework is one of our interests in the future.

Additionally, we plan to extend our work in a number of ways. We will first incorporate geometric information in the indexing system. Thus, both geometric and topological similarities will be taken into account during the process of fast candidate selection. We believe that this important addition will make the whole system more effective. Rather than evaluating our results using the classification per-

formed on the original dataset, we will use different sets of existing matching algorithms and measure the fitness of the framework with respect to these matching results. In addition, we plan to conduct a more comprehensive comparison of our approach to more leading indexing algorithms, including a test regarding the time efficiency of each system. Since our framework computes a proper topological similarity between graph pairs, one of our future goals is also to design a matching approach based on a similar idea. We will not only compute the similarity value between graphs, but also find the node correspondences using both topology and geometry.

Acknowledgments

References

[1] W.N. Anderson, T.D. Morley, Eigenvalues of the laplacian of a graph, Linear and Multilinear Algebra 18 (1985) 141–145.

[2] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, A.Y. Wu, An optimal algorithm for approximate nearest neighbor searching in fixed dimensions, Journal of the ACM 45 (6) (1998) 891–923.

[3] M. Attene, S. Biasotti, M. Spagnuolo, Shape understanding by contour-driven retiling, The Visual Computer 19 (2–3) (2003) 127–138.

[4] K. Balińska, D. Cvetković, Z. Radosavljevic, S. Simić, D. Stevanović, A survey on integral graphs, Univerzitet Beogradu Publikacije Elektrotehnickog Fakulteta Serija Matematika 13 (2002) 42–65.

[5] S. Barik, S. Pati, On algebraic connectivity and spectral integral variations of graphs, Linear Algebra and its Applications 397 (2005) 209–222.

[6] S. Biasotti, Reeb graph representation of surfaces with boundary, in: SMI'04: Proceedings of the Shape Modeling International 2004, IEEE Computer Society, Washington, DC, USA, 2004, pp. 371–374.

[7] S. Biasottia, S. Marini, M. Spagnuoloa, B. Falcidieno, Sub-part correspondence by structural descriptors of 3D shapes, Computer-Aided Design 38 (9) (2006) 1002–1019.

[8] S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkeler, H. Zhang, An eigenspace update algorithm for image analysis, Graphical Models and Image Processing: GMIP 59 (5) (1997) 321–332.

[9] K. Cole-McLaughlin, H. Edelsbrunner, J. Harer, V. Natarajan, V. Pascucci, Loops in Reeb graphs of 2-manifolds, Discrete and Computational Geometry (2004) 231–244.

[10] M.S. Costa, L.G. Shapiro, Relational indexing, in: SSPR'96: Proceedings of the 6th International Workshop on Advances in Structural and Syntactical Pattern Recognition, Springer-Verlag, London, UK, 1996, pp. 130–139.

[11] D.M. Cvetković, M. Doob, H. Sachs, Spectra of Graphs: Theory and Application, second ed., VEB Deutscher Verlag der Wissenschaften, Berlin, 1982.

[12] E.R.V. Dam, W.H. Haemers, Spectral characterizations of some distance-regular graphs, Journal of Algebraic Combinatorics 15 (2) (2002) 189–202.

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

[14] M. Fiedler, Algebraic connectivity of graphs, Czechoslovak Mathematics 23 (1973) 298–305.

[15] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, NY, USA, 1979.

[16] C.D. Godsil, B.D. McKay, Constructing cospectral graphsAequationes Mathematicae, vol. 25, Birkhäuser Basel, University of Waterloo, 1982, pp. 257–268.

[17] H.L.D.B. Goldgof, T.S. Huang, Matching and motion estimation of three-dimensional point and line sets using eigenstructure without correspondences, Pattern Recognition 25 (3) (1992) 271–286.

[18] R. Grone, R. Merris, V.S. Sunder, The Laplacian spectrum of a graph, SIAM Journal on Matrix Analysis and Applications 11 (1990) 218–238.

[19] W.H. Haemers, E. Spence, Enumeration of cospectral graphs, European Journal of Combinatorics 25 (2) (2004) 199–211.

[20] M. Hilaga, Y. Shinagawa, T. Kohmura, T.L. Kunii, Topology matching for fully automatic similarity estimation of 3D shapes, in: SIGGRAPH'01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, New York, NY, USA, 2001, pp. 203–212.

[21] C.M. Hoffmann, Group-theoretic algorithms and graph isomorphism, Springer-Verlag, Berlin, 1982.

[22] J.E. Hopcroft, J.K. Wong, Linear time algorithm for isomorphism of planar graphs, in: STOC'74: Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, ACM Press, New York, NY, USA, 1974, pp. 172–184.

[23] R. Horaud, T. Skordas, Structural matching for stereo vision, in: Ninth International Conference on Pattern Recognition, Rome, Italy, 1988, pp. 439–445.

[24] D. Huttenlocher, D. Klanderman, A. Rucklige, Comparing images using the Hausdorff distance, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (9) (1993) 850–863.

[25] O. Jesorsky, K. Kirchberg, R. Frischholz, Robust face detection using the Hausdorff distance, in: AVBPA'01: Proceedings of the Third International Conference on Audio and Video-Based Biometric Person Authentication, Springer-Verlag, London, UK, 2001, pp. 90–95.

[26] W. Kim, A. Kak, 3-D object recognition using bipartite matching embedded in discrete relaxation, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (3) (1991) 224–251.

[27] S. Kirkland, A characterization of spectral integral variation in two places for Laplacian matrices, Linear and Multilinear Algebra 52 (2) (2004) 79–98.

[28] J. Koenderink, A. van Doorn, The internal representation of solid shape with respect to vision, Biological Cybernetics 32 (1979) 211–216.

[29] S.W. Lee, J.H. Kim, Attributed stroke graph matching for seal imprint verification, Pattern Recognition Letters 9 (1989) 137–145.

[30] S.W. Lu, Y. Ren, C.Y. Suen, Hierarchical attributed graph representation and recognition of handwritten Chinese characters, Pattern Recognition 24 (7) (1991) 617–632.

[31] R. Merris, Laplacian matrices of graphs: a survey, Linear Algebra Applications 197 (1994) 143–176.

[32] B.T. Messmer, H. Bunke, A decision tree approach to graph and subgraph isomorphism detection, Pattern Recognition 32 (12) (1999) 1979–1998.

[33] B. Mohar, The Laplacian spectrum of graphs, in: Sixth International Conference on the Theory and Applications of Graphs, 1988, pp. 871–898.

[34] B. Mohar, Laplace eigenvalues of graphs: a survey, Discrete Mathematics 109 (1–3) (1992) 171–183.

[35] B. Mohar, Some applications of Laplace eigenvalues of graphsGraph Symmetry: Algebraic Methods and Applications, vol. 497, Kluwer, 1997, pp. 227–275.

[36] S. Pati, The third smallest eigenvalue of the Laplacian matrix, Electronic Journal of Linear Algebra 8 (2001) 128–139.

[37] D.H. Rouvray, A.T. Balaban, Chemical applications of graph theory, in: Applications of Graph Theory, Academic Press, London, 1979, pp. 177–221.

[38] W.J. Rucklidge, Locating objects using the hausdorff distance, in: ICCV'95: Proceedings of the Fifth International Conference on Computer Vision, IEEE Computer Society, Washington, DC, USA, 1995, p. 457.

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

[40] K. Sengupta, K. Boyer, Organizing large structural modelbases, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (4) (1995) 321–332.

[41] K. Sengupta, K.L. Boyer, Modelbase partitioning using property matrix spectra, Computer Vision and Image Understanding 70 (2) (1998) 177–196.

[42] L.G. Shapiro, R.M. Haralick, Organization of relational models for scene analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 4 (11) (1982) 595–602.

[43] L.S. Shapiro, J.M. Brady, Feature-based correspondence: an eigenvector approach, Image Vision Computing 10 (5) (1992) 283–288.

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

[45] K. Siddiqi, S. Bouix, A. Tannenbaum, S.W. Zucker, The Hamilton–Jacobi skeletons, International Journal of Computer Vision 48 (3) (2002) 215–231.

[46] K. Siddiqi, A. Shokoufandeh, S. Dickinson, S. Zucker, Shock graphs and shape matching, International Journal of Computer Vision 35 (1) (1999) 13–32.

[47] W. So, Rank one perturbation and its application to the Laplacian spectrum of a graph, Linear and Multilinear Algebra 46 (1999) 193–198.

[48] H. Sossa, R. Horaud, Model indexing: the graph-hashing approach, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1992, Champaign, IL, USA, pp. 811–814.

[49] A.H. Tewfik, M. Deriche, An eigenstructure approach to edge detection, IEEE Transactions on Image Processing 2 (3) (1993) 353–368.

[50] M. Turk, A.P. Pentland, Eigenfaces for recognition, Journal of Cognitive Neuroscience 3 (1) (1991) 71–86.

[51] A. Wong, S. Lu, M. Rioux, Recognition and shape synthesis of 3-d objects based on attributed hypergraphs, IEEE Transactions on Pattern Analysis and Machine Intelligence 11 (3) (1989) 279–290.

[52] E.K. Wong, Model matching in robot vision by subgraph isomorphism, Pattern Recognition 25 (3) (1992) 287–303.

[53] F. Yizheng, On spectral integral variations of graphs, Linear and Multilinear Algebra 50 (2) (2002) 133–142.

[54] J. Zhang, K. Siddiqi, D. Macrini, A. Shokoufandeh, S.J. Dickinson, Retrieving articulated 3-D models using medial surfaces and their graph spectra, in: International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition, St. Augustine, FL, USA, 2005, pp. 285–300.

[55] P. Zhu, R.C. Wilson, A study of graph spectra for comparing graphs, in: British Machine Vision Conference, Oxford, UK, 2005.