Shape Indexing through Laplacian Spectra

M. Fatih Demirci mdemirci@cs.uu.nl
Reinier H. van Leuken reinier@cs.uu.nl
Remco C. Veltkamp remco.veltkamp@cs.uu.nl

Department of Information and Computing Sciences, Universiteit Utrecht

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, whose 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 in which the components correspond to the sorted laplacian eigenvalues of the graph or subgraphs. We draw from recently-developed techniques in the field of spectral integral variation to overcome the problem of computing the laplacian spectrum for every subgraph individually. By doing a nearest neighbor search around the query spectra, similar but not necessarily isomorphic graphs are retrieved. 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. Our experiments, consisting of recognition trials in the domain of 2-D and 3-D object recognition, including a comparison with a competing indexing method, demonstrate both the robustness and efficacy of the approach.

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 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. 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. 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 [7].

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. 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 [13] 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.

Although indexing methods with (sub)graph isomorphism detection algorithms are effective, due to noise, occlusion, or segmentation errors, no (sub)graph isomorphism may 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.

A framework related to the approach reported in this paper is that of Shokoufandeh et al. [17]. 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 component of its $\delta-$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 neigbor 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.

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 [14]. Given a graph, the sorted eigenvalues of its laplacian matrix become the components of its signature, which is linear in the number of vertices. To perform indexing locally as well and thus to encode the topology of subgraphs in the framework, we adopt a technique analogous to that used in the decision tree approach [13]. Having established the signatures, the indexing now amounts to a nearest neighbor search around the query signatures in a model database. 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.

2 Definitions and 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 such that $u, v \in V$. 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 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 1 if there exists an edge between u and v, and 0 otherwise. 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 [15, 16, 12, 14] 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 $\Lambda = \mathrm{diag}\,(\lambda_1, \lambda_2, \ldots, \lambda_{|V|})$ 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 [8] and more recently Haemers and Spence [10] have shown that the laplacian matrix has more representational power than the adjacency matrix, in terms of resulting in less number of cospectral graphs. According to the results given in [10], 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 Winson [22]. 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%. Overall, these studies show that the laplacian spectrum is more representative and more informative than the adjacency spectrum. Our motivation for constructing the graph characterizations using the laplacian spectrum comes from these studies.

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

3 Encoding The Graph Structure

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. 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. Note that it is important to construct the signatures using the sorted eigenvalues, as the kth smallest eigenvalue reflects specific information about the graph, e.g., the relation between the second smallest laplacian eigenvalue λ_2 and the diameter, mean distance, minimum degree, and algebraic connectivity of the graph.

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.

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:*

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

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. Constructing the indexing mechanism based on graph signatures is, therefore, too weak. An ideal indexing framework should, in fact, select candidate database elements

[2]This theorem is obtained by Courant-Weyl([3], Theorem 2.1). The reader may also refer to [9].

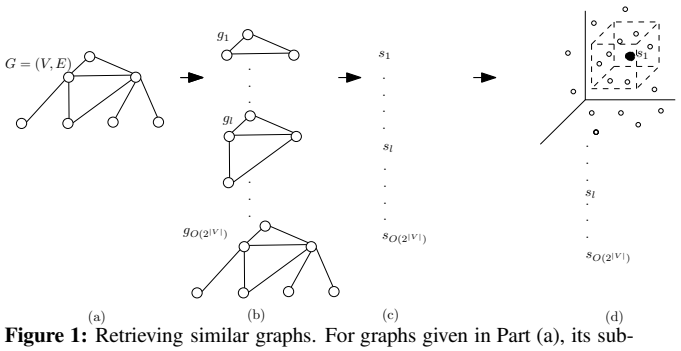

Figure 1: Retrieving similar graphs. For graphs given in Part (a), its subgraphs are constructed in Part (b). A signature is computed for each subgraph in Part (c). Given a signature, retrieving its similar graphs from a large database is formulated as a nearest neighbor search as shown in Part (d).

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 [13].

For a given database graph, 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 re-orderings of the graph's vertices, it is sufficient to compute 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. 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 Figure 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.

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 sig-

natures in the vector space. The database, therefore, can be pruned without losing structurally similar graphs.

4 More Efficient Indexing through Spectral Integral Variation

The local indexing procedure described above requires individual computation of the laplacian spectrum for each subgraph. Although for database graphs known a priori this process is performed offline, 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. 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 computing it. 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, on the other hand, provides more information.

It is shown in [20] 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.

In our framework, we will detect the changed eigenvalue(s) in the cases where spectral integral variation occurs. For this purpose, we draw on two theorems that appear in [20] and [11]. We refer to these papers or to the full version of this paper [6] for the specific use of these theorems.

After detecting changed laplacian eigenvalues when spectral integral variation occurs, we perform the following process for each given database graph offline. 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 $\hat{G}$ with $|V|$ vertices and k edges and compute its laplacian eigenvalues. 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 $\hat{G}$ are used as the signature. Next, when we add an edge to $\hat{G}$, we check whether spectral integral variation occurs and if so, we generate the eigenvalues using the theorems referred above. 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. In the experiments, we have generated the signatures for (sub)graphs with at least $k = 4$ edges.

Figure 2: Left: a view of a bat. Right: the shock graph constructed from the medial axis and superimposed on the left image.

Although the above formulation enables us to identify the changed laplacian eigenvalues when they are increased by integer quantities only, our empirical results show that it speeds up the signature generation step for a database with 1440 graphs known a priori by 6.47%.

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 2-D and 3-D object recognition, including a comparison with a competing indexing method. We first perform our experiments using silhouettes. For a given shape, its silhouette is represented by an undirected shock graph [19]. The graph is constructed from the discrete skeleton using the method described in [5]. An illustration of this type of graph is given in Figure 2, 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.

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. See [2] for details on the construction of Reeb graphs. The right of Figure 3 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 [21]. The database consists of 420 objects classified in 19 classes.

We first represent each object in each database as a graph. The average size of a graph in each database is 35. Given a graph, we compute the signatures for each of its subgraphs and populate the resulting signatures in the vec-

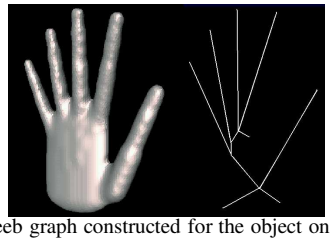

Figure 3: The Reeb graph constructed for the object on the left is shown on the right.

tor space. In our experimental setup, we applied the following leave-one-out procedure to the datasets for evaluation. 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 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.

The results for these experiments are presented in Table 1. For comparison purposes, the same experiments were conducted with the adjacency spectrum representation. For each query, the database graphs are ranked in decreasing order of the vote weights. In 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.

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 contains 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.

CRITERIA	HW(%)	AP	WP	PC(%)	PI(%)
SHOCK, LAPL.	37.3%	7.4	12	99%	90%
REEB, LAPL.	29.6%	4.3	9	97%	87%
SHOCK, ADJ.	18.0%	19.1	23	97%	82%
REEB, ADJ.	17.3%	10.2	22	94%	78%

Table 1: Results for indexing systems constructed using eigenvalues for Laplacian and Adjacency matrices. 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, 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.

Additionally, we compare our indexing framework to the one presented in [17]. This indexing algorithm was used in [21] on a subset of the McGill dataset with the object parts represented by directed acyclic graphs (DAG) through medial surfaces [18]. 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-legged animals, and fish). To be consistent with the test in [21], we also merge the categories "four-legged" and "dinosaurs" into a broader class, "four-limbs". The results reported in [21] indicate that on average 70% of the models from 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) more powerful representation obtained by laplacian matrices, 2) more effective signature construction by our algorithm, i.e., low loss of uniqueness in the signature, and, 3) better encoding of local topology.

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 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 varia-

tions to overcome the problem of computing the laplacian spectrum for every subgraph individually.

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 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 [4], we showed in this paper that representing graphs by their laplacian spectra is more discriminating than by adjacency spectra.

We plan to extend our work in a number of ways. We will first incorporate geometric information in the indexing system and combine it with the topological similarity 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 performed on the original dataset, we will use different sets of existing matching algorithms and measure the fitness of the framework. 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. Although our experiments present promising retrieval results for databases of sizes 1400 and 420, we will also evaluate our framework on large datasets (more than around 10000 images).

Our proposed indexing framework has been designed to account for both occlusion and local deformation. In one of our future works, we will perform a set of occlusion experiments and report the performance of the system. Finally, we will also study in detail how the concept of spectral integral variation changes the time efficiency of our indexing system. To measure this, we will compute the time it takes to construct the signature for each database graph with and without spectral integral variation. Improving the time efficiency of the signature construction process for database graphs known a priori is especially important for applications where new database entries are being inserted frequently.

Acknowledgments This research was supported by the FP6 IST projects 511572-2 PROFI and 506766 AIM@SHAPE. We thank Simone Marini, IMATI-CNR Italy, for providing the Reeb graphs.

References

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

[2] S. Biasotti. Reeb graph representation of surfaces with boundary. In *SMI'04*, pages 371–374, Washington, DC, USA, 2004.

[3] D. Cvetković, M. Doob, and H. Sachs. *Spectra of Graphs: Theory and Application*. VEB Deutscher Verlag der Wissenschaften, Berlin, 2nd edition, 1982.

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

[5] M. F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, and S. Dickinson. Object recognition as many-to-many feature matching. *IJCV*, 69(2):203–222, 2006.

[6] M. F. Demirci, R. van Leuken, and R. Veltkamp. Indexing through laplacian spectra, 2007. Computer Vision and Image Understanding, Special Issue on Similarity Matching in Computer Vision and Multimedia (to appear).

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

[8] C. Godsil and B. McKay. Constructing cospectral graphs. In *Aequationes Mathematicae*, pages 257– 268, 1982.

[9] R. Grone, R. Merris, and V. S. Sunder. The laplacian spectrum of a graph. *SIAM Journal on Matrix Analysis and Applications*, 11:218–238, 1990.

[10] W. H. Haemers and E. Spence. Enumeration of cospectral graphs. *Eur. J. Comb.*, 25(2):199–211, 2004.

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

[12] R. Merris. Laplacian matrices of graphs: a survey. In *Linear Algebra Applications*, volume 199, pages 381–389, 1994.

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

[14] B. Mohar. The laplacian spectrum of graphs. In *Sixth International Conference on the Theory and Applications of Graphs*, pages 871–898, 1988.

[15] B. Mohar. Laplace eigenvalues of graphs: a survey. *Discrete Math.*, 109(1-3):171–183, 1992.

[16] B. Mohar. Some applications of laplace eigenvalues of graphs, 1997.

[17] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S. Zucker. Indexing hierarchical structures using graph spectra. *PAMI*, 27(7), 2005.

[18] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker. The hamilton-jacobi skeletons. *IJCV*, 48(3):215–231, 2002.

[19] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *IJCV*, 30:1–24, 1999.

[20] W. So. Rank one perturbation and its application to the laplacian spectrum of a graph. *Linear and Multilinear Algebra*, 46:193–198, 1999.

[21] J. Zhang, K. Siddiqi, D. Macrini, A. Shokoufandeh, and S. J. Dickinson. Retrieving articulated 3-d models using medial surfaces and their graph spectra. In *EMMCVPR*, 2005.

[22] P. Zhu and R. C. Wilson. A study of graph spectra for comparing graphs. In *British Machine Vision Conference*, 2005.