PROFI

Project number:	FP6-511572
Project acronym:	PROFI
Title:	Perceptually-relevant Retrieval Of Figurative Images

Deliverable No: D7.1: Indexing scheme for retrieval by lay-out

Short description:
Given a query and a large database, the problem of layout indexing can be formulated as efficiently selecting candidate models, which have similar layouts as the query. The layout of an image with multiple shapes can be represented as a graph whose nodes correspond to shapes and whose edges show relations between the shapes. When graphs are used to represent images, the problem of layout indexing can thus be transformed into that of graph-based indexing.
In this report, we present an overall framework as well as a new graph-based indexing technique for large datasets. The algorithm is based on a powerful characterization of graphs through their laplacian matrices. We draw on an important theorem from graph theory to show that our graph characterization can be used to retrieve similar graphs or subgraphs from the database. Experiments performed in the context of a recognition task demonstrate both robustness and efficacy of the overall approach.

Due month:	M21
Delivery month:	M21
Lead partner:	Utrecht University
Partners contributed:	Utrecht University
Classification:	PU

Project funded by the European Community under the "Information Society Technologies" Programme

1 Results obtained

1.1 Objectives

PROFI work package (WP) 7 is layout indexing. The objective of layout indexing problem is to efficiently select candidate elements, which have similar layouts as the query. During the development of WP 7, we represent images as graphs, where the vertices correspond to shapes and the edges show the relations between the shapes. This representation allows us to formulate the problem of layout indexing as that of indexing based on graph structures. When working with graphs, indexing is defined as the problem of efficiently selecting a small set of database graphs, which share a subgraph with the query. In the context of graphs, while considerable research has been devoted to the problem of graph matching, rather less attention has been paid to graph-based indexing.

Towards this objective, we have designed an overall framework. Section 1.2 presents a new indexing framework for retrieving images by both shape and layout similarities in a large image database systems. The novelty of our approach lies in the combination of several competing approaches into a unified scheme. This, in turn, allows us to overcome the drawbacks of one approach by taking advantage of another. Our framework addresses layout and shape (both global and partial) indexing problems altogether. The system organizes database images in a multi-level structure, in which the relations between two consecutive-level entities are formed by parent/child edges with certain properties. Levels are designed to answer the specific query types for each case.

Within this framework, the technical aspect of the indexing is performed by a novel technique for indexing graph structures, given in the Appendix, which has been submitted as a scientific paper. 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. 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. Given the laplacian spectrum of a principal submatrix of a matrix, 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. More specifically, by performing a nearest neighbor search around the query spectra, similar but not necessarily isomorphic graphs are retrieved. In addition, for a query graph, a voting schema ranks database graphs into an indexing hypothesis to which a final matching process can be applied. We give the details of our indexing algorithm in the first appendix.

1.2 Extendable Multi-level Layout and Shape Indexing

1.2.1 Introduction

Shape matching is one of the fundamental problems in computer vision. In a typical matching problem the objective is to compute an overall measure of similarity 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. For one query, finding its best matches from a large database requires an effective and efficient indexing, in which the objective is to return

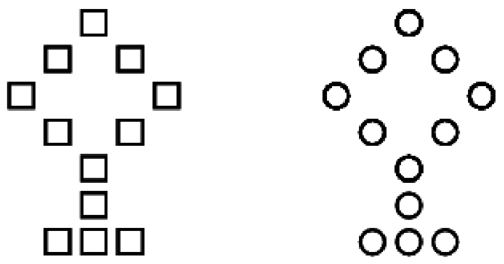


Figure 1: An example for one type of image retrieval problem where the overall layout similarity between two images is more significant than the individual shape similarities.

a small collection of candidate models to which the actual matching process is applied.

Given a query image and a large image database, each with multiple shapes, a good indexing algorithm filters out database images based on individual shape similarities as well as the spatial relations (or layout) that exist between the shapes. The overall similarity between two images is usually computed as the total similarity of their individual shapes and the closeness of their spatial relations. Thus, the best match for a particular query is the one that resembles both shape and layout similarities. In this scenario the indexing algorithm should perform the filtering process such that while the images, which have sufficiently similar shapes as the query, are kept, the others are eliminated. Computing the similarity between two shapes often requires the global shape characteristics to be similar. On the other hand, finding partial similarity involves the process of extracting parts and computing the similarity between them using local information. After retrieving the database images, which have sufficiently similar shapes as the query, they should be sorted based on the closeness of their layouts to the query.

Although this seems a quite common feature that any indexing approach should have, in some applications (one type of trademark registration, especially) the overall layout similarity is more important than the individual shape similarities (see Figure 1). The indexing algorithm, therefore, should return the images containing similar layouts regardless of the closeness their shapes. Clearly, the objectives of an indexing algorithm in these cases mentioned above are different, i.e., one image considered as false negative in one case may be false positive in the other. Specifically, given a query and a large database, the objective of an indexing algorithm for each case can be summarized as follows:

Case 1: Retrieve images that have overall layout similarity as the query. The shape similarities are not taken into consideration in this case.

Case 2: Retrieve images that have individual shape similarities as the query. Then sort them based on layout similarity.

Case 3: Retrieve images that have partial shape similarities as the query. Then sort them based on layout similarity.

Although case 2 and case 3 might seem closely related, in case 1, global information (center of mass, area, perimeter, elongation, etc.) is used to determine how similar two shapes are, while in case 2 only local information is considered. Many indexing approaches have been proposed for each of these cases separately. However, in some applications such as trademark registration, users are often interested in all three cases to determine if a newly designed trademark image is close to any of the registered, existing trademarks. A more comprehensive indexing approach should, therefore, address all of these problems in the same framework.

This paper presents a new indexing framework for retrieving images by both shape and layout similarities in a large image database systems. The novelty of our approach lies in the combination of several competing approaches into a unified scheme. This, in turn, allows us to overcome the drawbacks of one approach by taking advantage of another. Our framework contains three sub-parts, one for each case listed above. The system organizes database images and their shapes in a multi-level structure, in which the relations between two consecutive-level entities are formed by parent/child edges with certain properties. Levels are designed to answer the specific query types for each case. A variety of techniques have been proposed in the literature for shape representation [27]. These techniques are often divided into two classes: region-based and boundary-based. While the contour of shapes are used in boundary-based representation methods, the internal details along with the boundary details are taken into account in region-based algorithms.

1.2.2 Related Work

We distinguish between *physical* or *raw* images at pixel level (residing in the image database), and *logical* representations of these images (residing in the logical database). Computational performance cannot be ignored for large image sets: a linear scan through all image representations while querying makes interactive searching practically impossible for large databases. Therefore, indexing of the database has to be incorporated in the retrieval system to speed up the search. The actual indexing of the images takes place in the logical database. First we will consider some general indexing strategies.

In the early years of content based image retrieval, indexing was mostly done by space partitioning methods such as the kd-tree [2], R-tree [19] or variants such as the $R+$-tree [35] or $R*$-tree [1]. For a complete overview of these multidimensional access methods see the survey by Gaede and Günther [17]. In general, these methods either partition the data space into disjoint cells of possibly varying size (kd-tree and related work), or associate a region with each object in the data space (R-tree family).

When the feature or object space is metric, example-based space-partitioning techniques are well suited for indexing purposes. One of the first works in this field was by Yianilos [40]. All database objects are divided into concentric rings around one or multiple example objects and then stored in a tree. These example objects are often called vantage objects or vantage points. Other examples based on this strategy are the VP-tree [4], the M-Tree [8] and the MVP-Tree [5]. These and other techniques for searching in metric spaces are surveyed by Chaves et al. [13].

Instead of dividing the database objects in concentric rings around vantage objects and storing them in a tree, database objects can be embedded in a feature space. The dimensionality of this feature space is related to the number of vantage objects and where the features are based on the distances the database objects have to the vantage objects. Examples of these embedding techniques are Vantage Indexing [38], Fastmap [16], SparseMap [22] and MetricMap [39]. These methods are surveyed by Hjaltason and Samet [20]. A big advantage of these methods over tree-based indexing methods is that the required number of online distance calculations is reduced to the dimensionality of the embedding. Once the query has been positioned in the embedding space, all that is needed is a geometric range query or a nearest neighbour query where no more distance calculations are involved.

The methods mentioned above are all general shape indexing methods, i.e. they can be used in combination with any kind of distance measure. Some indexing strategies specifically facilitate queries by spatial similarity, i.e. queries containing layout information. We will now consider some of these spatial indexing techniques and highlight their main features in Table 1.

The spatial relations between two objects in an image can be divided into topological relations and directional relations. Egenhofer [14] describes 8 basic topological relations (*disjoint, contains, inside, meet, equal, covers, covered-by* and *overlap*). Directional relations are usually represented by the four primary directions (North, South, East and West) and the four secondary directions (NW, SE, SW and SE). An alternative for this representation is the angle between the line connecting the two centers of mass and the horizontal line. For instance, the latter method was used by El-Kwae et al. [15] and Gudivada et al. [18].

A method taking into account only directional information was proposed by Chang et al. [6], and is called 2D-Strings. To produce a 2D-string representation, the center of mass of each object in the image is projected on the x and y axes. By taking the objects from left to right and from below to above, two one-dimensional strings are obtained, in which the objects are represented by a class identifier. The shape matching problem is now transformed into string matching. Various extensions have been proposed such as the 2D G-String [7], 2D C-String [24]. These extensions deal mainly with overlapping objects with complex shapes.

Petrakis and Orphanoudakis [31] propose an indexing scheme based on 2D-Strings. For each image, all possible subsets of size 2 up to a predefined number K_{max} are created. These subsets are represented by a string taking into account both layout information and object specific information: the order (as in a 2D-String), inclusion properties, object size, roundness and orientation.

A major drawback of these *symbolic projection methods* is that in general they are not rotation invariant. El-Kwae et al. [15] propose a robust Framework for Retrieving Images by Spatial Similarity (FRISS). It can handle translation, scaling, perfect rotation (all objects in the image are rotated around a reference point with the same angle), multiple rotation (objects are rotated around a reference point with different angles). Furthermore, it takes into account topological relations between the objects and shape-based similarities.

A popular alternative to symbolic projection methods is the graph representation. Gudivada and Raghavan [18] propose spatial orientation graphs (SOG), in which each vertex represents an object and the edges between them are weighted with the slope of the line connecting the two centers of mass. The distance between two graphs is calculated by finding the angle between each pair of corresponding edges. ImageMap [30], proposed by Petrakis and Faloutsos, extends this idea. The images are represented as attributed relational graphs (ARG), storing object size, orientation, and roundness in the nodes and distance, angle, and contains-relationships in the edges. This approach first computes an $n \times n$ distance matrix, where each entry corresponds to the graph-edit distance between its corresponding graph pairs. The graphs are then embedded into an f-dimensional space (target space) using Fastmap [16] such that the distances in the target space are approximately equal to those in the original graph space. The method formulates the image retrieval problem as that of range search in the target space. Note however that the embedding process does not preserve the distances exactly, but the distances are distorted up to a certain degree. Although powerful, the method suffers from the limitations of the graph-edit distance approach. For instance, if the graphs are not trees, i.e, the images can not be represented as trees, then the graph distances can not be computed in polynomial-time using this approach. In addition, due to the fact that the graph-edit distance does not deal well with the occlusion, it is not clear how

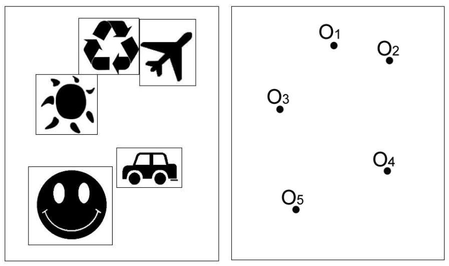


Figure 2: Layout indexing on an image consisting of iconic building blocks, example similar to the experiments conducted by El Kwae and Kabuka [15]. On the left the original image, on the right it's logical representation.

this indexing schema performs against noise and occlusions.

Yet another graph representation is proposed by Nabil et al. [28]. In the 2D-Projection Interval Representation, each object in the image is projected onto the x and y-axes. The intervallic relations between these projections are computed, and stored in the graph's edges. The topological relations that Egenhofer [14] pointed out are stored in these edges as well. Graph matching is based on two distance matrices, describing the distance between the different topological relations and the different intervallic relations.

The layout indexing techniques mentioned above along with our proposed framework (Emulsion) are summarized in Table 1. The columns, respectively, refer to the representation type, the type of information that is encoded (directional relations, topological relations, shape similarity), the type of transformations that can be handled (translation, scaling, rotation) and the cases as defined in Section 1.2.1 that are handled by the method.

One of our main contributions is the combination of both layout and shape based indexing. Most previous works handle only the layout indexing of iconic images, where the total image consists of a combination of iconic building blocks chosen from a predetermined set. There is no need then for finding shape-based correspondences between image components in the query and database objects, since the building blocks are simply labeled with their appropriate class labels. However, it also means a strong restriction on the types of images such methods can handle. See Figure 2 for an example. The only method in our table that incorporates both layout and shape indexing besides the proposed method is ImageMap, but it's disadvantages have been discussed already. Moreover, since it is possible to specify a query at each level of the hierarchy, our framework handles all three cases as described in Section 1.2.1, as opposed to all the other methods in the table.

The proposed method exploits a flexible yet powerful shape indexing method to reduce the set of candidate matches, and while doing so finds correspondences between image components in the query and the database objects that can be used to evaluate layout similarity efficiently.

1.2.3 Multi-Level Indexing Framework

The overview of our multi-level indexing framework is presented in Figure 3. Each database image in the system is first represented as a collection of polygonal curves, shown in Part (a). The type of a polygonal curve can be either closed (polygon) or open (polyline). The details of the polygonal approximation method are outside the scope of this paper. As mentioned in Section 1.2.1, we address three types of indexing problems, one in each level of our frame-

Table 1: Comparing 8 layout indexing schemas by various features.

Method	Layout repr.	Dir.	Top.	Shape	Trans.	Scal.	Rot.	Cases handled
SOG [18]	graph	Y	N	iconic	Y	Y	Y	2,3
2D-strings [6]	string	Y	N	iconic	Y	Y	N	2,3
2D-G-String [7]	string	Y	N	iconic	Y	Y	N	2,3
2D-C-String [24]	string	Y	N	iconic	Y	Y	N	2,3
FRISS [15]	graph	Y	Y	iconic	Y	Y	Y	2,3
2D-PIR [28]	graph	Y	Y	iconic	Y	Y	N	2,3
ImageMap [30]	graph	Y	Y	Y	Y	Y	Y	2,3
Emulsion	graph	Y	Y	Y	Y	Y	?	1,2,3

work. Before defining our indexing algorithms, we present a method to select the canonical (or representative) elements for datasets at group and component levels. By showing only such elements at these levels, our goal is to achieve the speedup for building the indexing methods. Here, the speedup one actually achieves depends on the structural homogenity of the databases. Note that the representative elements shown in group and component levels form the input datasets for component and shape-primitive levels, respectively. Therefore, the overal fitness of the framework depends on how good these levels are constructed. The arrows between the levels carry important information, which will be used during the recognition stage. The details of the framework are given below.

The group level (Part b of Figure 3) is designed for the first type of an indexing problem, in which an overall layout similarity between image pairs is given a higher significance than the individual shape similarities. For images represented as polygonal curves, this part creates graphs whose vertices represent polygonal curves and whose edges represent spatial relations between the vertices. Two vertices are connected by an edge if the normalized geometric distance between them is less than a predetermined threshold. Alternatively, one may construct the graphs such that each vertex is connected to its $k-$nearest neighbors. Part (a) of Figure 4 shows two images with several polygonal curves. The graphs corresponding to these images are shown in Part (b). Once the graphs are constructed, the problem of finding similarity between two images becomes that of graph matching, as depicted in part c of the figure. Our first objective in this level is to select canonical graphs and then build the indexing approach on such graphs. Thus, we first need to match two graphs and compute a similarity score between them.

Matching two graphs means to establish their vertex correspondences. In particular, we are interested in graph matching algorithms, which computes the similarity between graphs using both vertex and edge attributes. In addition, the graph matching algorithm should handle possible segmentation and articulation errors, yielding many-to-many matchings between vertex sets. Although the first indexing problem does not take into account vertex attributes and only considers overal layout to find the similarity between two images, using both vertex and edge attributes is essential for selecting canonical elements. These elements will represent graphs that are close in terms of their both vertex and edge properties.

Let $G = (V, E)$ be a graph constructed for one image. For vertex $v \in V$, we let A_v denote the attribute vector associated with vertex v. The entries of each such vector represent the set of features $A_v = \{\text{scale, location, significance, orientation}\}$, where

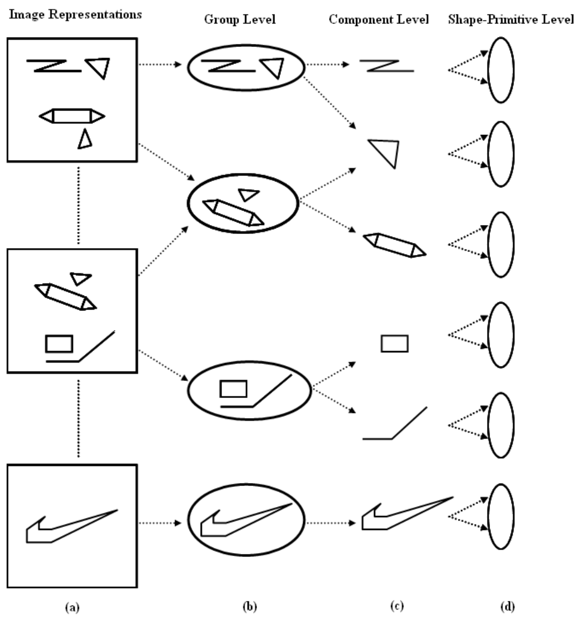


Figure 3: Overview of the shape indexing framework. Sample polygonal approximations for some database images are shown in part a. Part b shows the canonical groups computed by representing the groups as attributed graphs, using a graph matching algorithm to compute pairwise similarities, and applying a clustering algorithm to perform the final selection. Three types of indexing problems are addressed in parts b, c, and d. Relations between two consecutive-level entities are formed by parent/child edges with certain properties.

- scale: total length of the corresponding polygon's/polyline's line segments.
- location: location of the center of mass.
- significance: predetermined significance value for the polyline/polygone in the trademark image.
- orientation: the angle between the *major axis* and the horizantal line. Recall that the major axis of a polygon/polyline is the line joining two boundary points that are farthest away from each other.

Similarly, for every pair of vertices $u, v \in V$, if there is an edge $e = (u, v)$ between them, we let R_e denote the relation vector associated with edge e. The entries of this vector show the set of both geometrical and topological relations $R_e = \{$relative scale, geometric distance, relative orientation, edge strength$\}$. Here, the edge strength is defined for polygons only, and computed as the amount of overlapping area between them. Depending on the application domain, we can add more vertex and edge attributes. In any case, our proposed method can handle any set of attributes.

Armed with robust graph representations of each image, the problem of finding similarities between image pairs can now be transformed to that of many-to-many graph matching. We will use one such graph matching algorithm presented in [10]. The algorithm starts by representing the graphs as tree metrics and embedding tree metrics into a high-dimensional vector space with small distortion. Each vertex in the tree is represented as a point in the

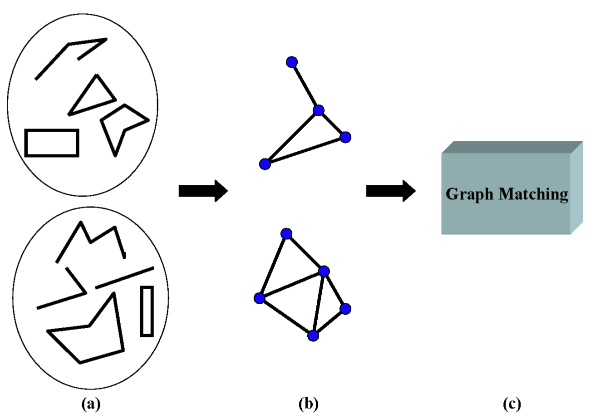


Figure 4: Formulating the problem of image matching as that of graph matching. Images with several polygonal curves shown in Part (a) are represented by attributed graphs whose vertices represent polygonal curves and whose edges represent spatial relations between the vertices (Part (b)). A graph matching algorithm is then used to perform the matching in Part (c).

high-dimensional space. Associated with each point is a set of histograms that encode the node's attributes as well as it's adjacent edges. Matching point distributions via the Earth Mover's Distance under transformation [33, 9] yields desired many-to-many matchings. The overal framework also returns a distance measure between the original graph pairs. Matching two graphs many-to-many might seem like a bad idea for the layout indexing problem since the overal layout of a graph may seem to be different from that of its representative. However, only graphs that are close to each other in the form of both their low level node features and their abstractions will share a representative. One important advantage for choosing such a graph matching algorithm is also to handle possible segmentation and articulation errors.

Now that a suitable graph matching algorithm has been selected, we can proceed with the selection of the canonical graphs to appear at the first level of our structure. Given a set of graphs $G = \{g_1, \ldots, g_n\}$ and a distance function $d_g : G \times G \rightarrow \mathbb{R} \geq 0$, our goal is to select a subset of graphs $G' \subset G$ that best describes the elements of G with respect to the distance function d_g. While we want to increase the total similarity score between the elements of G' and $G - G'$, we also want elements of G' to be as dissimilar to each other as possible. We use one such selection algorithm presented in [12, 11]. The algorithm expresses the problem as that of quadratic optimization integer programming and presents its relazation through semidefinite programming. Once the solution for the semidefinite program is found, the algorithm uses a rounding scheme to perform the selection. The reader is encouranged to look at the references for details. Since the pairwise graph distances are computed under similarity transformations, we store both the distance and transformation parameters in the edges between the database graphs and their representative sets, i.e., the edges between Part (b) and Part (c) of Figure 3.

After showing the canonical elements at the group level of our framework, we proceed with describing our indexing algorithms. Recall that we address three different indexing algorithms, one for each problem defined in Section 1: layout, shape and partial-shape indexing problems. Our layout indexing technique is presented in the first appendix. The second type of the indexing problem (shape indexing) is addressed in the component level. This indexing problem concerned with efficiently retrieving images that are globally similar to the query. Since we use polygonal representations of images, we capture general features of polygonal curves for shape description. Then we match two polygonal curves using their descriptions. Here, we are interested in a matching algorithm that optimally matches two complete components under similarity transformation, i.e., translation, rotation, and scaling. The similarity scores between component pairs are used to identify the subset of components that closely resembles the original set. As we did in the group level, we build our second indexing approach on this representative subset. Finally, in the last class of indexing problem we address in this paper (partial indexing), a partial similarity algorithm is used to filter out database images, which do not contain a similar part with the query. The shape-primitive level of the framework is designed to handle this type of indexing problem. In order to determine such similarity, we first need to decompose each polygonal curve shown in the second level of the framework into its shape-primitives. Obtaining such a decomposition in the literature is often referred to as 2D shape decomposition or shape-partition [29, 26]. Decomposition criteria can be defined in a number of ways. Medial-axis transform [25], boundary point clustering [36], collinearity [23], fuzzy subset theory [41], graph theoretic clustering [37], morphological method [32, 34], just to name a few. In our framework we require that the polygonal curves are decomposed into *natural* parts, corresponding to the human intuition. It has been shown that such a decomposition can be gained at the negative minima of curvature [21, 3]. A set of shape-primitives $P = \{p_1, \ldots, p_t\}$ is a decomposition of component C, if their union is C and all P_i are disjoint. More formally, P must satisfy:

$C = \{P | \cup_{i=1}^{t} p_i = C \text{ and } \forall_{i \neq j} p_i \cap p_j = \emptyset\}$.

Although by decomposing polygonal curves into their shape-primitives, the partial image matching problem is transformed into that of shape-primitive matching, performing the actual matching procedure between each shape-primitive appearing at shape-primitive level and each of the query shape-primitives is inefficient. We overcome this problem by employing the vantage indexing approach [38]. The first step in the vantage indexing is to compute distances between the database and m predetermined vantage objects. This step represents the database objects as a set of points in the $m-$dimensional vantage space, one point for each database entry. If the distance function obeys identity, positiveness, and triangle inequality then it is guaranteed that there will be no false negatives, i.e. recall is always 100%. Turning back to our multi-level structure, we must first select m vantage objects $A^* = \{A_1^*, \ldots, A_m^*\}$ among the shape primitives. Next, we need to compute the distance between each vantage object and the query shape-primitive under some distance measure that satisfies the properties mentioned above. This process creates a point for each query shape-primitive $p_i = (x_1, \ldots, x_m)$, such that $x_j = \delta(p_i, A_j^*)$, where δ is the distance measure defined on shape-primitives After computing the distances between query shape-primitives and the vantage objects, we perform a range search in the $m-$dimensional space or retrieve the $k-$nearest neighbors for nearest neighbor query. This algorithm returns a set of neighbors for each shape-primitive of the query. As we did in the previous level of our framework, we sort the database objects by the number of times they appear in these sets, which allows us to drop ambiguous database objects from further consideration.

Although this algorithm results in no false negatives, there are still false positives possible so precision is not necessarily high. The retrieval performance is influenced by the choice of vantage objects; with a good set of vantage objects precision values increase significantly. We use our method called Spacing-Based selection to find good vantage objects. This method avoids the selection of strongly correlated vantage objects and guarantees a dispersal (up to a

Table 2: Layout features

Relation	Representation
direction	scalar: angle between connecting line and horizontal line
distance	scalar: distance between c.o.m.
disjoint	boolean
touch	scalar: total length of touching edges
overlap	scalar: area of overlap(-1 for polylines)
covers	boolean
covered by	boolean
contains	boolean
inside	boolean
equal	boolean

certain predefined threshold) of the database objects in vantage space. This method was also under development while this report was written. We will finish developing the Spacing-Based selection method and use it in this structure.

1.2.4 Layout Similarity

Suppose the user specifies a query at either level 2 (component level) or level 3 (shape primitive level) of the Multi-Level Shape Indexing structure. The hierarchy is then used to find a reduced set of candidate database objects for the query based on shape similarity, by following the parent-child edges that connect the different layers of the hierarchy. However, no layout information is taken into account while producing this hypothesis. Therefore, we propose an additional step to rank or reduce this hypothesis, based on the layout information. Since the Multi-Level Shape Indexing structure hands us all correspondences between query components and candidate match components, we can evaluate layout similarity between the query and the candidate match efficiently. The same holds at the shape primitive level; we know the correspondences between query shape primitives and component shape primitives as well. In this section we describe in detail how the layout similarity between a query and a candidate image is evaluated.

In an offline preprocessing step, the complete layout information for each database image is computed. This results in a completely connected directed graph for each image, with n vertices and n^2 edges, where n is the number of components in the image. A vector with multiple features is associated with each edge (i, j), describing the layout relation between vertices i and j. The layout features are both topological and directional, see Table 2. The directional relation between two components is encoded as the angle between the line connecting the two centers of mass and the horizontal line. For the topological information however, we distinguish between eight different relations (see Figure 5): disjoint, touch, overlap, covered by/covers, contains/inside and equal. Note that overlap is undefined for polygons. Some topological relations are represented by a binary attribute, others are represented by a quantifier.

Note that these feature vectors can easily be extended to account for even more types of layout information, or be reduced to speed-up the search process. This is all dependent on the application.

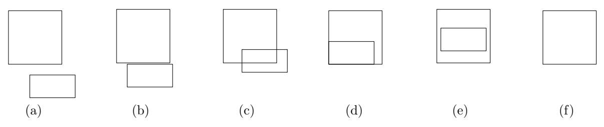


Figure 5: Different topological relations. (a) disjoint, (b) touch, (c), overlap, (d) covers/covers by, (e) contains/inside, (f) equal.

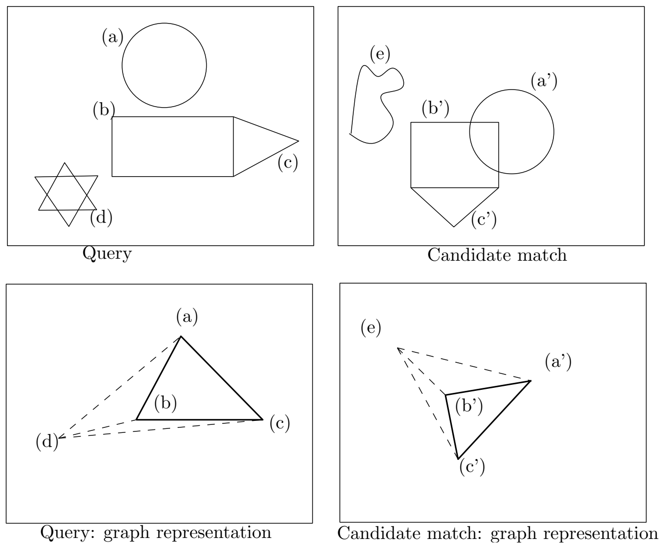


Figure 6: Evaluating layout similarity: layout information (topological and directional) is stored in the edges of the graph representation. Correspondences between components in query and candidate match are given by the Shape Indexing Framework. Only compare corresponding edges, i.e. (a, b) with (a', b'), (b, c) with (b', c') and (a, c) with (a', c')

1.2.5 Evaluating layout similarity

Given a query q, the Multi-Level Shape Indexing structure returns a set K containing all the candidate matches, which are sufficiently similar to the query in terms of shape similarity of the components. For each candidate match $k \epsilon K$, the Shape Indexing structure hands us all the correspondences between the query components and the components in k. We use these correspondences to evaluate layout similarity efficiently: we only evaluate layout similarity between corresponding edges. See Figure 6 for an example. Note however that the actual layout information is already present in the database, only the entries in the layout matrices are compared.

Based on the comparison of corresponding edges the candidate matches are ranked to produce the final hypothesis. The layout similarity between a query q and a candidate match k is based on the layout similarities between their corresponding edges. The total layout similarity between q and k is defined as

$$\mathrm{SIM}_{\mathrm{layout}}(q, k) = 1 - (1/c \sum_{i=1}^{c} d_{\mathrm{layout}}(e_i, e'_i))$$

where c is the number of corresponding edge pairs, e_i is an edge in q and e'_i is its corresponding

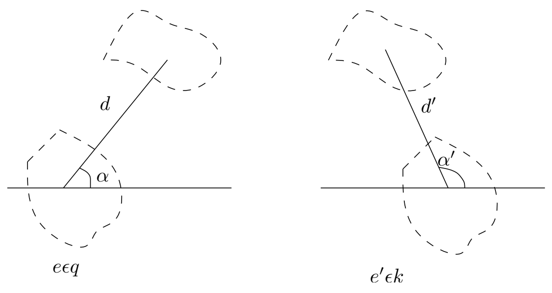


Figure 7: Illustrating the difference in directional relation between two corresponding edges. On the left, query edge e and on the right edge e' from a candidate match. The angles α and α' are found by rotating counter clockwise from the horizontal line to the lines connecting the centers of mass.

edge in k. The layout similarity between two corresponding edges $e\epsilon q$ and $e'\epsilon k$ is

$$d_{\text{layout}}(e, e') = w(d_{\text{dir}}(e, e') + (1 - w)d_{\text{top}}(e, e'))$$

where $d_{\text{dir}}(e, e')$ denotes the difference in directional relation between e and e', $d_{\text{top}}(e, e')$ denotes the difference in topological relation between e and e' and w denotes a weighting factor ($0 \leq w \leq 1$) to be specified by the user.

The directional difference, $d_{\text{dir}}(e, e')$, is defined as

$$d_{\text{dir}}(e, e') = |\cos\,\alpha\; d - \cos\,\alpha'\; d'|$$

where α is the angle associated with e, α' is the angle associated with e', d is the distance associated with e, and d' the distance associated with e'. The angles α and α' are found by rotating counter clockwise from the horizontal line to the lines connecting the centers of mass. For example, see Figure 7

The topological difference, $d_{\text{top}}(e, e')$, is based on the other eight features in the feature vector: the topological features. It is defined as

$$d_{\text{top}}(e, e') = 1/8 \sum_{i=1}^{8} d(f_i, f_i')$$

where f is the subvector containing the eight topological features associated with e and f' is this 8-dimensional subvector associated with e'. For topological relations represented by a boolean, $d(f_i, f_i')$ is defined as follows

$$d(f_i, f_i') = \begin{cases} 1 & \text{if } f_i = f_i', \\ 0 & \text{otherwise} \end{cases}$$

For topological relations represented by a scalar, $d(f_i, f_i')$ is defined as the extent to which the relation is preserved, i.e.

$$d(f_i, f_i') = \begin{cases} f_i/f_i' & \text{if } f_i \leq f_i' \\ f_i'/f_i & \textit{otherwise} \end{cases}$$

1.2.6 Query Handling Procedure

The structure of our framework allows users to start their query in any level. The selection of the level, in which, the query takes place is application-dependent. If, for instance, the user is interested in the images, which have the same layout as the query, as well as those, which have partial similarity to the query, then applying level 1 and level 3 indexing algorithms generates two hypothesis, one for the layout and one for the partial shape indexing. The intersection of these two hypothesis consists of images that contain partially similar query shapes, whose overall layouts are close to that of query. Note that for level 2 and level 3 shape indexing approaches, the framework allows the user to specify additional constrains besides the individual shape similarities. For example, the user may restrict the rotation angle computed for one particular shape to be at most α degree different from the other shapes of the query. Or, the scale factors of query shapes may be restricted to be the same as each other etc.

As noted before, the objective of this paper is to give the readers an overal idea on our multi-level shape and layout indexing framework. We are in the process of developing and evaluating this framework for trademark retrieval. We will add the algorithms proposed within the different work packages of Profi project such as shape matching and perceptual grouping to our framework.

1.3 Results

Although our technique is designed especially for layout indexing, the first experimental evaluation of the framework was performed using MPEG-7 dataset where each shape is represented as a shock graph. The results show that for a given query, the database can be pruned over 90 for detecting the right class and 99 to find all images belonging to the query class. Our future work for WP 7 include further exploration of our graph-based indexing method, creating a ground-truth for layout indexing, and performing experiments on this set. For more detailed presentation of results, see the accompanying paper in the appendix.

The paper in the Appendix is under review for CVIU Special Issue on “Similarity Matching in Computer Vision and Multimedia”. We are also in the process of prepearing one other layout indexing paper to be submitted to ACM International Conference on Image and Video Retrieval (CIVR 2007).

References

[1] N. Beckmann, H.P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust acces method for points and rectangles. In *Proceedings of ACM-SIGMOD*, pages 322–331, 1990.

[2] J.L. Bentley. Binary search trees used for associative searching. *Communications of the ACM*, 18(9):507–519, 1975.

[3] I. Biederman. Recognition–by–components: A theory of human image understanding. *Psychological Review*, 94:115–147, 1987.

[4] T. Bozkaya and M. Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, 1997.

[5] T. Bozkaya and M. Ozsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Trans. Database Syst.*, 24(3), 1999.

[6] S. K. Chang, Q. Y. Shi, and C. W. Yan. Iconic indexing by 2-d strings. *IEEE Transansctions on Pattern Analysis and Machiche Intelligence*, 9(3):413–428, 1987.

[7] S.K. Chang, E. Jungert, and Y. Li. Representation and retrieval of symbolic pictures using generalized 2D strings. In *SPIE Conference on Visual Communications and Image Processing*, volume 3, pages 1360–1372, November 1989.

[8] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of Very Large Data Bases conference*, 1997.

[9] S. D. Cohen and L. J. Guibas. The earth mover's distance under transformation sets. In *Proceedings, 7th International Conference on Computer Vision*, pages 1076–1083, Kerkyra, Greece, 1999.

[10] M. Fatih Demirci, Ali Shokoufandeh, Yakov Keselman, Lars Bretzner, and Sven Dickinson. Object recognition as many-to-many feature matching. *International Journal of Comput. Vision*, 69(2):203–222, 2006.

[11] T. Denton, J. Abrahamson, and A. Shokoufandeh. Approximation of canonical sets and their applications to 2d view simplification. In *CVPR (2)*, pages 550–557, 2004.

[12] T. Denton, M. F. Demirci, J. Abrahamson, A. Shokoufandeh, and S. J. Dickinson. Selecting canonical views for view-based 3-d object recognition. In *ICPR (2)*, pages 273–276, 2004.

[13] E.Chavez and G. Navarro. Searching in metric spaces. *ACM Computer Surveys*, 33(3):273–321, September 2001.

[14] M.J. Egenhofer and R.D. Franzosa. Point Set Topological Relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[15] E.A. El-Kwae and M.R. Kabuka. A Robust Framework for Content-Based Retrieval by Spatial Similarity in Image Databases. *ACM Transactions on Information Systems*, 17(2):174–198, April 1999.

[16] C. Faloutsos and K. Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Michael J. Carey and Donovan A. Schneider, editors, *Prooceedings of ACM SIGMOD '95*, pages 163–174, San Jose, California, 22–25 1995.

[17] V. Gaede and O. G&#252;nther. Multidimensional access methods. *ACM Comput. Surv.*, 30(2):170–231, 1998.

[18] V. N. Gudivada and V. V. Raghavan. Design and Evaluation of Algorithms for Image Retrieval by Spatial Similarity. *ACM Transactions on Information Systems*, 13(2):115–144, April 1995.

[19] A. Gutman. R-trees: A dynamic index structure for spatial searching. In *Proceedings of ACM-SIGMOD*, pages 47–54, June 1984.

[20] G.R. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. In *IEEE Transactions on pattern analysis and machine intelligence*, volume 25, may 2003.

[21] D. D. Hoffman and W. Richards. Parts of recognition. Technical Report AIM-732, 1983.

[22] G. Hristescu and M. Farach-Colton. Cluster-preserving embedding of proteins. Technical Report 99-50, 8, 1999.

[23] H.S. Kim, K.H. Park, and M. Kim. Shape decomposition by collinearity. *Pattern Recognition Letters*, 6:335–340, 1987.

[24] S.Y. Lee and F.J. Hsu. 2d c-string: a new spatial knowledge representation for image database systems. *Pattern Recognition*, 23(10):1077–1087, 1990.

[25] J. Lien and N. Amato. Simultaneous shape decomposition and skeletonization. Technical Report TR-05-015, Parasol Laboratory, Department of Computer Science, Texas A&M University, December 2005.

[26] S. Loncaric. A survey of shape analysis techniques. *Pattern Recognition*, 31(8):983–1001, 1998.

[27] B. M. Mehtre, M. S. Kankanhalli, and W. F. Lee. Shape measures for content based image retrieval: a comparison. *Inf. Process. Manage.*, 33(3):319–337, 1997.

[28] M. Nabil, A.H.H. Ngu, and J. Shepherd. Picture Similarity Retrieval Using the 2D Projection Interval Representation. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):533 – 539, 1996.

[29] T. Pavlidis. Algorithms for shape analysis of contours and waveforms. In *IEEE Transactions on pattern analysis and machine intelligence*, volume 2, pages 301–312, 1980.

[30] E.G.M. Petrakis, C. Faloutsos, and K.-I Lin. Imagemap: an image indexing method based on spatial similarity. In *IEEE Transactions on Knowledge and Data Engineering*, volume 14, pages 979– 987, 2002.

[31] E.G.M. Petrakis and S.C. Orphanoudakis. A Methology for the Representation, Indexing, and Retrieval of Images by Content. *Image and Vision Computing*, 8(11):504–512, October 1993.

[32] I. Pitas and A.N. Venetsanopoulos. Morphological shape decomposition. *IEEE Transactions on pattern analysis and machine intelligence*, 12(1):38–45, January 1990.

[33] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[34] D. Schonfeld and J. Goutsias. Optimal morphological filters for pattern restoration. In *SPIE*, volume 1199, pages 158–169, 1989.

[35] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The r -tree: A dynamic index for multi-dimensional objects. In *The VLDB Journal*, pages 507–518, 1987.

[36] L. G. Shapiro. A structural model of shape. In *IEEE Transactions on pattern analysis and machine intelligence*, volume 2, pages 111–126, 1980.

[37] L.G. Shapiro and R.M. Haralick. Decomposition of two-dimensional shapes by graph-theoretic clustering. *IEEE Transactions on pattern analysis and machine intelligence*, 1(1):10–20, January 1979.

[38] J. Vleugels and R. C. Veltkamp. Efficient image retrieval through vantage objects. *Pattern Recognition*, 35(1):69–80, 2002.

[39] X. Wang, J. Wang, K. Lin, D. Shasha, B. A. Shapiro, and K. Zhang. An index structure for data mining and clustering. *Knowledge and Information Systems*, 2(2):161–184, 2000.

[40] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA '93: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*, pages 311–321, 1993.

[41] L.A. Zadeh. Fuzzy sets. *Journal of Information and Control*, 8:338–353, 1965.

2 Deviations from plan

In the original plan, the due date for this Deliverable 7.1 was M12. This however was swapped with Deliberable 6.1, and the new due date for D7.1 is M21. This has been approved by the project officer.

3 Appendix

This appendix describes our indexing method for graph structures. This paper is under review for CVIU Special Issue on "Similarity Matching in Computer Vision and Multimedia".

Indexing through Laplacian Spectra

M. Fatih Demirci, Reinier H. van Leuken, Remco C. Veltkamp

{*mdemirci, renier, remco.veltkamp*} @ *cs.uu.nl*

Institute for Information and Computing Sciences,

Utrecht University, The Netherlands

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, where database 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. 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 2-D and 3-D 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.

Key words: 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 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, such as pattern recognition [26,25], chemical structures [33], or computer vision [45,23]. 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 the (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 [36] proposed a method to organize similar graphs in clusters where each cluster is indexed by a representative graph. Sossa and Horaud [42] 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 [28] 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. [38]. 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.

One of the primary aspects of graph theory is to derive the principal properties and structure of graphs from their graph spectra. It is well known that eigenvalues are closely related to main invariants of a graph. 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 [35] 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 [37] used eigenvectors of proximity graphs to compute the feature correspondences. Turk and Pentland [44] 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 [34] 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 [43], 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 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 [29]. The definition of a laplacian matrix along with other graph-theoretical concepts used in this paper are 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 we pay 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 [28]. 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. To overcome this, we use recently-developed techniques in the domain of spectral integral variation. Specifically, given the laplacian spectrum of graph G, we will explore an efficient method to generate the laplacian spectrum of graph $G + e$, where $G + e$ is a graph obtained by adding edge e to graph G. To our knowledge, the proposed framework is the first framework that uses spectral integral variation in an indexing algorithm.

This 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 [28], 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 [28]. 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 in Section 2, we describe our indexing mechanism and its complexity in Section 3. To avoid computing the signature of each subgraph individually, 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 Notation and Definitions

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

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

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 [30,31,27,29] 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

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

adjacency matrix, in terms of resulting in less number of 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 Winson [48]. 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 main motivation for constructing the graph characterizations using the laplacian spectrum comes from these studies.

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. 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. See the papers [14,32,30,31,27,29] for details about the relations between the laplacian spectrum and the graph structure.

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:*

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

[2] This theorem is obtained by Courant-Weyl([11], Theorem 2.1). The reader may also refer to [18].

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 [28].

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 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

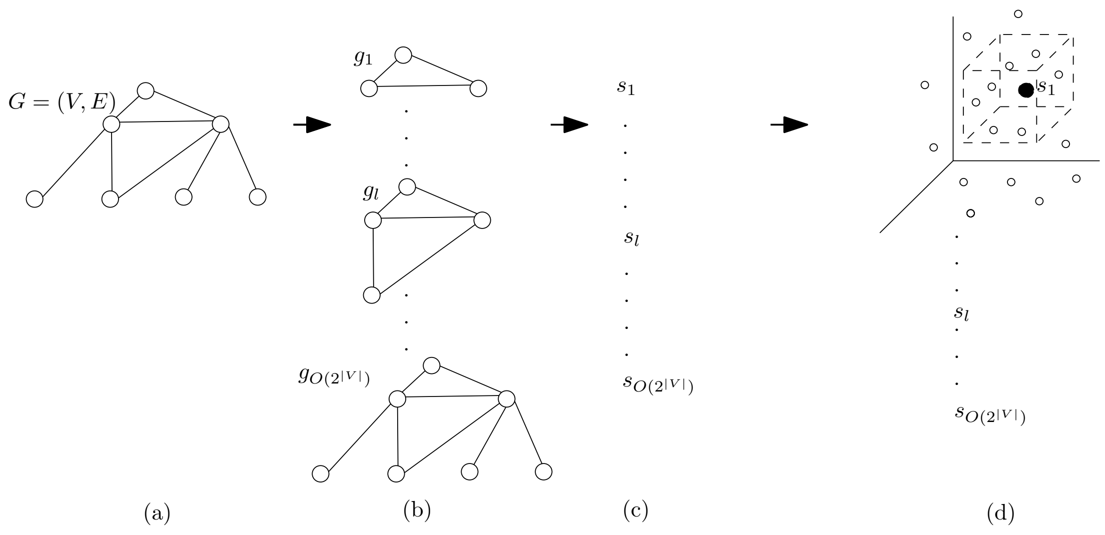


Fig. 1. Retrieving similar graphs. For graphs given in Part (a), its subgraphs are constructed in Part (b). A signature is computed for each subgraph in Part (c). Given a signature, retrieving its similar graphs from a large database is formulated as a nearest neighbor search as shown in Part (d).

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:

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

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 a certain size k, the weight of the votes based on S_q^k and S_d^k is computed using the directed Hausdorff distance 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 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 k is $O\left(\binom{n_d}{k}\right)$. Assume that there are m graphs in the database, the total number of signatures generated by the framework is bounded by

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

Notice, however, that during the signature generation for database graphs, the actual number of subgraphs for which we compute signatures is strictly less than $m \times \sum_{k=0}^{n_d} \binom{n_d}{k}$, 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. 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 [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 factor of $(1+\epsilon)$ of the distance to the true nearest neighbor. In general, given an integer $k \geq 1$, $(1+\epsilon)$-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.

Thus, for a query subgraph of size n_q, the total running time T_{n_q} of a k-nearest neighbor search using is

$$T_{n_q} = O\left(kn_q \log\left(m \times \binom{n_d}{n_q}\right)\right).$$

4 More Efficient Indexing through Spectral Integral Variation

The local indexing procedure described above requires individual computation of the laplacian spectrum for each subgraph. 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 them. 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 [41] 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.

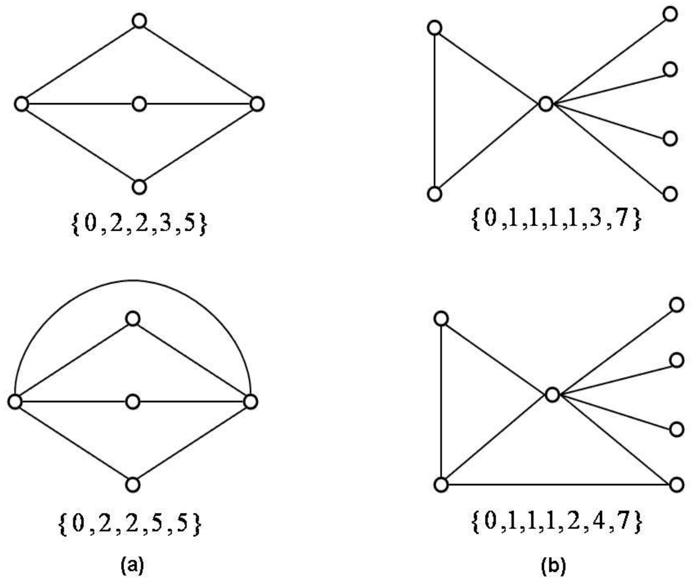


Fig. 2. Spectral integral variation in one and two places are shown in Part (a) and (b), respectively. Bottom graphs are formed by adding one edge to the graphs shown at the top. The laplacian spectrum is written below each graph. Observe that while only one eigenvalue increases by 2 in part (a), two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

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 cases:*

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

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

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

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

The proof for this lemma is given by Yizheng [46]. Part (a) and (b) of Figure 2 show two graphs where adding an edge results in spectral integral variation in one and two places, respectively.

In our framework, we will identify the changed eigenvalue(s) when spectral integral variation occurs. This, in turn, will allow us to generate the laplacian spectrum of $G + e$ given that of G. 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 [24,5]. 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 laplacian eigenvalue(s) when spectral integral variation occurs in one and two places. Theorem 2 appears in [41], while Theorem 3 is shown in [24].

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 $d_u \geq d_v$. Furthermore, let $1_x, 0_x$ denote the $x \times 1$ matrices whose entries are all 1,0, respectively, and let ${1^t}_x, {0^t}_x$ denote their transposes.

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

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

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:

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

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

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

$$L_{41}1_x - L_{42}1_x = 0. \tag{8}$$

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

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

and

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

After identifying the 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 given above. We then repeat this process and consider all distinct 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 Figure 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 6.47%. We will present our report on this part in Section 5.

In retrospect, our encoding of a graph's structure captures its local topology, thus

```
Input: $G = (V, E)$, $\hat{G} = (V, \hat{E})$, and $\Gamma_{(\hat{G})}$
Output: The laplacian spectrum $\Gamma_{(\hat{G})}$ for every distinct 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 distinct 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 then
            generate $\Gamma_{(\hat{G})}$ according to Theorem 2
        else if spectral integral variation occurs in two places then
            generate $\Gamma_{(\hat{G})}$ according to Theorem 3
        else compute $\Gamma_{(\hat{G})}$
    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.

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 2-D and 3-D 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 [40]. 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 Figure 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 Figure 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

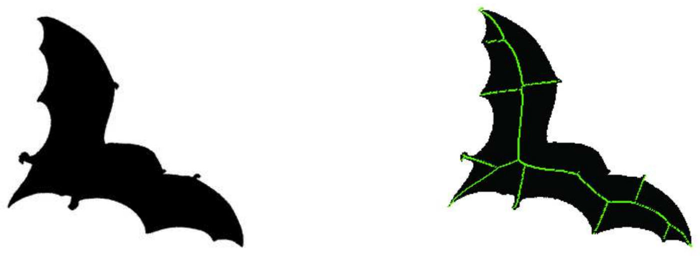


Fig. 4. Left: a view of a bat. Right: the shock graph constructed from the medial axis and superimposed on the left image.

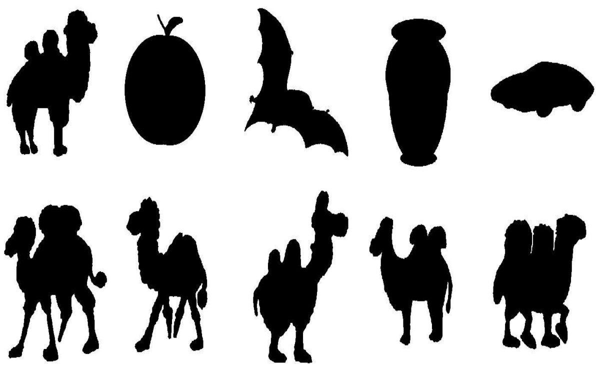


Fig. 5. Sample images of the MPEG-7 database are shown in the top row. The bottom row represents a set of different shapes of a particular class from the database.

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 $\mathrm{f}(\alpha) = \mathrm{f}(\gamma)$ 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 Figure 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 [47]. The database consists of 420 objects

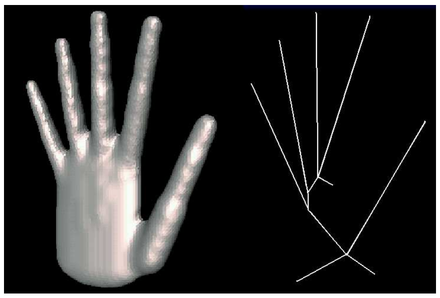


Fig. 6. The Reeb graph constructed for the object on the left is shown on the right.

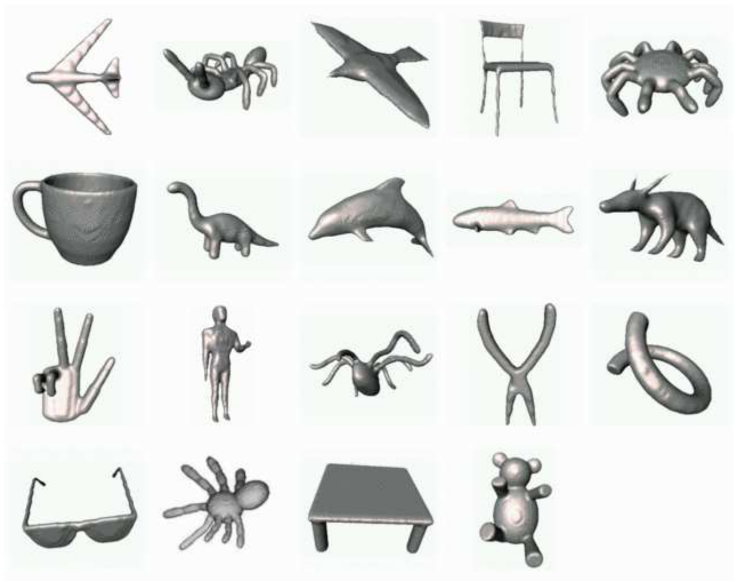


Fig. 7. Views of sample objects from the McGill 3D Shape Benchmark.

classified in 19 classes. Figure 7 shows representative views of objects from the database.

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 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 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. In Figure 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 in-

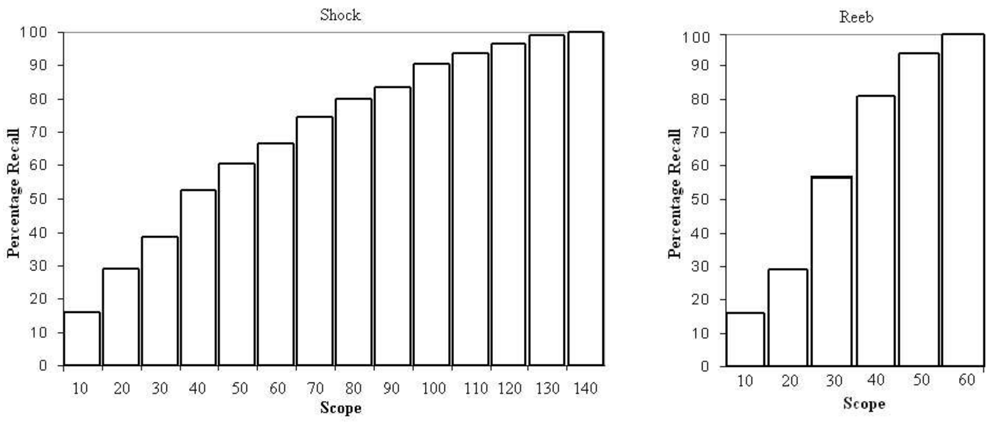


Fig. 8. Percentage recall values for various top ranked highest-weight candidate graphs for shock graphs of MPEG-7 and Reeb graphs of McGill datasets.

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	18.02%	19.1	23	257	97%	82%
REEB GRAPHS - ADJACENCY	17.3%	10.2	22	92	94%	78%

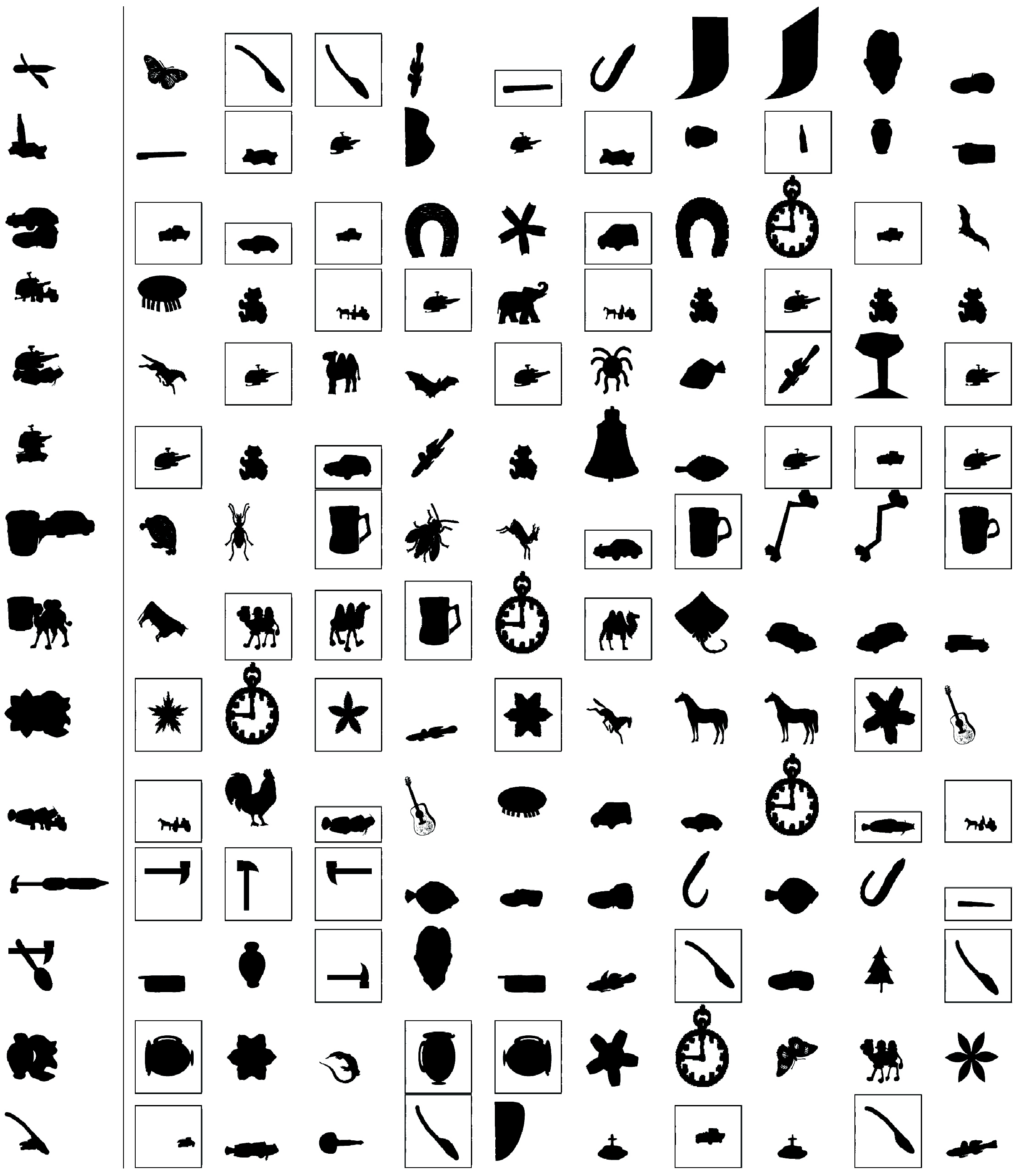
Table 1
Results for indexing system constructed using eigenvalues for Laplacian and Adjacency matrices. HW: Percentage of highest-weighted graph belonging to the same class as the query, AP: Average position of the closest matching graph from the query class, WP: Worst position of the closest matching graph from the query class, 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.

dexing framework to the one presented in [38]. This indexing algorithm was used in [47] on a subset of the McGill dataset with the object parts represented by directed acyclic graphs (DAG) through medial surfaces [39]. 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 [47], we also merge the categories “four-legged” and “dinosaurs” into a broader class, “four-limbs”. The results reported in [47] 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 minutes 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 3.00GHz computer. After involving spectral integral variation in the framework, the time we spent in this process decreases to 45.5 and 108.3 minutes, respectively. These results indicate that 1.8% and 6.47% time improvements are gained with spectral integral variation for these two datasets. One would expect that as the size of the database increases, the framework with spectral integral variation would yield an improved time efficiency. While the database size is an important factor, the number of laplacian integral graphs (graphs whose laplacian spectrum consists entirely of integers) in the database also effects the overall efficiency of the system using spectral integral variation. Balińska et al. [4] present an important survey on integral (graphs whose adjacency spectrum consists entirely of integers) and laplacian integral graphs. 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 14 occluded scenes, each with two images selected from different classes in the MPEG-7 dataset. In each scene, one shape occludes the other to a certain extent. The percentage of the occlusion varies from 2% to 16%, with on average 6.0%. Each of these 14 new scenes was used as a query against the complete database. We define our indexing schema to be effective if one of the shapes of the query appears in the highest vote-weight candidates. The results of this experiment is presented in Figure 9, where 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 in this experiment. 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, 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 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



1

Fig. 9. Results of occlusion experiments. 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.

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 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.

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 successfully 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 performed 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.

7 Acknowledgements

The authors acknowledge the support provided by FP6 IST projects 511572-2 PROFI and 506766 AIM@SHAPE. The authors would also like to thank Simone Marini for his Reeb graph representations and Esther Moet and Dennis Nieuwenhuisen for careful proofreading.

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

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

[4] K. Balińska, D. Cvetković, Z. Radosavljević, S. Simić, and D. Stevanović. A survey on integral graphs. *Univ. Beograd. Publ. Elektrotehn. Fak. Ser. Mat.*, 13:42–65, 2002.

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

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

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

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

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

[10] M.S. Costa and L.G. Shapiro. Relational indexing. In *SSPR*, pages 130–139, 1996.

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

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

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

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

[15] 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.

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

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

[18] 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.

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

[20] M. Hilaga, Y. Shinagawa, T. Kohmura, and 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*, pages 203–212, New York, NY, USA, 2001. ACM Press.

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

[22] J. E. Hopcroft and 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*, pages 172–184, New York, NY, USA, 1974. ACM Press.

[23] R. Horaud and T. Skordas. Structural matching for stereo vision. In *Nineth International Conference on Pattern Recognition*, pages 439–445, 1988.

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

[25] S. W. Lee and J. H. Kim. Attributed stroke graph matching for seal imprint verification. *Pattern Recognition Letters*, 9:137–145, 1989.

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

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

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

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

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

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

[32] S. Pati. The third smallest eigenvalue of the laplacian matrix. *Electronic Journal of Linear Algebra*, 8:128–139, August 2001.

[33] D.H. Rouvray and A.T. Balaban. Chemical applications of graph theory. In *Applications of Graph Theory*, pages 177–221, New York, 1979.

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

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

[36] L. G. Shapiro and R. M. Haralick. Organization of relational models for scene analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 4(11):595–602, November 1982.

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

[38] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S.W. Zucker. Indexing hierarchical structures using graph spectra. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(7), 2005.

[39] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker. The hamilton-jacobi skeletons. *International Journal of Commputer Vision*, 48(3):215–231, 2002.

[40] K. Siddiqi, A. Shokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *International Journal of Computer Vision*, 30:1–24, 1999.

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

[42] H. Sossa and R. Horaud. Model indexing: The graph-hashing approach. In *Proc. International Conference on Computer Vision and Pattern Recognition*, pages 811–814, 1992.

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

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

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

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

[47] 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 *International Workshop On Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 285–300, 2005.

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