Noname manuscript No.
(will be inserted by the editor)

Two-dimensional shape retrieval using the distribution of extrema of Laplacian eigenfunctions

Dongmei Niu · Peer-Timo Bremer · Peter Lindstrom · Bernd Hamann · Yuanfeng Zhou · Caiming Zhang

Received: date / Accepted: date

Abstract We propose a new method using the distribution of extrema of Laplacian eigenfunctions for two-dimensional (2D) shape description and matching. We construct a weighted directed graph, which we call signed natural neighbor graph, to represent a Laplacian eigenfunction of a shape. The nodes of this sparse graph are the extrema of the corresponding eigenfunction, and the edge weights are defined by signed natural neighbor coordinates derived from the local spatial arrangement of extrema. We construct the signed natural neighbor graphs defined by a small number of low-frequency Laplacian eigenfunctions of a shape to describe it. This shape descriptor is invariant under rigid transformations and uniform scaling, and is also insensitive to minor boundary deformations. When using our shape descriptor for matching two shapes, we determine their similarity by comparing the graphs induced by corresponding Laplacian eigenfunctions of the two shapes. Our experimental shape-matching results demonstrate that our method is effective for 2D shape retrieval.

Keywords Shape retrieval · Shape matching · Shape descriptor · Laplace operator · Signed natural neighbor graph · Graph matching

Dongmei Niu acknowledges fellowship support from the China Scholarship Council (CSC). Caiming Zhang appreciates the supports from the National Nature Science Foundation of China (61373078) and NSFC Joint Fund with Guangdong (U1201258).

D. Niu · Y. Zhou · C. Zhang
School of Computer Science and Technology, Shandong University, 1500 Shunhua Road, Jinan 250101, China
E-mail: dongmei_19881123@hotmail.com

Y. Zhou
E-mail: yfzhou@sdu.edu.cn

C. Zhang
E-mail: czhang@sdu.edu.cn

P.-T. Bremer · P. Lindstrom
Lawrence Livermore National Laboratory, 7000 East Avenue, L-422, Livermore, CA 94551, USA
E-mail: bremer5@llnl.gov

P. Lindstrom
E-mail: pl@llnl.gov

B. Hamann
Institute for Data Analysis and Visualization, Department of Computer Science, University of California, Davis, One Shields Avenue, Davis, CA 95616, USA
E-mail: hamann@cs.ucdavis.edu

1 Introduction

Two-dimensional (2D) shape retrieval is an important problem in computer vision and multimedia processing. Two major components of a shape retrieval algorithm are shape description and shape matching. A good shape descriptor should have the power of discriminating shapes from different classes. Besides that, a descriptor should not only be invariant under rigid transformations and uniform scaling, but also be insensitive to minor boundary deformations.

A deformation-invariant shape descriptor can be derived from the intrinsic properties of a shape. Since geodesic inner distance is related to the intrinsic property of a shape, geodesic inner distance is widely used in describing shapes [11, 19]. A drawback of this distance metric is that it is sensitive to local topology changes [15]. The Laplace operator of a shape is intrinsically determined by the shape itself, which is

also an isometric invariant being independent to the shape's representation. Eigenvalues and eigenfunctions of this operator are isometric invariants as well. Reuter et al. [29] proposed a shape descriptor, which they called ShapeDNA, as a vector consisting of a number of normalized Laplacian eigenvalues of a shape. This descriptor captures the major information of a shape but does not determine a shape uniquely up to isometry [8, 30]. Laplacian eigenfunctions provide insights into the structure and morphology of a shape [17, 28], which makes it possible to use these eigenfunctions to describe a shape. However, two problems of working with eigenfunctions are: (i) two eigenfunctions that correspond to close eigenvalues can switch their order for reasons such as numerical instabilities or deformations of geometry and (ii) the signs of the eigenfunctions are undefined [27].

We consider two 2D shapes to be similar if they have similarly shaped boundaries. The Laplacian eigenfunctions of a 2D shape are determined by the shape's boundary, which makes it possible to use the Laplacian eigenfunctions to discriminate shapes with different boundaries. We present a method using Laplacian eigenfunctions for 2D shape description and shape matching. We introduce a weighted directed graph, which we call signed natural neighbor graph (SNNG), to represent a Laplacian eigenfunction of a 2D shape. The nodes of this graph are the extrema of the eigenfunction, and the edge weights are defined by signed natural neighbor coordinates derived from the local spatial arrangement of extrema. The spatial distribution of the extrema captures global properties of the shape, and the edge weights reflect local shape features. We construct the SNNGs for a small number of low-frequency Laplacian eigenfunctions of a shape to describe it. This shape descriptor is invariant to rigid transformations and uniform scaling, and insensitive to minor boundary deformations. By assigning signs to the edge weights, our method handles the sign problem of the eigenfunctions. When matching two shapes, we first compute a best one-to-one match between the two shapes' Laplacian eigenfunctions to eliminate the ordering problem of eigenfunction. Based on the best one-to-one match, our method computes the dissimilarity between two shapes as the sum of the dissimilarities between the SNNGs of these shapes' corresponding eigenfunctions.

Boundary conditions are needed when computing eigenvalues and eigenfunctions of a 2D shape's Laplace operator. Our method uses Neumann boundary conditions because a small hole in the surface does not change the spectrum resulting from these conditions relative to using Dirichlet boundary conditions [29]. The Neumann boundary conditions that we are using force the derivatives of these eigenfunctions in the normal direction of the boundary curve to be zero. We refer to the Laplacian eigenfunctions computed with Neumann boundary conditions as Laplacian eigenfunctions.

The Laplacian eigenvalues cannot determine the shapes uniquely as nonisometric shapes exist that are also isospectral [8, 30]. However, there exist nonisometric, isospectral shapes whose Laplacian eigenfunctions have different distributions of extrema. Our method only depends on the Laplacian eigenfunctions, and it is independent of the Laplacian eigenvalues except for ordering.

The main contributions of this paper are: (i) introducing a novel 2D shape descriptor that is invariant to rigid transformations and uniform scaling and insensitive to minor boundary deformations; (ii) presenting a method using Laplacian eigenfunctions for 2D shape description and shape matching that handles the sign and ordering problems of eigenfunctions; (iii) converting the function comparison problem into a graph matching problem.

2 Related work

Many algorithms have been proposed in literature for describing and matching the shapes of 2D objects. These algorithms can be generally classified into two categories: contour-based and region-based methods [42]. Contour-based methods typically extract features only from boundary information, while region-based methods exploit all pixels within a shape. Contour-based methods are more popular than region-based methods as humans are thought to discriminate shapes mainly by their contour features [33]. However, region-based methods are more robust as they use the entire shape information [33].

There are many contour-based methods for shape description and matching. Mokhatarian et al. [20] proposed a descriptor that uses the maxima of curvature zero-crossing contours of the curvature scale space (CSS) image to describe a shape. This descriptor does not work well for convex shapes as there is no curvature zero crossing for convex shapes. Cui et al. [5] improved the CSS descriptor and applied it to remote sensing image registration. Adamek and O'Connor [1] presented a multiscale convexity concavity (MCC) descriptor using a 2D matrix to represent contour convexities and concavities at different scales. However, the optimal parameter of each scale is hard to determine. The curve normalization (CN) method proposed in [15] uses the second-order geometric moments and a number of normalized curves to describe a shape, which captures

both global and local features of the shape. The method described in [7] represents the contour of a shape by line segments, and matches two contours by aligning their segments. Grauman and Darrell [9] presented a method that represents a shape by a set of local features on the contour of the shape and compares two shapes by computing the minimum earth mover's distance (EMD) cost between the shapes' features. Some methods use geometric relationship between boundary sample points to describe shapes. One method of this type is shape context (SC) [2], which computes the distance between two shapes based on matching their boundary sample points. Ling and Jacobs [19] extended SC [2] to inner-distance shape context (IDSC) by defining the inner distance between landmark points within the shape silhouette. A limitation of this method is that the inner distance is sensitive to local topology changes. The contour points distribution histogram (CPDH) [35] descriptor represents a shape based on the distribution of boundary points under polar coordinates. The dissimilarity between two shapes is obtained by the EMD metric between their CPDHs. Xu et al. [40] defined a 2D shape descriptor, named contour flexibility, which depicts the deformable potential at each point on a curve. This descriptor can extract both local and global features of a shape. Isaacs and Roberts [12] proposed a spectral method that describes a 2D shape based on two metrics defined on the eigenfunctions of the weighted diffusion operator of the shape's contour. They demonstrated the ability of this shape descriptor to discriminate different shapes by clustering several prototypical 2D shapes. Wang et al. [39] proposed a method using a fixed number of sample points to represent the contour of an object. Each sample point is associated with a height function. This method is invariant to geometric transformations and insensitive to nonlinear deformations.

Many region-based methods have been proposed. The shape descriptor proposed in [13] describes a shape based on the magnitudes of Zernike moments (ZMs). Taking both the magnitude and phase coefficients of ZMs into account, Li et al. [18] defined a descriptor called invariant ZM descriptor. Zhang and Lu [41] proposed a generic Fourier descriptor that is extracted from a spectral domain by applying the 2D Fourier transform on polar-raster sampled shape image. This descriptor captures a shape's finer features in both radial and circular directions. The method described in [32] represents a shape as a graph implied by its skeleton. A drawback of this method is that the hierarchical relationships among parts of the shape can be violated in the matching process. Ion et al. [11] used normalized histograms of the eccentricity transform as descriptor for shape matching. This descriptor is robust to articulation because the eccentricity transform is computed based on geodesic distance. Shekar and Pilar [33] presented a decision level fused local morphological pattern spectrum (PS) and local binary pattern (LBP) approach for shape representation and classification. The distance between two shapes is computed based on the EMD metric. Laplace operator is an isometric invariant that makes it possible to describe a shape based on this operator. ShapeDNA [29], which is a deformation-invariant descriptor, describes a 2D shape as a number of the shape's normalized Laplacian eigenvalues. However, ShapeDNA cannot determine the shapes uniquely as non-isometric but isospectral shapes exist [8, 30].

3 Mathematical background

Let f be a twice-differentiable real function on Euclidean space. The Laplace operator Δ for f is a second-order differential operator,

$$\Delta f = \mathrm{div}(\mathrm{grad}(f)), \tag{1}$$

with $\mathrm{grad}(f)$ being the gradient of f and $\mathrm{div}(\mathrm{grad}(f))$ the divergence of the gradient [3].

The Helmholtz equation [38], which is also known as the Laplacian eigenvalue problem, is given as

$$-\Delta f = \lambda f. \tag{2}$$

The eigenvalues λ and eigenfunctions f represent the spectrum of the operator and the natural harmonics of a shape, respectively. The exact solutions to the Laplacian eigenvalue problem (Equation (2)) are only known for a limited number of domains, such as the rectangular domain and the disk domain. We approximate the solutions by interpolating the eigenvectors of a discrete Laplace operator defined at the vertices of a triangulation of the domain.

Eigenvalues and eigenfunctions of the Laplace operator have some properties that make them interesting for some shape processing tasks, like shape retrieval and shape matching. These properties are as follows:

- The Laplacian eigenvalues and eigenfunctions are invariant to rigid transformations of the shape. The Laplace operator uses the trace of the Hessian matrix [29] that is invariant to translation and rotation. Accordingly, the Laplacian eigenvalues and eigenfunctions are invariant to translation and rotation. Scaling a shape by a factor a uniformly results in scaled Laplacian eigenvalues by the factor $1/a^2$ [29]. However, relative positions of the eigenfunctions'

extrema are invariant to uniform scaling, and may be utilized to construct an invariant descriptor. Experiments discussed in [28] indicated that low-frequency Laplacian eigenfunctions are relatively insensitive to minor boundary deformations.

– There are n Laplacian eigenvalues for a triangle mesh with n vertices. These eigenvalues are non-negative that can be put into ascending order as follows:

$$0 \leq \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n. \quad (3)$$

The corresponding n eigenfunctions define n functions over the mesh, which occur in order of increasing frequency [37]. Low-frequency eigenfunctions correspond to dominant large-scale features of the shape while high-frequency eigenfunctions are related to finer details [24]. For most applications involved in spectral methods, eigenfunctions with low frequency are sufficient. For eigenvalues and eigenfunctions computed with Neumann boundary conditions, the smallest eigenvalue is zero and the corresponding eigenfunction is a constant function.

4 Method overview

The properties of Laplacian eigenfunctions introduced in section 3 make it possible to use these eigenfunctions to intrinsically describe a shape. We present a method for 2D shape description and matching based on characterizing and comparing low-frequency Laplacian eigenfunctions of the shapes. To compare two eigenfunctions, our method compares the distribution of these eigenfunctions' extrema. The distribution of Laplacian eigenfunctions' extrema is a good shape descriptor because it is insensitive to minor boundary deformations and captures important properties of the shape and the eigenfunctions. In addition, comparing the distribution of extrema is much easier than comparing two eigenfunctions directly. Fig. 1 shows the flowchart of our method.

The input of our method is a 2D shape database. Considering every shape in this database as a query, our method orders the other shapes in the database in ascending order of dissimilarity. To summarize our approach on a high level, it consists of these major steps:

Step 1. Triangulating the shapes in the database. To approximate the Laplacian eigenfunctions, our method first triangulates the shapes. To ensure a high degree of numerical accuracy of the eigenfunctions, a high-quality triangulation without skinny triangles is needed. Our method uses the "Triangle" program [34], which is a quality mesh generator, to triangulate the shapes.

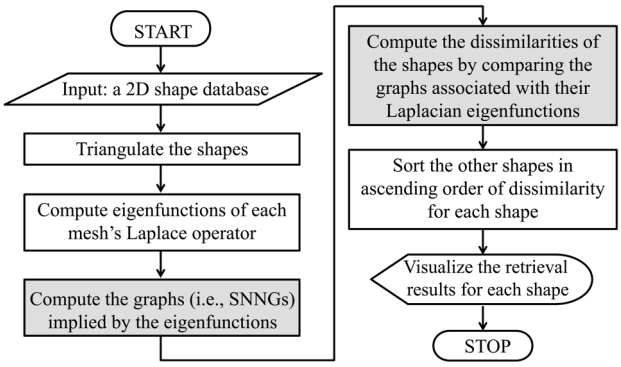


Fig. 1 Flowchart of our method. The boxes with light-gray background indicate new components that are crucial for our approach and represent innovations.

Step 2. Computing the low-frequency Laplacian eigenfunctions of the shapes. To compute the Laplacian eigenfunctions with high accuracy, we use the "ShapeDNA-tria" software [27, 29] to approximate the eigenfunctions with Neumann boundary conditions using cubic triangular finite elements. This approach is rather insensitive to the triangulation of a shape, as long as (i) the number of triangles used is sufficiently large and (ii) the angles in the triangulation satisfy a specific angle criterion [29].

Step 3. Shape descriptor construction. We construct a graph, which we call SNNG, to characterize the distribution of a Laplacian eigenfunction's extrema. We describe each shape as the SNNGs derived from the shape's low-frequency Laplacian eigenfunctions. The extrema of an eigenfunction are computed from a piecewise linear approximation of the eigenfunction, which limits the locations of extrema to vertices of the triangulation.

Step 4. Comparing the shapes. For each pair of shapes, we compute their dissimilarity as the sum of the dissimilarities between the SNNGs of their corresponding Laplacian eigenfunctions.

Step 5. Computing the retrieval results for each shape by ordering the other shapes in ascending dissimilarity.

Two most important components of a shape retrieval method are shape description and shape matching, i.e., Step 3 and Step 4 of our method. Details of these two steps are provided in the following two sections.

5 Shape description

Since Laplacian eigenfunctions are invariant to rigid transformations and insensitive to minor boundary deformations, and the relative positions of their extrema are invariant to uniform scaling of the shape, it is possible to use these eigenfunctions to describe a 2D shape. A finite number of Laplacian eigenfunctions is sufficient to discriminate different shapes, because low-frequency eigenfunctions are related to dominant large-scale features of a shape and high-frequency eigenfunctions correspond to finer details. Our method uses the first k non-trivial eigenfunctions to describe a shape. We construct a weighted directed graph, which we call SNNG, to characterize the distribution of an eigenfunction's extrema. A function defined on a shape has two types of extrema: maxima and minima. The Laplacian eigenfunctions that we have computed are all discrete functions in the sense that they are defined at the vertices of a shape's triangulation. For a discrete function of this type, a vertex is called a maximum/minimum when its function value is larger/smaller than the function values of its immediate edge-connected neighbor vertices.

Because our discrete representation constrains the eigenfunction extrema to lie at mesh vertices, these extrema may deviate from the true extrema given by the corresponding continuous eigenfunction. To estimate the error in position of the computed extrema, we performed computational experiments with many triangulations for the same shape, using different mesh densities; the true positions of the extrema were estimated using a dense triangulation. Our experiments indicated that the positional error is proportional to the size of the triangles in the coarse mesh, which implies that an arbitrary error tolerance can be achieved by using a triangulation with small-enough triangles. We also used different triangulation methods [34] for triangulating a shape, ensuring that the resulting triangulations had roughly the same density. It turned out that the error in position of the extrema and the shape matching results are largely insensitive to changes in the triangulation.

The worst time complexity of extracting the SNNG for a Laplacian eigenfunction is $\mathcal{O}(m \times n^3 \log(n))$, where n is the number of vertices in the triangulation and m is the eigenfunction's number of extrema, see section 7.4. The density of a triangulation affects the time required to extract the SNNG. We triangulated the shapes shown in section 7 with approximately 3,500 vertices. Based on our experiments, the average error of the locations of the computed extrema is smaller than 5.8% of the average edge length of the triangulation, and the matching results do not change at all when using triangulations with higher density.

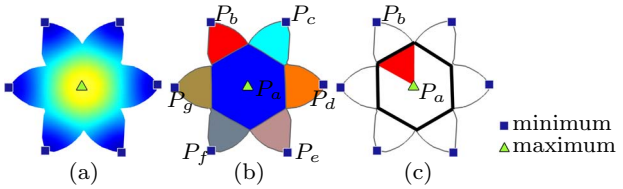


Fig. 2 The seventh Laplacian eigenfunction of a starfish shape. (a) The eigenfunction and its extrema. (b) The seven cells $C_a, C_b, \ldots, C_g$ for the seven extrema $P_a, P_b, \ldots, P_g$ defined by Definition 1 shown in different colors. (c) The subcell C_{ab} for the two extrema P_a and P_b defined by Definition 2. The region shown in red is C_{ab}, and the region shown with the black polygon is the cell C_a for P_a.

Let $\boldsymbol{P} = \{P_1, P_2, \ldots, P_n\}$ be the extrema of a function F defined over a planar shape S.

Definition 1 Following the definition of Voronoi cell [22], we define the cell for an extremum P_i as

$$C_i = \{X \in S : D(X, P_i) \leq D(X, P_j), j = 1, 2, \ldots, n\}, \quad (4)$$

where $D(X, Y)$ denotes the geodesic inner distance between the two surface points X and Y, i.e., the length of the shortest path between X and Y within the shape S.

We used the algorithm described in [26] to compute the cells associated with the extrema of a Laplacian eigenfunction. This algorithm computes the cells based on the geodesic inner distances between mesh vertices and extrema. We computed these distances using the toolbox described in [25]. This toolbox computes the shortest path between two mesh vertices as a set of mesh edges defining a polyline of minimum length. Fig. 2(a) shows the seventh Laplacian eigenfunction of a starfish shape. Minima and maxima of this eigenfunction are denoted by squares and triangles, respectively. In Fig. 2(b), we use different colors to show the cells defined by Definition 1 for these extrema.

Definition 2 Following the definition of a natural neighbor [36], we define the subcell C_{ij} for the extrema P_i and P_j as

$$C_{ij} = \{X \in S : D(X, P_i) \leq D(X, P_j) \leq D(X, P_k), \\ \forall k \notin \{i, j\}\}. \quad (5)$$

If $C_{ij} \neq \emptyset$, then we call the extremum P_j a natural neighbor of P_i.

Definition 3 Following the definition of natural neighbor coordinate [36], we define the natural neighbor coordinate of the extremum P_i with respect to the extremum P_j as

$$N_{ij} = \frac{|C_{ij}|}{|C_i|}, \quad (6)$$

where $|C|$ represents the area of cell C. We note that $N_{ij} \neq N_{ji}$.

The maximum P_a of the eigenfunction shown in Fig. 2 has six natural neighbors, which are the six minima. In Fig. 2(c), the region shown with the black polygon is C_a, which represents the set of points on the shape that are closest to P_a. The region shown in red is the subcell C_{ab}, which represents the set of points on the shape that are closest to P_a and second closest to P_b. The natural neighbor coordinate N_{ab} of P_a with respect to P_b is the area ratio of C_{ab} to C_a.

With the above definitions, we can now formally define an SNNG for a function F.

Definition 4 The SNNG for a function F defined over a 2D shape is a weighted directed graph. We use the notation $G(\boldsymbol{P}, W)$ to represent this graph, where $\boldsymbol{P}$ is the node set consisting of the extrema of F, and W is the weight matrix with W_{ij} being the weight of the directed edge (P_i, P_j). There are two types of extrema: maxima and minima. When comparing the SNNGs of two eigenfunctions, we define the weight of an edge as a signed number in order to avoid two extrema of the same type (i.e., two maxima or two minima) in an SNNG being mapped to two extrema of different types (i.e., a maximum and a minimum) in the other SNNG. For the directed edge (P_i, P_j), the following holds: If P_i and P_j are both maxima or minima, then $W_{ij} = N_{ij}$; otherwise, $W_{ij} = -N_{ij}$. If P_i and P_j are not adjacent, i.e., $N_{ij} = 0$, then $W_{ij} = 0$.

Compared with an unweighted graph, the absolute value of the weight W_{ij} denotes the connection strength of a neighbor P_j relative to the other neighbors of P_i, which can be indicative of local features of a shape. In contrast to an undirected graph, there are two directed edges between the nodes P_i and P_j, which are (P_i, P_j) and (P_j, P_i). The weights of (P_i, P_j) and (P_j, P_i) are W_{ij} and W_{ji}, respectively. The node P_j may be far away from P_i relative to the other neighbors of P_i, which results in a small value of $|W_{ij}|$; whereas the node P_i may be close to P_j relative to the other neighbors of P_j, which results in a large value of $|W_{ji}|$. By using a directed graph we can capture such asymmetric relationships.

The SNNGs for the shapes' Laplacian eigenfunctions are invariant to rigid transformations and uniform scaling and relatively insensitive to minor boundary deformations. The Laplacian eigenfunctions and the distributions of their extrema are invariant to rigid transformations and relatively insensitive to minor boundary deformations. The relative positions of the Laplacian eigenfunctions' extrema are invariant to the uniform scaling of the shape. We compute cells C_i, defined by Definition 1, and subcells C_{ij}, defined by Definition 2, for these eigenfunctions' extrema, using the geodesic inner distance for their constructions. As a consequence, the cell structures of two isometric shapes' Laplacian eigenfunctions are the same. Further, the edge weights of the SNNGs and the SNNGs themselves have the same invariance property.

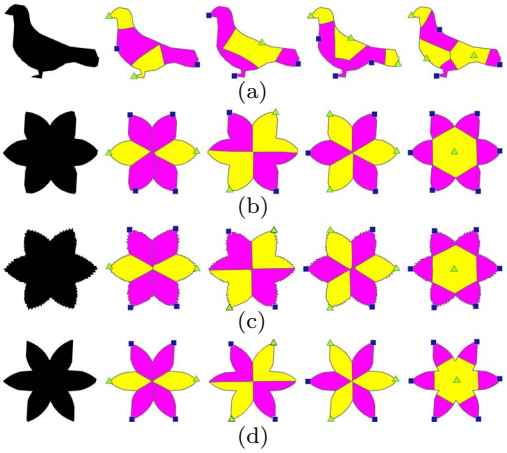


Fig. 3 Four shapes and the cell structures of their Laplacian eigenfunctions with indices ranging from four to seven based on Definition 1. The cells for the maxima and minima are shown in different colors. The eigenfunctions' cell structures of the three starfish shapes ((b)-(d)) are similar and are quite different from those of the pigeon shape ((a)).

Fig. 3 shows the cells for the Laplacian eigenfunctions' extrema defined by Definition 1 of a pigeon shape (Fig. 3(a)) and three near-isometric starfish shapes (Fig. 3(b)-(d)). The indices of these eigenfunctions range from four to seven. We use different colors to show the cells for the minima and maxima. Compared with the starfish shape shown in Fig. 3(b), the shape shown in Fig. 3(c) has more noise on the boundary, and the shape shown in Fig. 3(d) has a small deformation. Looking at Fig. 3, the three starfish shapes' corresponding Laplacian eigenfunctions have similar cell structures that are quite different from those of the pigeon shape's eigenfunctions, which means it is possible to use the Laplacian eigenfunctions to distinguish these shapes.

In Fig. 4, we show the Laplacian eigenfunctions with indices ranging from four to seven and their associated SNNGs of a pigeon shape and a starfish shape, respectively. We use different colors to show the edges connecting two extrema of the same type (i.e., both extrema are maxima or minima) and of different types (i.e., the two extrema are a maximum and a minimum). The width of a directed edge represents the absolute value of its weight. The thicker an edge is, the larger the absolute value of this edge's weight is.

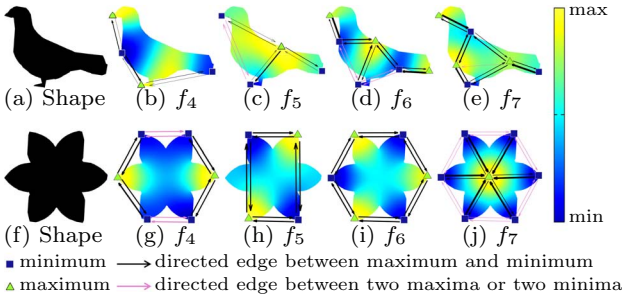


Fig. 4 Laplacian eigenfunctions with indices ranging from four to seven and their associated SNNGs of a pigeon shape ((a)) and a starfish shape ((f)). Edges that connect two extrema of the same type and different types are shown in different colors. The width of an edge represents the absolute value of the edge's weight. The thicker an edge is, the larger the absolute value of its weight is.

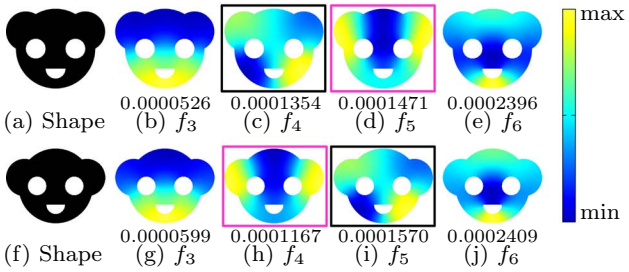


Fig. 5 Two near-isometric face shapes and their Laplacian eigenfunctions with indices ranging from three to six. The corresponding Laplacian eigenvalue of each eigenfunction is shown below the eigenfunction. Compared with the eigenfunctions of shape (f), the fourth and fifth eigenfunctions ((c) and (d)) of shape (a), the corresponding eigenvalues of which are close, switch their order.

An important property of the SNNG is the fact that the absolute values of the weights of a node P_i's outgoing edges sum to one:

$$\sum_{j=1}^{n} |W_{ij}| = 1. \tag{7}$$

If P_i and P_j are not adjacent, then $W_{ij} = 0$. The non-zero weights of the edges are intended to reflect the relationships between a node and its natural neighbors.

6 Shape comparison

Shape comparison is a crucial component of a shape retrieval method. Our method compares two shapes by comparing the SNNGs implied by the two shapes' first k non-trivial Laplacian eigenfunctions. The order of the k SNNGs is given by the order of their corresponding eigenvalues.

Our method compares two shapes by solving two nested problems. The outer problem is concerned with computing the best one-to-one match between the two shapes' k eigenfunctions using the Hungarian algorithm based on a dissimilarity matrix Z of the eigenfunctions. The total dissimilarity of the matched eigenfunctions is the dissimilarity between the two shapes. The inner problem is dealing with the issue of computing the dissimilarity matrix Z, i.e., the pairwise dissimilarity Z_{ij} of the SNNGs of a shape's i-th eigenfunction and another shape's j-th eigenfunction. More details are provided in sections 6.1 and 6.2, respectively.

6.1 Comparing two shapes

We order the Laplacian eigenfunctions using the order given by their associated eigenvalues, ordered in increasing magnitude. Since the first few eigenfunctions of similar shapes tend to be similar, it is reasonable to compare two shapes based on comparing their corresponding eigenfunctions. However, a problem arising when working with eigenfunctions is that two eigenfunctions of the same shape can switch their order for reasons such as numerical error in the computed eigenvalues or small geometry deformations [27]. Fig. 5 shows an example of this problem. The corresponding Laplacian eigenvalue of each eigenfunction is shown below the eigenfunction. Compared with the eigenfunctions of the shape shown in Fig. 5(f), the fourth and fifth eigenfunctions (shown in Fig. 5(c) and Fig. 5(d)) of the shape shown in Fig. 5(a), the corresponding eigenvalues of which are close, switch their order. When comparing two shapes, our method computes a best one-to-one match between the two shapes' Laplacian eigenfunctions in order to eliminate this problem.

We compare two shapes by computing the dissimilarity between them. We determine the dissimilarity between two shapes as the sum of the dissimilarities between the two shapes' corresponding Laplacian eigenfunctions with indices ranging from two to $k+1$. [1] We compute the dissimilarity between two eigenfunctions as the dissimilarity between their associated SNNGs. Let A and B be two 2D shapes, the dissimilarity between A and B is

$$Dis(A,B) = \sum_{i=2}^{k+1} d\left(A_i, B_{M(i)}\right), \tag{8}$$

where M is the best one-to-one match that maps the i-th Laplacian eigenfunction of shape A to the $M(i)$-th Laplacian eigenfunction of shape B, A_i and $B_{M(i)}$ represent the SNNGs of the i-th and $M(i)$-th eigenfunctions of A and B, respectively, and $d(X,Y)$

[1] We do not consider the first Laplacian eigenfunction because this eigenfunction is constant.

denotes the dissimilarity between the two SNNGs X and Y. Details about computing the dissimilarity d between two SNNGs are provided in section 6.2, and Equation (10) defines how to compute d.

Given two shapes A and B, the computation of the best one-to-one match M between their Laplacian eigenfunctions is a linear assignment problem [14]. The goal is to find an optimal assignment M that minimizes the sum of the dissimilarities between the matched eigenfunctions. This sum is our measure for defining the dissimilarity of A and B, see Equation (8).

The Hungarian algorithm solves the linear assignment problem in polynomial time [14, 21]. Given an $n \times n$ cost matrix Z with Z_{ij} being the cost of assigning the i-th resource to the j-th task, the Hungarian algorithm finds an optimal assignment, i.e., a permutation $\pi(1), \pi(2), \ldots, \pi(n)$ of the integers $1, 2, \ldots, n$, that minimizes the total cost $\sum_{i=1}^{n} Z_{i\pi(i)}$. Our method uses the Hungarian algorithm to compute the best one-to-one match M between two shapes' Laplacian eigenfunctions. The input matrix Z of the Hungarian algorithm is the dissimilarity matrix between the two shapes' first k non-trivial eigenfunctions, and Z_{ij} is the dissimilarity between A's i-th eigenfunction and B's j-th eigenfunction. The computed permutation defines the best one-to-one match M, and the i-th eigenfunction of A is matched with the $M(i)$-th eigenfunction of B. Since only the Laplacian eigenfunctions that have very similar eigenvalues can switch their order, we speed up our algorithm by enforcing that the i-th eigenfunction of a shape can only be mapped to the $(i-1)$-th, i-th or $(i+1)$-th eigenfunction of another shape. If $|i-j| > 1$, we set Z_{ij} to a large value.

6.2 Comparing two Laplacian eigenfunctions

We compute the dissimilarity between two shapes as the sum of the dissimilarities between the two shapes' Laplacian eigenfunctions. It is difficult to compare two functions defined on two different domains directly. Our method compares two eigenfunctions based on comparing the distribution of the two eigenfunctions' extrema, i.e., the two eigenfunctions' SNNGs.

The graph edit distance is commonly used to measure the dissimilarity between two graphs. It is defined as the minimum cost of transforming one graph to another [6]. However, the computation of graph edit distance is generally NP-hard. Inspired by the definition of graph edit distance, our method greedily computes the dissimilarity between two SNNGs in polynomial time by solving three nested tasks. The outer task is concerned with computing the two SNNGs' dissimilarity based on a heuristic mapping between the SNNGs' nodes. The middle task deals with the issue of computing this heuristic mapping based on the dissimilarities between the two SNNGs' nodes. The inner task is concerned with computing the dissimilarity between two nodes. We define a feature vector for each node and compare two nodes using the Hungarian algorithm based on the dissimilarity matrix between the elements of the two nodes' feature vectors. We now discuss these tasks in more details.

6.2.1 Computing the dissimilarity between two nodes

For a node in an SNNG, we use the weights of its outgoing edges as its feature vector. According to Equation (7), the sum of the absolute values of a feature vector's elements is one. Two nodes with neighbors that are spatially similarly distributed have similar feature vectors. Conversely, the feature vectors of two nodes whose neighbors are not distributed similarly are dissimilar. We compare two nodes in two steps. First, we use zero padding to ensure that the feature vectors of the two nodes have the same size. Second, we compute the dissimilarity between the nodes by comparing their feature vectors using the Hungarian algorithm [14, 21]. In this case, $\mathcal{F}_{ij}$ of the input matrix $\mathcal{F}$ of the Hungarian algorithm is the absolute difference between the i-th element of one node's feature vector and the j-th element of another node's feature vector. The corresponding total cost is the dissimilarity between the two nodes. The time complexity of these two steps is $\mathcal{O}(p^3)$, where p is the number of neighbors of a node in an SNNG. Typically, the value of p is small.

6.2.2 Mapping the nodes of two SNNGs

Having computed the dissimilarity between each pair of nodes of two SNNGs, we compute a mapping between their nodes. We assume that the two SNNGs have m and n nodes, respectively, where $m \leq n$. There are $n \times (n-1) \times \ldots \times (n-m+1)$ possible mappings. It would be time-consuming if one enumerated all the mappings to pick the mapping that minimizes the cost to transform one SNNG to the other. We use a greedy algorithm to find a heuristic mapping where the nodes mapped to each other have similar adjacency relationship.

Let $G(\boldsymbol{P}, W)$ and $G'(\boldsymbol{P'}, W')$ be two SNNGs. If the nodes P_i in $\boldsymbol{P}$ and P'_j in $\boldsymbol{P'}$ are mapped to each other, then some neighbors of P_i and some neighbors of P'_j will also be mapped to each other. We compute the mapping between the nodes in $\boldsymbol{P}$ and $\boldsymbol{P'}$ using the following steps:

Step 1. We compute the most similar pair of nodes (P_i, P'_j) with $P_i \in \boldsymbol{P}$ and $P'_j \in \boldsymbol{P'}$, and map P_i and P'_j to each other.

Step 2. We insert all possible pairs of nodes (P_s, P'_t) into a min-heap H, where P_s and P'_t are neighbors of P_i and P'_j, respectively, and both P_s and P'_t have not yet been mapped.

Step 3. We delete the most similar pair of nodes (P_x, P'_y) from the min-heap H. If both P_x and P'_y have not yet been mapped, we map P_x and P'_y to each other and insert all possible pairs of these two nodes' neighbors to the min-heap H. We repeat this step until the min-heap H is empty.

Step 4. When all the nodes in $\boldsymbol{P}$ or $\boldsymbol{P'}$ have been mapped, we stop. Otherwise, we compute the most similar pair of nodes (P_i, P'_j), where $P_i \in \boldsymbol{P}$, $P'_j \in \boldsymbol{P'}$ and both P_i and P'_j have not yet been mapped. We map P_i and P'_j to each other, and we repeat Step 2 through Step 4.

In each iteration of the third step, the greedy algorithm finds the most similar neighbors of the nodes that have been mapped and maps the found neighbors. We restrict this search to neighbors of mapped nodes in order to better preserve adjacency relationships between corresponding nodes.

Algorithm 1 Computing the mapping between the nodes of two SNNGs

1: **Input:** Two SNNGs $G(\boldsymbol{P}, W)$ and $G'(\boldsymbol{P'}, W')$
2: **Output:** A mapping $\boldsymbol{\mathcal{M}}$ between $\boldsymbol{P}$ and $\boldsymbol{P'}$
3: $\boldsymbol{\mathcal{D}} \leftarrow CompareNodes(\boldsymbol{P}, \boldsymbol{P'})$ ▷ $\boldsymbol{\mathcal{D}}(P_i, P'_j)$ represents the dissimilarity between the nodes P_i and P'_j, where $P_i \in \boldsymbol{P}$ and $P'_j \in \boldsymbol{P'}$
4: **repeat**
5: $(U, U') \leftarrow (P_i, P'_j)$, where $\boldsymbol{\mathcal{D}}(P_i, P'_j) = \min(\boldsymbol{\mathcal{D}})$
6: Insert the node pair (U, U') into $\boldsymbol{\mathcal{M}}$
7: For all $V' \in \boldsymbol{P'}$, $\boldsymbol{\mathcal{D}}(U, V') \leftarrow \infty$
8: For all $V \in \boldsymbol{P}$, $\boldsymbol{\mathcal{D}}(V, U') \leftarrow \infty$
9: **repeat**
10: For all $P_s \in \boldsymbol{N_U}$ and for all $P'_t \in \boldsymbol{N_{U'}}$, insert (P_s, P'_t) into the min-heap H ▷ The set $\boldsymbol{N_U}$ consists of the neighbors of the node U. The key of (P_s, P'_t) in the min-heap H is $\boldsymbol{\mathcal{D}}(P_s, P'_t)$.
11: For all $V \in \boldsymbol{N_U}$, delete U from $\boldsymbol{N_V}$
12: For all $V' \in \boldsymbol{N_{U'}}$, delete U' from $\boldsymbol{N_{V'}}$
13: $(U, U') \leftarrow$ the *root* (P_x, P'_y) of H
14: Insert (U, U') into $\boldsymbol{\mathcal{M}}$; Delete the *root* from H
15: For all $V' \in \boldsymbol{P'}$, $D(U, V') \leftarrow \infty$
16: For all $V \in \boldsymbol{P}$, $D(V, U') \leftarrow \infty$
17: Delete (X, X') from H, where $X = U$ or $X' = U'$
18: **until** $H = \emptyset$
19: **until** $\min(\boldsymbol{\mathcal{D}}) = \infty$

Algorithm 1 shows pseudo-code for this approach, i.e., for computing the mapping between the node sets $\boldsymbol{P}$ and $\boldsymbol{P'}$. The procedure $CompareNodes(\boldsymbol{P}, \boldsymbol{P'})$ computes the dissimilarity between each node pair (P_i, P'_j), where $P_i \in \boldsymbol{P}$ and $P'_j \in \boldsymbol{P'}$. The worst case of Algorithm 1 occurs when all nodes of two SNNGs are isolated nodes, which is impossible. In this case, the time complexity of the procedure described in lines 4 through 19 is $\mathcal{O}(m^2 \log(m))$, where m is the number of the eigenfunction's extrema. Since the eigenfunction with eigenvalue λ_i has at most i nodal domains [4] and we compare two shapes based on comparing their first k non-trivial eigenfunctions, the value of m is small, usually smaller than 12.

6.2.3 Computing the dissimilarity between two SNNGs

Having computed the mapping between two SNNGs' nodes, we obtain the mapping between the edges of the SNNGs. We determine the dissimilarity between two SNNGs based on the cost of transforming the SNNG with larger number of nodes to the other one. In order to perform this transformation, we need two groups of operations: modifying an edge's weight and deleting a node.

To compare two SNNGs, we use Algorithm 1 that maps the nodes P_i and P_j of an SNNG to the nodes P'_u and P'_v of another SNNG, respectively. Let W_{ij} and W'_{uv} be the weights of the edges (P_i, P_j) and (P'_u, P'_v), respectively. To transform (P_i, P_j) to (P'_u, P'_v), we need to modify W_{ij} to W'_{uv}, and the corresponding cost is $|W'_{uv} - W_{ij}|$.

Since two SNNGs can have different numbers of nodes, we might need to delete some nodes when transforming one SNNG to another. When deleting a node, we also delete the edges incident to this node. The cost of deleting a node is zero. We explain this cost by considering the situation of comparing two similar shapes, A and B, where A has a missing part when compared with B. Let S_B be the subregion of B that can be mapped to A, and $B - S_B$ be the remaining region. When transforming an SNNG of B to an SNNG of A, one needs to delete the nodes located in $B - S_B$. As the frequencies of the Laplacian eigenfunctions increase, the number of extrema located in $B - S_B$ increases. If the cost of deleting a node is non-zero, then the dissimilarity between the SNNGs of the two shapes' corresponding eigenfunctions will go up by the increase in the eigenfunctions' frequencies, which will result in a large dissimilarity between A and B. In order to compute a meaningful dissimilarity between A and B, we define the cost of node deletion as zero. The cost of transforming an SNNG of B to an SNNG of A can be viewed as the dissimilarity between the subgraph on S_B and the SNNG of A. We use this cost to estimate the dissimilarity between the two SNNGs based on the area

ratio of the mapped subregion S_B to the whole region of B.

We now consider the situation of comparing two SNNGs, $G(\boldsymbol{P}, W)$ and $G'(\boldsymbol{P'}, W')$, having n and m, $n \geq m$, nodes, respectively. Let $\bar{G}(\bar{\boldsymbol{P}}, \bar{W})$ be the subgraph of G induced by the nodes that are mapped to the nodes of G'. We reorder the nodes in $\bar{\boldsymbol{P}}$ to make the node $\bar{P}_i$ of $\bar{G}$ and the node P'_i of G' mapped to each other. Let $cost(G, G')$ be the total cost of the operations needed to transform G to G', i.e., let $cost(G, G')$ represent the dissimilarity between the subgraph $\bar{G}$ and G'. We compute $cost(G, G')$ as

$$cost(G, G') = \sum_{i=1}^{m} \sum_{j=1}^{m} |W'_{ij} - \bar{W}_{ij}|. \tag{9}$$

The dissimilarity between G and G' is

$$d(G, G') = \frac{cost(G, G')}{r}, \tag{10}$$

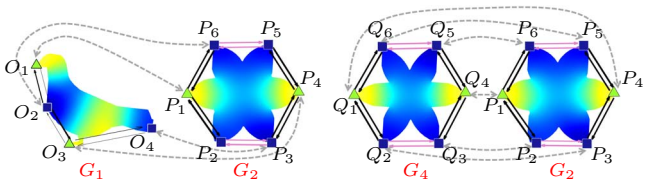
where r is the area ratio of the geometrical region defined by $\bar{G}$. We compute r as

$$r = \frac{\sum_{i=1}^{m} |\bar{C}_i|}{|G|}, \tag{11}$$

where $|\bar{C}_i|$ is the area of the cell $\bar{C}_i$ associated with the node $\bar{P}_i$ in $\bar{G}$, and $|G|$ is the area of the geometrical region defined by G. Since $m \leq n$, it holds that $r \leq 1$. For $m < n$, we obtain $r < 1$, and we use the dissimilarity between the subgraph $\bar{G}$ of G and G', $cost(G, G')$, to approximate the dissimilarity between G and G' based on r. For $m = n$, we obtain $r = 1$ and $d(G, G') = cost(G, G')$.

When deleting a node from an SNNG, it is possible that the feature vectors of the neighbors of this node are affected. When comparing the SNNGs of two similar shapes' Laplacian eigenfunctions, the deletion of the nodes only affects the feature vectors of a small number of other nodes. This is explained by the fact that we always delete the nodes in the region of a shape that corresponds to the missing part of the other shape. In addition, we compute the mapping between two SNNGs' nodes by first considering the most similar pair of nodes. Therefore, the deletion of nodes generally does not affect the mapping between the SNNGs' nodes of two similar shapes' Laplacian eigenfunctions.

Let G_1, G_2, G_3 and G_4 be the SNNGs defined by the fourth Laplacian eigenfunctions of the four shapes shown in Fig. 3. Fig. 6 shows the mappings between the nodes of some of these SNNGs. Fig. 6 (a) illustrates the mapping between the nodes of G_1 and G_2. The nodes O_1, O_2, O_3 and O_4 in G_1 are mapped to the nodes P_1, P_6, P_4 and P_3 in G_2, respectively. Let C_i^2 be the cell of the node P_i in G_2. The dissimilarity between G_1 and G_2 is $d(G_1, G_2) = cost(G_1, G_2)/r_{1,2}$, where $cost(G_1, G_2)$ is the dissimilarity between G_1 and the subgraph induced on G_2 by the mapped nodes (P_1, P_6, P_4 and P_3), and $r_{1,2} = (|C_1^2| + |C_3^2| + |C_4^2| + |C_6^2|)/|G_2|$. Since P_2 and P_5 in G_2 do not have corresponding nodes in G_1, the value of $r_{1,2}$ is smaller than one. Fig. 6 (b) shows the mapping between the nodes of G_4 and G_2. The dissimilarity between these two SNNGs is $d(G_2, G_4) = cost(G_2, G_4)/r_{2,4}$. The value of $r_{2,4}$ is one because these two SNNGs have the same number of nodes. We compute $cost(G_1, G_2)$ and $cost(G_2, G_4)$ using Equation (9), and we find that $cost(G_1, G_2) > cost(G_2, G_4)$. Therefore, $d(G_1, G_2) > d(G_2, G_4)$, which means G_4 is more similar to G_2.



(a) Mapping between G_1 and G_2 (b) Mapping between G_4 and G_2

Fig. 6 The mapping between the nodes of the SNNGs of the fourth Laplacian eigenfunctions of (a) the pigeon shape (shown in Fig. 3(a)) and the starfish shape (shown in Fig. 3(b)), (b) two near-isometric starfish shapes (shown in Fig. 3(d) and 3(b)).

Some Laplacian eigenfunctions of a symmetric shape can have symmetric structures, i.e., the distributions of these eigenfunctions' extrema are symmetric in the functions' domains, which results in symmetric SNNGs. In Fig. 6(b) both starfish shapes are symmetric and both G_2 and G_4 are symmetric SNNGs. The absolute values of the weights of the edges in each SNNG are similar, which means that the node Q_1 in G_4 can be mapped to the node P_1 or the node P_4 in G_2. When mapping Q_1 to P_1 and P_4, respectively, the corresponding dissimilarities between G_4 and G_2 are similar. In other words, the symmetry of the SNNG does not affect the quality of the mapping between the SNNGs' nodes.

7 Results, comparison and discussion

Experimental results are described and discussed in this section. For each shape in a shape database, our method orders the other shapes in ascending order of dissimilarity. We have tested our method for several standard shape databases commonly used to evaluate 2D shape retrieval methods: the Kimia-25 [32], Kimia-99 [31] and Kimia-216 [31] databases. By testing using different k low-frequency non-trivial Laplacian eigenfunctions to

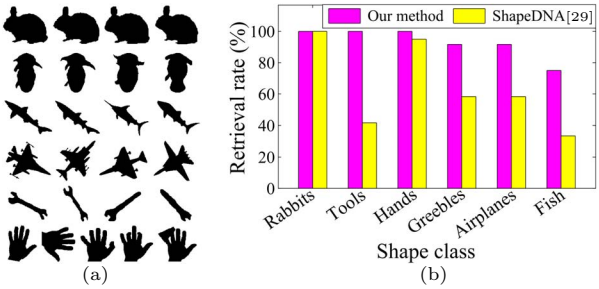


Fig. 7 (a) The Kimia-25 database. (b) The retrieval rates per class obtained by our method and ShapeDNA [29] for the Kimia-25 database. For all the classes, our method performs better than ShapeDNA.

compare the shapes, we found that our method could get good retrieval results when k was about 10. For the experimental results shown in this section, the value of k is 10. We implemented the ShapeDNA method proposed in the literature [29] [2], and compared our method with ShapeDNA and some recent methods for 2D shape retrieval.

7.1 Kimia-25 database

The Kimia-25 database [32] consists of 25 shapes grouped in six classes: five classes (rabbits, greebles, fish, airplanes and tools) with four shapes each, and one class (hands) with five shapes, as shown in Fig. 7(a). For a query, we call a retrieved shape a correct match when this retrieved shape and the query belong to the same class of shapes. Suppose that this class has x shapes, then there are $x-1$ correct matches (excluding the query itself). We compute the retrieval rate by counting the number of correct matches within the first $x-1$ retrieved shapes. The retrieval rate for a class of shapes is the average of the retrieval rates for the shapes within this class. Fig. 7(b) shows the retrieval rates for each class of shapes generated by our method and ShapeDNA [29]. The retrieval rate for each class obtained by our method is higher than that generated by ShapeDNA, and our method retrieves correct matches of 100% for the rabbits, tools and hands. Using every shape in this database as a query shape, Fig. 8(a) and 8(b) show the retrieval results obtained by our method and ShapeDNA, respectively. The first element of each row shows a query shape, and the remaining elements of that row show the first three or four retrieved shapes. The retrieved shapes shown with light-green background represent bad matches. Comparing Fig. 8(a) and 8(b), our method performs better than ShapeDNA for this database.

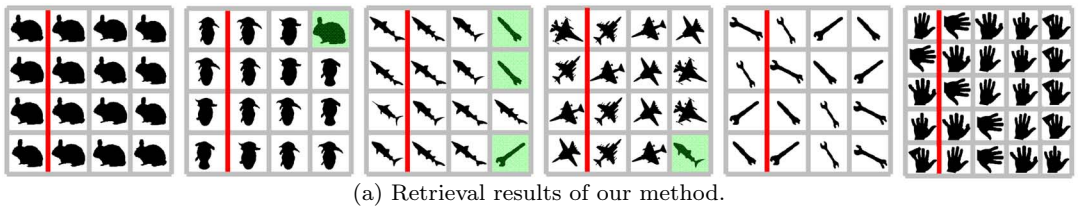
We consider each shape from the Kimia-25 database as a query, and rank the retrieved shapes in ascending order of dissimilarity. The retrieved shape at the r-th ranking position is the r-th closest shape retrieved. The comparison between the retrieval results on the Kimia-25 database generated by different algorithms is obtained by counting the number of the correct matches (excluding the query itself) at each ranking position r ($r = 1, 2, 3$). The highest possible score for r is 25, i.e., there are at most 25 correct matches at each ranking position r. Table 1 lists the comparison of performance between different algorithms. The results show that our method performs better than the first four methods, ShapeDNA [29], ECCobj2D [11], Sharvit et al. [32] and Gdalyahu et al. [7]. Though the total score of our method (score = 70) is lower than those of SC (score = 71) [2] and IDSC+DP (score = 74) [19], our method is slightly better than these two methods at the second ranking position.

The precision-recall curve is a common measure used to evaluate the performance of a shape retrieval method. Precision for a query is defined as the fraction of retrieved shapes that match the query, and recall is defined as the fraction of matching shapes that are retrieved. A precision-recall curve is defined by considering a number of points. The i-th point corresponds to the average recall and precision of all the shapes obtained by considering the first i retrieved shapes. Most shapes in the Kimia-25 database have three matching shapes. We plot the precision-recall curve based on the average recall and precision corresponding to the first one, two and three retrieved shapes, see Fig. 9. Fig 9 indicates that our method performs better than the first four methods and better than the last two methods when considering the first two retrieved shapes.

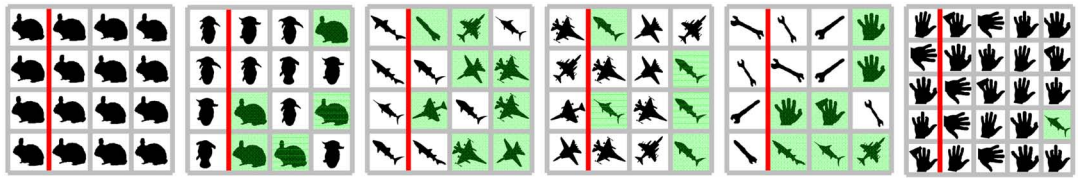
Our method uses the first k non-trivial Laplacian eigenfunctions to compare the shapes. To determine the value of k, we tested using different k to retrieve similar shapes for every shape in the Kimia-25 database. Fig. 10 shows the retrieval rates corresponding to different k. We compute the retrieval rate by recording the correct matches among the first four retrieved shapes for all the 25 shapes.[3] The highest retrieval rate (96.25%) is obtained when $k = 9, 10$, or 11, and the retrieval rates corresponding to k with values

[2] For the results generated by ShapeDNA that are shown in this paper, we tested using different numbers of Laplacian eigenvalues to do shape retrieval and chose the best results.

[3] The six shape classes from the Kimia-25 database consist of different numbers of shapes, and the largest number is five. For convenience, we compute the retrieval rate for the whole database by counting the correct matches among the first four retrieved shapes for all the 25 shapes.



(a) Retrieval results of our method.



(b) Retrieval results of ShapeDNA [29].

Fig. 8 The retrieval results for the Kimia-25 database generated by (a) our method and (b) ShapeDNA [29]. The first element of each row shows a query, and the remaining elements of that row show the retrieved top three or four closest matches (excluding the query itself). The retrieved shapes shown with light-green background represent bad matches.

Table 1 Comparison of retrieval results for the Kimia-25 database obtained by different algorithms for 2D shape retrieval (our method being highlighted in bold). The number of correct matches at each ranking position r ($r = 1, 2, 3$) is counted, and the highest possible score for r is 25. Our method performs better than the first four methods, and is slightly better than SC [2] and IDSC+DP [19] at the second ranking position.

Approach	r=1	r=2	r=3	Total
ShapeDNA [29]	17	19	14	50
ECCobj2D[11]	22	20	17	59
Sharvit et al. [32]	23	21	20	64
Gdalyahu et al. [7]	25	21	19	65
Our method	**25**	**25**	**20**	**70**
SC [2]	25	24	22	71
IDSC+DP [19]	25	24	25	74

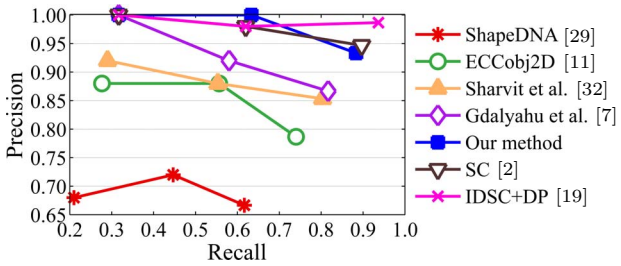


Fig. 9 Precision-recall curves for the Kimia-25 database. Each precision-recall curve is based on the average recall and precision of all 25 shapes that are obtained by considering the first one, two and three retrieved shapes.

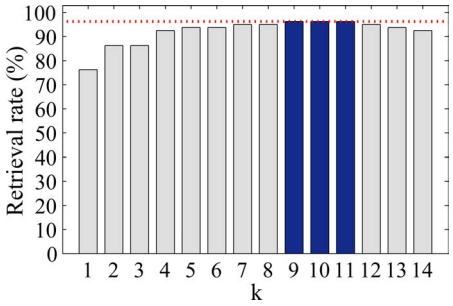


Fig. 10 Retrieval rates obtained by our method using first k non-trivial Laplacian eigenfunctions for the Kimia-25 database, $k = 1, 2, \ldots, 14$. The retrieval rate is computed by considering the correct matches among the first four retrieved shapes for all the 25 shapes. The highest retrieval rate (96.25%) is obtained when the value of k is 9, 10 or 11.

around 10 have minor differences. Besides that, the experimental results show that using more Laplacian eigenfunctions does not improve the retrieval rate but may result in a decrease in the retrieval rate. A reason is that similar shapes may have similar global features but different boundary details, and high-frequency Laplacian eigenfunctions are related to fine details of the shapes.

7.2 Kimia-99 database

The Kimia-99 database [31] consists of nine classes of shapes with 11 shapes per class, as shown in Fig. 11. From top to bottom, the shapes shown in rows are the images of the airplanes, quadrupeds, rabbits, men, greebles, hands, fish, rays and tools. Fig. 12 shows the retrieval rates for each class generated by our method and ShapeDNA [29]. The retrieval rate for each class obtained by our method is higher than that generated by ShapeDNA, and our method can retrieve correct matches of 100% for two classes, the rabbits and rays.

Table 2 lists the number of the correct matches (excluding the query itself) at each ranking position r ($r = 1, 2, \ldots, 10$) for all 99 shapes obtained by several different algorithms for 2D shape retrieval. The highest possible score for r is 99. Our method performs better than ShapeDNA and some recent methods (e.g.,

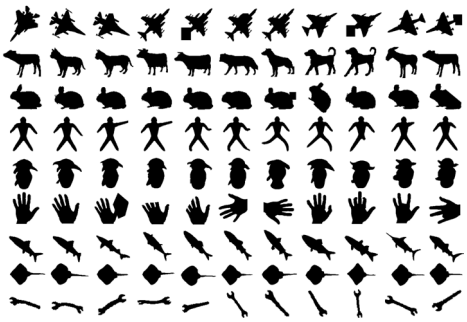


Fig. 11 The Kimia-99 database. Shapes shown in a row belong to a class.

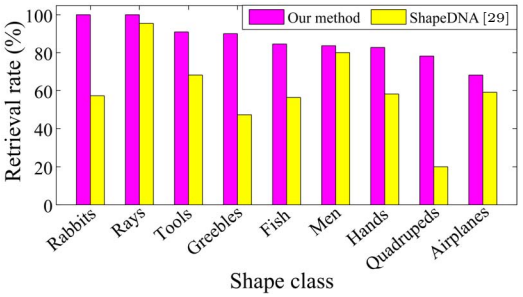


Fig. 12 The retrieval rates per class obtained by our method and ShapeDNA [29] for the Kimia-99 database. For all the classes, our method performs better than ShapeDNA.

Table 2 Comparison of retrieval results for the Kimia-99 database obtained by different algorithms for 2D shape retrieval (our method being highlighted in bold). The number of correct matches at each ranking position r ($r = 1, 2, \ldots, 10$) is counted. The highest possible score for r is 99.

Approach	r=1	r=2	r=3	r=4	r=5	r=6	r=7	r=8	r=9	r=10	Total
ShapeDNA [29]	82	80	76	66	64	56	54	38	37	43	596
CN [15]	97	86	87	75	76	70	55	59	46	44	695
ECCobj2D [11]	94	85	81	73	81	73	64	59	56	35	701
SC [2]	97	91	88	85	84	77	75	66	56	37	756
CPDH+EMD (Eucl) [35]	96	94	94	87	88	82	80	70	62	55	808
CPDH+EMD (shift) [35]	98	94	95	92	90	88	85	84	71	52	849
Our method	**98**	**97**	**96**	**94**	**95**	**87**	**82**	**84**	**72**	**54**	**859**
PS+LBP[33]	99	97	97	88	88	86	86	90	80	77	888
IDSC+DP[19]	99	99	99	98	98	97	97	98	94	79	958
Height function [39]	99	99	99	99	98	99	99	96	95	88	971

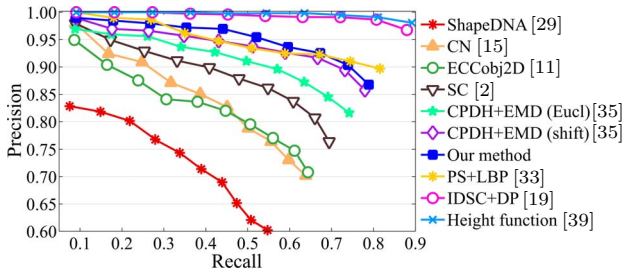
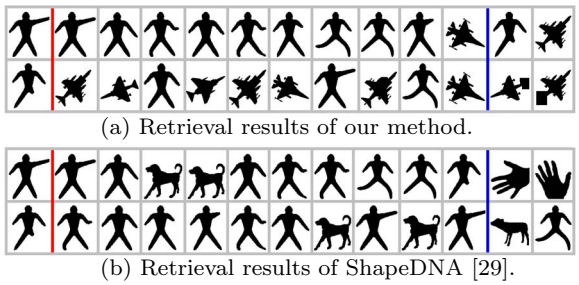


Fig. 13 Precision-recall curves for the Kimia-99 database. Each precision-recall curve is based on the average recall and precision of all 99 shapes that are obtained by considering the first one to 10 retrieved shapes.



(a) Retrieval results of our method.

(b) Retrieval results of ShapeDNA [29].

Fig. 14 The retrieval results for two man shapes from the Kimia-99 database generated by (a) our method and (b) ShapeDNA [29], respectively. Each row shows a query shape, followed by the top 12 matches. For the man shape with two legs, our method performs better than ShapeDNA, while for the shape with a missing leg, ShapeDNA performs better.

ECCobj2D [11] and CN [15]), but performs worse than the last three methods. Each shape in the Kimia-99 database has 10 matching shapes. Fig. 13 shows the precision-recall curves when considering the first one to 10 retrieved shapes.

A reason that our method cannot produce the highest retrieval score for the Kimia-99 database is that our method cannot handle a shape with large missing parts well. Considering the example shown in Fig. 14, the two man shapes shown in the first column of Fig. 14(a) are from the same class, the men, of the Kimia-99 database, and they both have 10 similar shapes in the database. Since we compare two shapes based on comparing the SNNGs defined by the Laplacian eigenfunctions with the nodes of the SNNGs being the eigenfunctions' extrema, the two man shapes has a large dissimilarity because the shape with a missing leg has a smaller number of extrema. Fig. 14 shows the first 12 matches retrieved by (a) our method and (b) ShapeDNA for these two shapes, respectively. Since the men class has 10 shapes with two legs and a shape with one leg, our method can retrieve the correct matches for the shape with two legs well, but performs worse than ShapeDNA for the bottom shape with a missing leg.

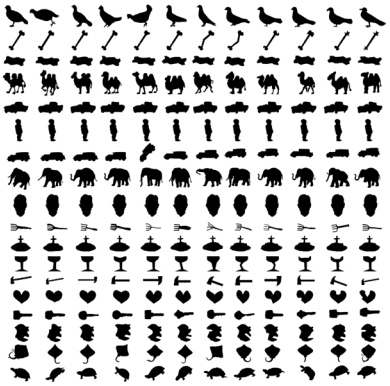


Fig. 15 The Kimia-216 database. Each row shows the shapes within a class.

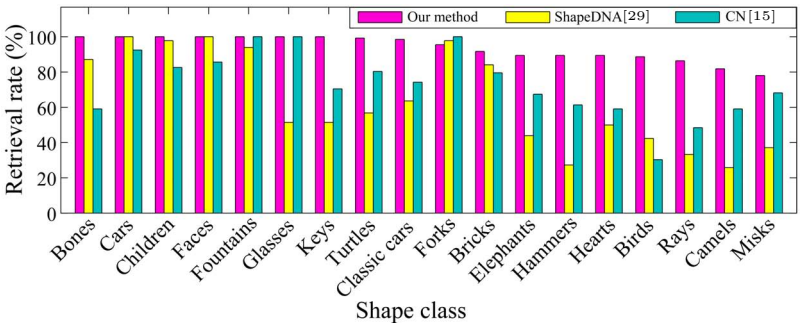


Fig. 16 The retrieval rates per class obtained by our method, ShapeDNA [29] and curve normalization (CN) [15] for the Kimia-216 database. For all classes except for the forks, our method performs as well as or better than the other two methods.

Table 3 Comparison of retrieval results for the Kimia-216 database obtained by different algorithms for 2D shape retrieval (our method being highlighted in bold). The number of correct matches at each ranking position r ($r = 1, 2, \ldots, 11$) is counted, and the highest possible score for r is 216.

Approach	r=1	r=2	r=3	r=4	r=5	r=6	r=7	r=8	r=9	r=10	r=11	Total
ShapeDNA [29]	181	174	163	148	143	139	120	120	116	109	97	1510
SC [2]	214	209	205	197	191	178	161	144	131	101	78	1809
CPDH+EMD (Eucl) [35]	214	215	209	204	200	193	187	180	168	146	114	2030
CPDH+EMD (shift) [35]	215	215	213	205	203	204	190	180	168	154	123	2070
PS+LBP[33]	216	209	205	195	195	197	188	180	179	163	152	2079
Our method	**216**	**215**	**214**	**211**	**212**	**213**	**209**	**197**	**191**	**181**	**169**	**2228**
Height function [39]	216	216	216	215	215	212	211	204	200	194	179	2278

7.3 Kimia-216 database

The Kimia-216 database [31] consists of 18 classes of shapes with 12 shapes per class. Fig. 15 shows all the 216 shapes in this database, and shapes shown in a row belong to a class. From top to bottom, the shapes shown in rows are the images of the birds, bones, bricks, camels, cars, children, classic cars, elephants, faces, forks, fountains, glasses, hammers, hearts, keys, misks, rays and turtles.

Fig. 16 shows the retrieval rates for each class obtained by our method, ShapeDNA [29] and CN [15] for the Kimia-216 database. Our method retrieves all the correct matches for seven classes of shapes, which are bones, cars, children, faces, fountains, glasses and keys; ShapeDNA finds all similar shapes for cars and faces; and CN retrieves correct matches of 100% for three classes of shapes, forks, fountains and glasses. Besides that, except for the forks, our method performs as well as or better than the other two methods.

Table 3 shows the number of the correct matches at each ranking position r ($r = 1, 2, \ldots, 11$), excluding the query itself, for all 216 shapes obtained by several different algorithms for 2D shape retrieval. The highest possible score for r is 216. Our method performs better than the first five algorithms considered here and slightly worse than the last algorithm. Each shape in this database has 11 matching shapes. The precision-recall curves are shown in Fig. 17, considering the first one to 11 retrieved shapes. Though the performance of our method is worse than that of the height function method [39], our method can be generalized more easily to handle 3D shapes, see section 7.5.

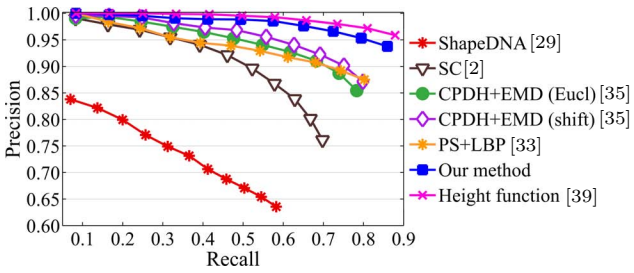


Fig. 17 Precision-recall curves for the Kimia-216 database. Each precision-recall curve is based on the average recall and precision of all 216 shapes that are obtained by considering the first one to 11 retrieved shapes.

7.4 Time complexity

The computationally most expensive components of our method are shape description and shape matching.

A core operation for extracting the SNNG of a Laplacian eigenfunction is computing the geodesic inner distance between two mesh vertices. We use the algo-

Table 4 Computation times (min) of our method for the Kimia-25, Kimia-99 and Kimia-216 databases. The second and third columns show the computation times for the extraction of the SNNGs and SNNG matching.

Database	SNNG extraction	SNNG matching	Total
Kimia-25	0.76	0.74	1.50
Kimia-99	3.00	7.73	10.73
Kimia-216	6.63	28.73	35.36

rithm described in [25]. This algorithm has $\mathcal{O}(n^2 \log(n))$ worst-case time complexity, with n being the number of vertices in the triangulation [23]. We extract the SNNG of a Laplacian eigenfunction in two steps. First, we use the algorithm described in [26] to compute the Voronoi segmentation (i.e., the cells associated with the extrema) of the triangulation with the extrema of the eigenfunction used as seeds. This algorithm computes the Voronoi segmentation based on geodesic inner distances between mesh vertices and their closest extrema. This results in a time complexity of $\mathcal{O}(n^3 \log(n))$ in the worst case. Second, we compute the natural neighbor coordinates of each extremum with respect to its natural neighbors based on the geodesic inner distances from the mesh vertices of its cell to its natural neighbors. The parameter m represents the eigenfunction's number of extrema. In the worst case, each extremum has $m-1$ natural neighbors, which rarely happens when $m > 3$. The time complexity of this worst case is $\mathcal{O}((m-1) \times n^3 \log(n))$. Therefore, the total time complexity of extracting the SNNG of a Laplacian eigenfunction is $\mathcal{O}(n^3 \log(n)) + \mathcal{O}((m-1) \times n^3 \log(n))$ in the worse case, i.e., $\mathcal{O}(m \times n^3 \log(n))$.

The number of nodal domains of the i-th Laplacian eigenfunction is at most i [4]. We use the first k non-trivial Laplacian eigenfunctions to represent a shape. Since we compare the i-th non-trivial eigenfunction of a shape with the $(i-1)$-th, i-th and $(i+1)$-th non-trivial eigenfunctions of another shape to handle the ordering problem of eigenfunctions, we extract the SNNGs of the first $k+1$ non-trivial eigenfunctions. These are the eigenfunctions with indices ranging from two to $k+2$. In our implementation, the value of k is 10, hence the value of m is at most 12. We implemented our method in Matlab and performed all experiments with a 3.2 GHz standard PC with 8 GB of memory. The second column of Table 4 shows the computation times required for extracting the Laplacian eigenfunctions' SNNGs for the shapes in the Kimia-25, Kimia-99 and Kimia-216 databases.

Our shape matching method discussed in section 6.2 has worst time complexity of $\mathcal{O}(m^2 \log(m))$ for comparing two SNNGs. We compute the dissimilarity between two shapes by comparing the SNNGs of their first k non-trivial Laplacian eigenfunctions. To handle the ordering problem of eigenfunctions, we consider the i-th eigenfunction of a shape can only be mapped to the $(i-1)$-th, i-th or $(i+1)$-th eigenfunction of another shape. For $i = 2$, the i-th eigenfunction of a shape can only be mapped to the i-th or $(i+1)$-th eigenfunctions of another shape since the $(i-1)$-th eigenfunction is constant. Therefore, to compute the dissimilarity between two shapes, we need to compare $3k-1$ pairs of eigenfunctions. The time complexity is $\mathcal{O}((3k-1) \times m^2 \log(m))$. We use 10 as the value of k. The resulting computation times required for comparing the shapes in the three databases are shown in the third column of Table 4. One could improve the computation times by implementing our method in more efficient programming languages (such as C and C++) or implementing it using a GPU architecture.

7.5 Discussion

A limitation of our method is the fact that it cannot handle the case of shapes with topology changes. For example, let us consider bending a rod shape until its two ends nearly meet and the shape almost forms a closed "O." We use the notation A to represent this shape. The inner distance between the two ends of shape A is the length of the rod. Now we keep bending the rod until its two ends touch. We call this shape B. The inner distance between the two ends of shape B is zero. The proposed method considers A and B as two different shapes, which means the proposed method is sensitive to a change like this. The MPEG-7 database [16] is a commonly used database for evaluating 2D shape retrieval methods. This database includes shapes that can have extremely dissimilarly shaped contours yet at the same time are considered to be similar, see Fig. 18. Our method considers these five shapes to be quite dissimilar.

Based on our experiments, our method performs better than ShapeDNA [29]. ShapeDNA describes a 2D shape using similar algorithmic foundations that we used. Our method also performs better than some recent methods, e.g., ECCobj2D [11] and CN [15]. Our method does not perform as well as some methods like the height function method [39]. However, compared with the height function method, our method can be generalized to handle 3D shapes.

The height function method [39] is a contour-based method. It samples the outer contour (i.e., the boundary curve) of a 2D shape with equidistantly spaced points. For each sample point, the height function is the ordered sequence of the distances of the other sample points to its tangent. For shape matching, this

Fig. 18 Five shapes stored in the MPEG-7 database. The contours of these shapes are extremely dissimilar, but in some cases it is desirable to consider these shapes as similar. Our method considers these shapes as dissimilar because their contours are extremely dissimilar.

method uses dynamic programming to find an optimal correspondence between two sampled contours. The dissimilarity between two sample points is computed as a weighted L^1 norm of the difference of the points' height functions. This method cannot be generalized easily to handle contours of shapes in 3D space because an order definition of the sample points on the contour of a 3D shape is required.

Our method can be generalized to 3D shapes, where one would describe a shape based on low-frequency Laplacian eigenfunctions of a tetrahedral mesh used to represent a shape. One would construct the SNNG for each eigenfunction to capture the distribution of extrema. Further, one can use more efficient methods that work only with a surface mesh via Laplace-Beltrami eigenfunctions.

In addition to the sign and ordering problems for eigenfunctions, a third issue can arise: Some eigenvalues can have multiplicities larger than one, and the corresponding eigenfunctions define a multi-dimensional eigenspace. The eigenfunctions of a multi-dimensional eigenspace form an orthogonal basis. A problem that arises as a consequence is that the eigenfunctions of such an eigenspace are not uniquely defined, i.e., any orthogonal basis of the eigenspace might be considered. Such multi-dimensional eigenspaces are extremely rare for asymmetric shapes [10, 27]. Our method currently does not handle this problem.

When matching the eigenfunctions of a multi-dimensional eigenspace of a shape A to the eigenfunctions of another shape B, one could handle the ambiguity of the eigenfunctions of A's eigenspace in the following way: For A's eigenspace, one could choose the orthogonal basis, and this basis would be chosen to minimize the total dissimilarity of its eigenfunctions and B's corresponding eigenfunctions.

8 Conclusions and possibilities for future research

We have presented a region-based method for 2D shape description and matching using 2D shape retrieval as driving application for performance evaluation. The major two components of our approach are the use of (i) a small number of a shape's associated low-frequency Laplacian eigenfunctions and (ii) a weighted directed graph to capture topological behavior of these eigenfunctions. The combination and effective integration of these two algorithmic components reflect the major innovation described in this paper. Our method is invariant under rigid transformations and uniform scaling, and is also insensitive to small boundary deformations. While there exist shape retrieval methods that produce equally good or sometimes even better retrieval results, the results obtained by our approach and prototype compete favorably relative to other published methods that are based on similar algorithmic foundations.

Concerning potential future research, one can consider several possibilities: (i) using other distance metrics between weighted directed graphs to improve our method's ability to handle shapes with large missing parts; (ii) using other structures to represent the Laplacian eigenfunctions and comparing retrieval results obtained when using different structures; (iii) generalizing the method to handle the multi-dimensional eigenspace situation; (iv) generalizing the proposed method to non-planar surface description in 3D space, to match and retrieve curved surfaces.

Acknowledgements We thank Martin Reuter for making available his "ShapeDNA-tria" software, Jonathan Shewchuk for his "Triangle" program and Gabriel Peyré for his toolboxes - "Toolbox Fast Marching" and "Toolbox Graph."

References

1. Adamek, T., O'Connor, N.E.: A multiscale representation method for nonrigid shapes with a single closed contour. IEEE Trans. Circuits Syst. Video Techn. **14**(5), 742–753 (2004)
2. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Trans. Pattern Anal. Mach. Intell. **24**(4), 509–522 (2002)
3. Chavel, I.: Eigenvalues in Riemannian Geometry, *Pure and applied mathematics*, vol. 115. Academic Press, Orlando (1984)
4. Courant, R., Hilbert, D.: Methods of Mathematical Physics, vol. 1. Interscience Publishers, New York (1953)
5. Cui, M., Wonka, P., Razdan, A., Hu, J.: A new image registration scheme based on curvature scale space curve matching. The Vis. Computer **23**(8), 607–618 (2007)

6. Gao, X., Xiao, B., Tao, D., Li, X.: A survey of graph edit distance. Pattern Anal. Appl. **13**(1), 113–129 (2010)
7. Gdalyahu, Y., Weinshall, D.: Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. IEEE Trans. Pattern Anal. Mach. Intell. **21**(12), 1312–1328 (1999)
8. Gordon, C., Webb, D., Wolpert, S.: Isospectral plane domains and surfaces via Riemannian orbifolds. Invent Math **110**(1), 1–22 (1992)
9. Grauman, K., Darrell, T.: Fast contour matching using approximate earth mover's distance. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 220–227 (2004)
10. Hu, J., Hua, J.: Pose analysis using spectral geometry. The Vis. Computer **29**(9), 949–958 (2013)
11. Ion, A., Artner, N.M., Peyré, G., Kropatsch, W.G., Cohen, L.D.: Matching 2D and 3D articulated shapes using the eccentricity transform. Comput. Vis. and Image Underst. **115**(6), 817–834 (2011)
12. Isaacs, J.C., Roberts, R.G.: Metrics of the Laplace-Beltrami eigenfunctions for 2D shape matching. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC), pp. 3347–3352 (2011)
13. Kim, W.Y., Kim, Y.S.: A region-based shape descriptor using Zernike moments. Signal Process.: Image Commun. **16**(1-2), 95–102 (2000)
14. Kuhn, H.W.: The Hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**(1–2), 83–97 (1955)
15. Laiche, N., Larabi, S., Ladraa, F., Khadraoui, A.: Curve normalization for shape retrieval. Signal Process.: Image Commun. **29**(4), 556–571 (2014)
16. Latecki, L.J., Lakämper, R., Eckhardt, U.: Shape descriptors for non-rigid shapes with a single closed contour. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 424–429 (2000)
17. Lévy, B.: Laplace-Beltrami eigenfunctions towards an algorithm that "understands" geometry. In: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006, invited talk, SMI '06, p. 13 (2006)
18. Li, S., Lee, M.C., Pun, C.M.: Complex Zernike moments features for shape-based image retrieval. IEEE Trans. on Syst., Man, and Cybern., Part A: Syst. and Humans **39**(1), 227–237 (2009)
19. Ling, H., Jacobs, D.W.: Shape classification using the inner-distance. IEEE Trans. Pattern Anal. Mach. Intell **29**(2), 286–299 (2007)
20. Mokhatarian, F., Abbasi, S., Kittler, J.: Efficient and robust retrieval by shape content through curvature scale space. In: A.W.M. Smeulders, R. Jain (eds.) Images databases and multimedia search, *Software engineering and knowledge engineering*, vol. 8, pp. 51–58. World Scientific (1997)
21. Munkres, J.: Algorithms for the assignment and transportation problems. J. of the Soc. of Ind. and Appl. Math. **5**(1), 32–38 (1957)
22. Okabe, A., Boots, B., Sugihara, K.: Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. John Wiley & Sons, Inc., New York (1992)
23. O'Rourke, J.: Computational geometry column 35. SIGACT News **30**(2), 31–32 (1999)
24. Peinecke, N., Wolter, F.E., Reuter, M.: Laplace spectra as fingerprints for image recognition. Comput.-Aided Des. **39**(6), 460–476 (2007)
25. Peyré, G.: Toolbox fast marching. MATLAB Central File Exchange Select (2009)
26. Peyré, G.: Toolbox graph. MATLAB Central File Exchange Select (2009)
27. Reuter, M.: Hierarchical shape segmentation and registration via topological features of Laplace-Beltrami eigenfunctions. Int. J. of Comput. Vis. **89**(2), 287–308 (2009)
28. Reuter, M., Biasotti, S., Giorgi, D., Patanè, G., Spagnuolo, M.: Discrete Laplace-Beltrami operators for shape analysis and segmentation. Comput. & Graph. **33**(3), 381–390 (2009)
29. Reuter, M., Wolter, F.E., Peinecke, N.: Laplace-Beltrami spectra as "Shape-DNA" of surfaces and solids. Comput-Aided Des **38**(4), 342–366 (2006)
30. Rustamov, R.M.: Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In: Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP'07, pp. 225–233 (2007)
31. Sebastian, T.B., Klein, P.N., Kimia, B.B.: Recognition of shapes by editing their shock graphs. IEEE Trans. Pattern Anal. Mach. Intell. **26**(5), 550–571 (2004)
32. Sharvit, D., Chan, J., Tek, H., Kimia, B.B.: Symmetry-based indexing of image databases. J. of Vis. Commun. and Image Represent. **9**(4), 366–380 (1998)
33. Shekar, B., Pilar, B.: Shape representation and classification through pattern spectrum and local binary pattern - a decision level fusion approach. In: Proceedings of the Fifth International Conference on Signal and Image Processing (ICSIP), pp. 218–224 (2014)

34. Shewchuk, J.R.: Triangle: engineering a 2D quality mesh generator and Delaunay triangulator. In: M.C. Lin, D. Manocha (eds.) Applied computational geometry: towards geometric engineering, *Lecture notes in computer science*, vol. 1148, pp. 203–222. Springer-Verlag (1996)
35. Shu, X., jun Wu, X.: A novel contour descriptor for 2D shape matching and its application to image retrieval. Image Vis. Comput. **29**(4), 286–294 (2011)
36. Sibson, R.: A brief description of natural neighbour interpolation(chapter 2). In: V. Barnett (ed.) Interpreting Multivariate Data, vol. 21, pp. 21–36. John Wiley & Sons, New York (1981)
37. Taubin, G.: Geometric signal processing on polygonal meshes. In: Eurographics 2000 State of the Art Report (STAR), pp. 81–96 (2000)
38. Taylor, M.E.: Partial Differential Equations I: Basic Theory. Applied functional analysis: applications to mathematical physics. U.S. Government Printing Office (1996)
39. Wang, J., Bai, X., You, X., Liu, W., Latecki, L.J.: Shape matching and classification using height functions. Pattern Recognit. Lett. **33**(2), 134–143 (2012)
40. Xu, C., Liu, J., Tang, X.: 2D shape matching by contour flexibility. IEEE Trans. Pattern Anal. Mach. Intell. **31**(1), 180–186 (2009)
41. Zhang, D., Lu, G.: Shape-based image retrieval using generic Fourier descriptor. Signal Process.: Image Commun. **17**(10), 825–848 (2002)
42. Zhang, D., Lu, G.: Review of shape representation and description techniques. Pattern Recognit. **37**(1), 1–19 (2004)