

Matching Shape Graphs

M. Fatih Demirci^{1,2} *, Reinier van Leuken¹, and Remco Veltkamp¹

¹ Department of Information and Computing Sciences, Universiteit Utrecht, The Netherlands

² Department of Computer Engineering, TOBB University of Economics and Technology, Turkey

Graphs are important structures to model complex relationships such as chemical compounds, proteins, geometric or hierarchical parts, and XML documents. Given a query graph, indexing has become a necessity to retrieve similar graphs quickly from large databases. We propose a novel technique for indexing databases, whose entries can be represented as graph structures. Our method starts by representing the topological structure of a graph as well as that of its subgraphs as vectors in which the components correspond to the sorted laplacian eigenvalues of the graph or subgraphs. By doing a nearest neighbor search around the query spectra, similar but not necessarily isomorphic graphs are retrieved.

Copyright line will be provided by the publisher

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. 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. An effective and efficient indexing mechanism is, therefore, 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 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. 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. Although powerful indexing techniques with (sub)graph isomorphism detection algorithms have been proposed in the literature (e.g. [1]), due to noise, occlusion, or segmentation errors, no (sub)graph isomorphism may exist between the query and the database. Furthermore, only a certain degree of similarity between two graphs may be present. The indexing problem, therefore, is reformulated as efficiently retrieving database graphs whose (sub)structure is similar to the query. Although considerable research has been devoted to the problem of inexact (or error-tolerant) graph matching, rather less attention has been paid to this type of indexing based on graph structures.

A framework related to the approach reported in this paper is that of Shokoufandeh et al. [2]. 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 δ -dimensional vector, where δ is the root degree. To account for occlusion and local deformation, these vectors are also computed for the root of each subtree. At indexing time, each non-leaf node of the query is represented as such a vector, and a nearest neighbor search is performed for each vector. Although effective, by summing up the largest eigenvalues one loses uniqueness, resulting in less representative graphs in the vector space.

2 Description of the Framework

In this section, we summarize our approach for the graph-based indexing problem. Details can be found in [3]. Instead of using the adjacency matrix for graph characterization as done by some earlier work, we characterize the graphs based on the laplacian spectrum, which is more natural, more important, and more informative about the input graphs [4]. The framework encodes the topology of a graph through the laplacian spectrum. Sorted eigenvalues of the laplacian matrix are assigned to the graph as its signature. To compute the similarity between two graphs, we compute the Euclidean distance between their signatures, which is inversely proportional to the structural similarity of the graphs. For a given query, retrieving similar graphs can be reduced to a nearest neighbor search among a set of points.

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

For a given database graph, rather than storing its signature in the system only, we compute the signatures of each subgraph of G in our algorithm. In this process, we gradually increase the size of the subgraphs. Since the sorted eigenvalues are invariant under consistent re-orderings of the graph's vertices, it is sufficient to compute the spectrum of permutation-similar

* Corresponding author E-mail: mfdemirci@etu.edu.tr, Phone: +90 312 292 4264, Fax: +90 312 292 4180

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).

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

3 Experiments

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. We used the MPEG-7 dataset CE-Shape-1 part B and McGill 3D Shape Benchmark [5] for these two domains. While the MPEG-7 database consists of 70 classes and 20 shapes per class, the McGill database consists of 420 objects classified in 19 classes. We used shock graph and Reeb graph representations for the MPEG-7 and McGill databases, respectively. 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 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.

The experimental results show that in 37.3% of the cases, the highest-weight database graph belongs to the correct shape class for shock graphs and this ratio is 29.6% for Reeb graphs. Moreover, the average position of the closest matching graph among the highest-weight candidates is 7.4 and 4.3 for shock and Reeb graphs, respectively. In addition, the worst position of the closest matching graph is 12 for shock graphs, while this number is 9 for Reeb graphs. These results present that to determine the correct class of the query, more than 99% and 97% of shock and Reeb graph datasets can be pruned by our indexing mechanism. In addition, according to the results, 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. A more comprehensive evaluation of the method including a comparison with a competing indexing method can be found in [3].

Acknowledgments

This research was supported by the FP6 IST projects 511572-2 PROFI and 506766 AIM@SHAPE.

References

- [1] B. Messmer and H. Bunke, *Pattern Recognition* **32**(12), 1979–1998 (1999).
- [2] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S. Zucker, *PAMI* **27**(7) (2005).
- [3] M. F. Demirci, R. van Leuken, and R. Veltkamp, *Indexing through laplacian spectra*, 2007, *Computer Vision and Image Understanding, Special Issue on Similarity Matching in Computer Vision and Multimedia* (conditionally accepted).
- [4] B. Mohar, *The laplacian spectrum of graphs*, in: *Sixth International Conference on the Theory and Applications of Graphs*, (1988), pp. 871–898.
- [5] J. Zhang, K. Siddiqi, D. Macrini, A. Shokoufandeh, and S. J. Dickinson, *Retrieving articulated 3-d models using medial surfaces and their graph spectra.*, in: *EMMCVPR*, (2005).