Layout Indexing of Trademark Images

Reinier H. van Leuken
Computer Science
Universiteit Utrecht
reinier@cs.uu.nl

M. Fatih Demirci
Computer Science
Universiteit Utrecht
mdemirci@cs.uu.nl

Victoria J. Hodge
Dept. of Computer Science
University of York
vicky@cs.york.ac.uk

Jim Austin
Dept. of Computer Science
University of York
austin@cs.york.ac.uk

Remco C. Veltkamp
Computer Science
Universiteit Utrecht
remco.veltkamp@cs.uu.nl

ABSTRACT

Ensuring the uniqueness of trademark images and protecting their identities are the most important objectives for the trademark registration process. To prevent trademark infringement, each new trademark must be compared to a database of existing trademarks. Given a newly designed trademark image, trademark retrieval systems are not only concerned with finding images with similar shapes but also locating images with similar layouts. Performing a linear-search, i.e., computing the similarity between the query and each database entry and selecting the closest one, is inefficient for large database systems. An effective and efficient indexing mechanism is, therefore, essential to select a small collection of candidates. This paper proposes a framework in which a graph-based indexing schema will be applied to facilitate efficient trademark retrieval based on spatial relations between image components, regardless of mutual shape similarity.

Our framework starts by segmenting trademark images into distinct shapes using a shape identification algorithm. Identified shapes are then encoded automatically into an attributed graph whose vertices represent shapes and whose edges show spatial relations (both directional and topological) between the shapes. Using a graph-based indexing schema, the topological structure of the 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. Having established the signatures, the indexing amounts to a nearest neighbour search in a model database. For a query graph and a large graph data set, the indexing problem is reformulated as that of fast selection of candidate graphs whose signatures are close to the query signature in the vector space. An extensive set of recognition trials, including a comparison with manually constructed graphs, show the efficacy of both the automatic graph construction process and the indexing schema.
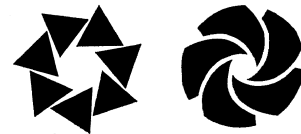
Figure 1: Two trademarks resemble each other based on the layout of their shapes despite the dissimilarity between their individual component shapes.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Content-based Image Retrieval, Trademark Retrieval, Indexing, Laplacian spectrum

1. INTRODUCTION

One of the highly active research areas within the broad field of shape matching and Content-based Image Retrieval (CBIR) is trademark retrieval. Trademarks (or, logos) [1] come in different forms, with varying kinds of unique properties. Textual information, shape, layout, and in some cases colour are probably the most important ones. Ensuring the uniqueness of trademarks and protecting their identities are the most important objectives for the trademark registration process. To prevent trademark infringement, each new trademark must be compared to the database of existing trademarks. Traditionally, this process is done by assigning keywords to shapes using predetermined vocabulary such as the Vienna classification and searching trademarks based on the keywords [21]. Since these kinds of methods involve heavy human interference, automatic trademark retrieval is of great importance.

Given a query image, most automatic trademark retrieval systems aim to find images with similar shapes without taking into account the spatial layout of the shapes. Although retrieving images containing similar shapes may seem as the

---

[1]defined by the UK patent office as *a sign which can distinguish the goods and services of one trader from those of another, and be represented graphically*
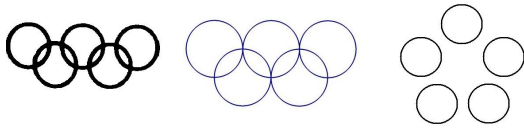
Figure 2: Three different configurations of 5 circles. Suppose the leftmost image, the Olympic logo, is used as a query. Because of a similarity in layout, the middle image should receive a higher vote than the rightmost image, despite the fact that pure shape similarity on components is the same.

Figure 3: A sample trademark image.

Figure 4: Possible shapes identified in figure 3.

primary goal, there are many cases where the layout similarity plays a more important role for ensuring uniqueness. An example of this scenario is given in Figure 1 in which the layout of the shapes reveals a strong figure in itself. The two trademarks resemble each other despite the dissimilarity between their individual shapes. In case these two trademarks are to be registered in the same or in a closely related product group or service category, a conflict of uniqueness arises.

Layout similarity between trademarks is also used to improve the quality of matching based on shape similarities. Consider Figure 2, where two candidates are returned with the same similarity scores against a given query. Although they both contain the same shapes, the middle candidate should be assigned a stronger similarity value since its shapes are in a configuration similar to that of the query. Hence, one may observe that applying layout similarity improves the overall quality of a trademark retrieval system.

The work presented in this paper proposes a framework in which a graph-based indexing schema will be applied to facilitate efficient trademark retrieval based on spatial relations between image shapes regardless of their mutual shape similarities. Our framework begins by segmenting trademark images into distinct shapes using a closed shape identification algorithm. A simple edge detector would not be sufficient as many images in our test set are noisy and this noise causes small gaps in the shape boundaries so these gaps need to be closed. In practice any closed shape identifier could be used here, such as region growing [29] or watershed [3].

We chose to refine and adapt Saund's closed shape identification algorithm [25] within the PROFI project [15]. Saund's approach was developed for the sketch retrieval domain but is equally applicable to the trademark retrieval domain. Our adapted algorithm integrates seamlessly with our other project software and provides two complementary versions. Using two complementary versions of the same technique ensures consistency which is essential in image retrieval. For this evaluation, we use a simplified version of our algorithm [15]. The simplified version aims to find just the basic shapes present in an image as the graph layout matching requires only the basic shapes compared to the more perceptual shapes perceived using Gestalt[2] principles [27, 16] discovered by our complementary version. In figure 3, we are interested in the three small triangles (to the left in figure 4) for our evaluation here but not the larger perceptual triangle formed by the three smaller triangles (to the right

---

[2]The Gestalt principles refer to the shape-forming capability of human vision. In particular, they refer to the visual recognition of figures and whole shapes rather than just 'seeing' a simple collection of lines and curves.
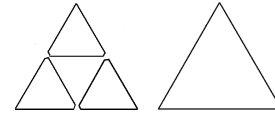
in figure 4). The fact that the three small triangles form a larger triangle will be discerned by the layout indexing so we do not need to find it here. The two approaches may therefore be viewed as complementary. The simple closed shape approach used here to find the basic shapes which feed into a layout indexing algorithm where the layout indexing infers the perceptual relations. The more perceptual closed shape approach described in [15] finds the perceptual shapes which may be used for shape matching where similarity is determined by "shape" and which requires higher level (perceptual) shapes for matching.

Given this set of shapes identified within a trademark, its layout is then encoded automatically into an attributed graph whose vertices represent shapes and whose edges show spatial relations (both directional and topological) between the shapes. Using a new graph-based indexing schema, the topological structure of the 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. Having established the signatures, the indexing now amounts to a nearest neighbour search in a model database. For a query graph and a large graph data set, the indexing problem is reformulated as that of fast selection of candidate graphs whose signatures are close to the query signature in the vector space.

The rest of the paper is organized as follows. After giving an overview of the related work in Section 2, we review the basics for our shape identification algorithm in Section 3. Section 4 presents a method to encode shape layouts in a graph. We give the details for our graph-based indexing algorithm in Section 5. Our framework is evaluated in the domain of trademark retrieval in Section 6. We close the paper with our conclusions and future work in Section 7.

2. RELATED WORK

Since our framework consists of encoding layout of a trademark in a graph and graph-based indexing, we will separately review the related work in these two concepts.

2.1 Encoding layout

The spatial relations between two objects in an image can be divided into topological and directional relations. Egenhofer [8] 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 alter-

native for this representation is the angle between the line connecting the two centres of mass and the horizontal line. For instance, the latter method was used by El-Kwae et al. [9] and Gudivada et al. [13].

A method for encoding layout taking into account only directional information was proposed by Chang et al. [5] and is called 2D-Strings. To produce a 2D-string representation, the centre 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 [4], 2D C-String [17]. These extensions deal mainly with overlapping objects with complex shapes.

Petrakis and Orphanoudakis [23] 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* such as 2D-strings is that in general they are not rotation invariant. Therefore, El-Kwae et al. [9] 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 that has also been applied in this paper is the graph representation. Gudivada and Raghavan [13] propose spatial orientation graphs (SOG's), in which each vertex represents an object and the edges between them are weighted with the slope of the line connecting the two centres of mass. The distance between two graphs is calculated by finding the angle between each pair of corresponding edges. ImageMap [22] proposed by Petrakis and Faloutsos extends this idea. The images are represented by attributed relational graphs (ARG's), 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 [10] 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. The embedding process in this method 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. Specifically, if the graphs are not trees then the graph distances cannot 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 this indexing schema performs against noise and occlusions.

2.2   Graph indexing and spectral methods

Our method deploys a graph representation for encoding a trademark's layout. The problem of retrieving similar graphs to a given query may be solved by finding graphs that are isomorphic to the query or one of its subgraphs. One important indexing method solving this problem 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 that no database graphs are isomorphic to the query. Messmer and Bunke [19] use this approach to organise the set of all permutations of the adjacency matrix of database graphs in a decision tree. At run time, the (sub)graph isomorphisms from the query to the database graphs are found by a decision tree traversal. A significant drawback of this method is its space requirement. All permutations of the adjacency matrix have to be encoded in decision trees, whose sizes grow exponentially with the size of the database graph. A set of pruning techniques is discussed to cut down the space complexity.

Although indexing methods with (sub)graph isomorphism detection algorithms are effective, due to noise, occlusion, or segmentation errors, no (sub)graph isomorphism may exist between the query and the database. Furthermore, only a certain degree of similarity between two graphs may be present. The indexing problem, therefore, is reformulated as efficiently retrieving database graphs whose (sub)structure is similar to the query. Although considerable research has been devoted to the problem of inexact (or error-tolerant) graph matching, rather less attention has been paid to this type of indexing based on graph structures.

An indexing framework related to the approach reported in this paper is that of Shokoufandeh et al. [26]. 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 neighbour 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.

3.   CLOSED SHAPE IDENTIFICATION

Prior to the encoding of layout in graphs, we require a shape identification algorithm to segment the trademark into separate closed shapes. For this evaluation, we use a simplified version of our adapted algorithm [15]. The simplified version aims to find just the basic shapes present in an image as the graph layout matching requires only the basic shapes.

Our closed shape algorithm requires an underlying technique to identify the line segments within an image and to detect the relationships between those line segments. The closed shape identifier then uses this output to identify the closed shapes. Therefore, we initially find the edges in an image and subdivide these into constant curvature segments

using the Sarkar & Boyer [24] edge detection algorithm and the Wuescher & Boyer [28] curve segmentation algorithm. These methods are used as they have been successfully used in the trademark system developed by Alwis [1]. The Sarkar & Boyer method finds the edge lines in an image and splits these lines into primitives. Wuescher & Boyer aggregates these primitives into more perceptually-oriented constant curvature segments. These segments thus provide the building blocks for our closed shape identifier. From these constant curvature segments, we produce a graph of segment relations. Each constant curvature segment becomes a node in the graph with two ends (first point (denoted as an x, y coordinate) and last point (also denoted as an x, y coordinate)). In our simplified implementation here, we find all segments that are end-point proximal within two pixels length. This effectively joins the graph by linking the proximal end-points. The resulting graph underpins the closed shape identification algorithm.

Our closed shape algorithm overlays this graph. Saund's approach focuses on managing the search of possible path continuations through the graph, particularly where the graph nodes represent junctions (crossroads, T-junctions etc) of lines in the original image. We use the same technique here. The closed path search commences from each end (first and last) of each node (line segment) identified by the underlying Wuescher & Boyer algorithm. For each end (first then last) in turn, all possible paths are followed. This effectively forms a search tree with paths through the tree representing the paths of candidate shapes. The search is managed through the use of Saund's local criteria [25] (scores) for ranking possible paths through junctions. Saund derived the scores from observations. These scores prioritise which node to expand next. As each leaf node in the tree is expanded, any new child nodes are compared with child nodes in the opposite side of the tree. If they are end-point proximal then a closed path has been identified and its nodes and pixels are added to the list of candidate paths. To produce the set of shapes for each image in this paper, we accept all candidate paths. However, closed paths that are subsumed by other closed paths with higher scores are discarded. Hence, each new closed path is compared to all existing stored paths. If the new path is equivalent to an existing path but has lower score then the new path is discarded. If the new path has higher score than the existing saved path then the saved path is discarded.

In our simplified approach used in this paper, we have included three changes from our usual method. We measure end-point proximity as within a 2 pixels length, normally we use Lowe's method [18] to extract endpoint proximity which is more perceptually plausible as it uses the ratio of line length to gap length [15] to decide when two lines are end-point proximal. We keep all candidate paths that are not subsumed, normally we use a minimum score threshold to identify plausible shapes as our previous approach [15] is aimed at identifying shapes perceived using Gestalt principles. Finally, we keep all paths as our set of shapes for the image, normally we use a global goodness score to assess perceptual relevance and discard perceptually irrelevant shapes (see [15]).

4. ENCODING LAYOUT IN GRAPHS

Our indexing schema can handle graphs carrying different kinds of layout information. In the experiments conducted for this paper, the vertices in the graphs correspond to the shapes in the trademark image, and the edges between them carry relations between them. Foremost, we encode the directional information (in the form of both primary and secondary directions). Rotational invariance can be achieved on demand, by neglecting for instance the difference between a south-north edge and a east-west edge. Furthermore, we are interested in detecting certain basic layout configurations that often occur in trademarks, such as triangular, circular or square configurations. If one or more of these types of layout are present in a trademark, they are encoded in the appropriate edges too.

The trademarks are segmented into individual shapes using the method described in the previous section. After this stage, their centroids are used as shape representatives to calculate the appropriate information and determine the edge labels. Each shape is connected to its n nearest neighbors, where n is a user defined parameter. The first step is to calculate the angle between the horizontal axis and a line connecting two centroids to determine the directional label for this edge. In principle there are eight possible directions (4 primary and 4 secondary), but the edges are undirected so there are four possible directions for each edge.

The next step is to detect the special pattern 'square'. This is done by performing a template match on the directional graph with a template representing a configuration of four shapes in a 2×2 square. Whenever this template is found, the edge labels are updated accordingly from the directional information to the special edge type square. Note that the square needs to be isolated to a certain extent; e.g. a grid is not a large collection of squares according to this definition. The same kind of template matching is performed for triangular and circular configurations. The decision of triangularity depends on the angles between the possibly triangular edges. Since every triplet of objects forms a triangle by definition, only the edges of a perfect triangle (or close to a perfect triangle) are labeled with the special triangle edge type. To detect circular configurations, the following circularity criterion is evaluated on the convex hull of the shape centroids: $4\pi A / \rho^2$, where A is the area and ρ is the perimeter of the convex hull. A threshold is set on the outcome of this circularity criterion to determine whether the edges on the convex hull need to be labeled with the special circular type or not.

In the experiments, the trademarks in our dataset are classified based on their layouts, not on the shapes they consist of. This classification was used to measure the retrieval performance. Since trademark retrieval and similarity are complex issues involving specific knowledge of perception and trademark logic, we presented our classification to a group of experts at Aktor Knowledge Technology who examine trademark similarity in commercial surroundings on a daily basis. It was only after their concise inspection of our dataset and classification, that we could be sure conducting our experiments and measuring performance are in correspondence with the real trademark similarities.

5. INDEXING VIA LAPLACIAN SPECTRA

Given a query graph and a large database, the objective of an indexing algorithm is to efficiently retrieve a small set of candidates, which share topological similarity with the query or one of its subgraphs. In our framework, we encode the topology of a graph through its laplacian spec-

trum. The laplacian matrix $L(G)$ of graph G is computed as $L(G) = D(G) - A(G)$, where $D(G)$ is the degree matrix and $A(G)$ is the adjacency matrix for G. The spectrum of a graph's laplacian matrix is obtained from its eigendecomposition. More formally, 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 $\{\lambda_1, \lambda_2, \ldots, \lambda_{|V|}\}$.

Our main motivation for encoding the topology of a graph using the lapcacian rather than the adjacency matrix as done by earlier work [26] comes from the fact that laplacian matrices are more natural, more important, and more informative about the input graphs [20]. Previously, Godsil and McKay [11] and more recently Haemers and Spence [14] have also shown that the laplacian matrix has more representational power than the adjacency matrix, i.e., it results in less number of cospectral graphs. Recall that two graphs are called cospectral (or, isospectral) if they have the same eigenvalues.

In our framework, we define the signature of a graph as the sorted eigenvalues of its laplacian matrix. 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 its similar graphs, therefore, can be reduced to a nearest neighbor search among a set of points.

Unfortunately, this formulation cannot deal with occlusion or segmentation errors as two graphs may share similar structures up to only some level. Although adding or removing edges changes the laplacian spectrum, the spectrum of the subgraphs that survive such alteration will not be affected. Our indexing mechanism, therefore, cannot depend on the signature of the whole graph only. Instead, we will combine the signatures of the subgraphs in the framework.

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 [3].

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

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

$$\ldots \leq \lambda_n(G) \leq \lambda_n(G').$$

In addition, it is known that $\sum_{i=1}^{n}(\lambda_i(G') - \lambda_i(G)) = 2$ [2]. 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. This shows that constructing an indexing mechanism based on graph signatures alone is too weak. An ideal indexing framework should, in fact, select candidate database elements based on both local and global similarities. To account for both local and global information, we will adopt

---

[3]This theorem is obtained by Courant-Weyl ([6], Theorem 2.1). The reader may also refer to [12].
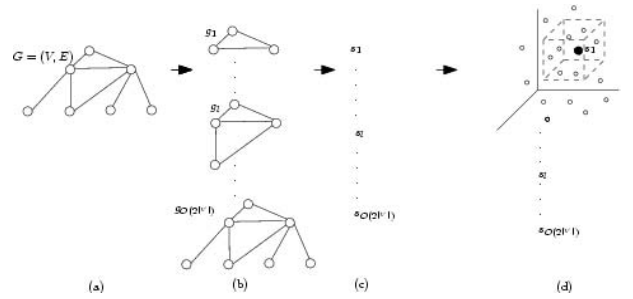

Figure 5: Retrieving similar graphs. For graphs given in Part (a), its subgraphs are constructed in Part (b). A signature is computed for each subgraph in Part (c). Given a signature, retrieving its similar graphs from a large database is formulated as a nearest neighbor search as shown in Part (d).

the following method analogous to that used in the decision tree approach [19].

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 retrieve its nearest neighbors of the same size from the database through a nearest neighbor search (see Figure 5). Each neighbor of s_q retrieved from the database gets a vote whose value is inversely proportional to the distance from s_q. Thus, as a result, each signature of the query generates a set of votes. Moreover, we weigh the votes according to the size of the subgraphs corresponding to the signatures, i.e., the bigger the size, the more weight the vote receives.

Our encoding of a graph's structure captures its local topology, thus allowing for its use in the case of occlusion and segmentation errors. Furthermore, the signature of a graph is invariant under the reorderings of its vertices. This, in turn, allows us to compare the signatures of a large number of graphs without solving the computationally expensive correspondence problem between their vertices. 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 to the query.

6. EXPERIMENTS

In this section we evaluate our framework in the context of a trademark retrieval experiment. We use a set of 450 trademark images from the UK PTO dataset used in the Artisan project [7]. Figure 6 shows some trademark images used in the experiments. We begin by representing the layout of
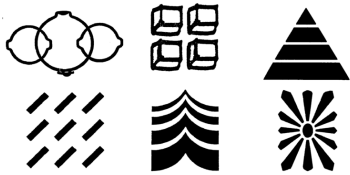
Figure 6: Some trademark images used in the experiments.

each image in the 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. We applied the following leave-one-out procedure to the datasets to evaluate the framework in the experiments. 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.

To check our segmentation and automatic graph construction procedures, the graph representation process was also performed manually in our experimental setup. Specifically, we manually selected shapes for each input trademark image, created a vertex for each shape, and connected two vertices by an edge if the layout of the corresponding shapes should be encoded in the graph based on the human perception. As a result, one manually and one automatically constructed graph datasets have been generated. The performance of the proposed indexing algorithm was evaluated for each dataset.

Precision and recall are two well-known performance measures to compute the quality of an indexing mechanism. In high precision, relevant items are in the top of the ranking, whereas in high recall, false negatives are avoided and the returned result contains all 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, our goal is to quickly determine the class of the query. In the second experiment, the objective is to return a small candidate set, which contains all the objects belonging to the query class. In both experiments, the indexing system ranks the database graphs in decreasing order of similarity from each query graph. According to the results, in 98.4% and 89.1% of the cases, the most similar database graph belongs to the correct shape class for manually constructed and automatically constructed datasets, respectively (nearest-neighbor rates). In addition, the worst position of the closest matching graph is 5 for manual graphs, while this number is 9 for automatic graphs. These numbers show that 98% of the datasets can be pruned by the indexing mechanism to determine the correct layout class for a query. In the second experiment, the system's performance was 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 71 of the candidate return set always contains all the graphs belonging to the query class for manual graphs; this number is 80 for automatic graphs. This indicates that for this task our framework prunes more than 84% and 82% of the manual and automatic graph datasets, respectively. In other words, the

recall in each dataset is 100% if the scope is set to the first 16% and 18% of the sorted candidate models for manual and automatic graphs respectively. We also computed how many of the models in the query's class appear within the top $K - 1$ matches, where K is the size of the query class (first tier). This number was 91.2% for manual graphs and 86.3% for automatic graphs. Repeating the same experiment but considering the top $2 \times K - 1$ matches (second tier) covers 98.1% and 91.7% members of the layout classes for manual and automatic graph datasets, respectively.

In Table 1, we have presented the matching results for a small subset of trademark images whose graphs were generated automatically using our approach. The first column of each row represents the query image; the remaining elements of each row show the top 10 closest database trademarks retrieved by our indexing algorithm. Squares are drawn around the wrong matches. In all but once case (row 5) the closest trademark image belongs to the same layout class as the query. Although the closest match for the query in row 5 was classified as a mismatch, one may notice that the query consists of three sets of small squares on top of each other and each set has the same layout as the mismatch. As another example, consider the query in row 8 of the left-right class and its first mismatch of the triangle class. Notice that three small triangles in the mismatch have the same layout as the query. Overall, rather than focusing on the mismatches that occur because of the result of a partial match, we observed that the wrong selections happen mainly due to the poor segmentation of the trademark. If different shapes in a trademark are connected, for instance, our segmentation algorithm detects them as one shape. Thus, the layout within these shapes are not encoded in the graphs. We will extend our segmentation technique to region-based and will use it within our framework in the future.

7. CONCLUDING REMARKS

In this paper we have presented a framework for retrieving trademark images based on spatial layout of the shapes. Besides pure shape similarity between trademark images, similarities in configuration of the shapes may also give rise to a conflict of uniqueness. The process of content based trademark retrieval, therefore, can be significantly improved by taking into account these layout features, enabling a stronger prevention of trademark infringement.

In our framework, trademark images are first segmented into closed, distinct shapes. This segmentation is line based; after an initial edge detection step, the shape boundaries are subdivided into constant curvature segments. These segments are then aggregated to more perceptually relevant primitives, which form the input blocks for the closed shape identifier. By searching for closed paths in the primitives, the shapes are returned and passed on to the next layer of our framework, the construction of a layout graph.

The centroids of the shapes are taken as shape representatives while constructing a graph that reflects the layout of the trademark. Each shape is represented by a vertex, and connected to a predefined number of nearest neighbors, and layout information is stored in the edges that connect these vertices. After the graph construction, the laplacian matrix is taken (by subtracting the adjacency matrix from the degree matrix) and its spectrum is computed. Every trademark is then stored in a database, by populating a vector space with the laplacian spectra. The laplacian spectrum

Query	Top 10 Matched Images

Table 1: Top matched models are sorted by the similarity to the query.

reflects important properties of the graph and its topology. Besides computing the laplacian spectrum of the complete graph, we also store this feature vector for every possible subgraph to perform partial matching.

We evaluated our framework on a test collection of 450 real trademark images and the results are promising. First and second tier results, averaged over all possible queries, were 86.3% and 81.7% respectively. It is one of our future works to extend the test collection and perform a comparison with other, known layout indexing techniques. Furthermore, we want to take into account topological information as well, besides the directional information and special configurations that are encoded now. Finally, we will use a region-based segmentation algorithm within our framework to reduce the number of mismatches that occured because of the current line-based segmentation procedure.

Acknowledgements

8. REFERENCES

[1] S. Alwis. *Content-Based Retrieval of Trademark Images.* PhD thesis, Dept. of Computer Science, University of York, UK, 1999.

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

[3] S. Beucher. Watersheds of functions and picture segmentation, acoustics, speech, and signal processing. In *IEEE International Conference on ICASSP'82*, pages 1928–1931, 1982.

[4] S. 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.

[5] 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.

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

[7] J. P. Eakins, K. Shields, and J. M. Boardman. Artisan: A shape retrieval system based on boundary family indexing. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 17–28, 1996.

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

[9] E. El-Kwae and M. 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.

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

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

[12] 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.

[13] 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.

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

[15] V. Hodge, J. Eakins, and J. Austin. Inducing a perceptual relevance shape classifier. In *ACM International Conference on Image and Video Retrieval, (CIVR07). July 9-11 2007*, 2007.

[16] K. Koffka. *Principles of Gestalt Psychology.* Harcourt Brace. New York, 1963.

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

[18] D. Lowe. Three dimensional object recognition from simple two dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987.

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

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

[21] W. I. P. Organisation. *CD-NIVILO ISBN 92-805-1280-7.* WIPO, 2003.

[22] E. 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.

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

[24] S. Sarkar and K. Boyer. On optimal infinite impulse response edge detection filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(11):1154–1171, 1991.

[25] E. Saund. Finding perceptually closed paths in sketches and drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(4):475–491, 2003.

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

[27] M. Wertheimer. Laws of organization in perceptual forms (1923)., 1938.

[28] D. Wuescher and K. Boyer. Robust contour decomposition using a constant curvature criterion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(1):41–51, 1991.

[29] S. Zucker. Region growing: Childhood and adolescence. *Computer Graphics & Image Processing*, 5:382–399, 1976.