Complex Fiedler vectors for shape retrieval

Reinier H. van Leuken[1], Olga Symonova[2] Remco C. Veltkamp[1], and Raffaele de Amicis[2]

[1] Universiteit Utrecht, The Netherlands
[2] Fondazione Graphitech, Trento, Italy

Abstract. Adjacency and Laplacian matrices are popular structures to use as representations of shape graphs, because their sorted sets of eigenvalues (spectra) can be used as signatures for shape retrieval. Unfortunately, the descriptiveness of these spectra is limited, and handling graphs of different size remains a challenge. In this work, we propose a new framework in which the shapes (3D models in our test corpus) are represented by multi-labeled graphs. A Hermitian matrix is associated to each graph, in which the entries are defined such that they contain all information stored in the graph edges. Additional constraints ensure that this Hermitian matrix mimics the well-studied spectral behaviour of the Laplcian matrix. We therefore use the Hermitian Fiedler vector as shape signature during retrieval. To deal with graphs of different size, we efficiently reuse the calculated Fiedler vector to decompose the graph into a limited number of non-overlapping, meaningful subgraphs. Retrieval results are based on both complete matching and subgraph matching.

1 Introduction

Although more and more 3D models populate the Internet nowadays, content based searching that is driven by shape matching remains a challenge. This research area can be divided in to areas: shape descriptors and corresponding matching algorithms at one hand, and indexing algorithms that facilitate efficient querying and database pruning on the other hand. In this paper we propose a shape descriptor that can be used as input for indexing algorithms directly. For more information about shape descriptors in general, see [1, 2].

Among all shape descriptors, graph based descriptors have an advantage of providing the possibility to detect both partial and overall similarity between shapes. In particular we would like to mention Reeb graphs, which represent the topological structure of 3D model and can store local geometrical characteristics. Reeb graphs for shape description and retrieval were proposed in [3–7]. However, it is still a challenge to perform (sub)graph matching. Different heuristics are used to reduce the NP-hard problem of finding (sub)graph isomorphism, e.g. bipartite graph matching through minimizing the distance between node labels [6], matching based on graph hierarchy [4, 7] and maximum common subgraph computation based on the node attributes [5]. An alternative approach to reduce the complexity of graph matching is to use the spectrum or eigenvectors of an

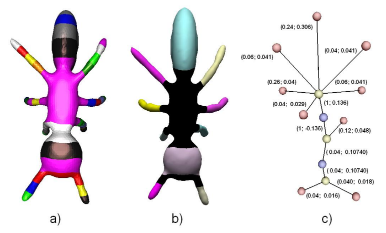


Fig. 1. ERG construction. (a) Shape segmentation. (b) Merging simple adjacent components. (c) ERG with edge attributes: shape index and segment weight.

associated matrix [8–12]. Unfortunately, the descriptiveness of the used matrices is limited, and it's often unclear how to handle graphs of different sizes.

Contributions. First, we propose a reeb-graph based shape descriptor that encodes topological and geometrical features of the model. Second, we propose to represent this descriptor by the complex-valued Fiedler vector of a Hermitian property matrix. This property matrix is constructed to contain all the features that are stored in the graph, while mimicking the well-known Laplacian matrix from a spectral point of view. Third, the calculated Fiedler vector is reused to partition the graph into non-overlapping, meaningful subgraphs that are used for partial similarity and to overcome the problem of graphs of different size. We show that our method outperforms existing spectral based indexing methods.

2 Segmentation

The Extended Reeb Graph (ERG) gives a notion about the structure of a model, and depending on the function which is used to construct the graph, it can be invariant to the position of a model in space as well as to posture variation. The ERG can be additionally enriched with geometrical shape characteristics, so that it gives a more complete shape description.

We define a mapping function f which is used to represent the manifold of a model, in our case the integral geodesic distance function [4]. The values of the mapping function are computed for each vertex of the triangular mesh.

Next, we specify N levelsets $f_i^{-1} = \{x | f(x) = f_{min} + i\frac{f_{max}-f_{min}}{N+1}\}$ for $i = 1..N$, and the intervals of the mapping function $\bigcup_{i=0}^{N}[f_i, f_{i+1}]$. These intervals define the decomposition of the model into connected components, see Figure 1a). The components can be divided into three groups, having one, two or more boundaries. In this context, a boundary is a levelset f_i^{-1} shared by two adjacent components. The components with one boundary represent minimum or maximum regions, components with two boundaries are regular regions, and components with more than two boundaries correspond to saddle regions of the mapping function [13]. We call the components with one and two boundaries *simple*. Their shape can be characterized by several geometric features [14, 15]. Adjacent simple components are merged into ons segment, see Figure 1b.

To construct the ERG, each component obtained after the described segmentation and merging process, is represented by a graph node. There is an edge between two nodes if the corresponding components are adjacent. If two saddle components are adjacent, we insert an intermediate node between them (body part): see Figure 1*c*. Simple and intermediate nodes are connected only to saddle nodes and vice versa.

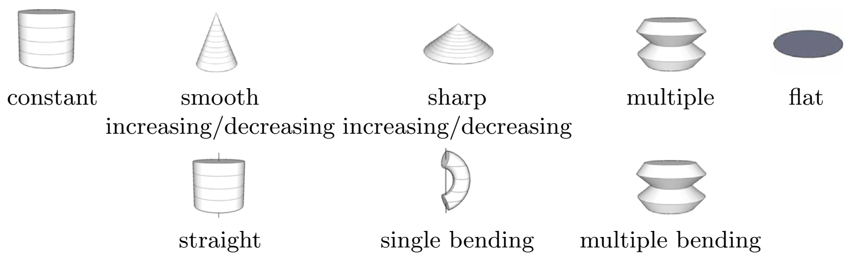


Fig. 2. Variations of the two shape criteria cross section alteration and bending.

To enrich the topological structure of the ERG with geometric characteristics, we use a recently proposed shape analysis of simple components that uses two criteria [15]. The first criterion is the change in area of the cross sections. All simple components are classified in one of the following seven shape classes: constant, smooth/sharp increasing/decreasing, cross area alterations and flat components. The second shape criterion is component bending, and defines three classes: zero, single and multiple bending. See figure 2 for prototypes of shape classes; see the top row for cross area alteration and the bottom row for bending.

Combined, these two criteria define 19 different shape classes, together with the intermediate nodes there are 20 shape types. Each class is represented by a unique index. Additionally, we calculate the relative size of each segment, defined as the ratio of the surface area of the segment to the surface area of the whole model. The index of the shape classification and the segment size are stored as attributes of the edge that connects the simple node with a saddle node.

3 Graph property matrices

We store the graph characteristics in a Hermitian matrix because it provides space for all topological information and the edge properties. A more popular and well-studied matrix, is the Laplacian matrix. We therefore take the Laplacian matrix as our starting point, and show how this matrix can be extended to a Hermitian matrix to contain more information, without losing the validity of known theorems. The Laplacian matrix is defined as $L(G) = D(G) - A(G)$, where $D(G)$ is the diagonal matrix containing node degrees, and $A(G)$ is the adjacency matrix; entry $A_{i,j}$ is 1 if nodes i and j are connected, 0 otherwise. The spectrum of the Laplacian matrix can be used as a signature representation for the graph, and thus for efficient retrieval purposes (indexing) [8]. One of the main reasons for this is that many graph properties and invariants are reflected by the Laplacian spectrum [16]. Moreover, cospectrality for non-isomorphic graphs tends to be rare [12] and similar Laplacian matrices have similar spectra due to the interlacing theorem for two graphs where one is a slightly modified version of the other [17].

In the case of a weighted graph, $L_w(G) = D_w(G) - A_w(G)$ can be obtained, by storing edge weights in the matrices instead of the number of edges. In order to preserve the useful properties of a normal Laplacian, every edge weight $w_{a,b}$ should be positive, it may only be zero if a and b are not adjacent, and should be equal to $w_{b,a}$ (symmetry).

3.1 Hermitian matrices

In order to encode more information about the graph in a matrix, we extend the ideas of [11] and use a Hermitian matrix to store graph characteristics. By imposing several constraints on the elements of the Hermitian matrix, we will define a complex analogue of the Laplacian matrix, such that it posesses all the properties reported in [16], [12] and [17].

A Hermitian matrix H (or self-adjoint matrix) is a square matrix with complex entries that is equal to its own conjugate transpose. Every Hermitian matrix has a real valued spectrum; the corresponding eigenvectors are complex. To impose the necessary constraints on H, it's off-diagonal elements are complex numbers written in polar form using Euler's formula, defined as $H_{a,b} = -W_{a,b}e^{iy_{a,b}}$, where each edge has the pair of properties $(W_{a,b}, y_{a,b})$. The first property, $W_{a,b}$, is used as the magnitude of the entry and should satisfy the same constraints as the edge weights of a Laplacian matrix. The second property, used as the phase of the complex matrix entry, must satisfy the following conditions:

$$y_{a,b} = -y_{b,a} \tag{1}$$

$$-\pi < y_{a,b} < \pi \tag{2}$$

The first condition (1) ensures that H is equal to its own conjugated transposed matrix. By obeying the second constraint (2), phase wrapping can be avoided. The real-valued on-diagonal entries are defined as $H_{aa} = \sum_{b \neq a} W_{a,b}$. In this way, the entries in each row of the matrix sum up to zero. We would like to stress that this is a necessary property to correctly mimic the spectral behavior of Laplacian matrices, contrary to the Hermitian matrix that is used in [11] (where additional node measurements on the diagonal are allowed). Furthermore, magnitudes should be calculated in such a way that an edge between two nodes can never be weighted 0, for it would destroy the connectivity of the graph.

As was described in section 2, an edge in the graph connects a saddle node and a simple node. As a consequence, the calculated shape index and the weight of the simple component can be stored as the attributes of the edge incident to the corresponding simple node. All 20 possible shape classifications are represented by one unique shape index. The index range is mapped to the interval $(0;1]$ to allow their use as magnitudes of complex entries of the Hermitian matrix. The positive shape index of the segment is invariant to edge direction, thus satisfying the edge weight constraints. We choose the values of shape indices based on visual shape similarity as well as on the alteration of one or both criteria used for shape analysis, assigning a greater influces to the variation of cross area than to the bending criterion. This choice can be explained by the fact that semantically equivalent subparts of articulated models frequently have different bending properties, e.g. straight and bended arms.

The relative size of the segment can be used as the phase value y_{ab} for the complex entries of the Hermitian matrix. To obey the antisymmetric condition (1) we set y_{ab} as the relative size $\mathrm{Size}_{S_{ab}}$ of the segment S_{ab} if $\deg(a)>\deg(b)$ and as $-\mathrm{Size}_{S_{ab}}$ if $\deg(a)<\deg(b)$. The value of $\mathrm{Size}_{S_{ab}}$ is bounded by the interval $(0;1)$. Scaling this interval up to $(0;\pi)$ and using the above definition of y_{ab} we satisfy the constraint (2) for the phase value of the complex entries of the Hermitian matrix.

4 Retrieval through complex Fiedler vectors

The Hermitian matrix with attributes calculated as described above, mimics the spectral behavior of the Laplacian matrix. We propose to use the eigenvector associated to the second smallest eigenvalue (i.e. the first non-zero eigenvalue) as a signature for the graphs. These graph entities are known as Fiedler eigenvalue and Fiedler eigenvector, they encode probably the most important graph information [16].

In order to preserve permutation invariance of the representation, the entries of the Fiedler vector need to be sorted. Therefore, before evaluating the similarity between two n-dimensional Fiedler vectors, they are transformed into $2n$-dimensional vectors, by sorting them lexicographically (first based on the real part, then on the imaginary part) and interleaving the real and imaginary parts. The similarity between graphs with n nodes, can then be defined as the Euclidean distance between their $2n$-dimensional, interlaced and sorted Fiedler vectors. In practice however, models are often represented by graphs of different sizes. Furthermore, partial similarity is not taken into account by this approach. We therefore propose to base retrieval both on complete and partial graphs.

4.1 Retrieval of complete graphs

A common solution to the problem of comparison of differently sized graphs through their spectra or eigenvectors is padding with zeros: enlarge the smaller spectrum or eigenvector to the size of the larger spectrum by inserting zeros. This is a semantically sensible approach, since it means that the smaller graph is enlarged by inserting isolated nodes. In the context of a given database, all the graphs are brought up to the maximum size present in that database, by inserting isolated dummy nodes. When all Fiedler vectors are of the same dimension, a simple range search or nearest neighbor search among these vectors produces a ranked list of relevant database objects with respect to a query. The distance between two graphs g_1 and g_2 (based on the complete graphs) is then defined as

$$d_{\text{full}}(g_1, g_2) = \sqrt{\sum_{i=1}^{2n} (F_1(i) - F_2(i))^2} \tag{3}$$

Although this is correct semantically, it can possibly distort the retrieval results. Suppose two similar graphs are of different size as a consequence of a different level of detail that was revealed during segmentation. When dummy nodes are inserted in the coarsely segmented graph simply to enlarge the dimension of its Fiedler vector, a larger distance to the finer segmented graph is found as a consequence of matching entries related to dummy nodes with entries related to real nodes. Despite this phenomenon, the approach still produces good results for similar graphs of nearly equal size or when difference between different graphs should be magnified. We therefore propose to use it, but in combination with a method for retrieving database models based on parts of the graphs.

4.2 Subgraph decomposition

Evaluating similarity of subgraphs is used to account for partial similarity. It also helps in solving the problem of handling graphs of different sizes, because combining similarity of parts may reduce the influence of a finer segmented region. Only subgraphs of the same size are used for similarity evaluation. The problem that remains, is to find subgraphs that are appropriate for this purpose. One proposed approach is extracting all possible subgraphs of all possible sizes, and match only subgraphs of the same size [8]. This is, however, a time consuming approach, and requires calculation of distances between many subgraphs that are not meaningful or irrelevant to the original graph. Therefore, we propose to use a graph partitioning based on [18] to find small, non-overlapping and meaningful subgraphs. The proposed partitioning approach reuses valuable information that is already calculated, i.e. the complex Fiedler vector. The approach is as follows:

1. Sort the entries of the Fiedler vectors lexicographically in decreasing order.
2. Associate a *score* with every node in the graph: $F(i) = \alpha \times \text{degree}(i) + \beta/\text{rank}(i)$, where $\text{rank}(i)$ is the position of the node in the sorted Fiedler vector, and α and β are two balancing factors to increase influence of either rank or degree. In our experiments, they are both set to 0.5.
3. Traverse through the list of the sorted Fiedler vector and select center nodes. A node is a center node if its score is higher than the scores of all its neighbors.
4. Remove the center node and its adjacent nodes from the list; together they form a subgraph. Continue processing the graph this way until all nodes are in a subgraph.
5. Decompose each subgraph s even further into every possible combination of the center node and its adjacent nodes of size 2 up to the size of s.

Steps 1-4 of the above procedure partition the graph in non-overlapping meaningful subgraphs. For an example, see Figure 3, where a segmented model of a goat is displayed, together with a view of its 3D graph and extracted meaningful subgraphs (back part, middle part and head part).

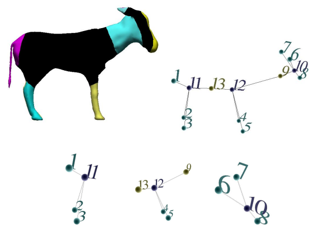


Fig. 3. Subgraph decomposition of a model of a goat.

To be even more robust against small alterations in the graph structure that are caused by segmentation and/or analysis differences, step 5 of the above procedure decomposes the subgraphs even further. To calculate the subgraph-based distance between two graphs g_1 and g_2, all their subgraphs are represented by their Fiedler vectors as described in the beginning of Section 4. The distance between g_1 and g_2, based on their subgraphs, is then defined as the normalized and weighted sum of all pairwise subgraph distances, where the distance between two subgraphs is calculated only if they are of the same size. For a given subgraph size s, the average is taken of all pairwise distances of subgraphs of this size:

$$d_{\mathrm{sg\_s}} = \frac{1}{n \times m} \sum_{i=1}^{n} \sum_{j=1}^{m} d_{\mathrm{full}}(g_1^i, g_2^j) \tag{4}$$

where g_1^i and g_2^j are the $i-th$ and $j-th$ subgraphs of size s of graphs g_1 and g_2, and where n and m are the number of subgraphs of size s that were extracted from g_1 and g_2.

A subgraph distance contributes more to the total similarity if the size of the subgraphs is larger. The total subgraph based distance between g_1 and g_2 is

$$d_{\mathrm{sg}}(g_1, g_2) = \frac{1}{\mathrm{MaxS} \times (|g_1| + |g_2|)} \sum_{d=1}^{\mathrm{MaxS}} s \times d_{\mathrm{sg\_s}} \tag{5}$$

where $|g_1|$ and $|g_2|$ are the number of nodes in g_1 and g_2 respectively, and where MaxS is the size of the smallest maximal subgraph that could be decomposed from either g_1 or g_2.

Using this distance function, a ranked list of all the database objects can be produced for a given query. Together with the ranked list based on the comparison of complete graphs (see Section 4.1), two complementary ranked lists are produced for one query. These ranked lists need to be aggregated into one. For this purpose, each database object is assigned a score i, calculated as $\text{score}(i) = \log(\text{r}_{\text{sg}}(i)) + \log(\text{r}_{\text{full}}(i))$, where $\text{r}_{\text{sg}}(i)$ and $\text{r}_{\text{full}}(i)$ are the ranks of object i in the ranked lists according to subgraph matching and complete matching respectively. Database objects are then re-ordered based on the scores. By taking the logarithms, a higher priority is given to the top of both ranked lists. By looking at ranks rather than distances while calculating the scores, the aggregation doesn't suffer from a difference in distance range or scale.

5 Experimental results

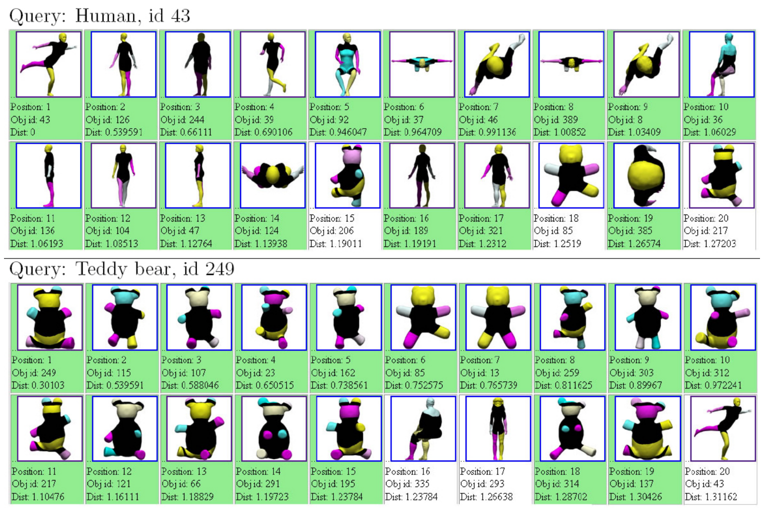
We tested our framework on the dataset that was used in the Watertight 3D Models track of the SHape REtrieval Contest 2007 (SHREC '07) [19]. This dataset consists of 400 models, divided over 20 classes of 20 models each. All models were preprocessed using the described segmentation and shape analysis software. After preprocessing, the implemented framework ranks the complete database of 400 models with respect to one query in around 0.2 seconds on a 3.00 Ghz Intel Xeon processor. Each model was used as a query once; the goal is to to see the complete class of the query at the top of the ranking.

However, before running the experiments on the whole dataset, we investigated the ability of the approach to discriminate between topologically similar, but geometrically and semantically different models. To this end, we have chosen the class of human and the class of Teddy bear models. Figure 4 shows retrieval results for some sample queries when only this subset of 40 models is considered. As can be seen from these results, the Teddies can be separated from the topologically similar humans, by means of the encoded shape analysis.

To be able to compare the results to other participants of this contest, the same performance measures are used. These are precision and recall rates when the scope is set to 20, 40, 60 or 80 items; Average Dynamic Recall [3]; success rates for the first (PF) and second (PS) items; Average Ranking and Last Place Ranking [4] Unfortunately, our method does not perform as well as most of the SHREC participants. Table 1 displays the above performance values for our method (abbreviated as *CFV*, for Complex Fiedler Vectors) and the method that performed best in SHREC 2007, a method by Tung et al. [7]. At this point however we would like to emphasize that our method is an indexing method rather than a matching method; all participating methods in SHREC are matching methods encoding a large amount of shape information. This means that with our method, parts of the database can be discarded upon querying time, whereas the matching methods need an exhaustive search through the database to obtain the ranking. Our method does improve with respect to other spectral based indexing methods; we compared it to using the Fiedler vectors of adjacency and Laplacian matrices.

[3] $ADR = \frac{1}{20}\sum_{i=1}^{20}\frac{RI(i)}{i}$, where $RI(i)$ is the number of relevant retrieved items in the top i of the ranking. $ADR \in [0,1]$, where 1 is the best retrieval result.

[4] $LPR = 1 - \frac{\text{Rank}-20}{380}$, where Rank is the rank of the last relevant item. $ADR \in [0,1]$, where 1 is the ideal retrieval result.



Query: Human, id 43
Position: 1 Obj id: 43 Dist: 0
Position: 2 Obj id: 126 Dist: 0.539591
Position: 3 Obj id: 244 Dist: 0.66111
Position: 4 Obj id: 39 Dist: 0.690106
Position: 5 Obj id: 92 Dist: 0.946047
Position: 6 Obj id: 37 Dist: 0.964709
Position: 7 Obj id: 46 Dist: 0.991136
Position: 8 Obj id: 389 Dist: 1.00852
Position: 9 Obj id: 8 Dist: 1.03409
Position: 10 Obj id: 36 Dist: 1.06029
Position: 11 Obj id: 136 Dist: 1.06193
Position: 12 Obj id: 104 Dist: 1.08513
Position: 13 Obj id: 47 Dist: 1.12764
Position: 14 Obj id: 124 Dist: 1.13938
Position: 15 Obj id: 206 Dist: 1.19011
Position: 16 Obj id: 189 Dist: 1.19191
Position: 17 Obj id: 321 Dist: 1.2312
Position: 18 Obj id: 85 Dist: 1.2519
Position: 19 Obj id: 385 Dist: 1.26574
Position: 20 Obj id: 217 Dist: 1.27203
Query: Teddy bear, id 249
Position: 1 Obj id: 249 Dist: 0.30103
Position: 2 Obj id: 115 Dist: 0.539591
Position: 3 Obj id: 107 Dist: 0.588046
Position: 4 Obj id: 23 Dist: 0.650515
Position: 5 Obj id: 162 Dist: 0.738561
Position: 6 Obj id: 85 Dist: 0.752575
Position: 7 Obj id: 13 Dist: 0.765739
Position: 8 Obj id: 259 Dist: 0.811625
Position: 9 Obj id: 303 Dist: 0.89967
Position: 10 Obj id: 312 Dist: 0.972241
Position: 11 Obj id: 217 Dist: 1.10476
Position: 12 Obj id: 121 Dist: 1.16111
Position: 13 Obj id: 66 Dist: 1.18829
Position: 14 Obj id: 291 Dist: 1.19723
Position: 15 Obj id: 195 Dist: 1.23784
Position: 16 Obj id: 335 Dist: 1.23784
Position: 17 Obj id: 293 Dist: 1.26638
Position: 18 Obj id: 314 Dist: 1.28702
Position: 19 Obj id: 137 Dist: 1.30426
Position: 20 Obj id: 43 Dist: 1.31162

Fig. 4. Retrieval results when only the subset of human and Teddy bear models is considered: top 20 retrieved items are displayed. Colored border means successful hit.

Table 1. Performance comparison of the proposed approach (CFV), using Laplacian matrices (Lapl), using Adjacency matrices (Adj) and the approach by Tung et al.

Measure	CFV	Lapl	Adj	Tung et al.	Measure	CFV	Lapl.	Adj.	Tung et al.
P20	0.42	0.22	0.20	0.71	R20	0.42	0.22	0.20	0.71
P40	0.29	0.16	0.16	0.41	R40	0.57	0.33	0.31	0.83
P60	0.22	0.14	0.14	0.29	R60	0.66	0.43	0.41	0.87
P80	0.18	0.12	0.12	0.23	R80	0.72	0.50	0.48	0.90
PS1	1	1	1	1	ADR	0.58	0.36	0.31	0.86
PS2	0.65	0.33	0.17	0.98	AVG	77	125	132	31
					LPR	0.48	0.25	0.17	0.74

6 Concluding remarks

In this paper we presented a new shape retrieval method using complex Fiedler vectors of Hermitian property matrices. These matrices represent the graph based shape descriptor, in which both topology and geometry of the models is encoded, such that the matrix mimics the Laplacian matrix from a spectral point of view. By combining partial and complete retrieval, the framework shows strong performance on a benchmark of 400 watertight 3D models with respect to other spectral based retrieval techniques.

It is one of our interests in the future to test the proposed approach on other domains. After the Hermitian matrix has been constructed with features of a certain database object (regardless its modality), the framework is fixed. It is interesting to see how this approach extends to retrieval of other multimedia objects, such as music or 2D images. Furthermore, we intend to investigate the how this method can be used as an efficient prefiltering step in combination with other methods to in order to increase performance and save computing time.

Acknowledgements This research was supported by FP7 ICT project 214993 FOCUS K3D and FP6 IST projects 511572-2 PROFI and 506766 AIM@SHAPE.

References

1. Biasotti, S., Falcidieno, B., Frosini, P., Giorgi, D., Landi, C., Marini, S., Patane, G., Spagnuolo, M.: 3D shape description and matching based on properties of real functions. In: Eurographics - Tutorials. (2007)
2. Tangelder, J.W., Veltkamp, R.C.: A survey of content based 3d shape retrieval methods. In: Shape Modeling International. (2004)
3. Biasotti, S., Giorgi, D., Spagnuolo, M., Falcidieno, B.: Reeb graphs for shape analysis and applications. Theoretical Computer Science (2007) To appear
4. Hilaga, M., Shinagawa, Y., Kohmura, T., Kunii, T.L.: Topology matching for fully automatic similarity estimation of 3d shapes. In: SIGGRAPH. (2001)
5. Marini, S., Spagnuolo, M., Falcidieno, B.: From exact to approximate maximum common subgraph. In: Graph-based Representations in Pattern Recognition. (2005)
6. Tierny, J., Vandeborre, J.P., Daoudi, M.: Reeb chart unfolding based 3D shape signatures. In: Eurographics. (2007)
7. Tung, T., Schmitt, F.: Augmented reeb graphs for content-based retrieval of 3d mesh models. In: Shape Modeling International. (2004)
8. Demirci, M.F., van Leuken, R.H., Veltkamp., R.C.: Indexing through laplacian spectra. Computer Vision and Image Understanding **110**(3) (2008) 312–325
9. Sengupta, K., Boyer, K.L.: Modelbase partitioning using property matrix spectra. Computer Vision and Image Understanding **70**(2) (1998) 177–196
10. Shokoufandeh, A., Macrini, D., Dickinson, S., Siddiqi, K., Zucker, S.: Indexing hierarchical structures using graph spectra. Pattern Analysis and Machine Intelligence **27**(7) (2005)
11. Wilson, R.C., Hancock, E.R., Luo, B.: Pattern vectors from algebraic graph theory. Pattern Analysis and Machine Intelligence **27** (2005) 1112–1124
12. Zhu, P., Wilson, R.C.: A study of graph spectra for comparing graphs. In: British Machine Vision Conference. (2005)
13. Biasotti, S., Falcidieno, B., Spagnuolo, M.: Extended reeb graphs for surface understanding and description. In: Discrete Geometry for Computer Imagery. (2000)
14. Mortara, M., Patane, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. Algorithmica **38**(1) (2004) 227–248
15. Symonova, O., De Amicis, R.: Shape analysis for augmented topological shape descriptor. In: Eurographics. (2007)
16. Mohar, B.: The laplacian spectrum of graphs. Graph Theory, Combinatorics and Applications **2** (1991) 871–898
17. Merris, R.: Laplacian matrices of graphs: a survey. Linear Algebra and its Applications **197**(1) (1994) 143–176
18. Qiu, H., Hancock, E.R.: Graph partition for matching. In: Graph-based Representations in Pattern Recognition. (2003)
19. Veltkamp, R.C., ter Haar, F.B.: SHREC2007: 3D Shape Retrieval Contest. Technical Report UU-CS-2007-015, Utrecht University (2007)