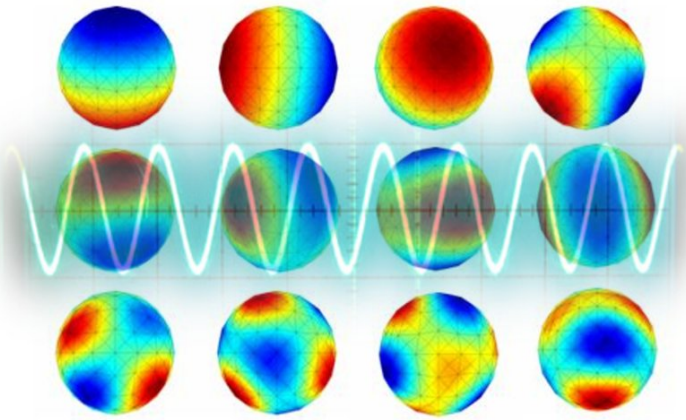
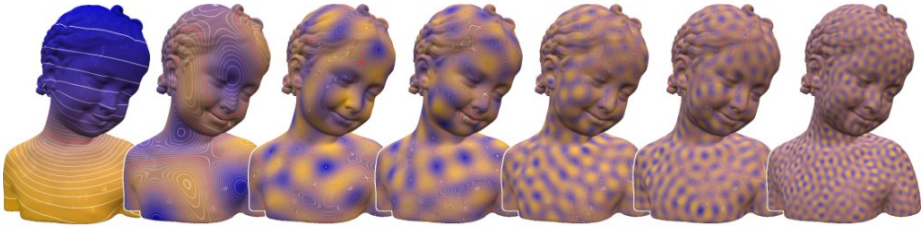
Spectral Mesh Processing

SIGGRAPH 2010 Course

Bruno Lévy (INRIA, France)
Hao (Richard) Zhang (Simon Fraser University, Canada)





About the instructors

Bruno Lévy
INRIA, centre Nancy Grand Est,
rue du Jardin Botanique,
54500 Vandoeuvre, France
Email: bruno.levy@inria.fr
http://alice.loria.fr/index.php/bruno-levy.html

Bruno Lévy is a researcher with INRIA. His main contribution concerns parameterization of triangulated surfaces (LSCM), and is now used by some 3D modeling software (including Maya, Silo, Blender, Gocad and Catia). He obtained his Ph.D in 1999, and was hired by INRIA in 2000. Since 2004, he has been leading Project ALICE - Geometry and Light, that aims at developping mathematical tools for next generation geometry processing. He served on the committee of ACM SPM, IEEE SMI, ACM/EG SGP, IEEE Visualization, Eurographics, PG, ACM Siggraph, and was program co-chair of ACM SPM in 2007 and 2008, and will be program co-chair of SGP 2010. He was recently awarded a "Starting Grant" from the European Research Council (3% acceptance, all disciplines of science).

Hao (Richard) Zhang
School of Computing Science
Simon Fraser University, Burnaby, Canada V5A 1S6
Email: haoz@cs.sfu.ca
http://www.cs.sfu.ca/~haoz

Hao (Richard) Zhang is an Associate Professor in the School of Computing Science at Simon Fraser University, Canada, and he co-directs the Graphics, Usability, and Visualization (GrUVi) Lab. He received his Ph.D. from the University of Toronto in 2003 and M. Math. and B. Math. degrees from the University of Waterloo. His research interests include geometry processing, shape analysis, and computer graphics. He gave a Eurographics State-of-the-art report on spectral methods for mesh processing and analysis in 2007 and wrote the first comprehensive survey on this topic. Recently, he has served on the program committees of Eurographics, ACM/EG SGP, ACM/SIAM GPM, and IEEE SMI. He was a winner of the Best Paper Award from SGP 2008.

Course description

Summary statement: In this course, you will learn the basics of Fourier analysis on meshes, how to implement it, and how to use it for filtering, remeshing, matching, compressing, and segmenting meshes.

Abstract: Spectral mesh processing is an idea that was proposed at the beginning of the 90's, to port the "signal processing toolbox" to the setting of 3D mesh models. Recent advances in both computer horsepower and numerical software make it possible to fully implement this vision. In the more classical context of sound and image processing, Fourier analysis was a corner stone in the development of a wide spectrum of techniques, such as filtering, compression, and recognition. In this course, attendees will learn how to transfer the underlying concepts to the setting of a mesh model, how to implement the "spectral mesh processing" toolbox and use it for real applications, including filtering, shape matching, remeshing, segmentation, and parameterization.

Background: Some elements of this course appeared in the "Geometric Modeling based on Polygon Meshes" course (SIGGRAPH 07, Eurographics 08), the "Mesh Parameterization, Theory and Practice" course (SIGGRAPH 07), and the "Mesh Processing" course (invited course, ECCV 08). Based on some feedback from the attendees, it seems that there is a need for a course focused on the spectral aspects. Whereas these previous courses just mentioned some applications of spectral methods, this course details the notions relevant to spectral mesh processing, from theory to implementation. Spectral methods for mesh processing had been presented as a Eurographics State-of-the-art report in 2007 and a subsequent survey was completed in 2009. Some coverages in the current notes, in particular those on applications, have been selected from the survey. This course, given at ACM SIGGRAPH ASIA 2009 and ACM SIGGRAPH 2010, starts with a more basic setup and include updates from some of the most recent developments in spectral mesh processing.

Intended audience: Researcher in the areas of geometry processing, shape analysis, and computer graphics, as well as practitioners who want to learn more about how to implement these new techniques and what are the applications.

Prerequisites: Knowledge about mesh processing, programming, mesh data structures, basic notions of linear algebra and signal processing.

Course syllabus

- (5 min) Introduction [Lévy]
- (45 min) "What is so spectral?" [Zhang]
  Intuition and theory behind spectral methods
  Different interpretations and motivating applications
- (55 min) Do your own Spectral Mesh processing at home [Lévy]
  DEC Laplacian, numerics of spectral analysis
  Tutorial on implementation with open source software

- (15 min) break

- (50 min) Applications - "What can we do with it ?" 1/2 [Zhang]
  Segmentation, shape retrieval, non-rigid matching, symmetry detection, ...
- (40 min) Applications - "What can we do with it ?" 2/2 [Lévy]
  Quadrangulation, parameterization, ...
- (15 min) Wrapup, conclusion, Q&A [Zhang and Lévy]

Spectral mesh processing involves the use of eigenvalues, eigenvectors, or eigenspace projections derived from appropriately defined mesh operators to carry out desired tasks. Early work in this area can be traced back to the seminal paper by Taubin [Tau95a] in 1995, where spectral analysis of mesh geometry based on a graph Laplacian allows us to understand the low-pass filtering approach to mesh smoothing. Over the past fifteen years, the list of geometry processing applications which utilize the eigenstructures of a variety of mesh operators in different manners have been growing steadily.

Most spectral methods for mesh processing have a basic framework in common. First, a matrix representing a discrete linear operator based on the topological and/or geometric structure of the input mesh is constructed, typically as a discretization of some continuous operator. This matrix can be seen as incorporating pairwise relations between mesh elements, adjacent ones or otherwise. Then an eigendecomposition of the matrix is performed, that is, its eigenvalues and eigenvectors are computed. Resulting structures from the decomposition are employed in a problem-specific manner to obtain a solution.

We will look at the various applications of spectral mesh processing towards the end of the course notes (Section 5), after we provide the intuition, motivation, and some theory behind the use of the spectral approach. The main motivation for developing spectral methods for mesh processing is the pursuit of Fourier analysis in the manifold setting, in particular, for meshes which are the dominant discrete representations of surfaces in the field of computer graphics. There are other desirable characteristics of the spectral approach including effective and information-preserving dimensionality reduction and its ability to reveal global and intrinsic structures in geometric data [ZvKDar]. These should become clear as we discuss the applications.

We shall start the course notes with a very gentle introduction in Section 1. Instead of diving into 3D data immediately, we first look at the more classic case of processing a 2D shape represented by a contour. The motivating application is shape smoothing. We reduce the 2D shape processing problem to the study of 1D functions or signals specifying the shape's contour, naturally exposing the problem in a signal-processing framework. Our choice to start the coverage in the discrete setting is intended to not involve the heavy mathematical formulations at the start so as to better provide an intuition. We present Laplacian smoothing and show how spectral processing based on 1D discrete Laplace operators can perform smoothing as well as compression. The relationship between this type of spectral processing and the classical Fourier transform is revealed and a natural extension to the mesh setting is introduced.

Having provided an intuition, we then instill more rigor into our coverage and more formally present spectral mesh processing as a means to perform Fourier analysis on meshes. In particular, we deepen our examination on the connection between the continuous and the discrete settings, focusing on the Laplace operator and its eigenfunctions. While Section 2 provides some theoretical background in the continuous setting (eigenfunctions of the Laplace-Beltrami operator) and establishes connections with other domains (machine learning and spectral graph theory). Section 3 is concerned with ways to discretize the

Laplace operator. Since spectral mesh processing in practice often necessitates the computation of eigenstructures of large matrices, Section 4 presents ways to make that process efficient. Finally, we discuss various applications of spectral mesh processing in Section 5. For readers who wish to obtain a more thorough coverage on the topic, we refer to the survey of Zhang et al. [ZvKDar].

1 A gentle introduction

Consider the seahorse shape depicted by a closed contour, shown in Figure 1. The contour is represented as a sequence of 2D points (the contour vertices) that are connected by straight line segments (the contour segments), as illustrated by a zoomed-in view. Now suppose that we wish to remove the rough features over the shape of the seahorse and obtain a smoothed version, such as the one shown in the right of Figure 1.

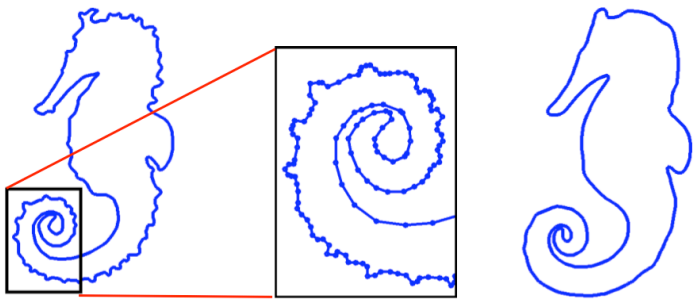


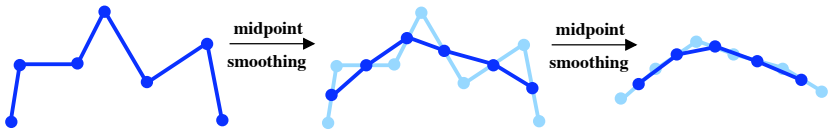
Figure 1: A seahorse with rough features (left) and a smoothed version (right).

1.1 Laplacian smoothing

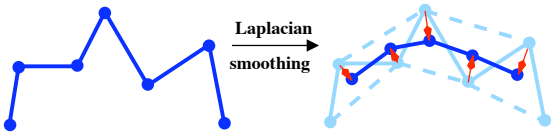
A simple procedure to accomplish this is to repeatedly connect the midpoints of successive contour segments; we refer to this as *midpoint smoothing*. Figures 2(a) illustrates two such steps. As we can see, after two steps of midpoint smoothing, each contour vertex $\mathbf{v}_i$ is moved to the midpoint of the line segment connecting the midpoints of the original contour segments adjacent to $\mathbf{v}_i$. Specifically, let $\mathbf{v}_{i-1} = (x_{i-1}, y_{i-1})$, $\mathbf{v}_i = (x_i, y_i)$, and $\mathbf{v}_{i+1} = (x_{i+1}, y_{i+1})$ be three consecutive contour vertices. Then the new vertex $\hat{\mathbf{v}}_i$ after two steps of midpoint smoothing is given by a *local averaging*,

$$\hat{\mathbf{v}}_i = \frac{1}{2}\left[\frac{1}{2}\left(\mathbf{v}_{i-1} + \mathbf{v}_i\right)\right] + \frac{1}{2}\left[\frac{1}{2}\left(\mathbf{v}_i + \mathbf{v}_{i+1}\right)\right] = \frac{1}{4}\mathbf{v}_{i-1} + \frac{1}{2}\mathbf{v}_i + \frac{1}{4}\mathbf{v}_{i+1}. \quad (1)$$

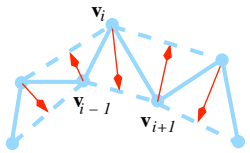
The vector between $\mathbf{v}_i$ and the midpoint of the line segment connecting the two vertices adjacent to $\mathbf{v}_i$, shown as a red arrow in Figure 2(b), is called the



(a) Two steps of midpoint smoothing.



(b) Laplacian smoothing.



(c) 1D Laplacians (red).

Figure 2: One step of Laplacian smoothing (b) is equivalent to two steps of midpoint smoothing (a). The 1D discrete Laplacian vectors (c) are in red.

1D discrete Laplacian at $\mathbf{v}_i$, denoted by $\delta(\mathbf{v}_i)$,

$$\delta(\mathbf{v}_i) = \frac{1}{2}(\mathbf{v}_{i-1} + \mathbf{v}_{i+1}) - \mathbf{v}_i. \tag{2}$$

As we can see, after two steps of midpoint smoothing, each contour vertex is displaced by half of its 1D discrete Laplacian, as shown in Figure 2(c); this is referred to as *Laplacian smoothing*. The smoothed version of the seahorse in Figure 1 was obtained by applying 10 steps of Laplacian smoothing.

1.2 Signal representation and spectral transform

Let us denote the contour vertices by a coordinate vector V, which has n rows and two columns, where n is the number of vertices along the contour and the two columns correspond to the x and y components of the vertex coordinates. Let us denote by x_i (respectively, y_i) the x (respectively, y) coordinate of a vertex $\mathbf{v}_i$, $i = 1, \ldots, n$. For analysis purposes, let us only consider the x component of V, denoted by X; the treatment of the y component is similar.

We treat the vector X as a discrete 1D signal. Since the contour is closed, we can view X as a periodic 1D signal defined over uniformly spaced samples along a circle, as illustrated in Figure 3(a). We sort the contour vertices in counterclockwise order and plot it as a conventional 1D signal, designating an arbitrary element in X to start the indexing. Figure 3(b) shows such a plot for the x-coordinates of the seahorse shape ($n = 401$) from Figure 1. We can now express the discrete 1D Laplacians (2) for all the vertices using an $n \times n$ matrix

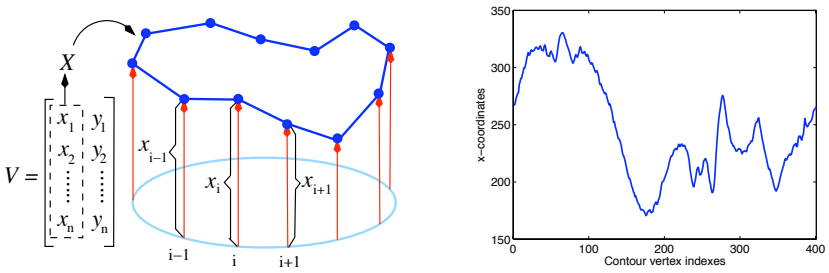


Figure 3: Left: The x-component of the contour coordinate vector V, X, can be viewed as a 1D periodic signal defined over uniform samples along a circle. Right: it is shown by a 1D plot for the seahorse contour from Figure 1.

L, called the *1D discrete Laplace operator*, as follows:

$$\delta(X) = LX = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & \dots & \dots & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \dots & \dots & 0 & -\frac{1}{2} & 1 \end{bmatrix} X. \tag{3}$$

The new contour $\hat{X}$ resulting from Laplacian smoothing (1) is then given by

$$\hat{X} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_{n-1} \\ \hat{x}_n \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & 0 & \dots & \dots & 0 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & 0 & \dots & \dots & 0 & \frac{1}{4} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = SX. \tag{4}$$

The smoothing operator S is related to the Laplace operator L by

$$S = I - \frac{1}{2}L.$$

To analyze the behavior of Laplacian smoothing, in particular what happens in the limit, we rely on the set of basis vectors formed by the eigenvectors of L. This leads to a framework for *spectral analysis* of geometry. From linear algebra, we know that since L is symmetric, it has real eigenvalues and a set of real and orthogonal set of eigenvectors which form a basis. Any vector of size n can be expressed as a linear sum of these basis vectors. We are particularly interested in such an expression for the coordinate vector X. Denote by $\mathbf{e}_1$, $\mathbf{e}_2$,...,$\mathbf{e}_n$ the normalized eigenvectors of L, corresponding to eigenvalues λ_1, λ_2, ..., λ_n, and let E be the matrix whose columns are the eigenvectors. Then we

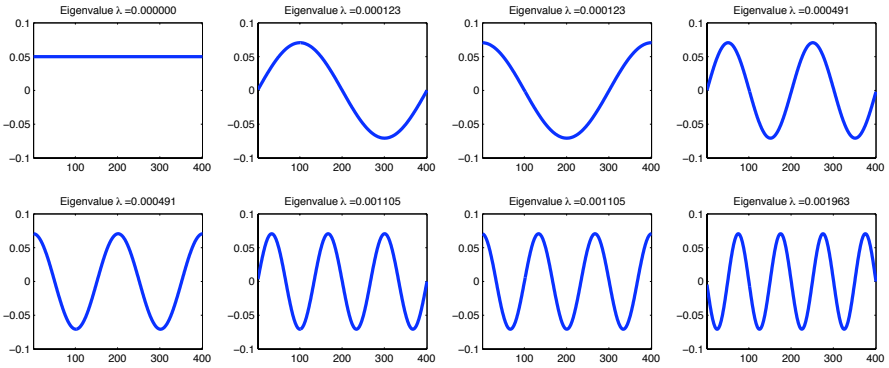


Figure 4: Plots of first 8 eigenvectors of the 1D discrete Laplace operator ($n = 401$) given in equation (3). They are sorted by the eigenvalue λ.

can express X as a linear sum of the eigenvectors,

$$X = \sum_{i=1}^{n} \mathbf{e}_i \tilde{x}_i = \begin{bmatrix} E_{11} \\ E_{21} \\ \vdots \\ E_{n1} \end{bmatrix} \tilde{x}_1 + \ldots + \begin{bmatrix} E_{1n} \\ E_{2n} \\ \vdots \\ E_{nn} \end{bmatrix} \tilde{x}_n = \begin{bmatrix} E_{11} & \ldots & E_{1n} \\ E_{21} & \ldots & E_{2n} \\ \vdots & \vdots & \vdots \\ E_{n1} & \ldots & E_{nn} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{bmatrix} = E\tilde{X}. \tag{5}$$

The above expression represents a transform from the original signal X to a new signal $\tilde{X}$ in terms of a new basis, the basis given by the eigenvectors of L. We call this a *spectral transform*, whose coefficients $\tilde{X}$ can be obtained by

$$\tilde{X} = E^{\mathrm{T}} X, \qquad \text{where } E^{\mathrm{T}} \text{ is the transpose of } E,$$

and for each i, the *spectral transform coefficient*

$$\tilde{x}_i = \mathbf{e}_i^{\mathrm{T}} \cdot X. \tag{6}$$

That is, the spectral coefficient $\tilde{x}_i$ is obtained as a *projection* of the signal X along the direction of the i-th eigenvector $\mathbf{e}_i$. In Figure 4, we plot the first 8 eigenvectors of L, sorted by increasing eigenvalues, where $n = 401$, maching the size of the seahorse shape from Figure 1. The indexing of elements in each eigenvector follows the same contour vertex indexing as X, the coordinate vector; it was plotted in Figure 3(b) for the seahorse. It is worth noting that aside from an agreement on indexing, the Laplace operator L and the eigenbasis vectors do not depend on X, which specifies the geometry of the contour. L, as defined in equation (3), is completely determined by n, the size of the input contour, and a vertex ordering.

As we can see, the eigenvector corresponding to the zero eigenvalue is a constant vector. As eigenvalue increases, the eigenvectors start to oscillate as sinusoidal curves at higher and higher frequencies. Note that the eigenvalues of

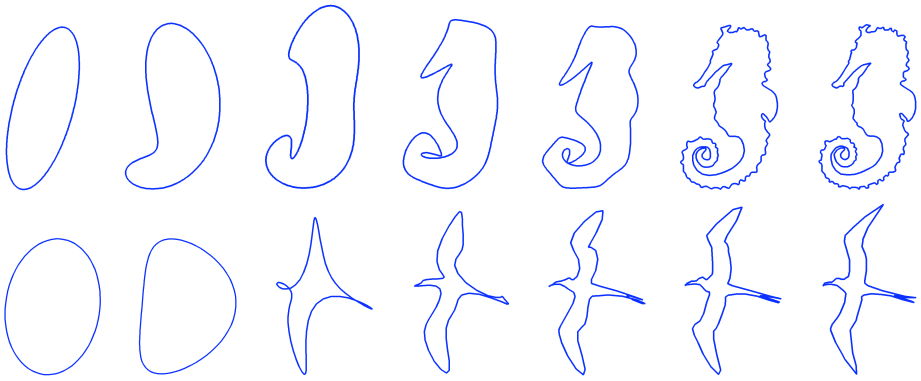


Figure 5: Shape reconstruction via Laplacian-based spectral analysis.

L repeat (multiplicity 2) after the first one, hence the corresponding eigenvectors of these repeated eigenvalues are not unique. One particular choice of the eigenvectors reveals a connection of our spectral analysis to the classical Fourier analysis; this will be discussed in Section 1.5.

1.3 Signal reconstruction and compression

With the spectral transform of a coordinate signal defined as in equation (5), we can now look at compression and filtering of a 2D shape represented by a contour. Analogous to image compression using JPEG, we can obtain compact representations of a contour by retaining the leading (low-frequency) spectral transform coefficients and eliminating the rest. Given a signal X as in equation (5), the signal reconstructed by using the k leading coefficients is

$$X^{(k)} = \sum_{i=1}^{k} \mathbf{e}_i \tilde{x}_i, \qquad k \leq n. \tag{7}$$

This represents a compression of the contour geometry since only k out of n coefficients need to be stored to approximate the original shape. We can quantify the information loss by measuring the L_2 error

$$||X - X^{(k)}|| = || \sum_{i=k+1}^{n} \mathbf{e}_i \tilde{x}_i|| = \sqrt{\sum_{i=k+1}^{n} \tilde{x}_i^2},$$

which is simply the Euclidean distance between X and $X^{(k)}$. The last equality is easy to obtain if we note the orthogonality of the eigenvectors, i.e., $\mathbf{e}_i^{\mathrm{T}} \cdot \mathbf{e}_j = 0$ whenever $i \neq j$. Also, since the $\mathbf{e}_i$'s are normalized, $\mathbf{e}_i^{\mathrm{T}} \cdot \mathbf{e}_i = 1$.

In Figure 5, we show some results of this type of shape reconstruction (7), with varying k for the seahorse and a bird shape. As more and more high-frequency spectral coefficients are removed, i.e., with decreasing k, we obtain

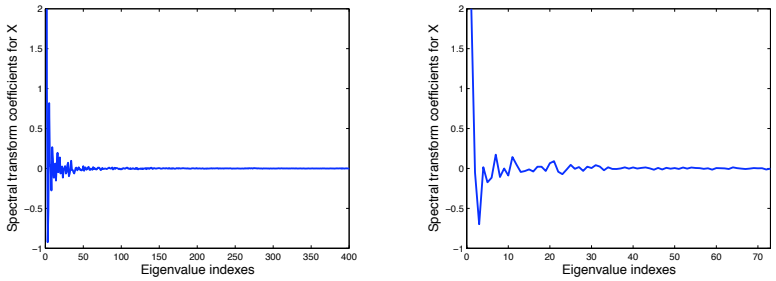


Figure 6: Plot of spectral transform coefficients for the x component of a contour. Left: seahorse. Right: bird. The models are shown in Figure 5.

smoother and smoother reconstructed contours. How effectively a 2D shape can be compressed this way may be visualized by plotting the spectral transform coefficients, the $\tilde{x}_i$'s in (6), as done in Figure 6. In the plot, the horizontal axis represents eigenvalue indexes, $i = 1, \ldots, n$, which roughly correspond to frequencies. One can view the magnitude of the $\tilde{x}_i$'s as the energy of the input signal X at different frequencies.

A signal whose energies are sharply concentrated in the low-frequency end can be effectively compressed at a high compression rate, since as a consequence, the total energy at the high-frequency end, representing the reconstruction error, is very low. Such signals will exhibit fast decay in its spectral coefficients. Both the seahorse and the bird models contain noisy or sharp features so they are not as highly compressible as a shape with smoother boundaries. This can be observed from the plots in Figure 6. Nevertheless, at 2:1 compression ratio, we can obtain a fairly good approximation, as one can see from Figure 5.

1.4 Filtering and Laplacian smoothing

Compression by truncating the vector $\tilde{X}$ of spectral transform coefficients can be seen as a *filtering* process. When a discrete filter function f is applied to $\tilde{X}$, we obtain a new coefficient vector $\tilde{X}'$, where $\tilde{X}'(i) = f(i) \cdot \tilde{X}(i)$, for all i. The filtered signal X' is then reconstructed from $\tilde{X}'$ by $X' = E\tilde{X}'$, where E is the matrix of eigenvectors as defined in equation (5). We next show that Laplacian smoothing is one particular filtering process. Specifically, when we apply the Laplacian smoothing operator S to a coordinate vector m times, the resulting coordinate vector becomes,

$$X^{(m)} = S^m X = (I - \frac{1}{2}L)^m X = \sum_{i=1}^{n} (I - \frac{1}{2}L)^m \mathbf{e}_i \tilde{x}_i = \sum_{i=1}^{n} \mathbf{e}_i (1 - \frac{1}{2}\lambda_i)^m \tilde{x}_i. \quad (8)$$

Equation (8) provides a characterization of Laplacian smoothing in the spectral domain via filtering and the filter function is given by $f(\lambda) = (1 - \frac{1}{2}\lambda)^m$. A few such filters with different m are plotted in the first row of Figure 7. The

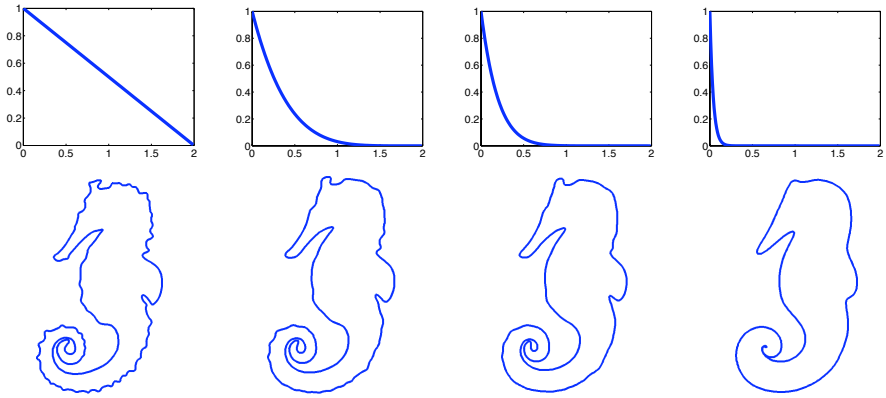


Figure 7: First row: filter plots, $(1 - \frac{1}{2}\lambda)^m$ with $m = 1, 5, 10, 50$. Second row: corresponding results of Laplacian smoothing on the seahorse.

corresponding Laplacian smoothing leads to attenuation of the high-frequency content of the signal and hence achieves denoising or smoothing.

To examine the limit behavior of Laplacian smoothing, let us look at equation (8). Note that it can be shown (via the Gerschgorin's Theorem [TB97]) that the eigenvalues of the Laplace operator are in the interval $[0, 2]$ and the smallest eigenvalue $\lambda_1 = 0$. Since $\lambda \in [0, 2]$, the filter function $f(\lambda) = (1 - \frac{1}{2}\lambda)^m$ is bounded by the unit interval $[0, 1]$ and attains the maximum $f(0) = 1$ at $\lambda = 0$. As $m \to \infty$, all the terms in the right-hand side of equation (8) will vanish except for the first, which is given by $\mathbf{e}_1\tilde{x}_1$. Since $\mathbf{e}_1$, the eigenvector corresponding to the zero eigenvalue is a normalized, constant vector, we have $\mathbf{e}_1 = [\frac{1}{\sqrt{n}} \ \frac{1}{\sqrt{n}} \ \ldots \ \frac{1}{\sqrt{n}}]^{\mathrm{T}}$. Now taking the y-component into consideration, we get the limit point for Laplacian smoothing as $\frac{1}{\sqrt{n}}[\tilde{x}_1 \ \tilde{y}_1]$. Finally, noting that $\tilde{x}_1 = \mathbf{e}_1^{\mathrm{T}} \cdot X$ and $\tilde{y}_1 = \mathbf{e}_1^{\mathrm{T}} \cdot Y$, we conclude that the limit point of Laplacian smoothing is the *centroid* of the set of original contour vertices.

1.5 Spectral analysis vs. discrete Fourier transform

The sinosoidal behavior of the eigenvectors of the 1D discrete Laplace operator (see plots in Figure 4) leads one to believe that there must be a connection between the discrete Fourier transform (DFT) and the spectral transform we have defined so far. We now make that connection explicit. Typically, one introduces the DFT in the context of Fourier series expansion. Given a discrete signal $X = [x_1 \ x_2 \ \ldots \ x_n]^{\mathrm{T}}$, its DFT is given by

$$\tilde{X}(k) = \frac{1}{n}\sum_{j=1}^{n} X(k)e^{-i2\pi(k-1)(j-1)/n}, k = 1, \ldots, n.$$

And the corresponding inverse DFT is given by

$$X(j) = \sum_{k=1}^{n} \tilde{X}(k) e^{i2\pi(j-1)(k-1)/n}, j = 1, \ldots, n,$$

or

$$X = \sum_{k=1}^{n} \tilde{X}(k)\mathbf{g}_k, \text{ where } \mathbf{g}_k(j) = e^{i2\pi(j-1)(k-1)/n}, k = 1, \ldots, n.$$

We see that in the context of DFT, the signal X is expressed as a linear combination of the complex exponential DFT basis functions, the $\mathbf{g}_k$'s. The coefficients are given by the $\tilde{X}(k)$'s, which form the DFT of X. Fourier analysis, provided by the DFT in the discrete setting, is one of the most important topics in mathematics and has wide-ranging applications in many scientific and engineering disciplines. For a systematic study of the subject, we refer the reader to the classic text by Bracewell [Bra99].

The connection we seek, between DFT and spectral analysis with respect to the Laplace operator, is that the DFT basis functions, the $\mathbf{g}_k$'s, form a set of eigenvectors of the 1D discrete Laplace operator L, as defined in (3). A proof of this fact can be found in Jain's classic text on image processing [Jai89], where a stronger claim with respect to circulant matrices was made. Note that a matrix is circulant if each row can be obtained as a shift (with circular wrap-around) of the previous row. It is clear that L is circulant.

Specifically, if we sort the eigenvalues of L in ascending order, then they are

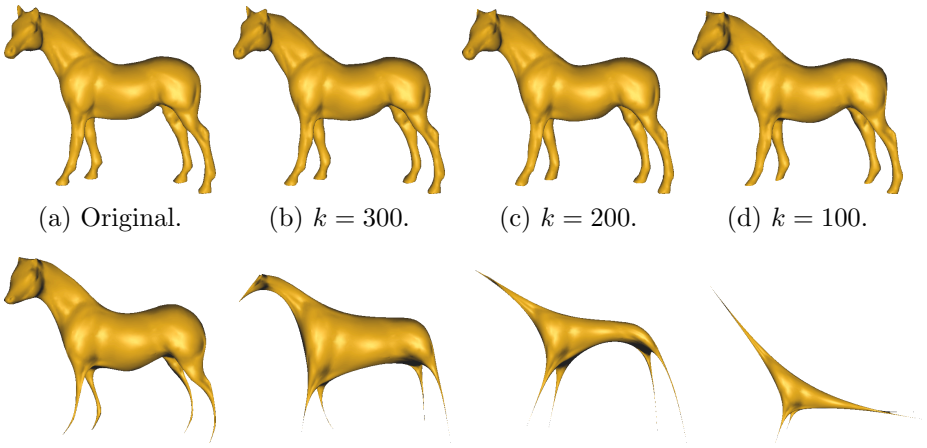
$$\lambda_k = 2\sin^2 \frac{\pi\lfloor k/2 \rfloor}{n}, k = 2, \ldots, n. \tag{9}$$

The first eigenvalue λ_1 is always 0. We can see that every eigenvalue of L, except for the first, and possibly the last, has a multiplicity of 2. That is, it corresponds to an eigensubspace spanned by two eigenvectors. If we define the matrix G of DFT basis as $G_{kj} = e^{i2\pi(j-1)(k-1)/n}$, $1 \leq k, j \leq n$, then the first column of G is an eigenvector corresponding to λ_1 and the k-th and $(n+2-k)$-th columns of G are two eigenvectors corresponding to λ_k, for $k = 2, \ldots, n$. The set of eigenvectors of L is not unique. In particular, it has a set of real eigenvectors; some of these eigenvectors are plotted in Figure 4.

1.6 Towards spectral mesh transform

Based on the above observation, one way to extend the notion of Fourier analysis to the manifold or surface setting, where our signal will represent the geometry of the surfaces, is to define appropriate discrete Laplace operators for meshes and rely on the eigenvectors of the Laplace operators to perform Fourier analysis. This was already observed in Taubin's seminal paper [Tau95b].

To extend spectral analysis of 1D signals presented in Section 1.2 to surfaces modeled by triangle meshes, we first need to extend the signal representation. This is quite straightforward: any function defined on the mesh vertices can be seen as a discrete mesh signal. Typically, we focus on the coordinate signal for



(a) Original. (b) $k = 300$. (c) $k = 200$. (d) $k = 100$.

(e) $k = 50$. (f) $k = 10$. (g) $k = 5$. (h) $k = 3$.

Figure 8: Shape reconstruction based on spectral analysis using a typical mesh Laplace operator, where k is the number of eigenvectors or spectral coefficients used. The original model has 7,502 vertices and 15,000 faces.

the mesh, which, for a mesh with n vertices, is an $n \times 3$ matrix whose columns specify the x, y, and z coordinates of the mesh vertices.

The main task is to define an appropriate Laplace operator for the mesh. Here a crucial difference to the classical DFTs is that while the DFT basis functions are fixed as long as the length of the signal in question is determined, the eigenvectors of a mesh Laplace operator would change with the mesh connectivity and/or geometry. Formulations for the construction of appropriate mesh Laplace operators will be the subjects of Sections 2 and 3.

Now with a mesh Laplace operator chosen, defining the spectral transform of a mesh signal X with respect to the operator is exactly the same as the 1D case for L in Section 1.2. Denote by $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ the normalized eigenvectors of the mesh Laplace operator, corresponding to eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, and let E be the matrix whose columns are the eigenvectors. The vector of spectral transform coefficients $\tilde{X}$ is obtained by $\tilde{X} = E^{\mathrm{T}} X$. And for each i, we obtain the spectral coefficient by $\tilde{x}_i = \mathbf{e}_i^{\mathrm{T}} \cdot X$ via projection.

As a first visual example, we show in Figure 8 some results of spectral reconstruction, as define in (7), of a mesh model with progressively more spectral coefficients added. As we can see, higher-frequency contents of the geometric mesh signal manifest themselves as rough geometric features over the shape's surface; these features can be smoothed out by taking only the low-frequence spectral coefficients in a reconstruction. A tolerance on such loss of geometric features would lead to a JPEG-like compression of mesh geometry, as first proposed by Karni and Gotsman [KG00a] in 2000. More applications using spectral transforms of meshes will be given in Section 5.

2 Fourier analysis for meshes

The previous section introduced the idea of Fourier analysis applied to shapes, with the example of a closed curve, for which the frequencies (sine waves) are naturally obtained as the eigenvectors of the 1D discrete Laplacian. We now study how to formalize the idea presented in the last subsection, i.e. porting this setting to the case of arbitrary surfaces. Before diving into the heart of the matter, we recall the definition of the Fourier transform, that is to say the continuous version of the discrete signal processing framework presented in Subsections 1.2 and 1.4.

2.1 Fourier analysis

As in Taubin's article [Tau95b], we start by studying the case of a closed curve, but staying in the continuous setting. Given a square-integrable periodic function $f : x \in [0,1] \mapsto f(x)$, or a function f defined on a closed curve parameterized by normalized arclength, it is well known that f can be expanded into an infinite series of sines and cosines of increasing frequencies:

$$f(x) = \sum_{k=0}^{\infty} \tilde{f}_k H^k(x) \quad ; \quad \left\{ \begin{array}{lcl} H^0 & = & 1 \\ H^{2k+1} & = & \cos(2k\pi x) \\ H^{2k+2} & = & \sin(2k\pi x) \end{array} \right. \tag{10}$$

where the coefficients $\tilde{f}_k$ of the decomposition are given by:

$$\tilde{f}_k = < f, H^k > = \int_0^1 f(x) H^k(x) dx \tag{11}$$

and where $< ., . >$ denotes the inner product (i.e. the "dot product" for functions defined on $[0,1]$). See [Arv95] or [Lev06] for an introduction to functional analysis. The "Circle harmonics" basis H^k is orthonormal with respect to $< ., . >$: $< H^k, H^k >= 1$, $< H^k, H^l >= 0$ if $k \neq l$.

The set of coefficients $\tilde{f}_k$ (Equation 11) is called the Fourier Transform (FT) of the function f. Given the coefficients $\tilde{f}_k$, the function f can be reconstructed by applying the inverse Fourier Transform FT^{-1} (Equation 10). Our goal is now to explain the generalization of these notions to arbitrary manifolds. To do so, we can consider the functions H^k of the Fourier basis as the eigenfunctions of $-\partial^2/\partial x^2$: the eigenfunctions H^{2k+1} (resp. H^{2k+2}) are associated with the eigenvalues $(2k\pi)^2$:

$$-\frac{\partial^2 H^{2k+1}(x)}{\partial x^2} = (2k\pi)^2 \cos(2k\pi x) = (2k\pi)^2 H^{2k+1}(x)$$

This construction can be extended to arbitrary manifolds by considering the generalization of the second derivative to arbitrary manifolds, i.e. the Laplace operator and its variants, introduced below. Before studying the continuous theory, we first do a step backward into the discrete setting, in which it is easier

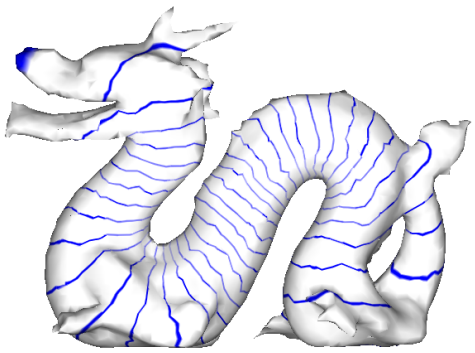


Figure 9: The Fiedler vector gives a natural ordering of the nodes of a graph. The displayed contours show that it naturally follows the shape of the dragon.

to grasp a geometric meaning of the eigenfunctions. In this setting, eigenfunctions can be considered as orthogonal non-distorting 1D parameterizations of the shape, as explained further.

2.2 The discrete setting: Graph Laplacians

Spectral graph theory was used for instance in [IL05] to compute an ordering of vertices in a mesh that facilitates out-of-core processing. Such a natural ordering can be derived from the Fiedler eigenvector of the Graph Laplacian. The Graph Laplacian $L = (a_{i,j})$ is a matrix defined by:

$$\begin{array}{lcll} a_{i,j} & = & w_{i,j} > 0 & \text{if } (i,j) \text{ is an edge} \\ a_{i,i} & = & -\sum_j w_{i,j} & \\ a_{i,j} & = & 0 & \text{otherwise} \end{array}$$

where the coefficients $w_{i,j}$ are weights associated to the edges of the graph. One may use the uniform weighting $w_{i,j} = 1$ or more elaborate weightings, computed from the embedding of the graph. There are several variants of the Graph Laplacian, the reader is referred to [ZvKDar] for an extensive survey. Among these discrete Laplacians, the so-called Tutte Laplacian applies a normalization in each row of L and is given by $T = (t_{i,j})$, where

$$\begin{array}{lcll} t_{i,j} & = & \frac{w_{i,j}}{\sum_j w_{i,j}} > 0 & \text{if } (i,j) \text{ is an edge} \\ t_{i,i} & = & -1 & \\ t_{i,j} & = & 0 & \text{otherwise} \end{array}$$

The Tutte Laplacian was employed in the original work of Taubin [Tau95a], among others [GGS03a, Kor03].

The first eigenvector of the Graph Laplacian is $(1, 1 \ldots 1)$ and its associated eigenvalue is 0. The second eigenvector is called the Fiedler vector and

has interesting properties, making it a good permutation vector for numerical computations [Fie73, Fie75]. It has many possible applications, such as finding natural vertices ordering for streaming meshes [IL05]. Figure 9 shows what it looks like for a snake-like mesh (it naturally follows the shape of the mesh).

More insight on the Fiedler vector is given by the following alternative definition. The Fiedler vector $u = (u_1 \dots u_n)$ is the solution of the following constrained minimization problem:

$$\begin{array}{ll} \text{Minimize:} & F(u) = u^t L u = \sum_{i,j} w_{i,j}(u_i - u_j)^2 \\ \text{Subject to:} & \sum_i u_i = 0 \quad \text{and} \quad \sum_i u_i^2 = 1 \end{array} \tag{12}$$

In other words, given a graph embedded in some space, and supposing that the edge weight $w_{i,j}$ corresponds to the lengths of the edges in that space, the Fielder vector $(u_1 \dots u_n)$ defines a (1-dimensional) embedding of the graph on a line that tries to respect the edge lengths of the graph.

This naturally leads to the question of whether embedding in higher-dimensional spaces can be computed (for instance, computing a 2-dimensional embedding of a surface corresponds the the classical surface parameterization problem). This general problem is well known by the *automatic learning* research community as a *Manifold learning* problem, also called *dimension reduction*.

One of the problems in manifold learning is extracting from a set of input (e.g. a set of images of the same object) some meaningful parameters (e.g. camera orientation and lighting conditions), and sort these images with respect to these parameters. From an abstract point of view, the images leave in a high-dimensional space (the dimension corresponds to the number of pixels of the images), and one tries to *parameterize* this image space. The first step constructs a graph, by connecting each sample to its nearest neighbors, according to some distance function. Then, different classes of methods have been defined, we quickly review the most popular ones:

Local Linear Embedding [RS00a] tries to create an embedding that best approximates the barycentric coordinates of each vertex relative to its neighbors. In a certain sense, Floater's Shape Preserving Parameterization (see [FH04]) is a particular case of this approach.

Isomap [TdSL00] computes the geodesic distances between each pair of vertex in the graph, and then uses MDS (*multidimensional scaling*) [You85] to compute an embedding that best approximates these distances. Multidimensional scaling simply minimizes an objective function that measures the deviation between the geodesic distances in the initial space and the Euclidean distances in the embedding space (GDD for Geodesic Distance Deviation), by computing the eigenvectors of the matrix $D = (d_{i,j})$ where $d_{i,j}$ denotes the geodesic distance between vertex i and vertex j. This is a multivariate version of Equation 12, that characterizes the Fiedler vector (in the univariate setting). Isomaps and Multidimensional scaling were used to define parameterization algorithms in [ZKK02], and more recently in the *ISO-charts* method [ZSGS04], used in Microsoft's DirectX combined with the packing algorithm presented in [LPM02].

At that point, we understand that the eigenvectors play an important role in determining meaningful parameters. Just think about the simple linear case: in PCA (principal component analysis), the eigenvectors of the covariance matrix characterize the most appropriate hyperplane on which the data should be projected. In dimension reduction, we seek for eigenvectors that will fit non-linear features. For instance, in MDS, these eigenvectors are computed in a way that makes the embedding space mimic the global metric structure of the surface, captured by the matrix $D = (d_{i,j})$ of all geodesic distances between all pairs of vertices in the graph.

Instead of using the dense matrix D, methods based on the Graph Laplacian only use local neighborhoods (one-ring neighborhoods). As a consequence, the used matrix is sparse, and extracting its eigenvectors requires lighter computations. Note that since the Graph Laplacian is a symmetric matrix, its eigenvectors are orthogonal, and can be used as a vector basis to represent functions. This was used in [KG00b] to define a compact encoding of mesh geometry. The basic idea consists in encoding the topology of the mesh together with the coefficients that define the geometry projected onto the basis of eigenvectors. The decoder simply recomputes the basis of eigenvectors and multiplies them with the coefficients stored in the file. A survey of spectral geometry compression and its links with graph partitioning is given in [Got03]. Spectral graph theory also enables to exhibit ways of defining valid graph embeddings. For instance, Colin-de-verdière's number [dV90] was used in [GGS03b] to construct valid spherical embeddings of genus 0 meshes. Other methods that use spectral graph theory to compute graph embeddings are reviewed in [Kor02]. Spectral graph theory can also be used to compute topological invariants (e.g. Betti numbers), as explained in [Fei96]. As can be seen from this short review of spectral graph theory, the eigenvectors and eigenvalues of the graph Laplacian contain both geometric and topological information.

However, as explained in [ZSGS04], only using the connectivity of the graph may lead to highly distorted mappings. Methods based on MDS solve this issue by considering the matrix D of the geodesic distances between all possible pairs of vertices. However, we think that it is also possible to inject more geometry in the Graph Laplacian approach, and understand how the global geometry and topology of the shape may emerge from the interaction of local neighborhoods.

This typically refers to notions from the *continuous* setting, i.e. functional analysis and operators. The next section shows its link with the Laplace-Beltrami operator, that appears in the wave equation (Helmholtz's equation). We will also exhibit the link between the so-called *stationary waves* and spectral graph theory.

2.3 The Continuous Setting: Laplace Beltrami

The Laplace operator (or Laplacian) plays a fundamental role in physics and mathematics. In $\mathbb{R}^n$, it is defined as the divergence of the gradient:

$$\Delta = \text{div grad} = \nabla.\nabla = \sum_i \frac{\partial^2}{\partial x_i^2}$$

Intuitively, the Laplacian generalizes the second order derivative to higher dimensions, and is a characteristic of the irregularity of a function as $\Delta f(P)$ measures the difference between $f(P)$ and its average in a small neighborhood of P.

Generalizing the Laplacian to curved surfaces require complex calculations. These calculations can be simplified by a mathematical tool named exterior calculus (EC) [1]. EC is a coordinate free geometric calculus where functions are considered as abstract mathematical objects on which operators act. To use these functions, we cannot avoid instantiating them in some coordinate frames. However, most calculations are simplified thanks to higher-level considerations. For instance, the divergence and gradient are known to be coordinate free operators, but are usually defined through coordinates. EC generalizes the gradient by d and divergence by δ, which are built independently of any coordinate frame.

Using EC, the definition of the Laplacian can be generalized to functions defined over a manifold $\mathcal{S}$ with metric g, and is then called the Laplace-Beltrami operator:

$$\Delta = \text{div grad} = \delta d = \sum_i \frac{1}{\sqrt{|g|}} \frac{\partial}{\partial x_i} \sqrt{|g|} \frac{\partial}{\partial x_i}$$

where $|g|$ denotes the determinant of g. The additional term $\sqrt{|g|}$ can be interpreted as a local "scale" factor since the local area element dA on $\mathcal{S}$ is given by $dA = \sqrt{|g|} dx_1 \wedge dx_2$.

Finally, for the sake of completeness, we can mention that the Laplacian can be extended to k-forms and is then called the Laplace-de Rham operator defined by $\Delta = \delta d + d\delta$. Note that for functions (i.e. 0-forms), the second term $d\delta$ vanishes and the first term δd corresponds to the previous definition.

Since we aim at constructing a function basis, we need some notions from functional analysis, quickly reviewed here. A similar review in the context of light simulation is given in [Arv95].

2.3.1 Functional analysis

In a way similar to what is done for vector spaces, we need to introduce a dot product (or inner product) to be able to define function bases and project functions onto those bases. This corresponds to the notion of Hilbert space, outlined

[1] To our knowledge, besides Hodge duality used to compute minimal surfaces [PP93], one of the first uses of EC in geometry processing [GY02] applied some of the fundamental notions involved in the proof of Poincaré's conjecture to global conformal parameterization. More recently, a Siggraph course was given by Schroeder *et al.* [SGD05], making these notions usable by a wider community.

below.

Hilbert spaces

Given a surface S, let X denote the space of real-valued functions defined on S. Given a norm $\|.\|$, the function space X is said to be *complete* with respect to the norm if Cauchy sequences converge in X, where a Cauchy sequence is a sequence of functions $f_1, f_2, \ldots$ such that $\lim_{n,m->\infty} \|f_n - f_m\| = 0$. A complete vector space is called a *Banach space*.

The space X is called a *Hilbert space* in the specific case of the norm defined by $\|f\| = \sqrt{< f, f >}$, where $< ., . >$ denotes an inner product. A possible definition of an inner product is given by $< f, g >= \int_S f(x)g(x)dx$, which yields the L_2 norm.

One of the interesting features provided by this additional level of structure is the ability to define function bases and project onto these bases using the inner product. Using a function basis (Φ_i), a function f will be defined by $f = \sum \alpha_i \Phi_i$. Similarly to the definition in geometry, a function basis (Φ_i) is *orthonormal* if $\|\Phi_i\| = 1$ for all i and $< \Phi_i, \Phi_j >= 0$ for all $i \neq j$. Still following the analogy with geometry, given the function f, one can easily retrieve its coordinates α_i with respect to an orthonormal basis (Φ_i) by *projecting* f onto the basis, i.e. $\alpha_i =< f, \Phi_i >$.

Operators

We now give the basic definitions related with operators. Simply put, an operator is a function of functions (i.e. from X to X). An operator L applied to a function $f \in X$ is denoted by Lf, and results in another function in X. An operator L is said to be *linear* if $L(\lambda f) = \lambda Lf$ for all $f \in X, \lambda \in \mathbb{R}$. An *eigenfunction* of the operator L is a function f such that $Lf = \lambda f$. The scalar λ is an *eigenvalue* of L. In other words, the effect of applying the operator to one of its eigenfunctions means simply *scaling* the function by the scalar λ.

A linear operator L is said to be *Hermitian* (or with Hermitian symmetry)[2] if $< Lf, g >=< f, Lg >$ for each $f, g \in X$. An important property of Hermitian operators is that their eigenfunctions associated to different eigenvalues have real eigenvalues and are orthogonal. This latter property can be easily proven as follows, by considering two eigenfunctions f, g associated with the different eigenvalues λ, μ respectively:

$$\begin{aligned} < Lf, g > &= < f, Lg > \\ < \lambda f, g > &= < f, \mu g > \\ \lambda < f, g > &= \mu < f, g > \end{aligned}$$

which gives the result ($< f, g >= 0$) since $\lambda \neq \mu$.

As a consequence, considering the eigenfunctions of an Hermitian operator is a possible way of defining an orthonormal function basis associated to a given

[2] the general definition of Hermitian operators concerns complex-valued functions, we only consider here real-valued functions.

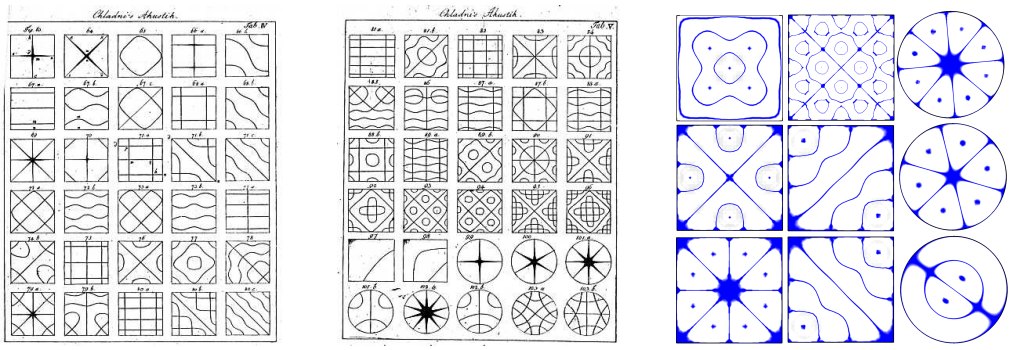


Figure 10: Left: In the late 1700's, the physicist Ernst Chladni was amazed by the patterns formed by sand on vibrating metal plates. Right: numerical simulations obtained with a discretized Laplacian.

function space X. The next section shows this method applied to the Laplace-Beltrami operator. Before entering the heart of the matter, we will first consider the historical perspective.

2.3.2 Chladni plates

In 1787, the physicist Ernst Chladni published the book entitled "Discoveries Concerning the Theories of Music". In this book, he reports his observations obtained when putting a thin metal plate into vibration using a bow, and spreading sand over it. Sand accumulates in certain zones, forming surprisingly complex patterns (see Figure 10).

This behavior can be explained by the theory of *stationary waves*. When the metal plate vibrates, some zones remain still, and sand naturally concentrates in these zones. This behavior can be modeled as follows, by the spatial component of Helmholtz's wave propagation equation:

$$\Delta f = \lambda f \tag{13}$$

In this equation, Δ denotes the Laplace-Beltrami operator on the considered object. In Cartesian 2D space, $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$. We are seeking for the eigenfunctions of this operator. To better understand the meaning of this equation, let us first consider a vibrating circle. This corresponds to the univariate case on the interval $[0, 2\pi]$ with cyclic boundary conditions (i.e. $f(0) = f(2\pi)$). In this setting, the Laplace-Beltrami operator simply corresponds to the second order derivative. Recalling that $\sin(\omega x)'' = \omega^2 sin(\omega x)$, the eigenfunctions are simply $\sin(Nx), \cos(Nx)$ and the constant function, where N is an integer. Note that the so-constructed function basis is the one used in Fourier analysis.

From the spectrum of the Laplace-Beltrami operator, it is well known that one can extract the area of S, the length of its border and its genus. This leads to the question asked by Kac in 1966: *Can we hear the shape of a drum*

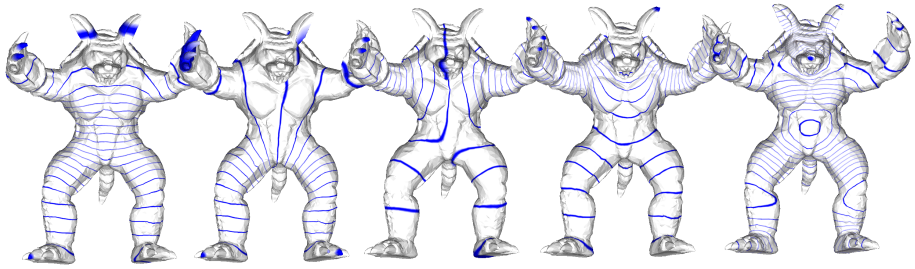


Figure 11: Contours of the first eigenfunctions. Note how the protrusions and symmetries are captured. The eigenfunctions combine the geometric and topological information contained in the shape signal.

? [Kac66]. The answer to this question is "no": one can find drums that have the same spectrum although they do not have the same shape [Cip93] (they are then referred to as *isospectral* shapes). However, the spectrum contains much information, which led to the idea of using it as a signature for shape matching and classification, as explained in the "shape DNA" approach [RWP05a].

We are now going now to take a look at the eigenfunctions. Mathematicians mostly studied bounds and convergence of the spectrum. However, some results are known about the *geometry* of the eigenfunctions [JNT]. More precisely, we are interested in the so-called *nodal sets*, defined to be the zero-set of an eigenfunction. Intuitively, they correspond to the locations that do not move on a Chladni plate, where sand accumulates (see Figure 10). A nodal set partitions the surface into a set of *nodal domains* of constant sign. The nodal sets and nodal domains are characterized by the following theorems:

1. the n-th eigenfunction has at most n nodal domains

2. the nodal sets are curves intersecting at constant angles

Besides their orthogonality, these properties make eigenfunction an interesting choice for function bases. Theorem 1 exhibits their multi-resolution nature, and from theorem 2, one can suppose that they are strongly linked with the geometry of the shape. Note also that these theorems explain the appearance of Chladni plates. This may also explain the very interesting re-meshing results obtained by Dong *et. al* [DBG+05], that use a Morse-smale decomposition of one of the eigenfunctions.

In the case of simple objects, a closed form of the eigenfunctions can be derived. This made it possible to retrieve the patterns observed by Chladni in the case of a square and a circle. For curved geometry, Chladni could not make the experiment, since sand would not remain in the nodal set. However, one can still study the eigenfunctions. For instance, on a sphere, the eigenfunction correspond to *spherical harmonics* (see e.g. [JNT]), often used in computer graphics to represent functions defined on the sphere (such as radiance fields or

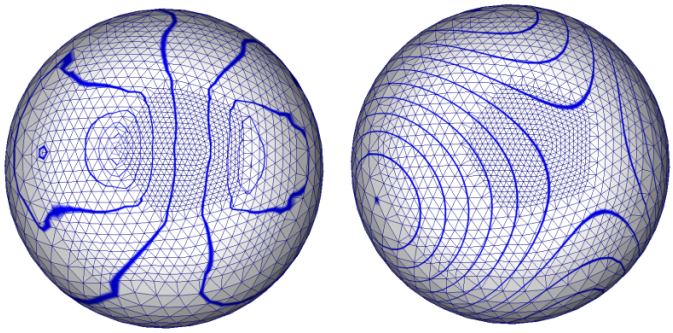


Figure 12: Contours of the 4th eigenfunction, computed from the Graph Laplacian (left) and cotangent weights (right) on an irregular mesh.

Bidirectional Reflectance Distribution Functions). In other words, on a sphere, the eigenfunctions of the Laplace-Beltrami operator define an interesting hierarchical function basis. One can now wonder whether the same approach could be used to create function bases on more complex geometries. In the general case, a closed form cannot be derived, and one needs to use a numerical approach, as explained in the next section.

3 Discretizing the Laplace operator

The previous section mentioned two different settings for spectral analysis :

- *the discrete setting* is concerned with graphs, matrices and vectors;
- *the continuous setting* is concerned with manifolds, operators and functions.

The continuous setting belongs to the realm of differential geometry, a powerful formalism for studying relations between manifolds and functions defined over them. However, computer science can only manipulate discrete quantities. Therefore, a practical implementation of the continuous setting requires a *discretization*. This section shows how to convert the continuous setting into a discrete counterpart using the standard Finite Element Modeling technique. In Geometry Processing, another approach consists in trying to define a discrete setting that mimics the property of the continous theory. As such, we will also show how spectral geometry processing can be derived using Discrete Exterior Calculus (DEC).

The eigenfunctions and eigenvalues of the Laplacian on a (manifold) surface $\mathcal{S}$, are all the pairs (H^k, λ_k) that satisfy:

$$-\Delta H^k = \lambda_k H^k \tag{14}$$

Figure 13: Some functions of the Manifold Harmonic Basis (MHB) on the Gargoyle dataset

The "−" sign is here required for the eigenvalues to be positive. On a closed curve, the eigenfunctions of the Laplace operator define the function basis (sines and cosines) of Fourier analysis, as recalled in the previous section. On a square, they correspond to the function basis of the DCT (Discrete Cosine Transform), used for instance by the JPEG image format. Finally, the eigenfunctions of the Laplace-Beltrami operator on a sphere define the Spherical Harmonics basis.

In these three simple cases, two reasons make the eigenfunctions a function basis suitable for spectral analysis of manifolds:

1. Because the Laplacian is symmetric ($< \Delta f, g >=< f, \Delta g >$), its eigenfunctions are orthogonal, so it is extremely simple to project a function onto this basis, i.e. to apply a Fourier-like transform to the function.

2. For physicists, the eigenproblem (Equation 14) is called the Helmoltz equation, and its solutions H^k are stationary waves. This means that the H^k are functions of constant wavelength (or spatial frequency) $\omega_k = \sqrt{\lambda_k}$.

Hence, using the eigenfunctions of the Laplacian to construct a function basis on a manifold is a natural way to extend the usual spectral analysis to this manifold. In our case, the manifold is a mesh, so we need to port this construction to the discrete setting. The first idea that may come to the mind is to apply spectral analysis to a discrete Laplacian matrix (e.g. the cotan weights). However, the discrete Laplacian is not a symmetric matrix (the denominator of the $a_{i,j}$ coefficient is the area of vertex $i's$ neighborhood, that does not necessarily correspond to the area of vertex j's neighborhood). Therefore, we lose the symmetry of the Laplacian and the orthogonality of its eigenvectors. This makes it difficult to project functions onto the basis. For this reason, we will clarify the relations between the continuous setting (with functions and operators), and the discrete one (with vectors and matrices) in the next section.

In this section, we present two ways of discretizing the Laplace operator on a mesh. The first approach is based on Finite Element Modeling (FEM) such as done in [WBH+07], and converge to the continuous Laplacian under certain conditions as explained in [HPW06] and [AFW06]. Reuter *et al.* [RWP05b] also use FEM to compute the spectrum (i.e. the eigenvalues) of a mesh, which provides a signature for shape classification. The cotan weights were also used in [DBG+06b] to compute an eigenfunction to steer a quad-remeshing process.

The cotan weights alone are still dependent on the sampling as shown in Figure 18-B, so they are usually weighted by the one ring or dual cell area of each vertex, which makes them loose their symmetry. As a consequence, they are improper for spectral analysis (18-C). An empirical symmetrization was proposed in [Lev06] (see Figure 18-D). The FEM formulation enables to preserve the symmetry property of the continuous counterpart of the Laplace operator.

It is also possible to derive a symmetric Laplacian by using the DEC formulation (Discrete Exterior Calculus). Then symmetry is recovered by expressing the operator in a proper basis. This ensures that its eigenfunctions are both geometry aware and orthogonal (Figure 18-E). Note that a recent important proof [WMKG07] states that a "perfect" discrete Laplacian that satisfies all the properties of the continuous one cannot exist on general meshes. This explains the large number of definitions for a discrete Laplacian, depending on the desired properties.

3.1 The FEM Laplacian

In this subsection, we explain the Finite Element Modeling (FEM) formulation of the discrete Laplacian. The Discrete Exterior Calculus (DEC)formululation, presented during the course, is described in the next subsection. We have chosen to include in these course nodes the full derivations for the FEM Laplacian for the sake of completeness.

To setup the finite element formulation, we first need to define a set of *basis functions* used to express the solutions, and a set of *test functions* onto which the eigenproblem (Equation 14) will be projected. As it is often done in FEM, we choose for both basis and test functions the same set $\Phi^i (i = 1 \dots n)$. We use the "hat" functions (also called P_1), that are piecewise-linear on the triangles, and that are such that $\Phi^i(i) = 1$ and $\Phi^i(j) = 0$ if $i \neq j$. Geometrically, Φ^i corresponds to the barycentric coordinate associated with vertex i on each triangle containing i. Solving the finite element formulation of Equation 14 relative to the Φ^i's means looking for functions of the form: $H^k = \sum_{i=1}^{n} H_i^k \Phi^i$ which satisfy Equation 14 in projection on the Φ^j's:

$$\forall j, < -\Delta H^k, \Phi^j >= \lambda_k < H^k, \Phi^j >$$

or in matrix form:

$$-Q\mathbf{h}^k = \lambda B \mathbf{h}^k \tag{15}$$

where $Q_{i,j} =< \Delta\Phi^i, \Phi^j >$, $B_{i,j} =< \Phi^i, \Phi^j >$ and where $\mathbf{h}^k$ denotes the vector $[H_1^k, \dots H_n^k]$. The matrix Q is called the *stiffness matrix*, and B the *mass matrix*.

This appendix derives the expressions for the coefficients of the stiffness matrix Q and the mass matrix B. To do so, we start by parameterizing a triangle $t = (i, j, k)$ by the barycentric coordinates (or hat functions) Φ_i and Φ_j of a point $P \in t$ relative to vertices i and j. This allows to write $P = k + \Phi_i \mathbf{e}_j - \Phi_j \mathbf{e}_i$ (Figure 14). This yields an area element $dA(P) = \mathbf{e}_i \wedge \mathbf{e}_j d\Phi_i d\Phi_j = 2|t| d\Phi_i d\Phi_j$,

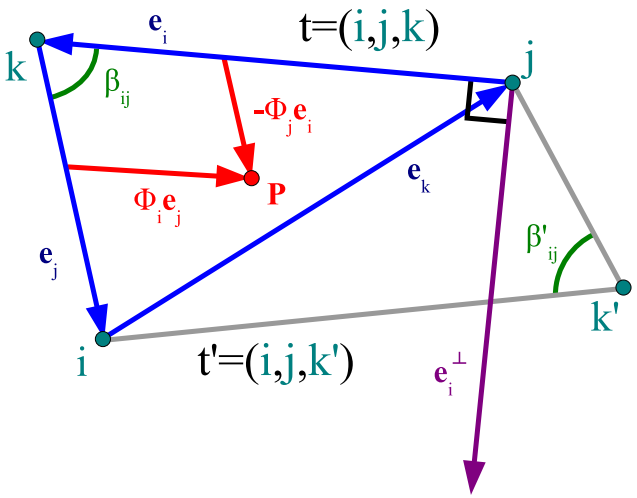


Figure 14: Notations for matrix coefficients computation. Vectors are typed in bold letters ($\mathbf{e}_i$)

where $|t|$ is the area of t, so we get the integral:

$$\int_{P\in t} \Phi_i\Phi_j dA = 2|t| \int_{\Phi_i=0}^{1} \int_{\Phi_j=0}^{1-\Phi_i} \Phi_i\Phi_j d\Phi_i d\Phi_j =$$

$$|t| \int_{\Phi_i=0}^{1} \Phi_i(1-\Phi_i)^2 d\Phi_i = |t| \left(\frac{1}{2} - \frac{2}{3} + \frac{1}{4}\right) = \frac{|t|}{12}$$

which we sum up on the 2 triangles sharing (i,j) to get $B_{i,j} = (|t| + |t'|/12$. We get the diagonal terms by:

$$\int_{P\in t} \Phi_i^2 dA = 2|t| \int_{\Phi_i=0}^{1} \int_{\Phi_j=0}^{1-\Phi_i} \Phi_i^2 d\Phi_j =$$

$$2|t| \int_{\Phi_i=0}^{1} \Phi_i^2(1-\Phi_i) d\Phi_i = 2|t| \left(\frac{1}{3} - \frac{1}{4}\right) = \frac{|t|}{6}$$

which are summed up over the set $St(i)$ of triangles containing i to get $B_{i,i} = (\sum_{t\in St(i)} |t|)/6$.

To compute the coefficients of the stiffness matrix Q, we use the fact that d and δ are adjoint to get the more symmetric expression:

$$Q_{i,j} =< \Delta\Phi^i, \Phi^j >=< \delta d\Phi^i, \Phi^j >=< d\Phi^i, d\Phi^j >= \int_{\mathcal{S}} \nabla\Phi^i . \nabla\Phi^j$$

In t, the gradients of barycentric coordinates are the constants :

$$\nabla\Phi^i = \frac{-\mathbf{e}_i^{\perp}}{2|t|} \qquad \nabla\Phi^i . \nabla\Phi^j = \frac{\mathbf{e}_i . \mathbf{e}_j}{4|t|^2}$$

Where $\mathbf{e}_i^{\perp}$ denotes $\mathbf{e}_i$ rotated by $\pi/2$ around t's normal. By integrating on t we get:

$$\int_t \nabla\Phi^i.\nabla\Phi^j dA = \frac{\mathbf{e}_i.\mathbf{e}_j}{4|t|} = \frac{||\mathbf{e}_i||.||\mathbf{e}_j||\cos(\beta_{ij})}{2||\mathbf{e}_i||.||\mathbf{e}_j||\sin(\beta_{ij})} = \frac{\cot(\beta_{ij})}{2}$$

Summing these expressions, the coefficients of the stiffness matrix Q are given by:

$$Q_{i,i} = \sum_{t\in St(i)} \nabla\Phi^i.\nabla\Phi^i = \sum_{t\in St(i)} \frac{\mathbf{e}_i^2}{4|t|}$$

$$Q_{i,j} = \int_{t\cup t'} \nabla\Phi^i.\nabla\Phi^j = \frac{1}{2}\left(\cot(\beta_{ij}) + \cot(\beta'_{ij})\right)$$

Note that this expression is equivalent to the numerator of the classical cotan weights.

$$\begin{cases} Q_{i,j} &= \left(\text{cotan}(\beta_{i,j}) + \text{cotan}(\beta'_{i,j})\right)/2 \\ Q_{i,i} &= -\sum_j Q_{i,j} \end{cases}$$
$$\begin{cases} B_{i,j} &= \left(|t| + |t'|\right)/12 \\ B_{i,i} &= \left(\sum_{t\in St(i)} |t|\right)/6 \end{cases} \tag{16}$$

where t, t' are the two triangles that share the edge (i,j), $|t|$ and $|t'|$ denote their areas, $\beta_{i,j}$, $\beta'_{i,j}$ denote the two angles opposite to the edge (i,j) in t and t', and $St(i)$ denotes the set of triangles incident to i.

To simplify the computations, a common practice of FEM consists in replacing this equation with an approximation:

$$-Q\mathbf{h}^k = \lambda D\mathbf{h}^k \quad \left(\text{or} \quad -D^{-1}Q\mathbf{h}^k = \lambda\mathbf{h}^k\right) \tag{17}$$

where the mass matrix B is replaced with a diagonal matrix D called the *lumped mass matrix*, and defined by:

$$D_{i,i} = \sum_j B_{i,j} = \left(\sum_{t\in St(i)} |t|\right)/3. \tag{18}$$

Note that D is non-degenerate (as long as mesh triangles have non-zero areas). FEM researchers [Pra99] explain that besides simplifying the computations this approximation fortuitously improves the accuracy of the result, due to a cancellation of errors, as pointed out in [Dye06]. The practical solution mechanism to solve Equation 17 will be explained further in the section about numerics.

Remark: The matrix $D^{-1}Q$ in (Equation 17) exactly corresponds to the usual discrete Laplacian (cotan weights). Hence, in addition to direct derivation of triangle energies [PP93] or averaged differential values [MDSB03], the discrete Laplacian can be derived from a lumped-mass FEM formulation.

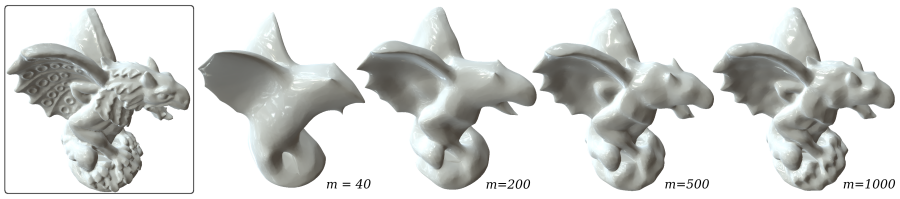


Figure 15: Reconstructions obtained with an increasing number of eigenfunctions.

As will be seen further, the FEM formulation and associated inner product will help us understand why the orthogonality of the eigenvectors seems to be lost (since $D^{-1}Q$ is not symmetric), and how to retrieve it.

Without entering the details, we mention some interesting features and degrees of freedom obtained with the FEM formulation. The function basis Φ onto the eigenfunction problem is projected can be chosen. One can use piecewise polynomial functions. Figure 16 shows how the eigenfunctions look like with the usual piecewise linear basis (also called P1 in FEM parlance) and degree 3 polynomial basis (P3). Degree 3 triangles are defined by 10 values (1 value per vertex, two values per edge and a central value). As can be seen, more complex function shapes can be obtained (here displayed using a fragment shader). It is also worth mentioning that the boundaries can be constrained in two different ways (see Figure 17). With Dirichlet boundary conditions, the value of the function is constant on the boundary (contour lines are parallel to the boundary). With Neumann boundary condistions, the gradient of the eigenfunction is parallel to the boundary (therefore, contour lines are orthogonal to the boundary).

3.2 The DEC Laplacian

For a complete introduction to DEC we refer the reader to [DKT05], [Hir03] and to [AFW06] for proofs of convergence. We quickly introduce the few notions and notations that we are using to define the inner product $< ., . >$ and generalized second-order derivative (i.e. Laplacian operator).

A k-simplex s_k is the geometric span of $k+1$ points. For instance, $0, 1$ and 2-simplices are points, edges and triangles respectively. In our context, a mesh can be defined as a 2-dimensional simplicial complex S, i.e. a collection of n_k k-simplices ($k = 0, 1, 2$), with some conditions to make it manifold. A discrete k-form ω^k on S is given by a real value $\omega^k(s_k)$ associated with each oriented k-simplex (that corresponds to the integral of a smooth k-form over the simplex). The set $\Omega^k(S)$ of k-forms on S is a vector-space of dimension n_k. With a proper numbering of the k-simplices, ω^k can be assimilated to a vector of size n_k, and linear operators from $\Omega^k(S)$ to $\Omega^l(S)$ can be assimilated to (n_k, n_l) matrices.

The *exterior derivative* $d_k : \Omega^k(S) \rightarrow \Omega^{k+1}(S)$ is defined by the signed adjacency matrix: $(d_k)_{s_k, s_{k+1}} = \pm 1$ if s_k belongs to the boundary of s_{k+1}, with the sign depending on their respective orientations.

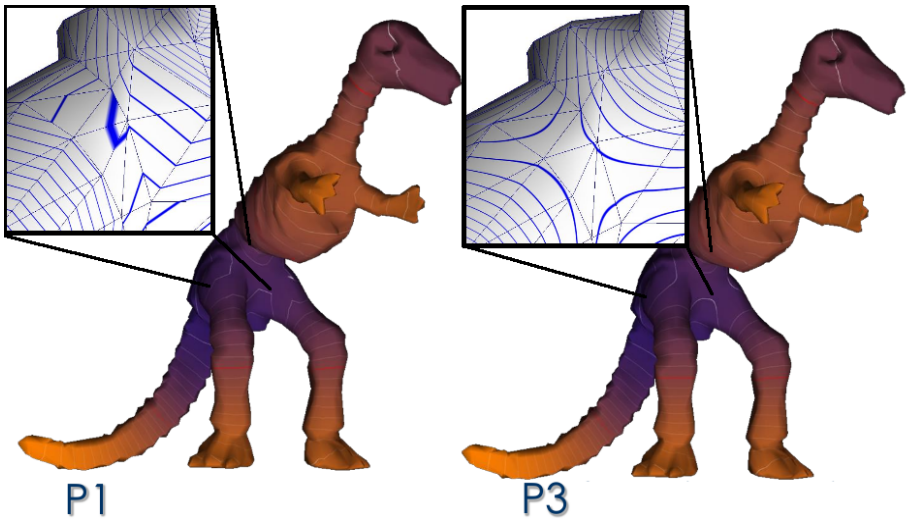


Figure 16: Contour lines of eigenfunctions obtained with a linear basis (left) and a cubic basis (right)

DEC provides $\Omega^k(S)$ with a L_2 *inner product*:

$$< \omega_1^k, \omega_2^k >= (\omega_1^k)^T \star_k \omega_2^k \tag{19}$$

where $\star_k$ is the so-called *Hodge star*. As a matrix, the Hodge star is diagonal with elements $|s_k^*|/|s_k|$ where s_k^* denotes the circumcentric dual of simplex s_k, and $|.|$ is the simplex volume. In particular, for $v$ertices, $e$dges and $t$riangles:

$$(\star_0)_{vv} = |v^*| \ ; \ (\star_1)_{ee} = \frac{|e^*|}{|e|} = \cot \beta_e + \cot \beta'_e \ ; \ (\star_2)_{tt} = |t|^{-1}$$

where β_e and β'_e denote the two angles opposite to e.

The *codifferential* $\delta_k : \Omega^k(S) \to \Omega^{k-1}(S)$ is the adjoint of the exterior derivative for the inner product:

$$< \delta_k \omega_1^k, \omega_2^{k-1} >=< \omega_1^k, d_{k-1} \omega_2^{k-1} >$$

which yields $\delta_k = - \star_{k-1}^{-1} d_{k-1}^T \star_k$.

Finally the *Laplace de Rham* operator on k-forms is given by: $\Delta_k = d_{k-1}\delta_k + \delta_{k+1} d_k$. In this paper, we are only interested in the Laplacian $\Delta = \Delta_0$, i.e. the Laplace de Rham operator for 0-forms . Since δ_0 and d_2 are undefined (zero by convention), $\Delta = \delta_1 d_0 = - \star_0^{-1} d_1^T \star_1 d_0$ and its coefficients are given by:

$$\Delta_{ij} = - \frac{\cot(\beta_{ij}) + \cot(\beta'_{ij})}{|v_i^*|} \quad ; \quad \Delta_{ii} = - \sum_j \Delta_{ij}$$

For surfaces with borders, if the edge ij is on the border, the term $\cot(\beta'_{ij})$ vanishes and the dual cell v_i^* is cropped by the border. This matches the FEM formulation with Neumann boundary conditions (not detailed here).

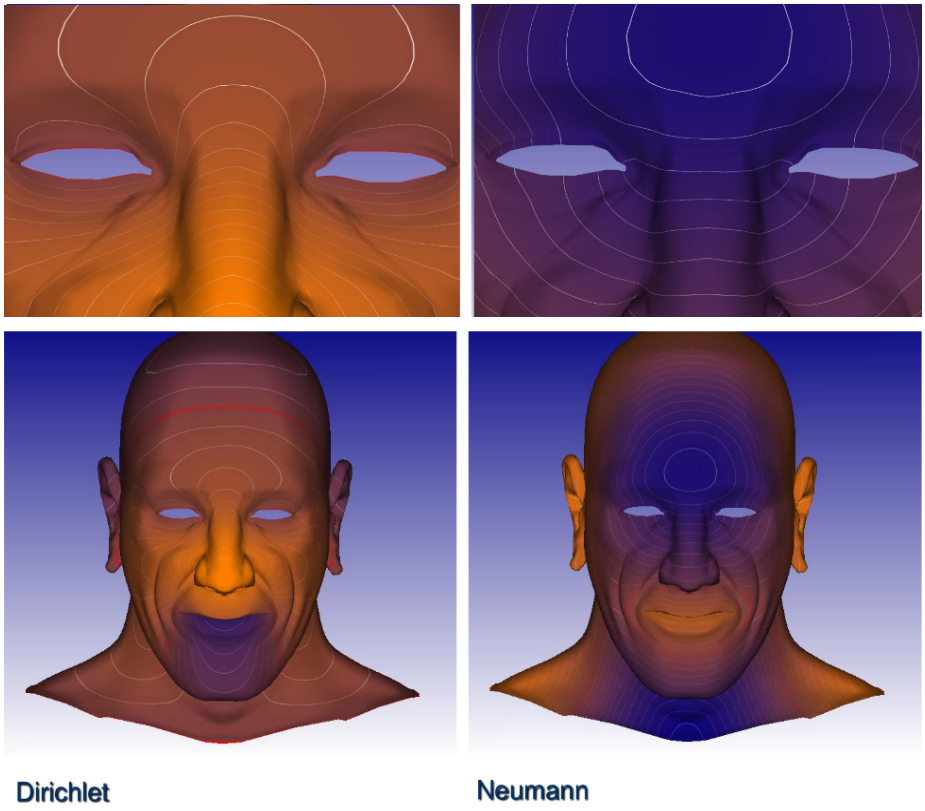


Figure 17: Contour lines of eigenfunctions obtained with Dirichlet boundary conditions (left) and Neumann boundary conditions (right)

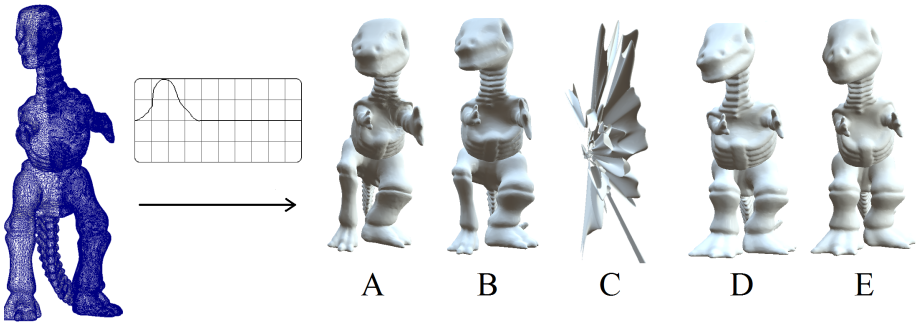


Figure 18: Filtering an irregularly sampled surface (twice denser on the right half) with different discrete laplacians. Combinatorial Laplacian (A), unweighted cotan $cot(\beta_{ij}) + cot(\beta'_{ij})$ (B) and symmetrized weighted cotan $(cot(\beta_{ij})+cot(\beta'_{ij}))/(A_i+A_j)$ (D) produce deformed results (right leg is bigger). Weighted cotan $(cot(\beta_{ij})+cot(\beta'_{ij}))/A_i$ is not symmetric which does not allow for correct reconstruction (C). Only symmetric weights $(cot(\beta_{ij})+cot(\beta'_{ij}))/\sqrt{A_i A_j}$ is fully mesh-independent (E).

Remark: The matrix Δ corresponds to the standard discrete Laplacian, except for the sign. The sign difference comes from the distinction between Laplace-Beltrami and Laplace de Rham operators.

The so-defined Laplacian Δ apparently looses the *symmetry* of its continuous counterpart ($\Delta_{ij} \neq \Delta_{ji}$). This makes the eigenfunction basis no longer orthonormal, which is problematic for our spectral processing purposes (Figure 18-C). To recover symmetry, consider the canonical basis (ϕ_i) of 0-forms: $\phi_i = 1$ on vertex i and $\phi_i = 0$ on other vertices. This basis is orthogonal but not normal with respect to the inner product defined in Equation 19 ($< \phi_i, \phi_i >= (\phi_i)^T \star_0 \phi_i \neq 1$). However, since the Hodge star $\star_0$ is a diagonal matrix, one can easily normalize (ϕ_i) as follows:

$$\bar{\phi}_i = \star_0^{-1/2} \phi_i$$

In this orthonormalized basis $(\bar{\phi}_i)$, the Laplacian $\bar{\Delta}$ is symmetric, and its coefficients are given by:

$$\bar{\Delta} = \star_0^{-1/2} \Delta \star_0^{1/2} \quad ; \quad \bar{\Delta}_{ij} = -\frac{\cot \beta_{ij} + \cot \beta'_{ij}}{\sqrt{|v_i^*||v_j^*|}} \tag{20}$$

4 Computing eigenfunctions

Now that we have seen two equivalent discretizations of the Laplacien, namely FEM (Finite Elements Modeling) and DEC (Discrete Exterior Calculus), we will now focus on the problem of computing the eigenfunctions in practice. An

implementation of the algorithm is available from the WIKI of the course (see web reference at the beginning of this document).

Computing the eigenfunctions means solving for the eigenvalues λ_k and eigenvectors $\bar{H}^k$ for the symmetric positive semi-definite matrix $\bar{\Delta}$:

$$\bar{\Delta}\bar{H}^k = \lambda_k \bar{H}^k$$

However, eigenvalues and eigenvectors computations are known to be extremely computationally intensive. To reduce the computation time, Karni *et al.* [KG00c] partition the mesh into smaller charts, and [DBG+06b] use multiresolution techniques. In our case, we need to compute multiple eigenvectors (typically a few thousands). This is known to be currently impossible for meshes with more than a few thousand vertices [WK05]. In this section, we show how this limit can be overcome by several orders of magnitude.

To compute the solutions of a large sparse eigenproblems, several iterative algorithms exist. The publicly available library ARPACK (used in [DBG+06b]) provides an efficient implementation of the Arnoldi method. Yet, two characteristics of eigenproblem solvers hinder us from using them directly to compute the MHB for surfaces with more than a few thousand vertices:

- first of all, we are interested in the lower frequencies, i.e. eigenvectors with associated eigenvalues lower than ω_m^2. Unfortunately, iterative solvers performs much better for the other end of the spectrum. This can be explained in terms of filtering as lower frequencies correspond to higher powers of the smoothing kernel, which may have a poor condition number;
- secondly, we need to compute a large number of eigenvectors (typically a thousand), and it is well known that computation time is superlinear in the number of requested eigenpairs. In addition, if the surface is large (millions of vertices), the MHB does not fit in system RAM.

4.1 Band-by-band computation of the MHB

We address both issues by applying spectral transforms to the eigenproblem. To get the eigenvectors of a spectral band centered around a value λ_S, we start by shifting the spectrum by λ_S, by replacing $\bar{\Delta}$ with $\bar{\Delta} - \lambda_S Id$. Then, we can swap the spectrum by inverting this matrix in $\Delta^{SI} = (\bar{\Delta} - \lambda_S Id)^{-1}$. This is called the *Shift-Invert* spectral transform, and the new eigenproblem to solve is given by:

$$\Delta^{SI}\bar{H}^k = \mu_k \bar{H}^k$$

It is easy to check that $\bar{\Delta}$ and Δ^{SI} have the same eigenvectors, and that their eigenvalues are related by $\lambda_k = \lambda_S + 1/\mu_k$. Applying an iterative solver to Δ^{SI} will return the high end of the spectrum (largest μ's), corresponding to a band of eigenvalues of $\bar{\Delta}$ centered around λ_S . It is then possible to split the MHB computation into multiple bands, and obtain a computation time that is linear in the number of computed eigenpairs. In addition, if the MHB does not fit in

RAM, each band can be streamed into a file. This gives the following algorithm:

```
(1)   λ_S ← 0   ;   λ_last ← 0
(2)   while(λ_last < ω_m^2)
(3)     compute an inverse Δ^SI of (Δ̄ − λ_S Id)
(4)     find the 50 first eigenpairs (H̄^k, μ_k) of Δ^SI
(5)     for k = 1 to 50
(6)       λ_k ← λ_S + 1/μ_k
(7)       if (λ_k > λ_last) write(H̄^k, λ_k)
(8)     end //for
(9)     λ_S ← max(λ_k) + 0.4(max(λ_k) − min(λ_k))
(10)    λ_last ← max(λ_k)
(11)  end //while
```

Before calling the eigen solver, we pre-compute Δ^{SI} with a sparse direct solver (Line 3). Note that $\bar{\Delta} - \lambda_S Id$ may be singular. This is not a problem since the spectral transform still works with an *indefinite* factorization. The factorized $\bar{\Delta} - \lambda_S Id$ is used in the inner loop of the eigen solver (Line 4). To factorize $\bar{\Delta} - \lambda_S Id$, we used sparse direct solvers (TAUCS, SUPERLU). For large models (millions of vertices), we used the sparse OOC (out-of-core) symmetric indefinite factorization [MIT06] implemented in the future release of TAUCS, kindly provided by S. Toledo. We then recover the λ's from the μ's (Line 6) and stream-write the new eigenpairs into a file (Line 7). Since the eigenvalues are centered around the shift λ_S, the shift for the next band is given by the last computed eigenvalue plus slightly less than half the bandwidth to ensure that the bands overlap and that we are not missing any eigenvalue (Line 9). If the bands do not overlap, we recompute a larger band until they do.

Note that this is different from the shift-invert spectral transform implemented by ARPACK, dedicated to iterative solvers. Ours makes use of the factorization of the matrix, resulting in much better performances.

5 Applications

The eigendecomposition of a discrete mesh Laplace operator provides a set of eigenvalues and eigenvectors, which can be directly used by an application to accomplish different tasks. Moreover, the eigenvectors can also be used as a basis onto which a signal defined on a triangle mesh is projected. The resulting spectral transform coefficients can be further analyzed or manipulated. Here we discuss a subset of the applications which utilize the spectral transform or the eigenvectors of mesh Laplace or more general linear mesh operators.

5.1 Use of eigenvectors

Eigenvectors are typically used to obtain an embedding of the input shape in the spectral domain, which we call a *spectral embedding*. After obtaining the eigendecomposition of a specific operator, the coordinates of vertex

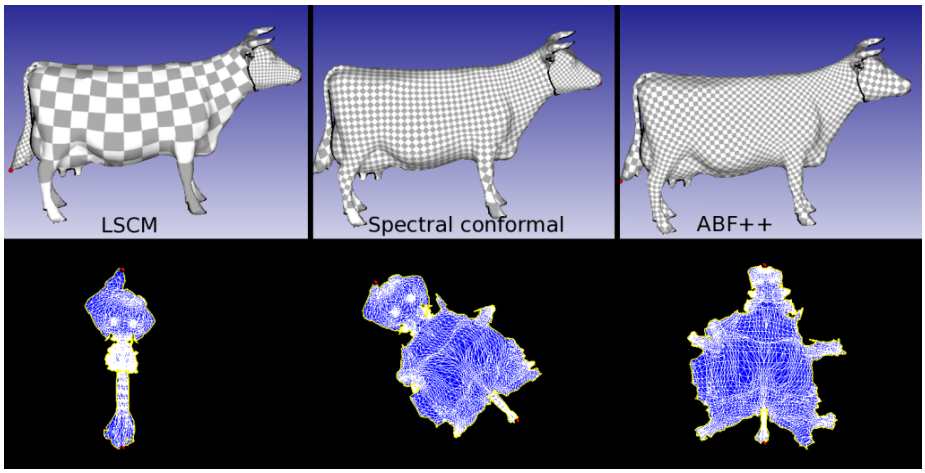


Figure 19: Compared parameterization results obtained with LSCM (left), spectral parameterization (center) and ABF++ (right).

i in a k-dimensional spectral embedding are given by the i-th row of matrix $V_k = [\mathbf{v}_1, \ldots, \mathbf{v}_k]$, where $\mathbf{v}_1, \ldots, \mathbf{v}_k$ are the first k eigenvectors from the spectrum (possibly after scaling). Whether the eigenvectors should be in ascending or descending order of eigenvalues depends on the operator that is being used. The choice of k also varies between applications. For planar mesh parameterization, k is typically 2, while in other applications, a single eigenvector possessing desirable properties may be selected for a task.

In computer vision and machine learning, spectral methods usually employ a different operator, the so-called *affinity matrix* [SM00, Wei99]. Each entry W_{ij} of an affinity matrix W represents a numerical relation, the affinity, between two data points i and j, e.g., pixels in an image, vertices in a mesh, or two face models in the context of face recognition. Note that the affinity matrix differs from the Laplacian in that affinities between all data pairs are defined. Therefore this matrix is not sparse in general. In practice, this non-sparse structure implies more memory requirements and more expensive computations. The use of affinity matrices for spectral mesh processing has received wide appeal in computer graphics as well, e.g., for mesh segmentation.

5.1.1 Parameterization and remeshing

In the context of mesh parameterization, spectral methods have the interesting property of connecting local entities in a way that lets a global behavior emerge. This property can be used to compute a good base complex [DBG+06a], [HZM+08], or to directly define a parameterization [ZSGS04], [MTAD08], as explained below. Figure 19 shows that Mullen et.al's spectral parameterization achieves a result comparable to ABF++. The interesting point is that it uses a simple and elegant formulation of the problem (just compute the Fiedler vector

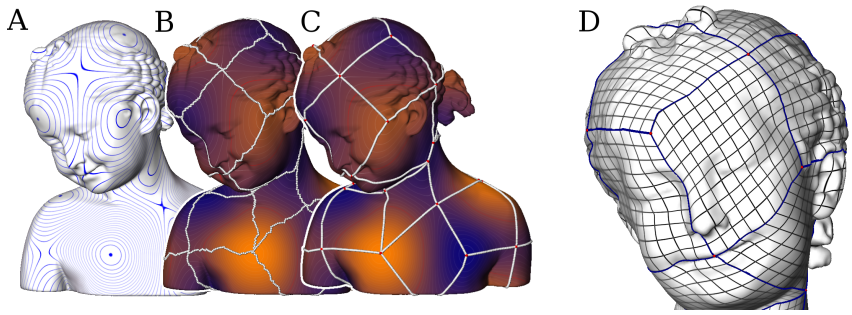


Figure 20: Spectral Surface Quadrangulation first computes a Laplace eigenfunction (A), then extracts its Morse complex (B), smoothes it (C) and uses it to partition the mesh into quads, that can be parameterized (D).

of the LSCM matrix).

Besides corresponding to a "Fourier basis on meshes", eigenfunctions can be used to define parameterizations. For instance, in the context of data analysis, LLE (Local Linear Embedding) [RS00b] computes an embedding of a graph that preserves metric relations in local neighborhoods. Similarly, the MDS method (*multidimensional scaling*) [You85] computes an embedding that best approximates all the geodesic distances between the vertices of a graph. Multidimensional scaling simply minimizes an objective function that measures the deviation between the geodesic distances in the initial space and the Euclidean distances in the embedding space (GDD for Geodesic Distance Deviation), by computing the eigenvectors of the matrix $D = (d_{i,j})$ where $d_{i,j}$ denotes the geodesic distance between vertex i and vertex j. Isomaps and Multidimensional scaling were used to define parameterization algorithms in [ZKK02], and more recently in the *ISO-charts* method [ZSGS04]. Spectral analysis also has the interesting property of defining an orthogonal basis. This property can be used to avoid the degenerate configurations encountered by linear conformal parameterization methods [LPRM02], [DMA02]. Those linear methods use "vertex pinning" to avoid the trivial constant solution, whereas the spectral method introduced in [MTAD08] computes the first solution orthogonal to the trivial one, that is to say, the eigenvector associated with the first non-zero eigenvalue.

However, the spectral parameterization methods listed above still need to partition the mesh into charts. More recently, Dong et al. used the Laplacian to decompose a mesh into quadrilaterals [DBG+05, DBG+06a], in a way that facilitates constructing a globally smooth parameterization. As shown in Figure 20, their method first computes one eigenfunction of the Laplacian (the 38^{th} in this example), then extract the Morse complex of this function, filters and smoothes the Morse complex, and uses it to partition the mesh into quads. These quads are parameterized, and inter-chart smoothness can be further optimized using global relaxation [KLS03, THCM04]. More recently, a generalization was proposed [HZM+08], still based on Laplacian eigenfunctions, and steered by a

guidance vector field.

5.1.2 Clustering and segmentation

One of the most well-known techniques in this regard is *spectral clustering* [BN03, KVV00, NJW02]. Interested readers should refer to the recent survey by von Luxburg [vL06] and the comparative study by Verma and Meilă [VM03]. Ng et al. [NJW02] presented a clustering method where the entries of the first k eigenvectors corresponding to the largest eigenvalues of an affinity matrix are used to obtain the transformed coordinates of the input data points. Additionally, the embedded points are projected onto the unit k-sphere. Points that possess high affinities tend to be grouped together in the spectral domain, where a simple clustering algorithm, such as k-means, can reveal the final clusters. Other approaches differ only slightly from the core solution paradigm, e.g., in terms of the operator used and the dimensionality of the embedding.

The ubiquity of the clustering problem makes spectral clustering an extremely useful technique. Kolluri et al. [KSO04] used spectral clustering to determine the inside and outside tetrahedra in the context of a Delaunay-based surface reconstruction algorithm. Liu and Zhang [LZ04] performed mesh segmentation via spectral clustering. Basically, an affinity matrix is constructed where the affinity measure depends on both geodesic distances and curvature information. Next, the eigenvectors given by the eigendecomposition of this matrix guide a clustering method, which provides patches of faces that define the different segments of the mesh returned by the segmentation algorithm. It is shown by example that it can be advantageous to perform segmentation in the spectral domain, e.g., in terms of higher-quality cut boundaries.

In a follow-up work, Zhang and Liu [ZL05] presented a mesh segmentation approach based on a recursive 2-way spectral clustering method. Only the first two largest eigenvectors of the affinity matrix are computed. This provides a one-dimensional embedding of mesh faces given by the quotient of the entries of the two eigenvectors. The most salient cut in this embedding is located, given by a part salience measure [HS97]. The cut provides a segmentation of the faces into two parts. This process is recursively repeated in order to obtain a hierarchical binary partitioning of the input mesh.

In yet another follow-up work, Liu and Zhang [LZ07] proposed an algorithm for mesh segmentation via recursive bisection where at each step, a sub-mesh embedded in 3D is spectrally projected to 2D and then a contour is extracted from the planar embedding. Two operators are used in combination to compute the projection: the well-known graph Laplacian and a geometric operator designed to emphasize concavity. The two embeddings reveal distinctive shape semantics of the 3D model and complement each other in capturing the structural or geometrical aspects of a segmentation. Transforming the shape analysis problem to the 2D domain also facilitates segmentability analysis and sampling, where the latter is needed to identify two samples residing on different parts of the sub-mesh. These two samples are used by the Nyström method in the construction of a 1D face sequence for finding an optimal cut, as in [ZL05]. Similar

to the technique presented in Section 4, the Nyström method is designed to efficiently compute eigenvectors of large matrices. However, it relies on sampling and extrapolation and only approximates the eigenvectors.

Recently, de Goes et al. [dGGV08] presented a hierarchical segmentation method for articulated bodies. Their approach relies on the diffusion distance, which is a multi-scale metric based on the heat kernel and computed from the eigenvectors of the Laplace-Beltrami operator. The diffusion distance is used to compute a bijection between medial structures and segments of the model. The medial structures yield a means to further refine the segmentation in an iterative manner and provide a full hierarchy of segments for the shape.

Huang et al. [HWAG09] also performed hierarchical shape decomposition via spectral analysis. However, the operator they use encapsulates shape geometry beyond the static setting. The idea is to define a certain deformation energy and use the eigenvectors of the Hessian of the deformation energy to characterize the space of possible deformations of a given shape. The *eigenmodes* corresponding to the low-end of the spectrum of the Hessian capture low-energy or in their formulation, more rigid deformations, called "typical" deformations. The optimal shape decomposition they compute is one whose optimal articulated (piecewise rigid) deformation defined on the parts of the decomposition best conforms to the basis vectors of the space of typical deformations. As a result, their method tends to identify parts of a shape which would likely remain rigid during the "typical" deformations.

5.1.3 Shape correspondence and retrieval

Depending on the requirement of the problem at hand, the mesh operator we use to compute a spectral embedding can be made to incorporate any intrinsic measure on a shape in order to obtain useful invariance properties, e.g., with respect to part articulation or bending. In Figure 21, we show 3D spectral embeddings of a few human and hand models obtained from an operator derived from geodesic distances over the mesh surfaces. As geodesic distance is bending-tolerant, the resulting embeddings are normalized with respect to bending and can facilitate shape retrieval under part articulation [EK03, JZ07].

Elad and Kimmel [EK03] used MDS to compute bending-invariant signatures for meshes. Geodesic distances between mesh vertices are computed by fast marching. The resulting spectral embedding effectively normalizes the mesh shapes with respect to translation, rotation, as well as bending transformations. The similarity between two shapes is then given by the Euclidean distance between the moments of the first few eigenvectors, usually less than 15, and these similarity distances can be used for shape classification.

Inspired by works in computer vision on spectral point correspondence [SB92], Jain and Zhang [JZvK07] relied on higher-dimensional embeddings based on the eigenvectors of an affinity matrix to obtain point correspondence between two mesh shapes. The first k eigenvectors of the affinity matrix encoding the geodesic distances between pairs of vertices are used to embed the model in a k-dimensional space; typically $k = 5$ or 6. After this process is performed for

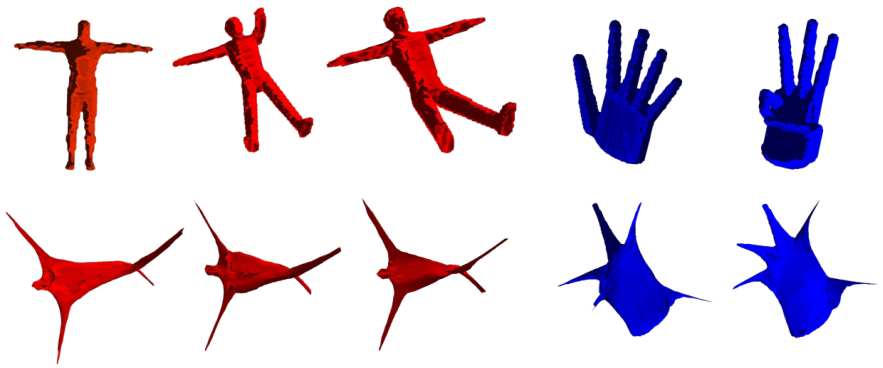


Figure 21: Spectral embeddings (bottom row) of some articulated 3D shapes (top row). Since the mesh operator is constructed from geodesic distances, the embeddings are normalized with respect to shape bending.

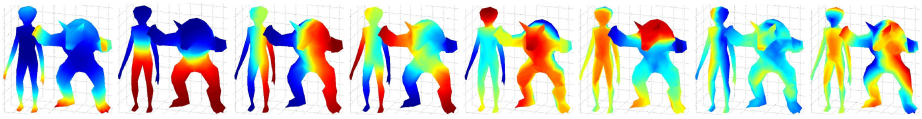


Figure 22: Eigenvector plots for two similar shapes, both with 252 vertices. The entries in an eigenvector are color-mapped. As we can see, there is an "eigenvector switching" occurring between the fifth and sixth eigenvectors. Such "switching" is difficult to detect from the magnitude of the eigenvalues. The first 8 eigenvalues for the two shapes are [205.6, 11.4, 4.7, 3.8, 1.8, 0.4, 0.26, 0.1] and [201.9, 10.9, 6.3, 3.4, 1.8, 1.2, 0.31, 0.25], respectively.

two models, the two embeddings are non-rigidly aligned via thin-plate splines and the correspondence between the two shapes is given by the proximity of the vertices after such alignment. Any measure for the cost of a correspondence, e.g., sum of distances between corresponding vertices, can be used as a similarity distance for shape retrieval.

One of the key observations made in [JZvK07] is the presence of "eigenvector switching" due to non-uniform scaling of the shapes. Specifically, the eigenmodes of similar shapes do not line up with respect to the magnitude of their corresponding eigenvalues, as illustrated in Figure 22. As a result, it is unreliable to sort the eigenvectors according to the magnitude of their respective eigenvalues, as has been done in all works on spectral correspondence so far. Jain and Zhang [JZvK07] relied on a heuristic to "unswitch" the eigenmodes and thin-plate splines to further align the shapes in the spectral domain [JZvK07]. Recent work of Mateus et al. [MCBH07] addressed the issue using an alignment by the Expectation-Maximization (EM) algorithm instead.

The method by Leordeanu and Hebert [LH05] focuses on the global characteristics of correspondence computation and aims at finding *consistent cor-*

respondences between two sets of shape or image features, where the spectral approach has also found its utility. They build a graph whose nodes represent possible feature pairings and edge weights measure how agreeable the pairings are. The principal eigenvector of an affinity matrix W, one corresponding to the largest eigenvalue, is inspected to detect how strongly the graph nodes belong to the main cluster of W. The idea is that correct correspondences are likely to establish links among each other and thus form a strongly connected cluster.

To define informative shape descriptors, another possibility consists in using the heat diffusion equation and the heat kernel. Heat diffusion is governed by the following differential equation :

$$\frac{\partial f}{\partial t} = k\Delta f$$

where k is a constant. This equation admits a solution that can be written as :

$$f(t,x) = \int K(t,y,x)dy$$

where $K(t,x,y)$ denotes the *heat kernel*. Intuitively, considering that a Dirac of heat was emitted from point x at time $t = 0$, the heat kernel $K(t,x,y)$ indicates the quantity of heat that can be measured at point y after time t. The heat kernel admits an expression as an (infinite) sum of eigenfunctions :

$$K(t,x,y) = \sum_{i=0}^{\infty} e^{-\lambda_i t}\phi_i(x)\phi_i(y)$$

Using this expression, the idea of using the *auto-diffusion function* was simultaneously proposed in [SOG] and [GBAL]. The auto-diffusion function corresponds to the amount of heat that remains at point x after time t for a given t, in other words $K(t,x,x)$. This defines a scalar field on the surface that provably has good properties to be used as a shape descriptor [SOG]. In particular, protrusions correspond to extrema of the function. The value of t acts as a smoothing parameter (smaller values of t preserve most details, and higher values of t tend to a constant function). Another application of the auto-diffusion function is to compute a Morse-Smale complex for shape segmentation [GBAL].

5.1.4 Global intrinsic symmetry detection

Ovsjanikov et al. [OSG08] propose an approach to detect the intrinsic symmetries of a shape which are invariant to isometry preserving transformations. They show that if the shape is embedded into a signature space defined by the eigenfunctions of the Laplace-Beltrami operator, then the intrinsic symmetries are transformed into extrinsic Euclidean symmetries (rotations or reflections). However, it is possible to restrict the search of symmetries only to reflections, avoiding the search for rotational symmetries, a task that can be hard in high-dimensional space. This result allows to obtain the intrinsic symmetries by first computing the eigenvectors of the operator, then embedding the shape into

the signature space, and finally finding point-to-point correspondences between symmetric points. The signature adopted is derived from the GPS embedding of Rustamov [Rus07].

5.2 Use of spectral transforms

The spectral mesh transforms are closely related to the Fourier transform that is the foundation of signal processing theory. Conceivably, any application of the classical Fourier transform in signal or image processing has the potential to be realized in the mesh setting. In geometry processing, the mesh signal considered is often the embedding function that specifies the 3D coordinates of each vertex. This signal serves to represent mesh geometry.

5.2.1 Geometry compression

Karni and Gotsman [KG00c] proposed an approach to compress the geometry of triangle meshes. Firstly, the set of eigenvectors of the Tutte Laplacian is computed. Next, the mesh vertex coordinates are projected into the spectral space spanned by the computed eigenvectors. Part of the coefficients obtained by this transformation is eliminated in order to reduce the storage space required for mesh geometry. The coefficients related to the eigenvectors associated to larger eigenvalues are firstly removed, which would correspond to high frequency detail, when following an analogy with Fourier analysis.

The main drawback of this method is that many eigenvectors need to be computed. Karni and Gotsman propose to partition the mesh into smaller sets of vertices. Although that alleviates the problem of computing the eigenvectors for large matrices, it still requires a good partitioning of the mesh for the efficiency of the compression, and artifacts along the partition boundaries are evident when higher compression rates are employed.

5.2.2 Watermarking

Ohbuchi et al. [OTMM01, OMT02] also employed the spectral transform approach, but to insert watermarks into triangle meshes. In their method, the eigenvectors of the graph Laplacian are used as the basis for the projection. After transforming the geometry of the mesh and obtaining the spectral coefficients, a watermark is inserted into the model by modifying coefficients at the low-frequency end of the spectrum. In this way, modifications on the geometry of the mesh are well-spread over the model and less noticeable than if they were directly added to the vertex coordinates. This method also requires the computation of eigenvectors of the Laplace operator, which is prohibitive in the case of large meshes. Mesh partitioning is again used to address this problem.

5.2.3 Fourier descriptors

2D Fourier descriptors have been quite successful as a means to characterize 2D shapes. Using eigendecomposition with respect to the mesh Laplacians,

one can compute analogous Fourier-like descriptors to describe mesh geometry. However, there have not seen such mesh Fourier descriptors being proposed for shape analysis so far. There have been methods, e.g., [VS01], which rely on 3D Fourier descriptors for 3D shape retrieval. In this case, the mesh shapes are first voxelized and 3D Fourier descriptors are extracted from the resulting volumetric data. We suspect that the main difficulties with the use of mesh Fourier descriptors for shape matching include computational costs and the fact that when the eigenmodes vary between the two mesh shapes to be matched, it becomes doubtful whether their associated eigenspace projections can serve as reliable shape descriptors. Also, even when the shapes are very similar, eigenvector switching, as depicted in Figure 22, can occur when the eigenvectors are ordered by the magnitude of their eigenvalues.

References

[AFW06] D. N. Arnold, R. S. Falk, and R. Winther. Finite element exterior calculus, homological techniques, and applications. *Acta Numerica 15*, 2006.

[Arv95] James Arvo. The Role of Functional Analysis in Global Illumination. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 115–126, New York, NY, 1995. Springer-Verlag.

[BN03] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Neural Computations*, 15(6):1373–1396, 2003.

[Bra99] Ronald N. Bracewell. *The Fourier Transform And Its Applications*. McGraw-Hill, 1999.

[Cip93] Barri Cipra. You can't always hear the shape of a drum. *What's Happening in the Mathematical Sciences*, 1, 1993.

[DBG+05] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Quadrangulating a mesh using laplacian eigenvectors. Technical report, June 2005.

[DBG+06a] S. Dong, P.-T. Bremer, M. Garland, V. Pascucci, and J. C. Hart. Spectral mesh quadrangulation. *ACM Transactions on Graphics (SIGGRAPH 2006 special issue)*, 2006.

[DBG+06b] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John C. Hart. Spectral surface quadrangulation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 1057–1066, New York, NY, USA, 2006. ACM Press.

[dGGV08] Fernando de Goes, Siome Goldenstein, and Luiz Velho. A hierarchical segmentation of articulated bodies. *Computer Graphics Forum (Symposium on Geometry Processing)*, 27(5):1349–1356, 2008.

[DKT05] Mathieu Desbrun, Eva Kanzo, and Yiying Tong. Discrete differential forms for computational modeling. *Siggraph '05 course notes on Discrete Differential Geometry, Chapter 7*, 2005.

[DMA02] Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. In *Proceedings of Eurographics*, pages 209–218, 2002.

[dV90] Y. Colin de Verdiere. Sur un nouvel invariant des graphes et un critere de planarite. *J. of Combinatorial Theory*, 50, 1990.

[Dye06] Ramsey Dyer. Mass weights and the cot operator (personal communication). Technical report, Simon Fraser University, CA, 2006.

[EK03] A. Elad and R. Kimmel. On bending invariant signatures for surfaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1285–1295, 2003.

[Fei96] J. Feidman. Computing betti numbers via combinatorial laplacians. In *Proc. 28th Sympos. Theory Comput.*, pages 386–391. ACM, 1996.

[FH04] M. S. Floater and K. Hormann. *Surface parameterization: a tutorial and survey*. Springer, 2004.

[Fie73] Miroslav Fiedler. Algebraic connectivity of graphs. *Czech. Math. Journal*, 23:298–305, 1973.

[Fie75] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czech. Math. Journal*, 25:619–633, 1975.

[GBAL] Katarzyna Gebal, Andreas Baerentzen, Henrik Aanaes, and Rasmus Larsen. Shape analysis using the auto diffusion function. *Computer Graphics Forum (Proc. of Symp. on Geom. Proc.)*.

[GGS03a] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes. *ACM Trans. Graph.*, 22(3):358–363, 2003.

[GGS03b] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3d meshes, 2003.

[Got03] Craig Gotsman. On graph partitioning, spectral analysis, and digital mesh processing. In *Shape Modeling International*, pages 165–174, 2003.

[GY02] X. Gu and S.-T. Yau. Computing conformal structures of surfaces. *Communications in Information and Systems*, 2(2):121–146, 2002.

[Hir03] Anil Hirani. Discrete exterior calculus. *PhD thesis*, 2003.

[HPW06] Klaus Hildebrandt, Konrad Polthier, and Max Wardetzky. On the convergence of metric and geometric properties of polyhedral surfaces. *Geom Dedicata*, 2006.

[HS97] D. D. Hoffman and M. Singh. Salience of visual parts. *Cognition*, 63:29–78, 1997.

[HWAG09] Qixing Huang, Martin Wicke, Bart Adams, and Leonidas J. Guibas. Shape decomposition using modal analysis. 28(2):to appear, 2009.

[HZM+08] Jin Huang, Muyang Zhang, Jin Ma, Xinguo Liu, Leif Kobbelt, and Hujun Bao. Spectral quadrangulation with orientation and alignment control. *ACM Transactions on Graphics (SIGGRAPH Asia conf. proc.*, 2008.

[IL05] Martin Isenburg and Peter Lindstrom. Streaming meshes. In *IEEE Visualization*, page 30, 2005.

[Jai89] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

[JNT] Dmitry Jakobson, Nikolai Nadirashvili, and John Toth. Geometric properties of eigenfunctions.

[JZ07] Varun Jain and Hao Zhang. A spectral approach to shape-based retrieval of articulated 3D models. *Computer Aided Design*, 39:398–407, 2007.

[JZvK07] Varun Jain, Hao Zhang, and Oliver van Kaick. Non-rigid spectral correspondence of triangle meshes. *International Journal on Shape Modeling*, 2007. to appear.

[Kac66] Mark Kac. Can you hear the shape of a drum? *Amer. Math. Monthly*, 73, 1966.

[KG00a] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *Proc. of ACM SIGGRAPH*, pages 279–286, 2000.

[KG00b] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *SIGGRAPH*, pages 279–286, 2000.

[KG00c] Zachi Karni and Craig Gotsman. Spectral compression of mesh geometry. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 279–286, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[KLS03] A. Khodakovsky, N. Litke, and P. Schröder. Globally smooth parameterizations with low distortion. *ACM TOG (SIGGRAPH)*, 2003.

[Kor02] Y. Koren. On spectral graph drawing, 2002.

[Kor03] Y. Koren. On spectral graph drawing. In *Proc. of the International Computing and Combinatorics Conference*, pages 496–508, 2003.

[KSO04] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Proc. of Eurographics Symposium on Geometry Processing*, pages 11–21, 2004.

[KVV00] R. Kannan, S. Vempala, and A. Vetta. On clustering - good, bad, and spectral. In *FOCS*, pages 367–377, 2000.

[Lev06] Bruno Levy. Laplace-beltrami eigenfunctions: Towards an algorithm that understands geometry. In *IEEE International Conference on Shape Modeling and Applications*, 2006.

[LH05] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference of Computer Vision (ICCV)*, volume 2, pages 1482–1489, October 2005.

[LPM02] Bruno Levy, Sylvain Petitjean, and Nicolas Ray Nicolas Jerome Maillot. Least squares conformal maps for automatic texture atlas generation. In ACM, editor, *SIGGRAPH conf. proc*, 2002.

[LPRM02] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. In *Proc. of ACM SIGGRAPH 02*, pages 362–371, 2002.

[LZ04] R. Liu and H. Zhang. Segmentation of 3D meshes through spectral clustering. In *Pacific Graphics*, pages 298–305, 2004.

[LZ07] Rong Liu and Hao Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum (Special Issue of Eurographics 2007)*, 26:385–394, 2007.

[MCBH07] Diana Mateus, Fabio Cuzzolin, Edmond Boyer, and Radu Horaud. Articulated shape matching by robust alignment of embedded representations. In *ICCV '07 Workshop on 3D Representation for Recognition (3dRR-07)*, 2007.

[MDSB03] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.

[MIT06] Omer Meshar, Dror Irony, and Sivan Toledo. An out-of-core sparse symmetric indefinite factorization method. *ACM Transactions on Mathematical Software*, 32:445–471, 2006.

[MTAD08] Patrick Mullen, Yiying Tong, Pierre Alliez, and Mathieu Desbrun. Spectral conformal parameterization. In *ACM/EG Symposium of Geometry Processing*, 2008.

[NJW02] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Neural Information Processing Systems*, volume 14, pages 849–856, 2002.

[OMT02] R. Ohbuchi, A. Mukaiyama, and S. Takahashi. A frequency-domain approach to watermarking 3D shapes. *Computer Graphics Forum*, 21(3):373–382, 2002.

[OSG08] Maks Ovsjanikov, Jian Sun, and Leonidas Guibas. Global intrinsic symmetries of shapes. *Computer Graphics Forum (Symposium on Geometry Processing)*, 27(5):1341–1348, 2008.

[OTMM01] R. Ohbuchi, S. Takahashi, T. Miyazawa, and A. Mukaiyama. Watermarking 3D polygonal meshes in the mesh spectral domain. In *Proc. of Graphics Interface*, pages 9–18, 2001.

[PP93] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1), 1993.

[Pra99] G. Prathap. Towards a science of fea: Patterns, predictability and proof through some case studies. *Current Science*, 77:1311–1318, 1999.

[RS00a] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[RS00b] Sam Roweis and Lawrence Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, Dec 2000.

[Rus07] R. M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *Proc. of Eurographics Symposium on Geometry Processing*, pages 225–233, 2007.

[RWP05a] M. Reuter, F.-E. Wolter, and N. Peinecke. Laplace-beltrami spectra as "shape-dna" of surfaces and solids. *CAD Journal*, 2005.

[RWP05b] Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. Laplace-spectra as fingerprints for shape matching. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 101–106, New York, NY, USA, 2005. ACM Press.

[SB92] L. S. Shapiro and J. M. Brady. Feature-based correspondence: an eigenvector approach. *Image and Vision Computing*, 10(5):283–288, 1992.

[SGD05] P. Schröder, E. Grinspun, and M. Desbrun. Discrete differential geometry: an applied introduction. In *SIGGRAPH Course Notes*, 2005.

[SM00] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[SOG] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Computer Graphics Forum (Proc. of Symp. on Geom. Proc.)*.

[Tau95a] G. Taubin. A signal processing approach to fair surface design. In *Proc. of ACM SIGGRAPH*, pages 351–358, 1995.

[Tau95b] Gabriel Taubin. A signal processing approach to fair surface design. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 351–358, New York, NY, USA, 1995. ACM Press.

[TB97] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997.

[TdSL00] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[THCM04] M. Tarini, K. Hormann, P. Cignoni, and C. Montani. Polycube-maps. *ACM TOG (SIGGRAPH)*, 2004.

[vL06] Ulrike von Luxburg. A tutorial on spectral clustering. Technical Report TR-149, Max Plank Institute for Biological Cybernetics, August 2006.

[VM03] D. Verma and M. Meila. A comparison of spectral clustering algorithms. Technical Report UW-CSE-03-05-01, University of Washington, 2003.

[VS01] D. V. Vranić and D. Saupe. 3D shape descriptor based on 3D Fourier transform. In *Proc. EURASIP Conf. on Digital Signal Processing for Multimedia Communications and Services*, 2001.

[WBH+07] Max Wardetzky, Miklos Bergou, David Harmon, Denis Zorin, and Eitan Grinspun. Discrete quadratic curvature energies. *Computer Aided Geometric Design (CAGD)*, 2007.

[Wei99] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proc. of the International Conference on Computer Vision*, pages 975–983, 1999.

[WK05] Jianhua Wu and Leif Kobbelt. Efficient spectral watermarking of large meshes with orthogonal basis functions. In *The Visual Computer*, 2005.

[WMKG07] Max Wardetzky, Saurabh Mathur, Felix Kalberer, and Eitan Grinspun. Discrete laplace operators: No free lunch. *Eurographics Symposium on Geometry Processing*, 2007.

[You85] F. W. Young. Multidimensional scaling. *Encyclopedia of Statistical Sciences*, 5:649–658, 1985.

[ZKK02] Gil Zigelman, Ron Kimmel, and Nahum Kiryati. Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, 8(2), 2002.

[ZL05] H. Zhang and R. Liu. Mesh segmentation via recursive and visually salient spectral cuts. In *Proc. of Vision, Modeling, and Visualization*, 2005.

[ZSGS04] Kun Zhou, John Snyder, Baining Guo, and Heung-Yeung Shum. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. In *Symposium on Geometry Processing*, pages 47–56, 2004.

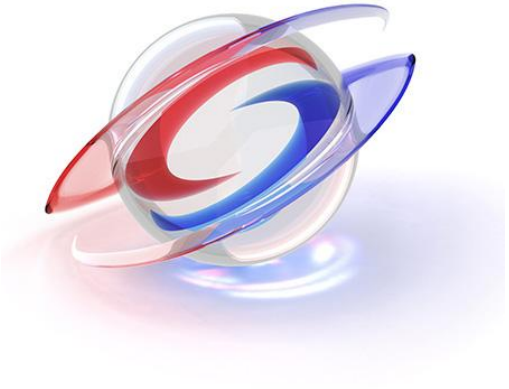
[ZvKDar] Hao Zhang, Oliver van Kaick, and Ramsay Dyer. Spectral mesh processing. *Computer Graphics Forum*, 2009, to appear. http://www.cs.sfu.ca/~haoz/pubs/zhang_cgf09_spect_survey.pdf.

SIGGRAPH 2010

“*Spectral Mesh Processing*”

Bruno Lévy and Richard Hao Zhang

Syllabus



- Introduction [Levy]

- What is so spectral? [Zhang]
  - Intuition and theory behind spectral methods
  - Different interpretations and motivating applications
- Do your own Spectral Mesh processing at home [Levy]
  - DEC Laplacian, numerics of spectral analysis
  - Tutorial on implementation with open source software

- ---------- break ----------

Syllabus

- Applications - What can we do with it ?" 1/2 [Zhang]
  - Segmentation, shape retrieval, non-rigid matching, symmetry detect
  - . . .
- Applications - What can we do with it ?" 2/2 [Levy]
  - Quadrangulation, parameterization, . . .
- Wrapup, conclusion, Q&A [Zhang and Levy]

SIGGRAPH 2010

Spectral Mesh Processing

“***What is So Spectral ?”***

Richard Hao Zhang

Why is “Spectral” Special?

Spectral Mesh Processing

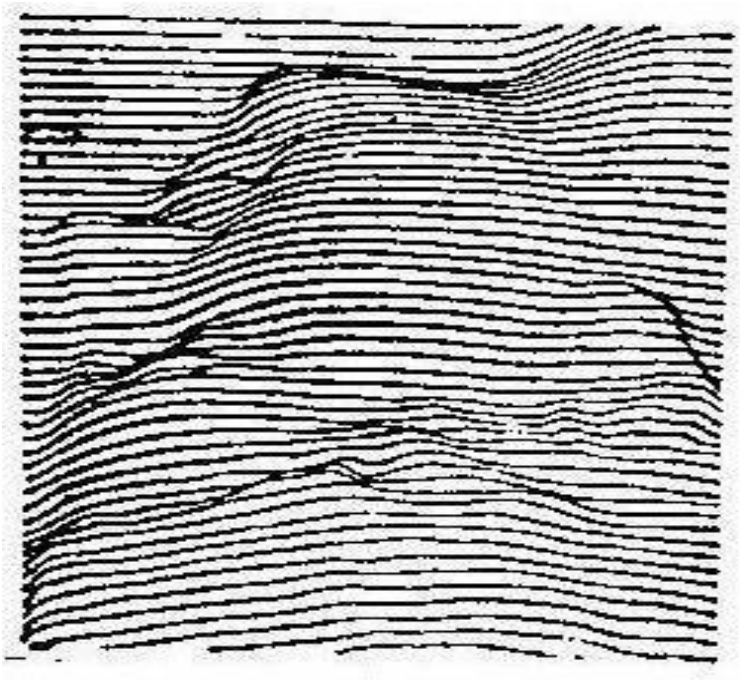
Hao (Richard) Zhang
School of Computing Science
Simon Fraser University, Canada

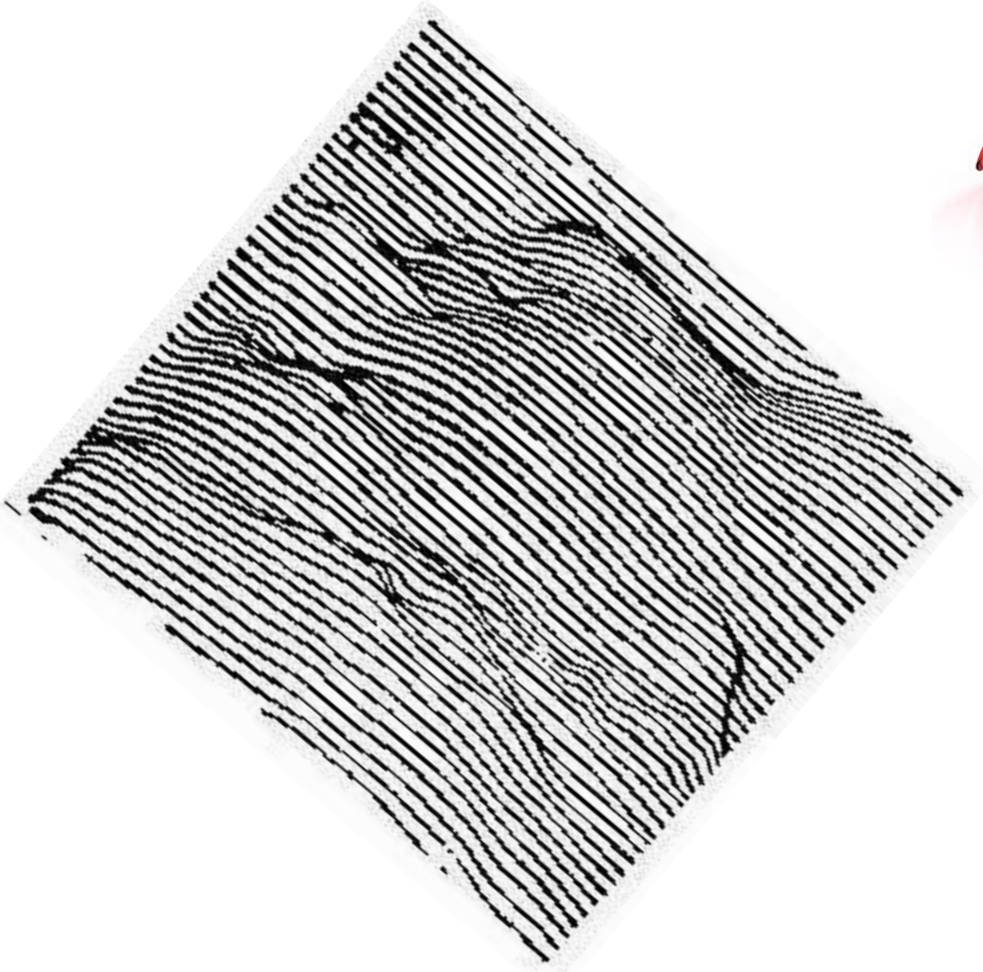
An Introduction to Spectral Mesh Processing

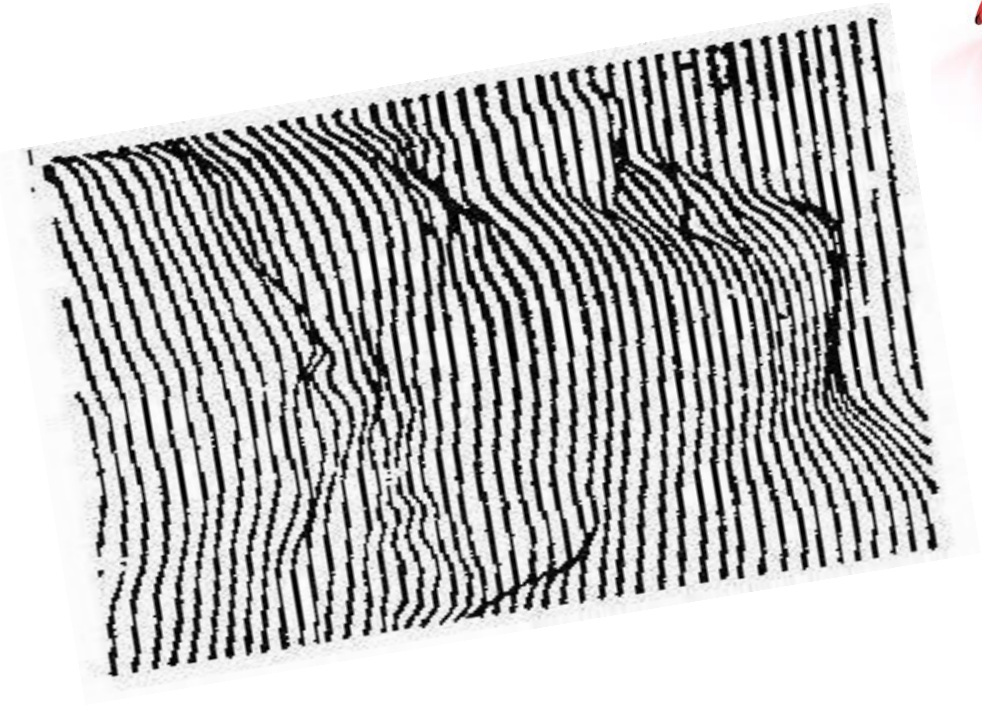
Spectral Mesh Processing

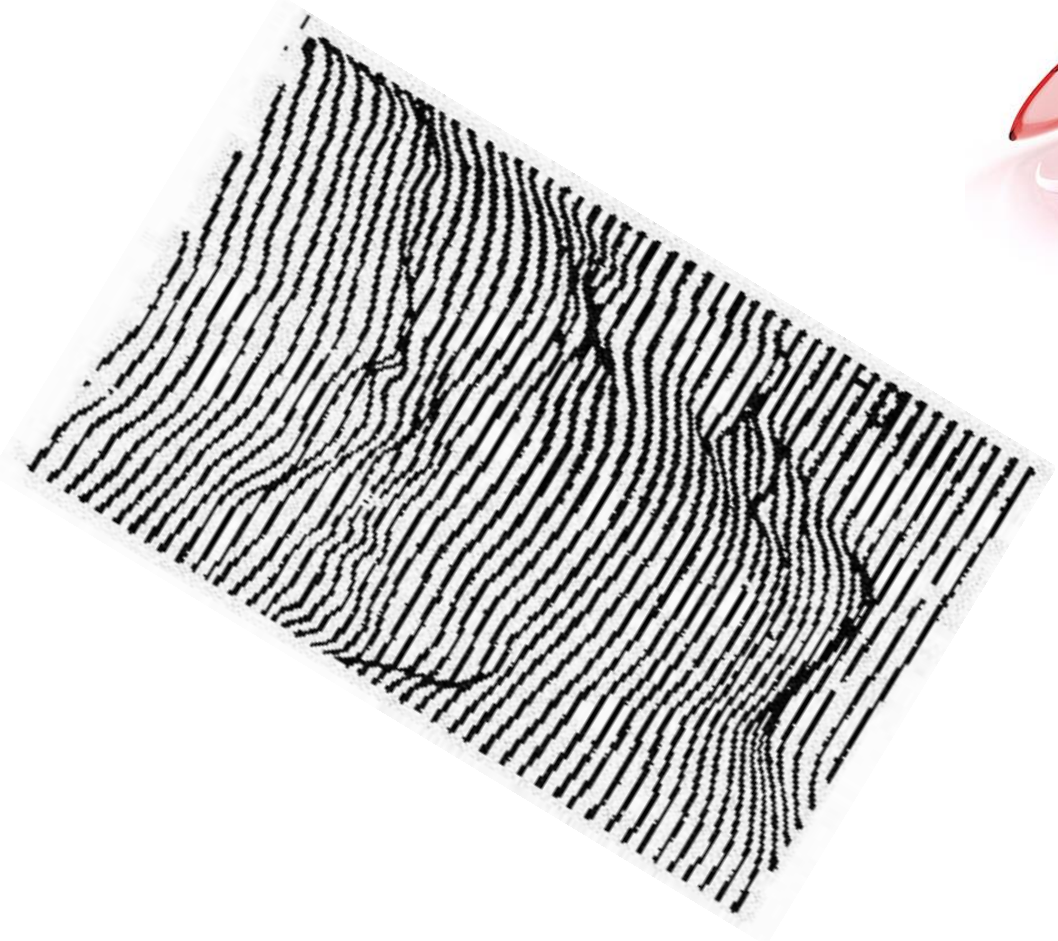
Hao (Richard) Zhang
School of Computing Science
Simon Fraser University, Canada

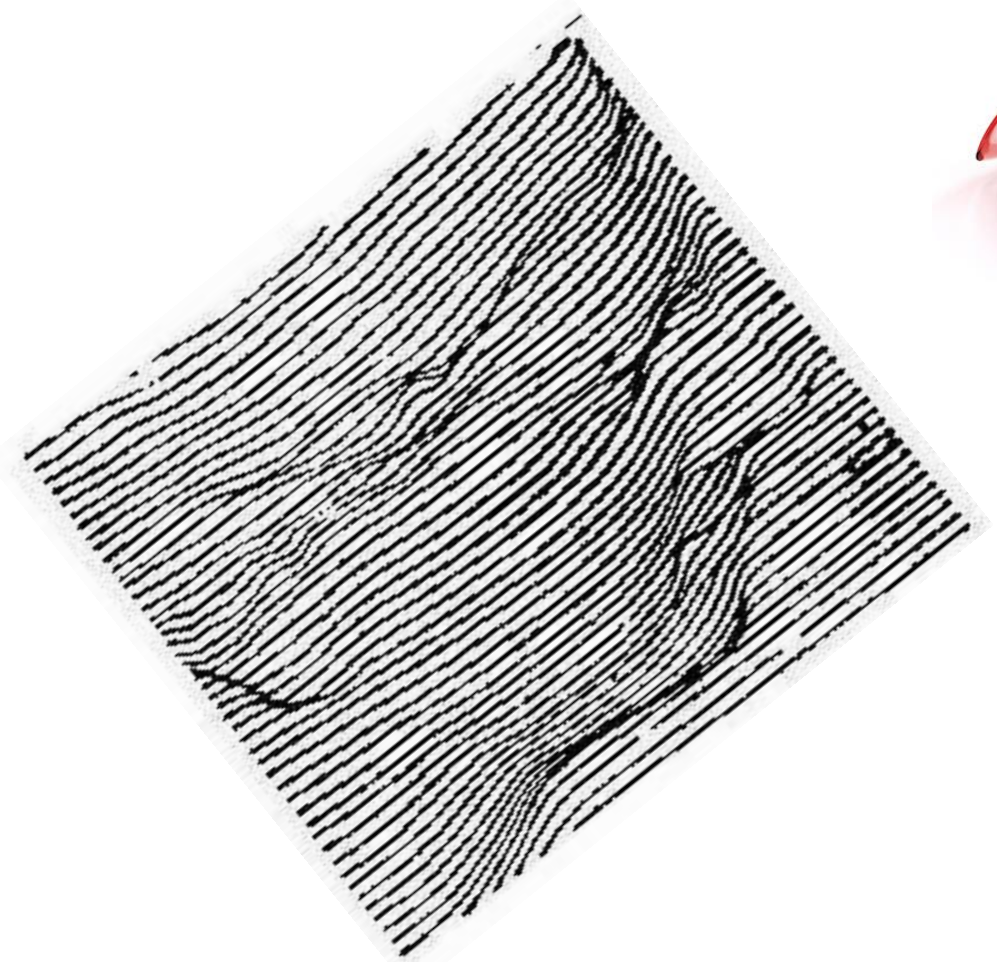
What do you see?

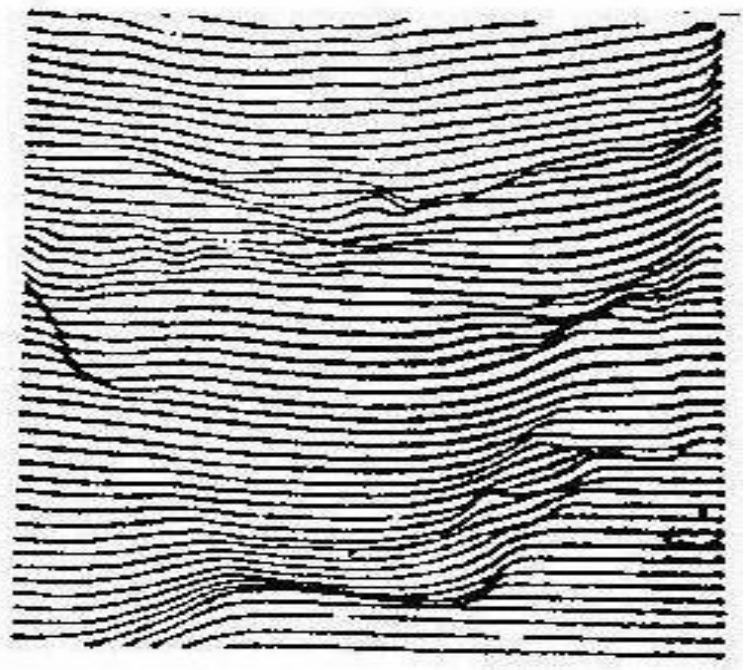




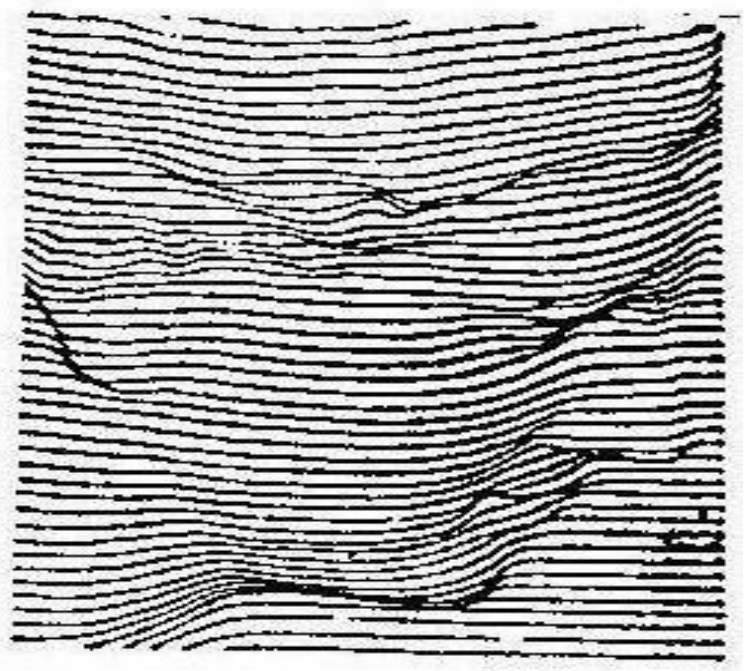






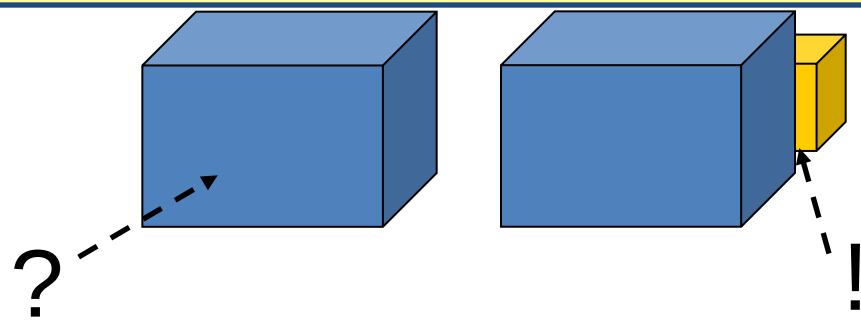


“Puzzle” solved from a new view



Different view, different domain

The same problem, phenomenon or data set, when viewed **from a different angle**, or **in a new domain**, may better reveal its underlying structure or solution

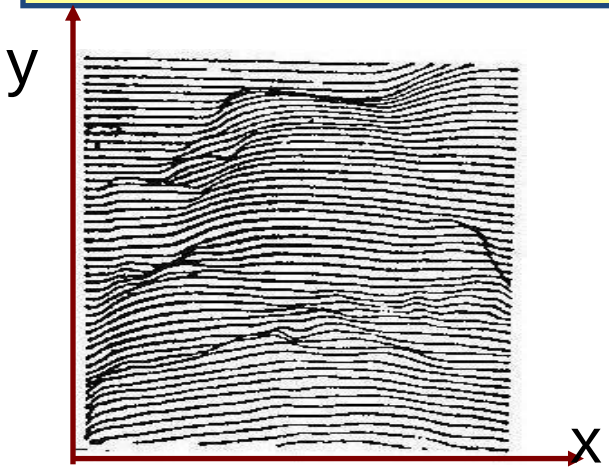


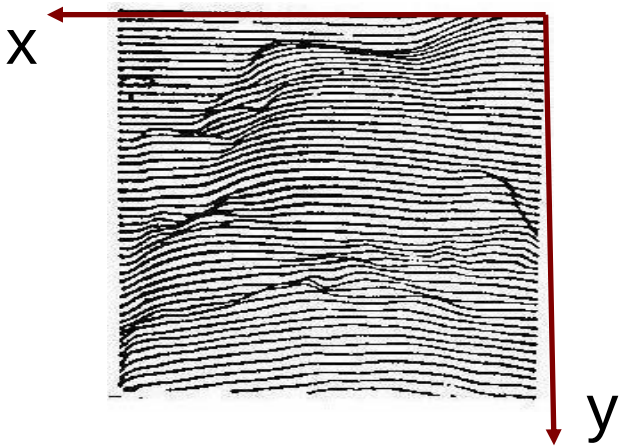
Spectral approach: a solution paradigm

Solving problems in a different domain using a **transform — spectral transform**

Spectral approach: a solution paradigm

Solving problems in a different domain using a **transform — spectral transform**



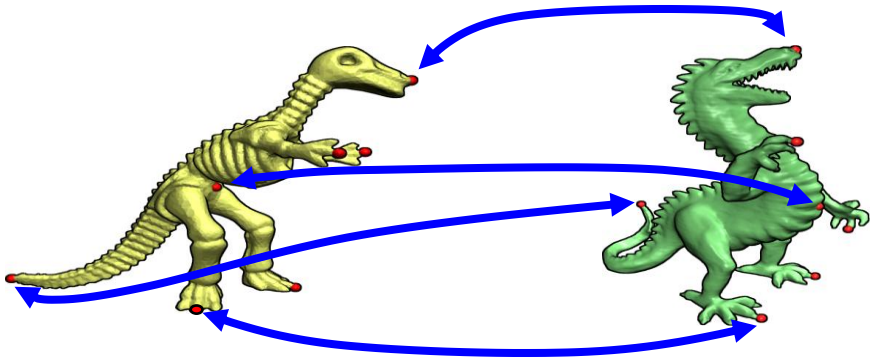


Talking about different views …

- The spectral approach itself under different views!
  - Signal processing perspective
  - Geometric perspective
  - Machine learning — dimensionality reduction

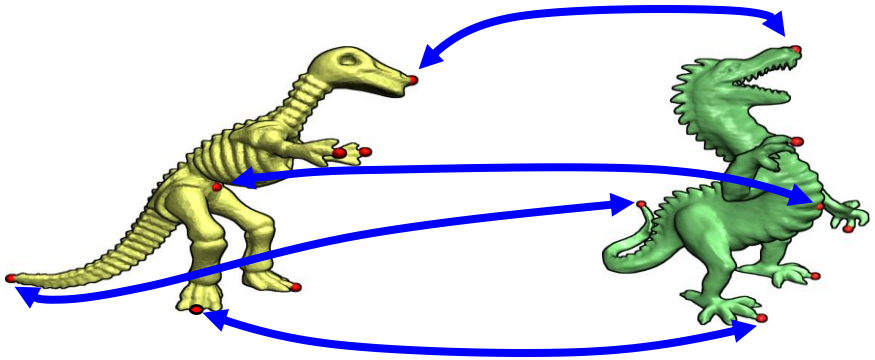
A first example

- Correspondence problem



A first example

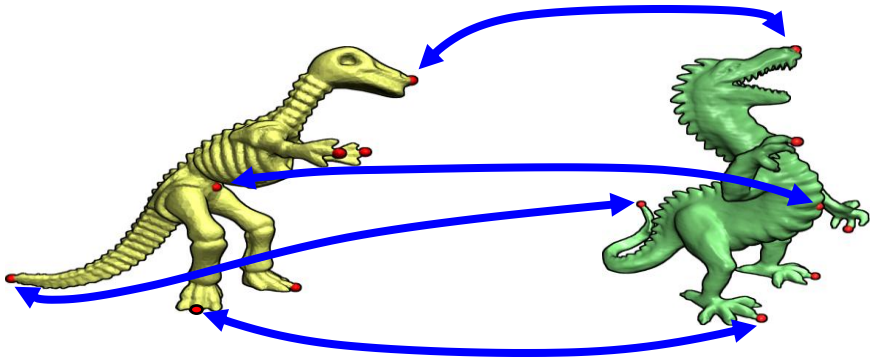
- Correspondence problem



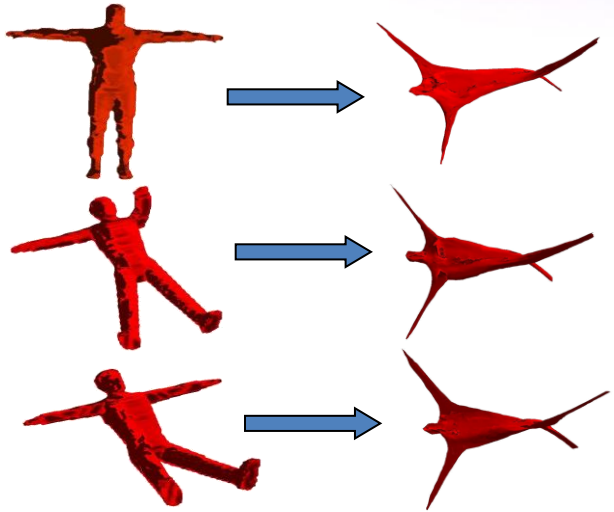
- Rigid alignment easy, but how about shape pose?

A first example

- Correspondence problem

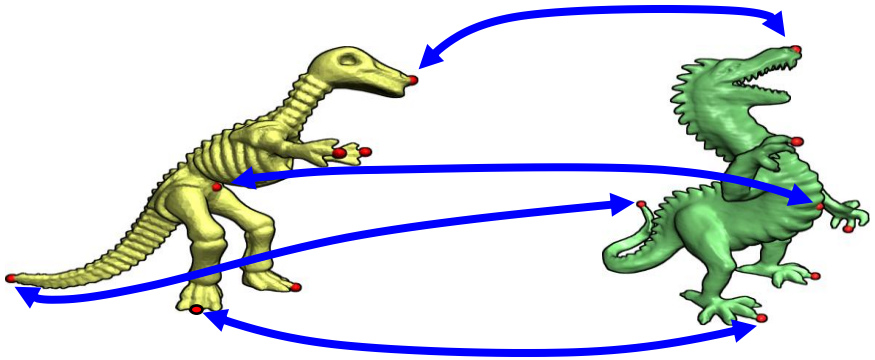


- **Spectral transform** to normalize shape pose

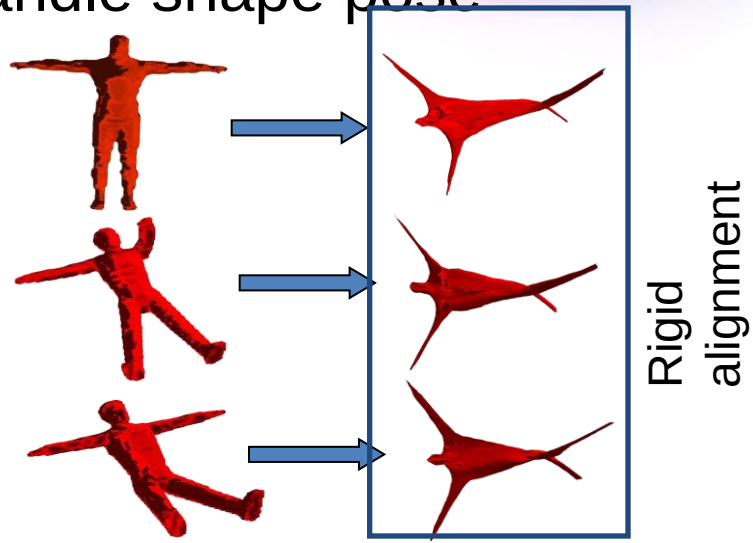


A first example

- Correspondence problem



- **Spectral transform** to handle shape pose



Spectral mesh processing

Use eigenstructures of appropriately defined linear mesh operators for geometry analysis and processing

Spectral = use of **eigenstructures**

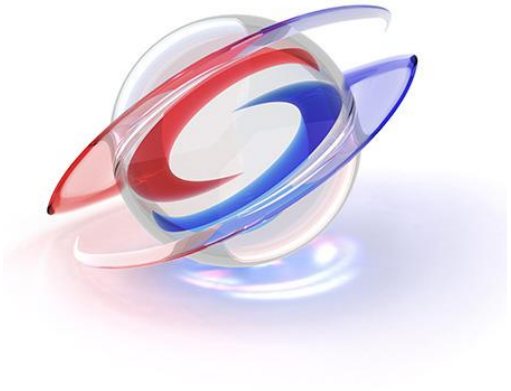
Eigenstructures

- Eigenvalues and eigenvectors: $A\mathbf{u} = \lambda\mathbf{u},\ \mathbf{u} \neq 0$
- Diagonalization or eigen-decomposition: $A = U\Lambda U^{\mathsf{T}}$
- Projection into eigensubspace: $\mathbf{y}' = U_{(k)}U_{(k)}^{\mathsf{T}}\mathbf{y}$
- A transform into eigenspace: $\hat{\mathbf{y}} = U^{\mathsf{T}}\mathbf{y}$

This lecture

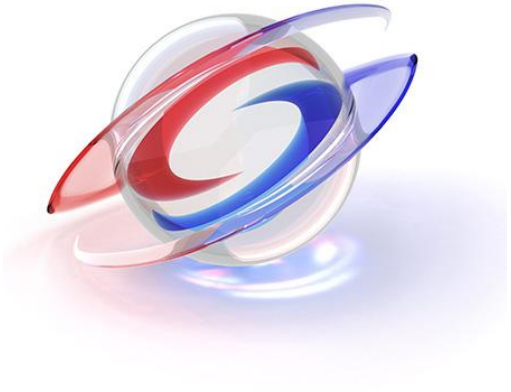
- Overview of spectral methods for mesh processing
- What is the spectral approach exactly?
  - Three perspectives or views
  - Motivations
  - Sample applications
- Difficulties and challenges

Outline



- Overview
- A signal processing perspective
  - A very gentle introduction — boredom alert ☺
  - Signal compression and filtering
  - Relation to discrete Fourier transform (DFT)
- Geometric perspective: global and intrinsic
- Dimensionality reduction
- Difficulties and challenges

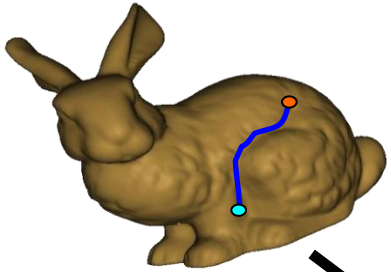
Outline

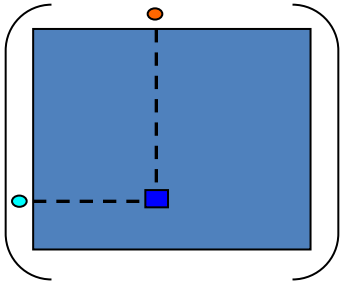


- Overview
- A signal processing perspective
  - A very gentle introduction — boredom alert ☺
  - Signal compression and filtering
  - Relation to discrete Fourier transform (DFT)
- Geometric perspective: global and intrinsic
- Dimensionality reduction
- Difficulties and challenges

Overall structure

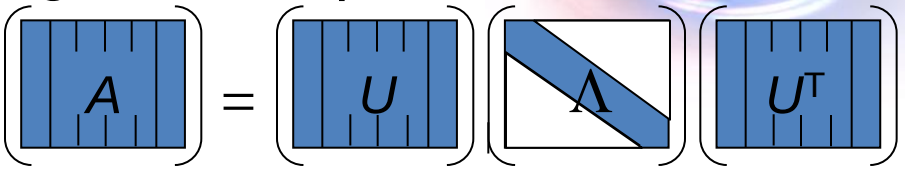
Input mesh *y*





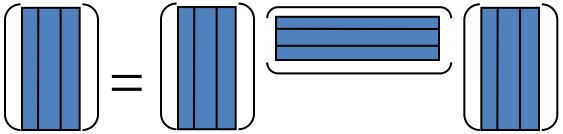
Some mesh operator *A*

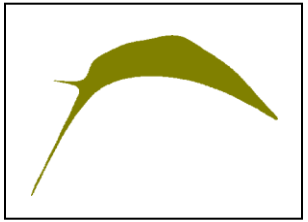
Eigendecomposition: $A = U\Lambda U^T$



e.g.

$\mathbf{y'} = U_{(3)}U_{(3)}^T\mathbf{y}$





Classification of applications

- Eigenstructure(s) used
  - Eigenvalues: **signature** for shape characterization
  - Eigenvectors: form **spectral embedding** (a transform)
  - Eigenprojection: also a transform — **DFT-like**
- Dimensionality of spectral embeddings
  - 1D: mesh sequencing
  - 2D or 3D: graph drawing or mesh parameterization
  - Higher D: clustering, segmentation, correspondence
- Mesh operator used
  - Combinatorial vs. geometric, 1st-order vs. higher order

Classification of applications

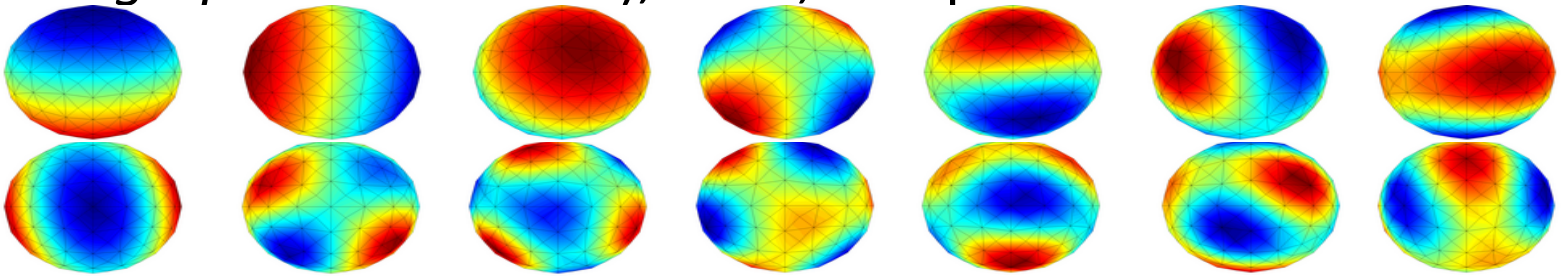
- Eigenstructure(s) used
  - Eigenvalues: **signature** for shape characterization
  - Eigenvectors: form **spectral embedding** (a transform)
  - Eigenprojection: also a transform — **DFT-like**
- Dimensionality of spectral embeddings
  - 1D: mesh sequencing
  - 2D or 3D: graph drawing or mesh parameterization
  - Higher D: clustering, segmentation, correspondence
- Mesh operator used
  - Combinatorial vs. geometric, 1st-order vs. higher order

Classification of applications

- Eigenstructure(s) used
  - Eigenvalues: **signature** for shape characterization
  - Eigenvectors: form **spectral embedding** (a transform)
  - Eigenprojection: also a transform — **DFT-like**
- Dimensionality of spectral embeddings
  - 1D: mesh sequencing
  - 2D or 3D: graph drawing or mesh parameterization
  - Higher D: clustering, segmentation, correspondence
- Mesh operator used
  - Combinatorial vs. geometric, 1st-order vs. higher order

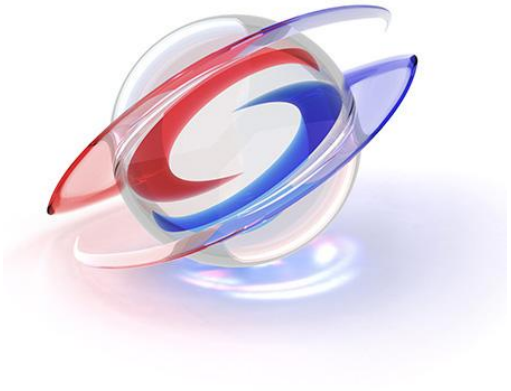
Much more details in survey

- H. Zhang, O. van Kaick, and R. Dyer, “Spectral Mesh Processing”, Computer Graphics Forum (shorter version as *Eurographics 2007 STAR*), 2009, accepted.



Eigenvector (of a combinatorial operator) plots on a sphere

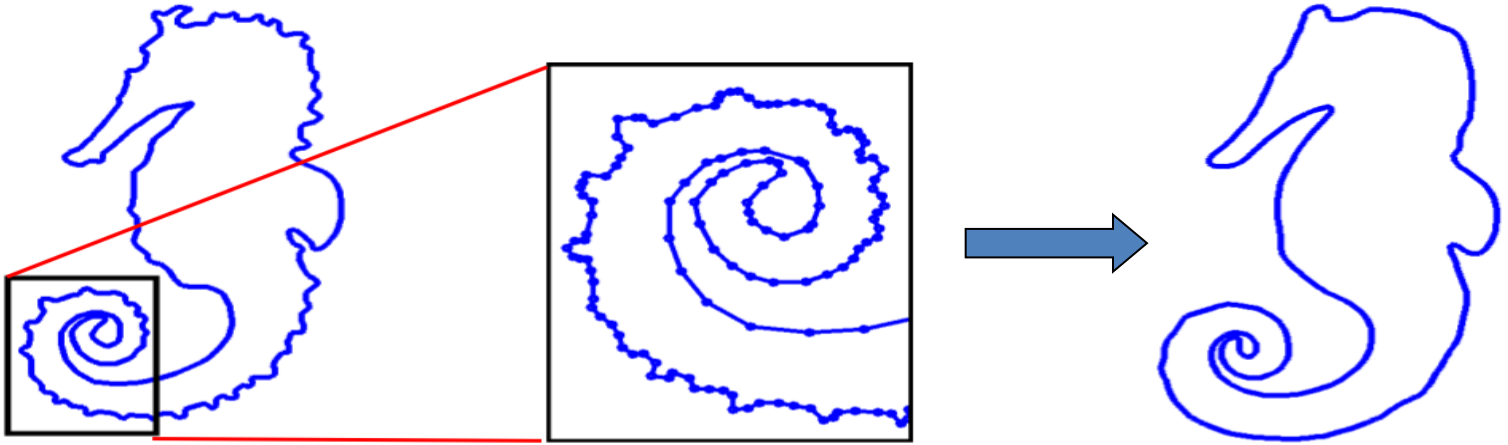
Outline



- Overview
- A signal processing perspective
  - A very gentle introduction — boredom alert ☺
  - Signal compression and filtering
  - Relation to discrete Fourier transform (DFT)
- Geometric perspective: global and intrinsic
- Dimensionality reduction
- Difficulties and challenges

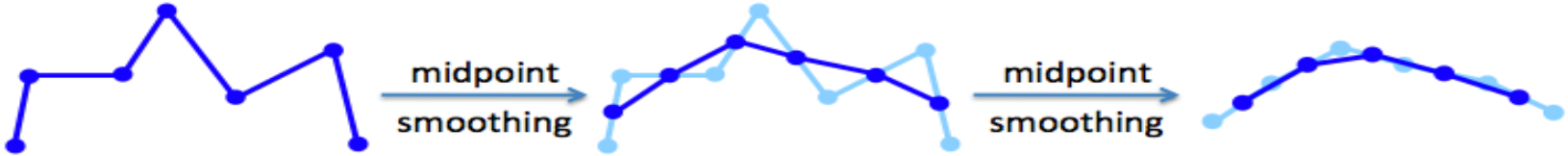
The smoothing problem

- Smooth out rough features of a contour (2D shape)



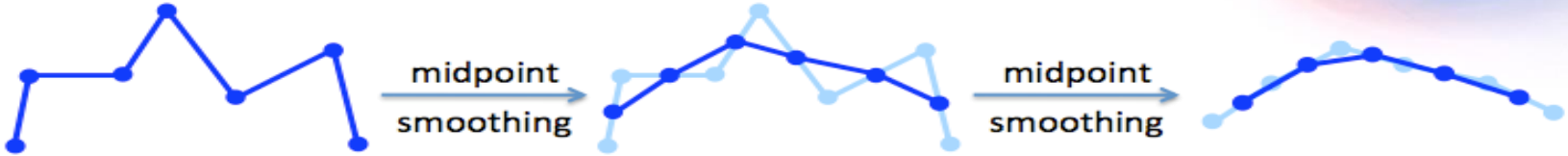
Midpoint smoothing

- Successive connection of midpoints

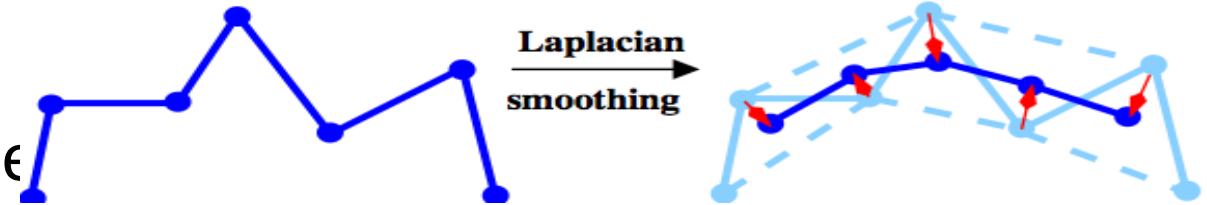


Midpoint → Laplacian smoothing

- Two steps of midpoint smoothing

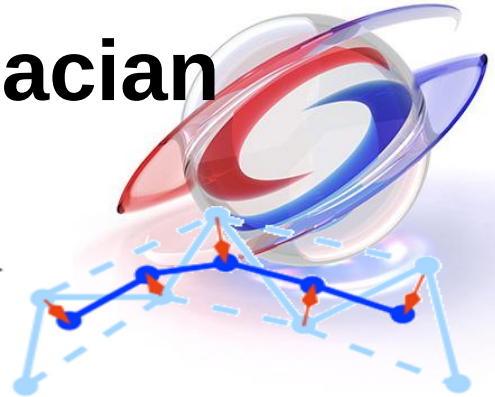


- One ste



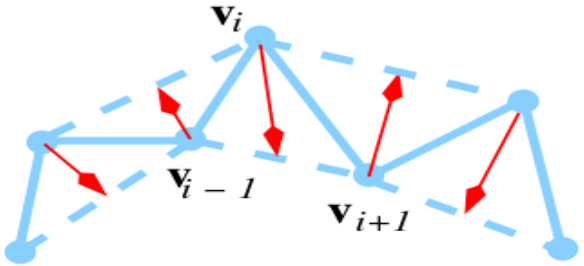
Laplacian smoothing and Laplacian

Laplacian smoothing



- Local averaging

$$\hat{\mathbf{v}}_i = \frac{1}{2}\left[\frac{1}{2}(\mathbf{v}_{i-1} + \mathbf{v}_i)\right] + \frac{1}{2}\left[\frac{1}{2}(\mathbf{v}_i + \mathbf{v}_{i+1})\right] = \frac{1}{4}\mathbf{v}_{i-1} + \frac{1}{2}\mathbf{v}_i + \frac{1}{4}\mathbf{v}_{i+1}$$

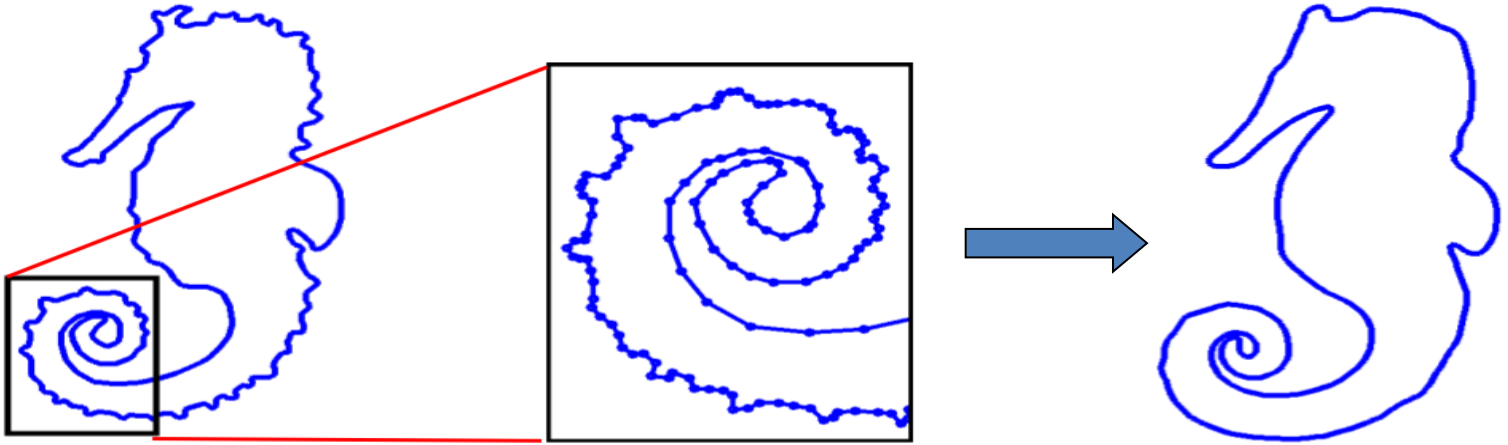


$$\delta(\mathbf{v}_i) = \frac{1}{2}(\mathbf{v}_{i-1} + \mathbf{v}_{i+1}) - \mathbf{v}_i$$

- **1D discrete Laplacian**

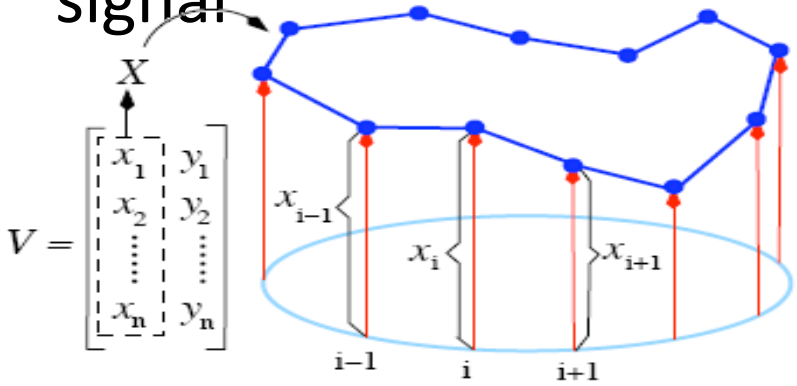
Laplacian smoothing

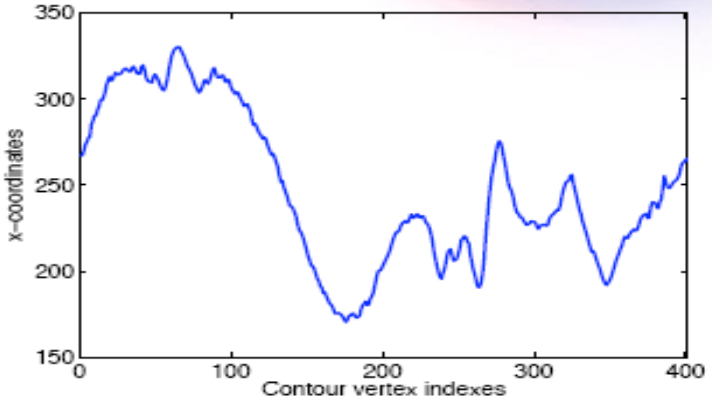
- Obtained by 10 steps of Laplacian smoothing



Signal representation

- Represent a contour using a discrete periodic 2D signal





x-coordinates of the seahorse contour

Laplacian smoothing in matrix form

$$\hat{X} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \vdots \\ \hat{x}_{n-1} \\ \hat{x}_n \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{4} & 0 & \dots & \dots & 0 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & 0 & \dots & \dots & 0 & \frac{1}{4} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = SX.$$

Smoothing operator

x component only

y treated same way

1D discrete Laplacian operator

$$\delta(X) = LX = \begin{bmatrix} 1 & -\frac{1}{2} & 0 & \dots & \dots & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 1 & -\frac{1}{2} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \dots & \dots & 0 & -\frac{1}{2} & 1 \end{bmatrix} X.$$

$$S = I - \frac{1}{2}L.$$

- Smoothing and Laplacian operator

Spectral or eigen-decomposition

- Spectral decomposition of the underlying linear space
- Use of eigenvectors of the Laplacian operator *L*
- *L* is real and symmetric
  - Real eigenvalues
  - Real and orthogonal set of eigenvectors
  - Eigenvectors span the vector space R^n

Spectral analysis of signal/geometry

- Express signal *X* as a linear sum of eigenvectors

$$X = \sum_{i=1}^{n} \mathbf{e}_i \tilde{x}_i = \begin{bmatrix} E_{11} \\ E_{21} \\ \vdots \\ E_{n1} \end{bmatrix} \tilde{x}_1 + \ldots + \begin{bmatrix} E_{1n} \\ E_{2n} \\ \vdots \\ E_{nn} \end{bmatrix} \tilde{x}_n = \begin{bmatrix} E_{11} & \ldots & E_{1n} \\ E_{21} & \ldots & E_{2n} \\ \vdots & \vdots & \vdots \\ E_{n1} & \ldots & E_{nn} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{bmatrix} = E\tilde{X}.$$

$$\tilde{X} = E^{\mathrm{T}} X$$

$$\tilde{x}_i = \mathbf{e}_i^{\mathrm{T}} \cdot X.$$

Projection of *X* along eigenvector

The spectral transform

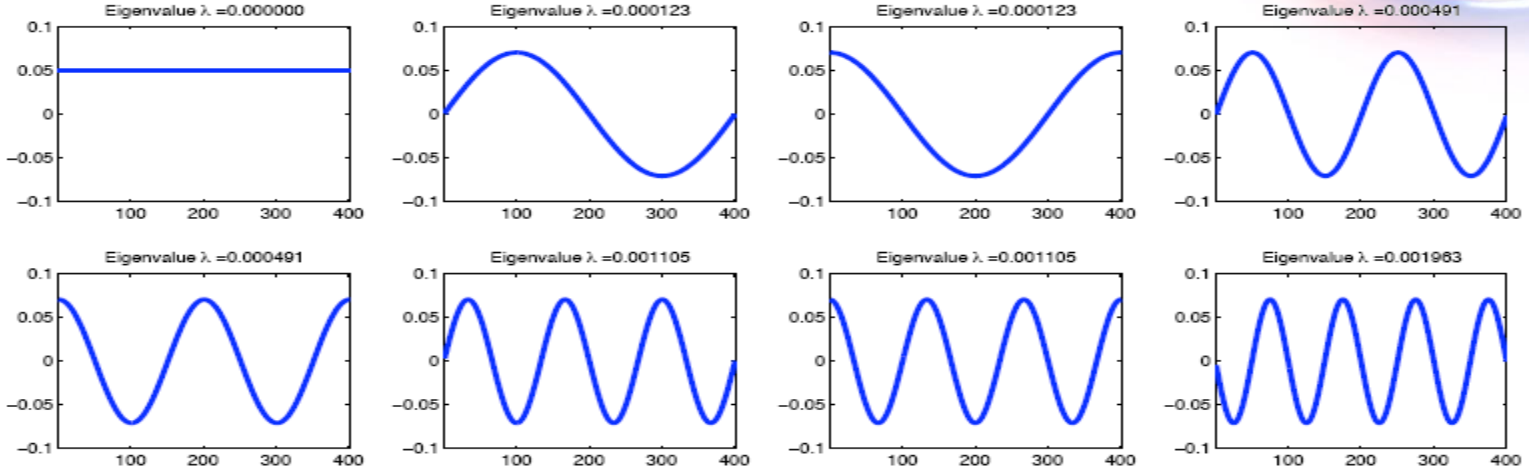
X → $\tilde{X}$

Spatial domain

Spectral domain

Plot of eigenvectors

- First 8 eigenvectors of the 1D Laplacian



More oscillation as eigenvalues (frequencies) increase

Relation to DFT

- Smallest eigenvalue of *L* is zero
- Each remaining eigenvalue (except for the last one when *n* is even) has multiplicity 2
- The plotted real eigenvectors are not unique to *L*
- One particular set of eigenvectors are the DFT basis
- Oscillatory behaviour of the two bases are similar

Signal reconstruction and compression

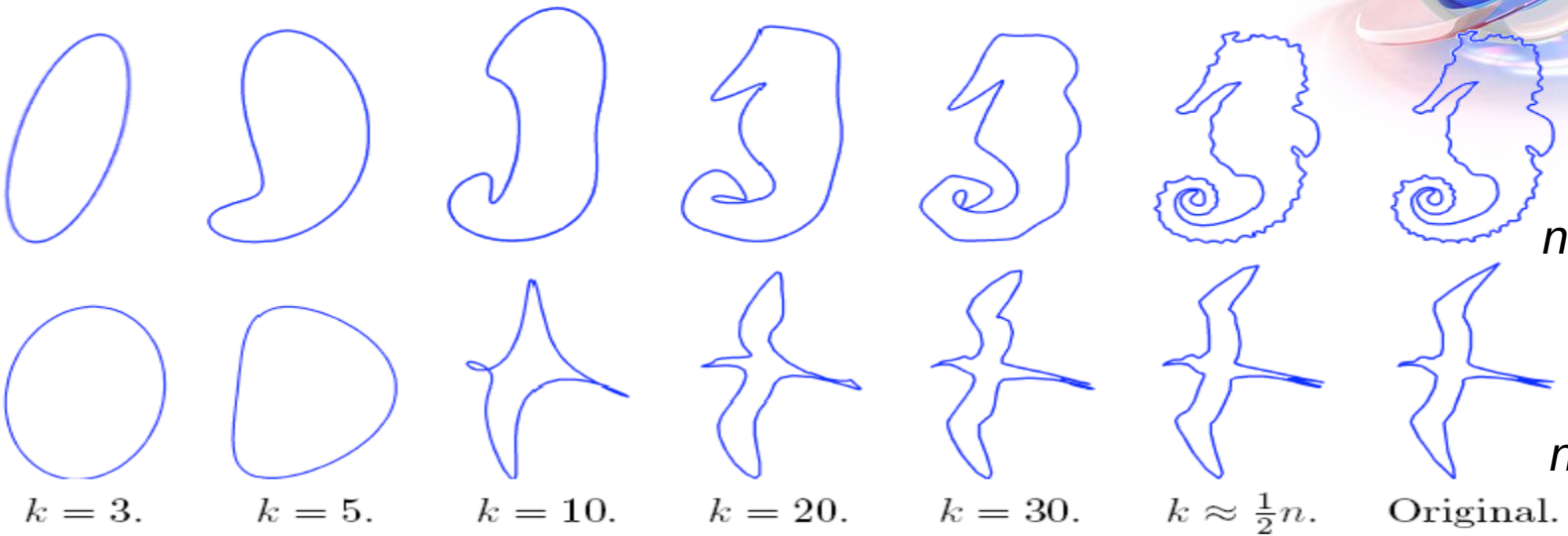
- Reconstruction using *k* leading coefficients

$$X^{(k)} = \sum_{i=1}^{k} \mathbf{e}_i \tilde{x}_i, \qquad k \leq n.$$

- A form of compression with information loss given by L_2

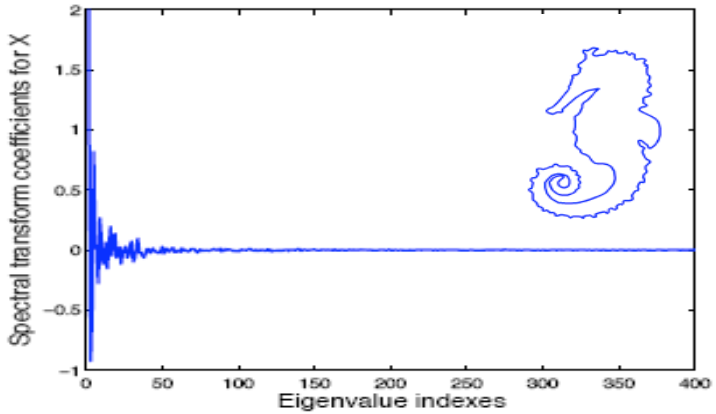
$$||X - X^{(k)}|| = || \sum_{i=k+1}^{n} \mathbf{e}_i \tilde{x}_i|| = \sqrt{\sum_{i=k+1}^{n} \tilde{x}_i^2}$$

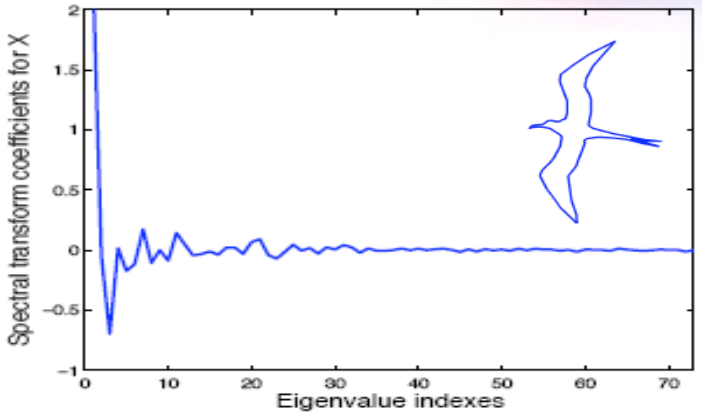
Examples



Eigenvalue plots

- Fairly fast eigenvalue decay





Filtering and Laplacian smoothing

- Recall the Laplacian smoothing operator

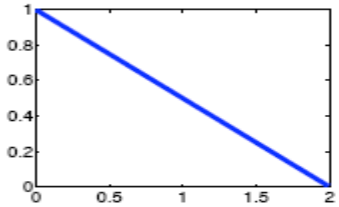
$$S = I - \frac{1}{2}L.$$

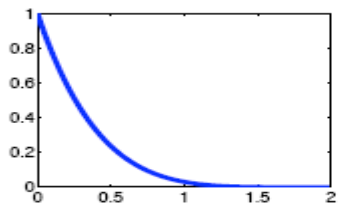
- Repeated application of *S*

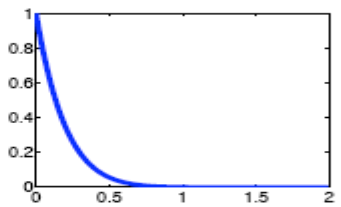
$$X^{(m)} = S^m X = (I - \frac{1}{2}L)^m X = \sum_{i=1}^{n} (I - \frac{1}{2}L)^m \mathbf{e}_i \tilde{x}_i = \sum_{i=1}^{n} \mathbf{e}_i (1 - \frac{1}{2}\lambda_i)^m \tilde{x}_i.$$

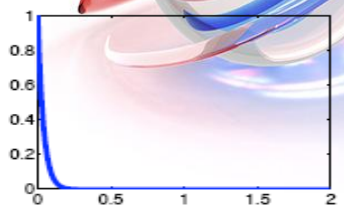
A filter applied to spectral coefficients

Examples









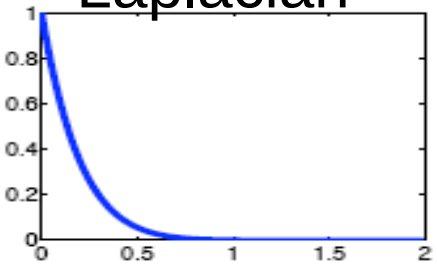
m = 1 *m* = 5 *m* = 10 *m* = 50

Filter: $f(\lambda) = (1 - \frac{1}{2}\lambda)^m$

More by Bruno

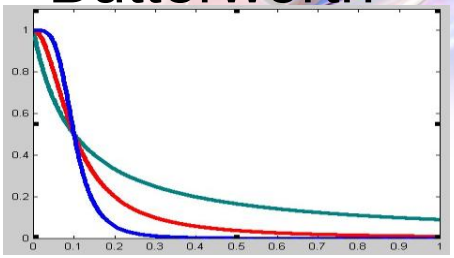
Aside: other filters

Laplacian

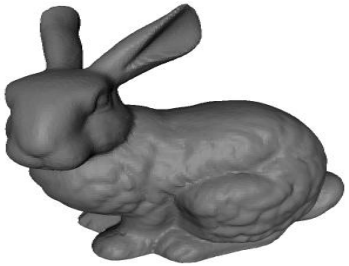


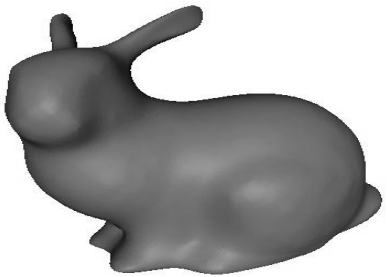
$$f(\lambda) = (1 - \tfrac{1}{2}\lambda)^m$$

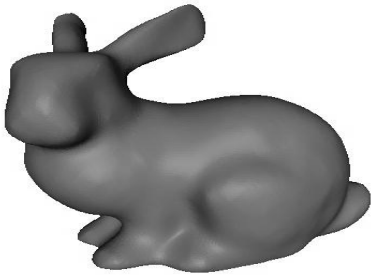
Butterworth



$$f(\lambda) = \frac{1}{1 + (\lambda\delta)^{2m}}$$



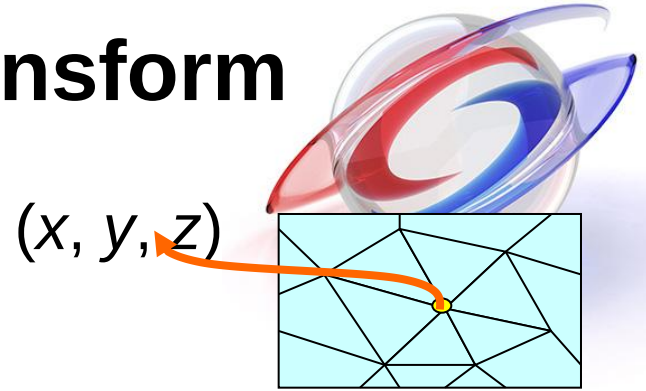




Computational issues

- No need to compute spectral coefficients for filtering
  - Polynomial filters (Laplacian): matrix-vector multiplication
  - Rational polynomial filters (Butterworth): solution of linear system
- Spectral compression needs explicit spectral transform
- Efficient computation by Bruno

Towards spectral mesh transform



- Signal representation
  - Vectors of *x*, *y*, *z* vertex coordinates
- Laplacian operator for meshes
  - Encodes connectivity and geometry
  - Combinatorial: **graph Laplacians** and variants
  - Discretization of the continuous Laplace-Beltrami operator — coverage by Bruno
- The same kind of spectral transform and analysis

Spectral mesh compression

(a) Original.

(b) $k = 300$.

(c) $k = 200$.

(d) $k = 100$.

(e) $k = 50$.

(f) $k = 10$.

(g) $k = 5$.

(h) $k = 3$.

Main references

G. Taubin. A signal processing approach to fair surface design. In *Proc. of ACM SIGGRAPH*, pages 351–358, 1995.

Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *Proc. of ACM SIGGRAPH*, pages 279–286, 2000.

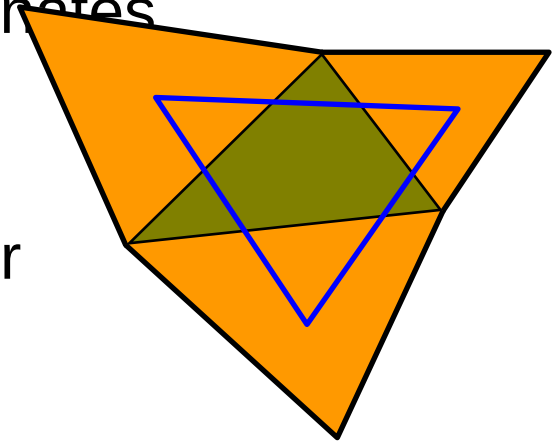
A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, 1989.

Outline

- Overview
- A signal processing perspective
  - A very gentle introduction — boredom alert ☺
  - Signal compression and filtering
  - Relation to discrete Fourier transform (DFT)
- Geometric perspective: global and intrinsic
- Dimensionality reduction
- Difficulties and challenges

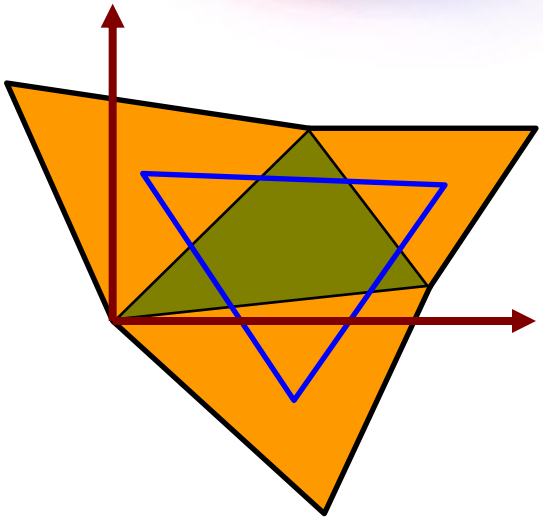
A geometric perspective: classical

- Classical Euclidean geometry
  - Primitives not represented in coordinates
  - Geometric relationships deduced in a pure and self-contained manner
  - Use of axioms



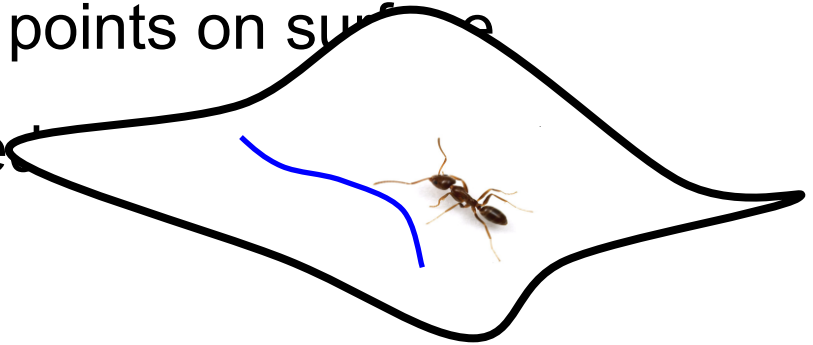
A geometric perspective: analytic

- Descartes' analytic geometry
  - Algebraic analysis tools introduced
  - Primitives referenced in global frame — **extrinsic** approach



Intrinsic approach

- Riemann's intrinsic view of geometry
  - Geometry viewed purely from the surface perspective
  - Metric: "distance" between points on surface
  - Many spaces can be treated simultaneously: **isometry**



Spectral methods: intrinsic view

- Spectral approach takes the intrinsic view
  - **Intrinsic geometric/mesh information** captured via a linear mesh operator
  - **Eigenstructures** of the operator present the intrinsic geometric information in an **organized manner**
  - Never need all the eigenstructures, the dominant ones often suffice

Global, by definition

Courant-Fisher Theorem:

Let $S \in \Re^{n \times n}$ be a symmetric matrix. Then its eigenvalues $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$ must satisfy the following,

$$\lambda_i = \min_{\substack{V \subseteq \Re^n \\ \dim V = i}} \max_{\substack{\mathbf{v} \in V \\ \|\mathbf{v}\|_2 = 1}} \mathbf{v}^{\mathrm{T}} S \mathbf{v}$$

$$\mathbf{v}^{\mathrm{T}} S \mathbf{v} \rightarrow \frac{\mathbf{v}^{\mathrm{T}} S \mathbf{v}}{\mathbf{v}^{\mathrm{T}} \mathbf{v}}$$

Rayleigh quotient

$$\lambda_1 = \min_{\|\mathbf{v}\|_2 = 1} \mathbf{v}^{\mathrm{T}} S \mathbf{v} \quad \text{and} \quad \lambda_n = \max_{\|\mathbf{v}\|_2 = 1} \mathbf{v}^{\mathrm{T}} S \mathbf{v}$$

where V is a subspace of $\Re^n$ with the given dimension. For only the smallest and largest eigenvalues,

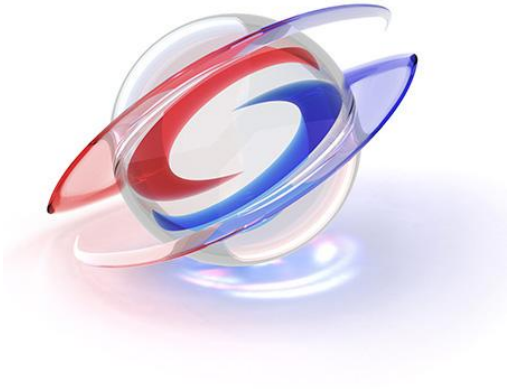
Corollary to Courant-Fisher

Let $S \in \Re^{n \times n}$ be a symmetric matrix. Then its eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots. \leq \lambda_n$ must satisfy the following,

$$\lambda_i = \min_{\substack{\|\mathbf{v}\|_2 = 1 \\ \mathbf{v}^{\mathrm{T}}\mathbf{v}_k = 0,\ \forall 1 \leq k \leq i\text{-}1}} \mathbf{v}^{\mathrm{T}} S \mathbf{v}$$

where $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{i-1}$ are eigenvectors of S corresponding to the smallest eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_{i-1}$, respectively.

Interpretation



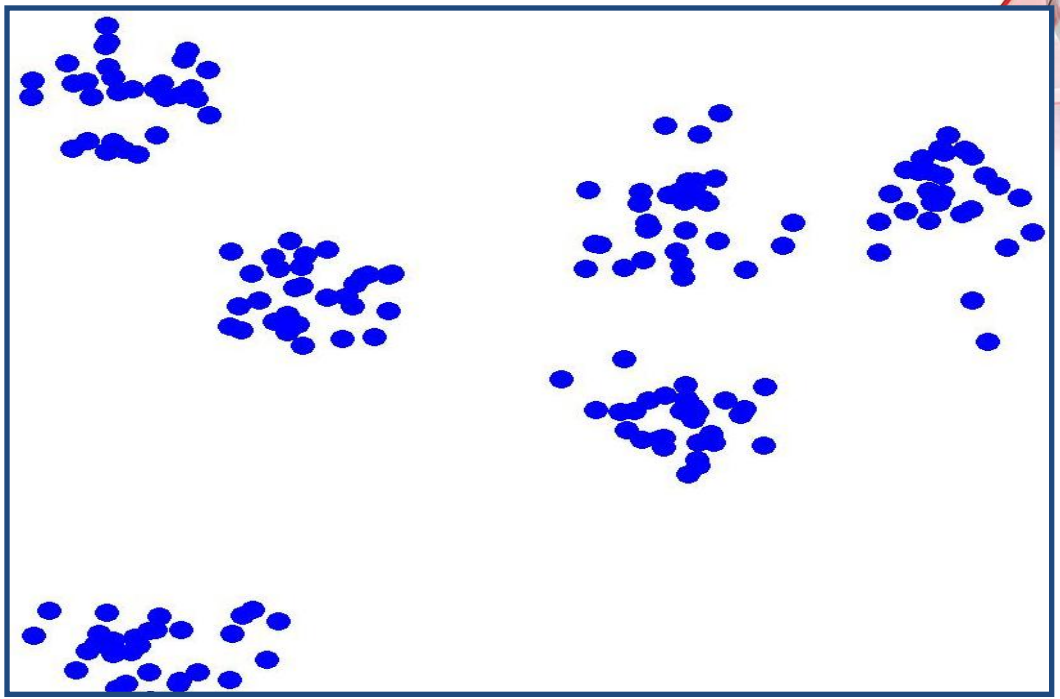
$$\lambda_i = \min_{\substack{\|\mathbf{v}\|_2 = 1 \\ \mathbf{v}^{\mathrm{T}}\mathbf{v}_k = 0,\ \forall 1 \le k \le i\text{-}1}} \mathbf{v}^{\mathrm{T}} S \mathbf{v}$$

- Smallest eigenvector minimizes the Rayleigh quotient
- *k*-th smallest eigenvector minimizes Rayleigh quotient, among the vectors orthogonal to all previous eigenvectors
- Solutions to **global optimization** problems

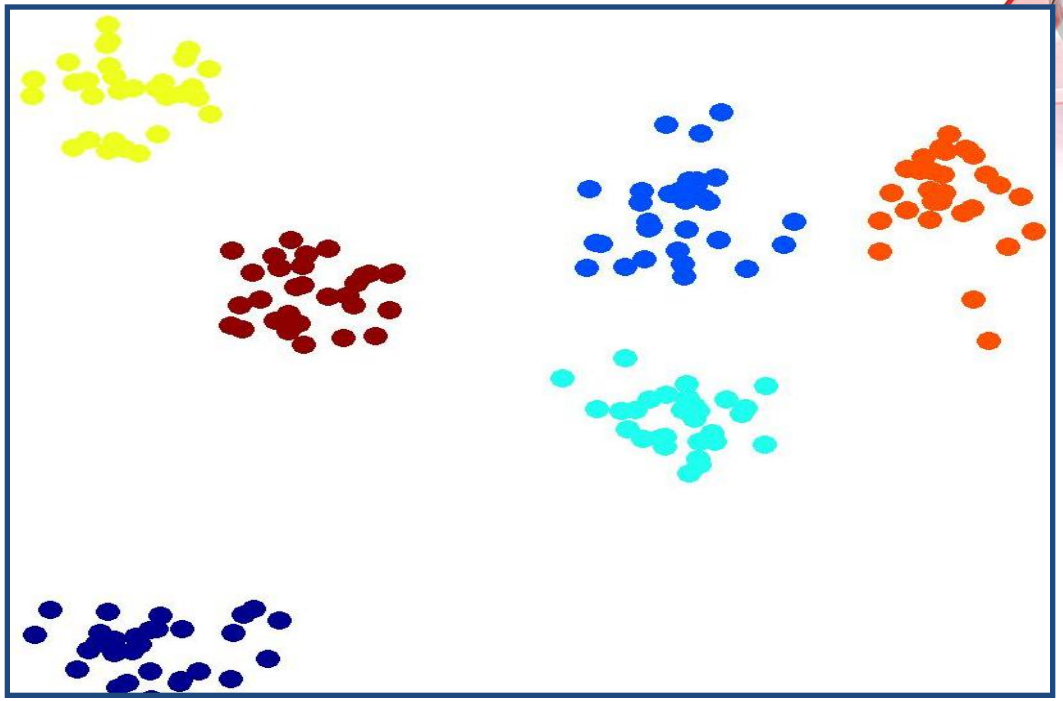
Capture of global information

- Eigenvalues
  - Spectral graph theory: graph eigenvalues closely related to almost all major global graph invariants
  - Often adopted as compact global shape descriptors
- Eigenvectors
  - Useful extremal properties, e.g., heuristic for NP-hard problems — normalized cuts and sequencing
  - Spectral embeddings capture global information

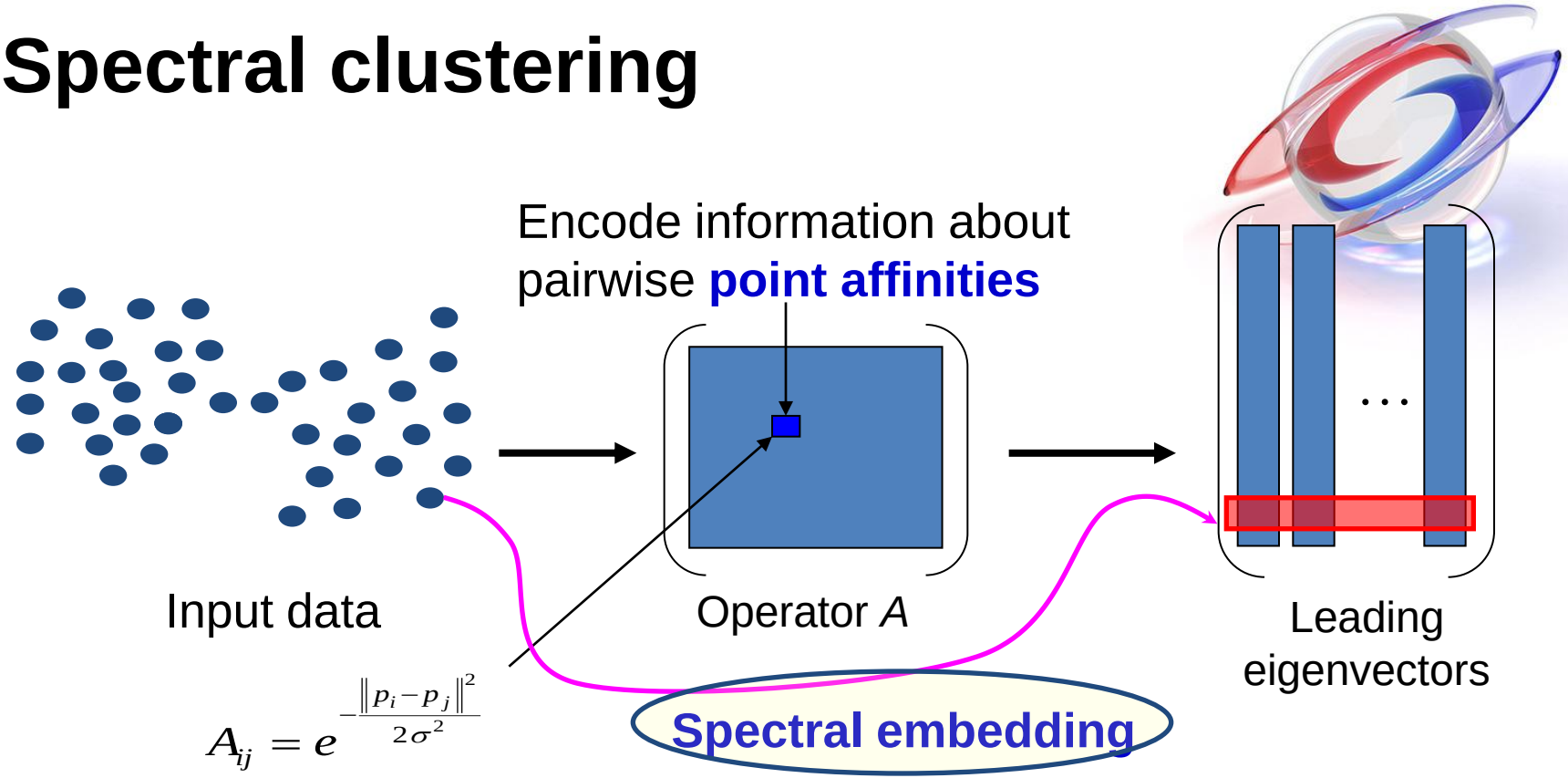
Example: clustering problem



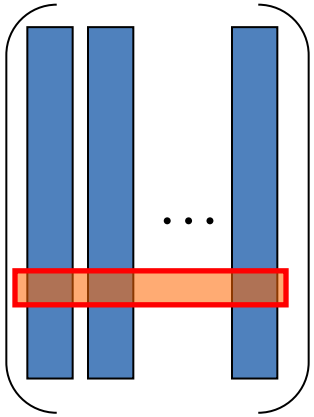
Example: clustering problem



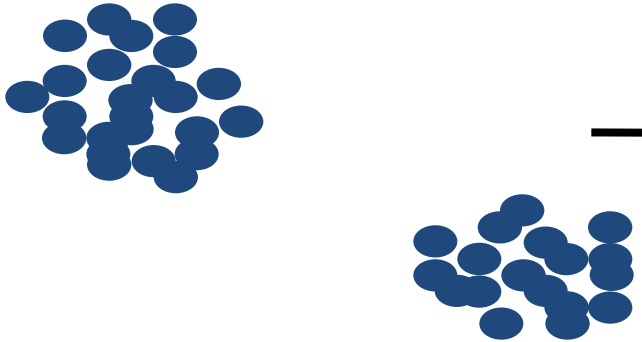
Spectral clustering



Spectral clustering



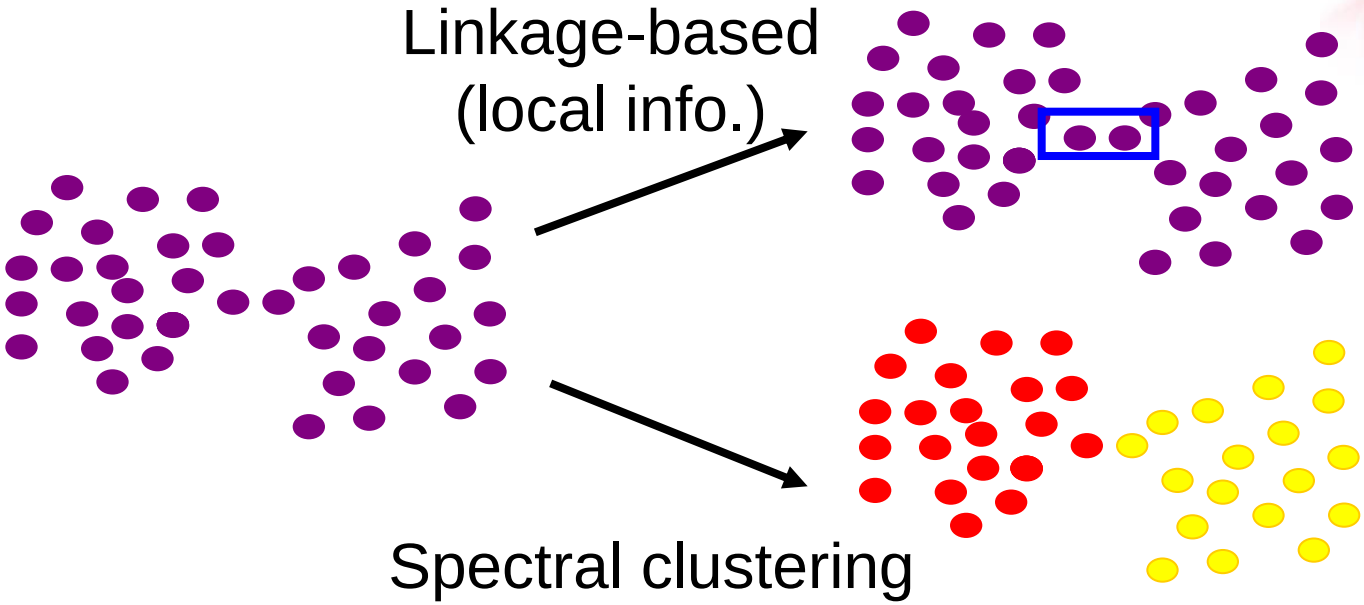
eigenvectors



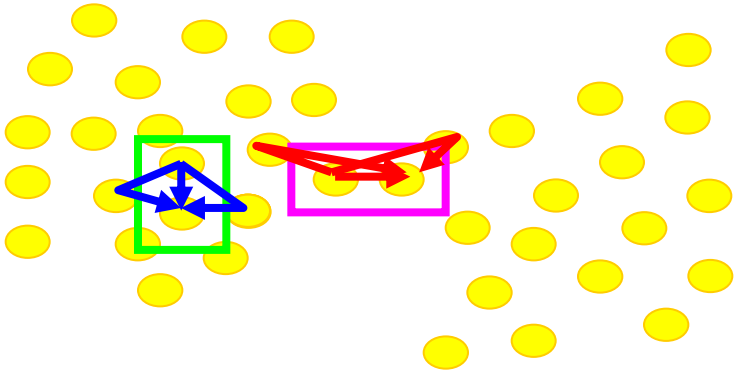
In spectral domain

Perform any clustering (e.g., *k*-means) in spectral domain

What does it work this way?



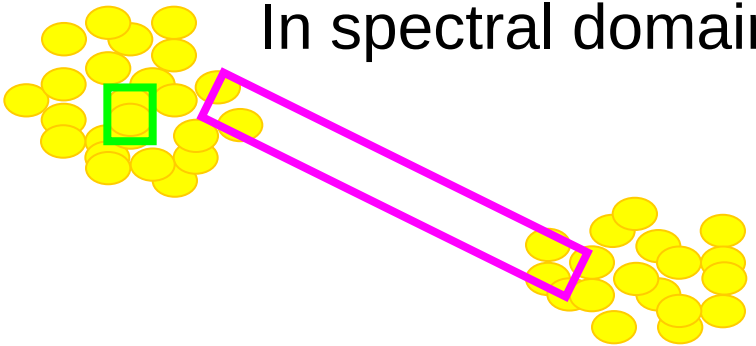
Local vs. global distances



Would be nice to cluster according to c_{ij}

- A **good distance**: Points in same cluster closer in transformed domain
- Look at **set of shortest paths — more global**
- **Commute time distance** c_{ij} = expected time for random walk to go from *i* to *j* and then back to *i*

Local vs. global distances



Commute time and spectral

- Consider eigen-decompose the graph Laplacian K

$$K = U\Lambda U^{\mathsf{T}}$$

- Let K' be the **generalized inverse** of K,

$$K' = U\Lambda' U^{\mathsf{T}},$$

$\Lambda'_{ii} = 1/\Lambda_{ii}$ if $\Lambda_{ii} \neq 0$, otherwise $\Lambda'_{ii} = 0$.

- Note: the Laplacian is singular

Commute time and spectral

- Let z_i be the i-th row of $U\Lambda'^{1/2}$ — the spectral embedding
- Essentially scaling each eigenvector by the inverse square root of its eigenvalue
- Then

$$||z_i - z_j||^2 = c_{ij}$$

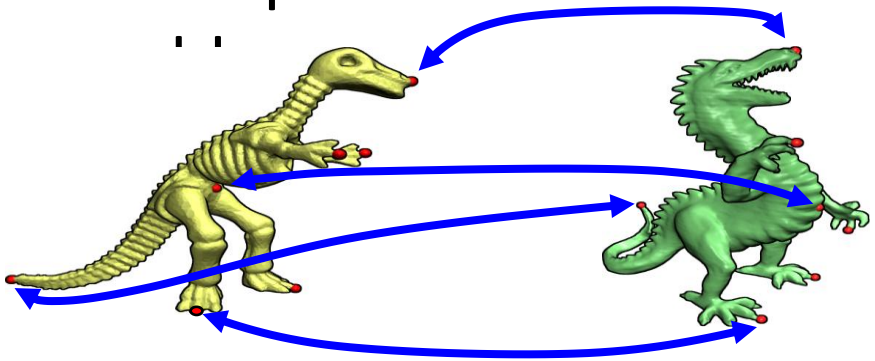
the communite time distance

[Klein & Randic 93, Fouss et al. 06]

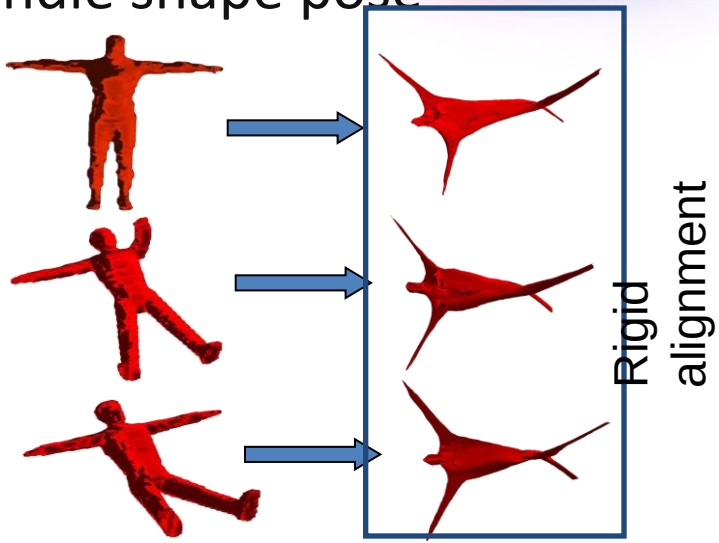
- Full set of eigenvectors used, but select first k in practice

Example: intrinsic geometry

- Our first example: the Correspondence



- Spectral transform to handle shape pose



More in the afternoon

Main references

F. R. K. Chung. *Spectral Graph Theory*. AMS, 1997.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

Ulrike von Luxburg. A tutorial on spectral clustering. Technical Report TR-149, Max Plank Institute for Biological Cybernetics, August 2006.

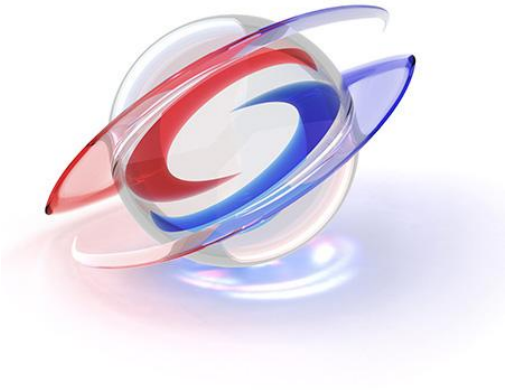
Varun Jain, Hao Zhang, and Oliver van Kaick. Non-rigid spectral correspondence of triangle meshes. *International Journal on Shape Modeling*, 13(1):101–124, 2007.

H. Qiu and ER Hancock, *Clustering and embedding using commute times*, IEEE Transactions on Pattern Analysis and Machine Intelligence **29** (2007), no. 11, 1873–1890.

Outline

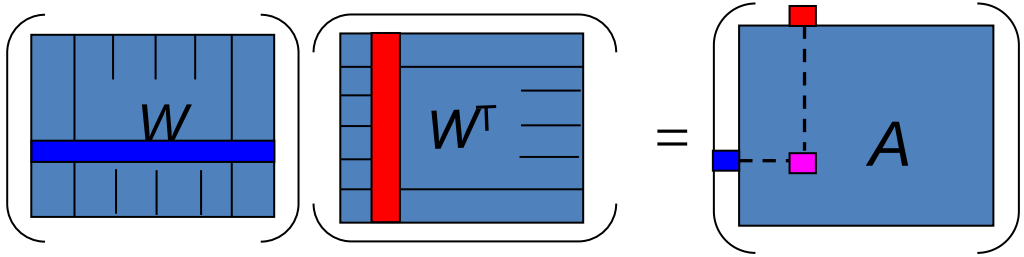
- Overview
- A signal processing perspective
  - A very gentle introduction — boredom alert ☺
  - Signal compression and filtering
  - Relation to discrete Fourier transform (DFT)
- Geometric perspective: global and intrinsic
- Dimensionality reduction
- Difficulties and challenges

Spectral embedding



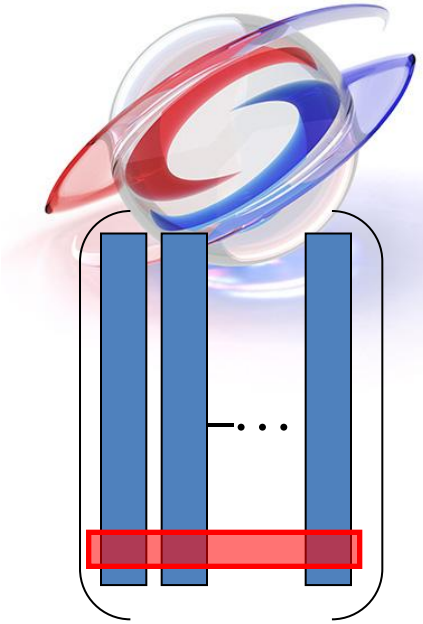
- Spectral decomposition $A = U\Lambda U^T$

- Full spectral embedding given by scaled eigenvectors (each scaled by squared root of eigenvalue) completely captures the operator



$$W = U\Lambda^{1/2}$$

Dimensionality reduction



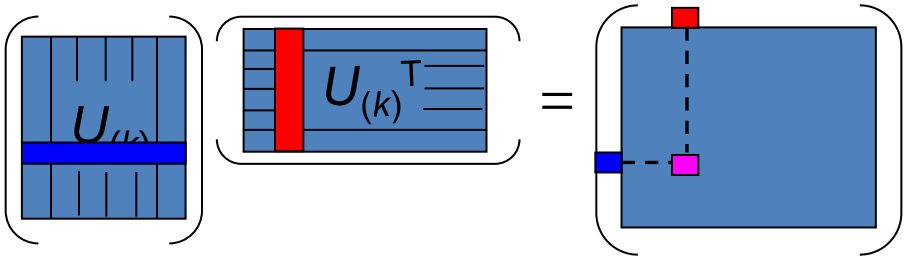
- Full spectral embedding is high-dimensional
- Use few dominant eigenvectors

  dimensionality reduction

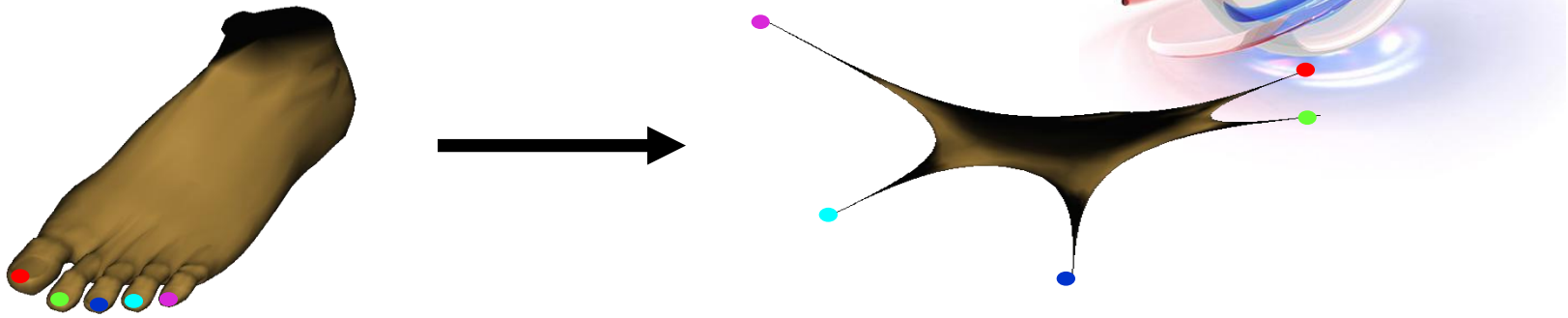
  - **Information-preserving**
  - **Structure enhancement** (Polarization Theorem)
  - Low-D representation: **simplifying solutions**

Eckard & Young: Info-preserving

- $A \in \Re^{n \times n}$: symmetric and positive semi-definite
- $U_{(k)} \in \Re^{n \times k}$: leading eigenvectors of A, scaled by square root of eigenvalues
- Then $U_{(k)}U_{(k)}^{\mathsf{T}}$ is the **best rank-*k* approximation** of A in Frobenius norm



Low-D leads to simpler problems



- Mesh projected into the eigenspace formed by the first two eigenvectors of a mesh Laplacian
- Reduce 3D analysis to contour analysis [Liu & Zhang 07]

Main references

Hao Zhang, Oliver van Kaick, and Ramsay Dyer. Spectral mesh processing. *Computer Graphics Forum*, 2009, to appear. http://www.cs.sfu.ca/~haoz/pubs/zhang_cgf09_spect_survey.pdf.

Rong Liu and Hao Zhang. Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum (Special Issue of Eurographics 2007)*, 26:385–394, 2007.

[EY36] ECKART C., YOUNG G.: The approximation of one matrix by another of lower rank. *Psychometrika 1* (1936), 211–218.

[BH03] BRAND M., HUANG K.: A unifying theorem for spectral embedding and clustering. In *Proc. of Int. Conf. on AI and Stat.* (Key West, Florida, 2003).

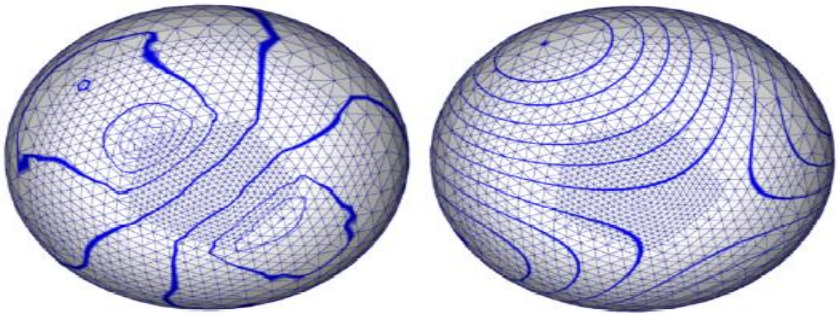
Outline

- Overview
- A signal processing perspective
  - A very gentle introduction — boredom alert ☺
  - Signal compression and filtering
  - Relation to discrete Fourier transform (DFT)
- Geometric perspective: global and intrinsic
- Dimensionality reduction
- Difficulties and challenges

Not quite DFT

- Basis for DFT is fixed given *n*, e.g., regular and easy to compare (Fourier descriptors)

- Spectral mesh transform operator-dependent

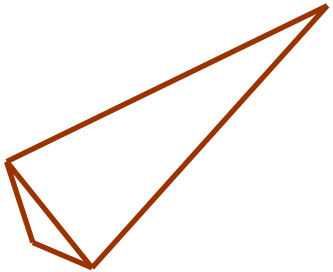


Different behavior of eigen-functions on the same sphere

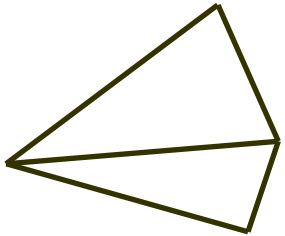
Which operator to use?

No free lunch

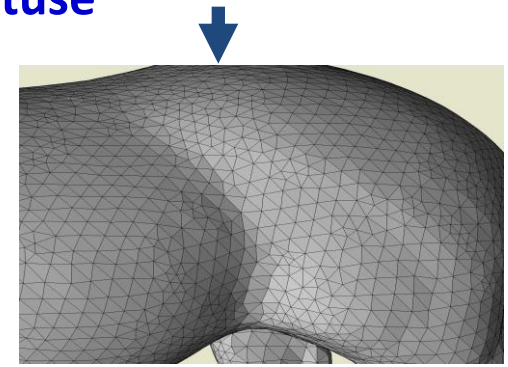
- No mesh Laplacian on general meshes can satisfy a list of all desirable properties
- Remedy: use nice meshes — **Delaunay** or **non-obtuse**



Delaunay but obtuse



Non-obtuse



No DFT again

- Computational issues: FFT vs. eigen-decomposition
- Regularity of vibration patterns lost
  - Difficult to characterize eigenvectors, eigenvalue not enough
  - Non-trivial to compare two sets of eigenvectors — how to pair up?

More in the afternoon

Main references

Max Wardetzky, Saurabh Mathur, Felix Kalberer, and Eitan Grinspun. Discrete laplace operators: No free lunch. *Eurographics Symposium on Geometry Processing*, 2007.

[VL08] VALLET B., LÉVY B.: Spectral geometry processing with manifold harmonics. *Computer Graphics Forum (Special Issue of Eurographics) 27*, 2 (2008), 251–260.

DYER, R., ZHANG, H., AND MÖLLER, T. 2007. Delaunay mesh construction. In *Symp. Geometry Processing*, 271–282.

[LZ06] LI J., ZHANG H.: Nonobtuse remeshing and decimation. In *SGP* (2006), pp. 235–238.

Conclusion

- Spectral mesh processing
  - Use eigenstructures of appropriately defined linear mesh operators for geometry analysis and processing
  - Solve a problem in a different domain via a transform — spectral transform
  - Fourier analysis on meshes
  - Captures global and intrinsic shape characteristics
  - Dimensionality reduction: effective and simplifying

SIGGRAPH 2010

“*Spectral Mesh Processing*”

Bruno Lévy and Richard Hao Zhang

Do Your Own Spectral Mesh Processing at Home

Bruno Lévy

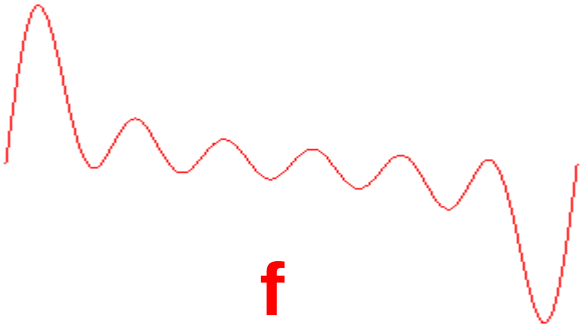
INRIA - ALICE

Introduction

I. Harmonics
II. DEC formulation
III. Filtering
IV. Numerics

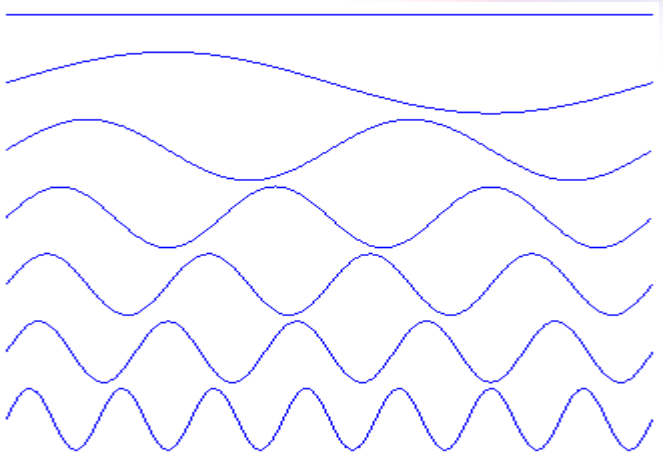
Introduction

Fourier transform



f

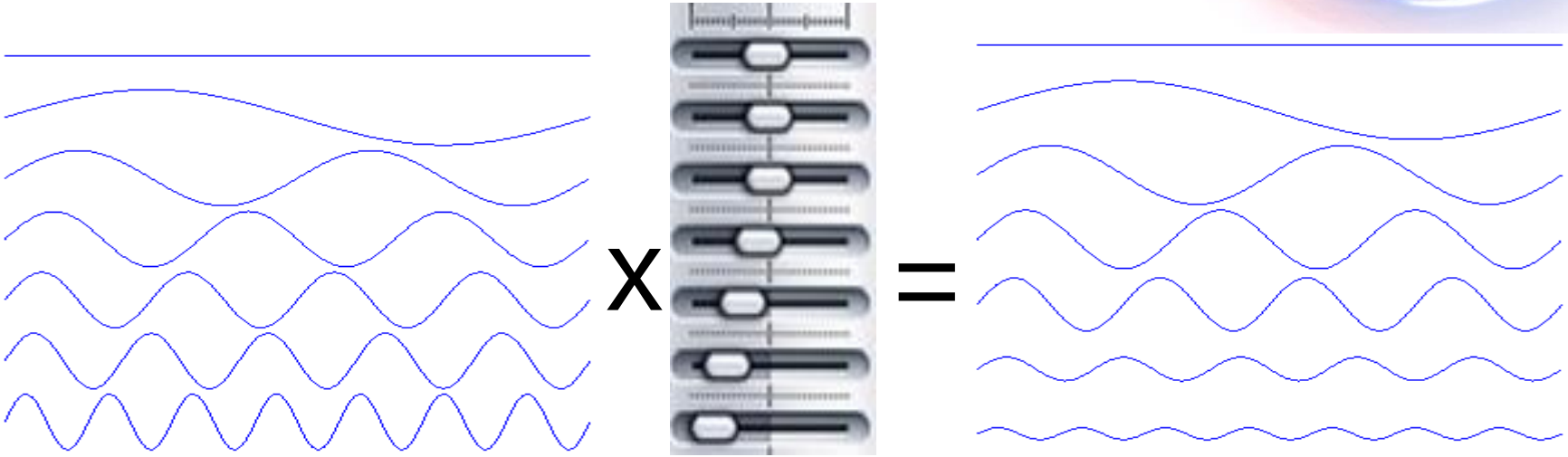
$=\Sigma$



sin(kx)

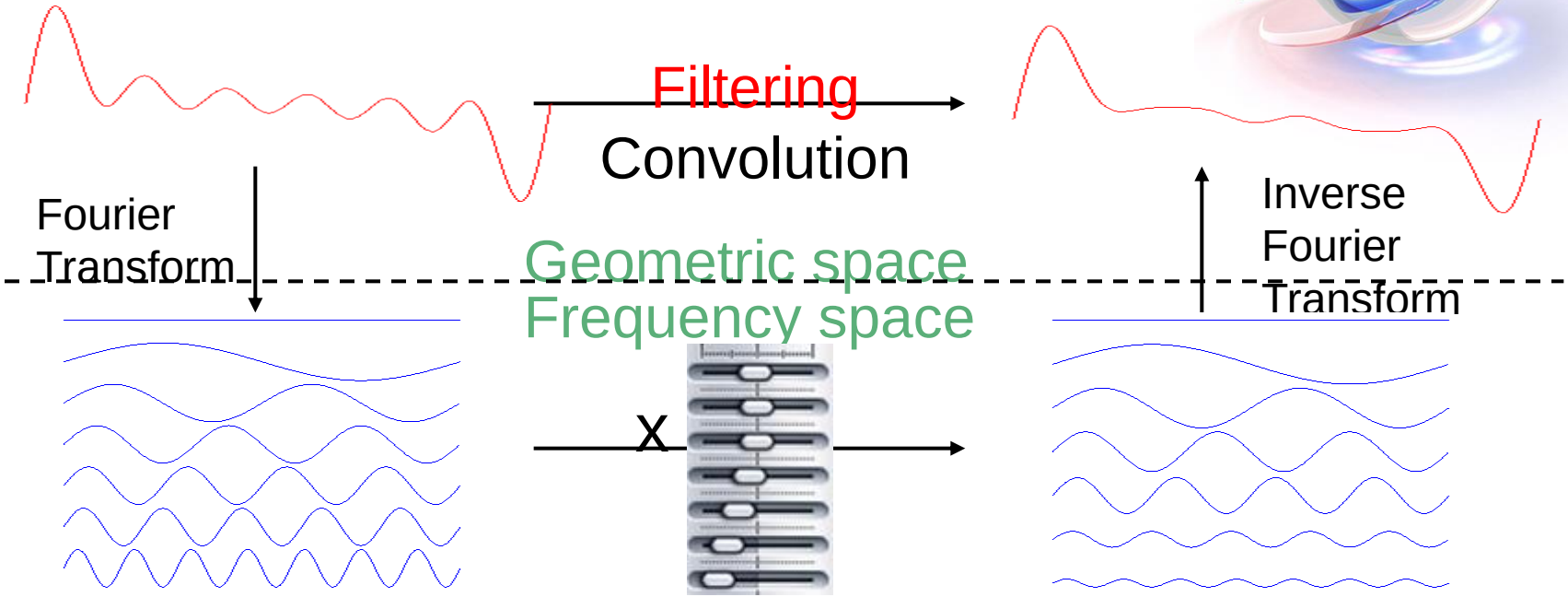
Introduction

Filtering



Introduction

Filtering



Introduction

Filtering on a mesh

Filtering

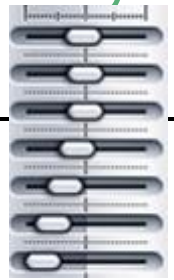
[Taubin 95]

Geometric space

Frequency space

?

x



?

Introduction

Filtering on a mesh

Filtering

[Taubin 95]

Geometric space

Frequency space

?

x

?

Introduction

Filtering on a mesh

Filtering

[Taubin 95]

Geometric space

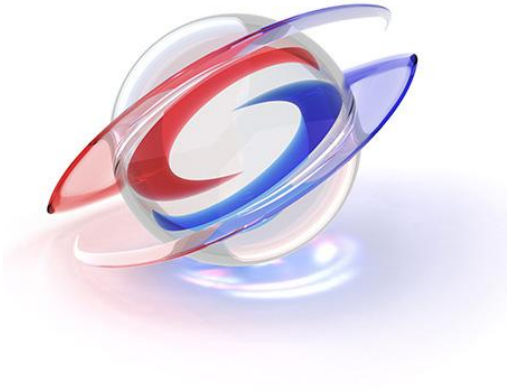
Frequency space

?

[Karni00] mesh compression
[Zhang06] shape matching
[Reuter03] shape signatures
[Dong06] quadrangulation

?

I Harmonics



Introduction

I. **Harmonics**

II. DEC formulation

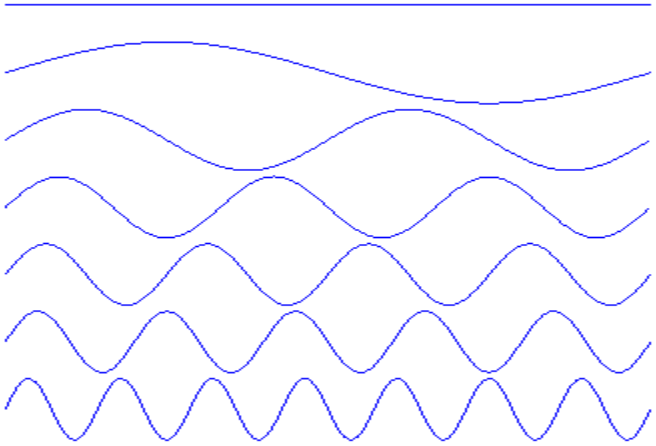
III. Filtering

IV. Numerics

Results and conclusion

I Harmonics

Question



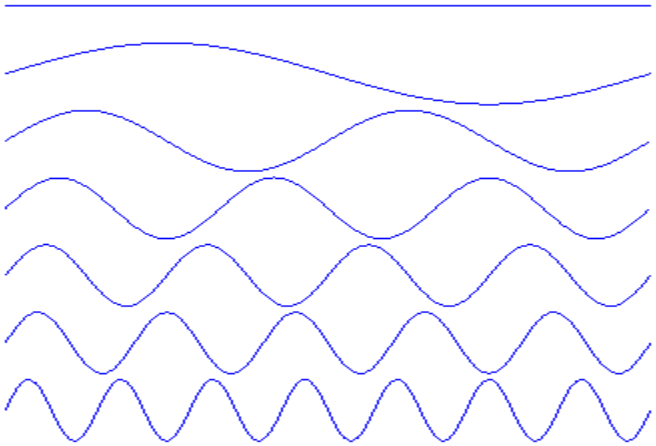
sin(kx)

on

?

I Harmonics

Harmonics and vibrations



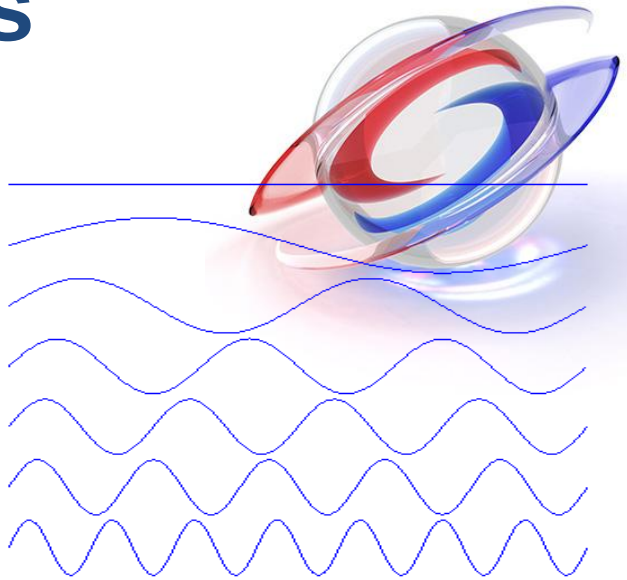
sin(kx) are the stationary vibrating modes = harmonics of a string

I Harmonics

Manifold Harmonics

Harmonics

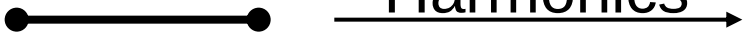
Harmonics

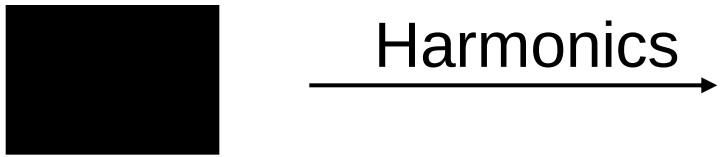


?

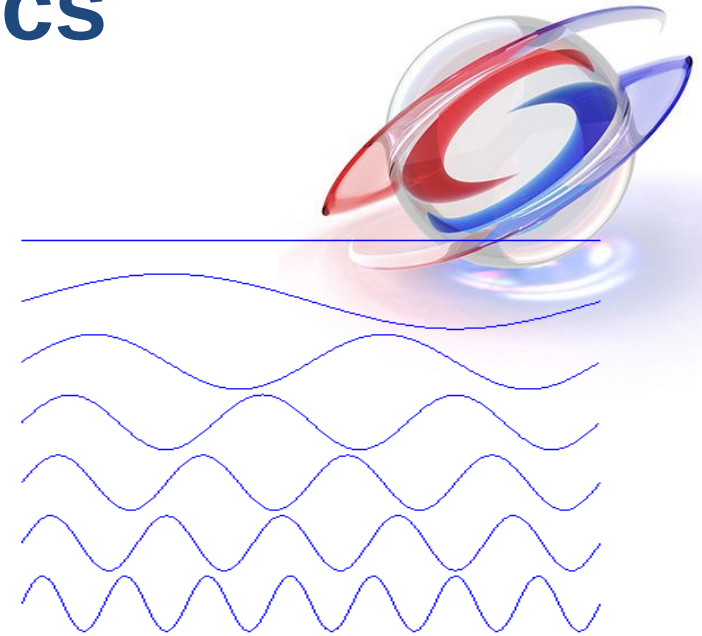
I Harmonics

Square Harmonics



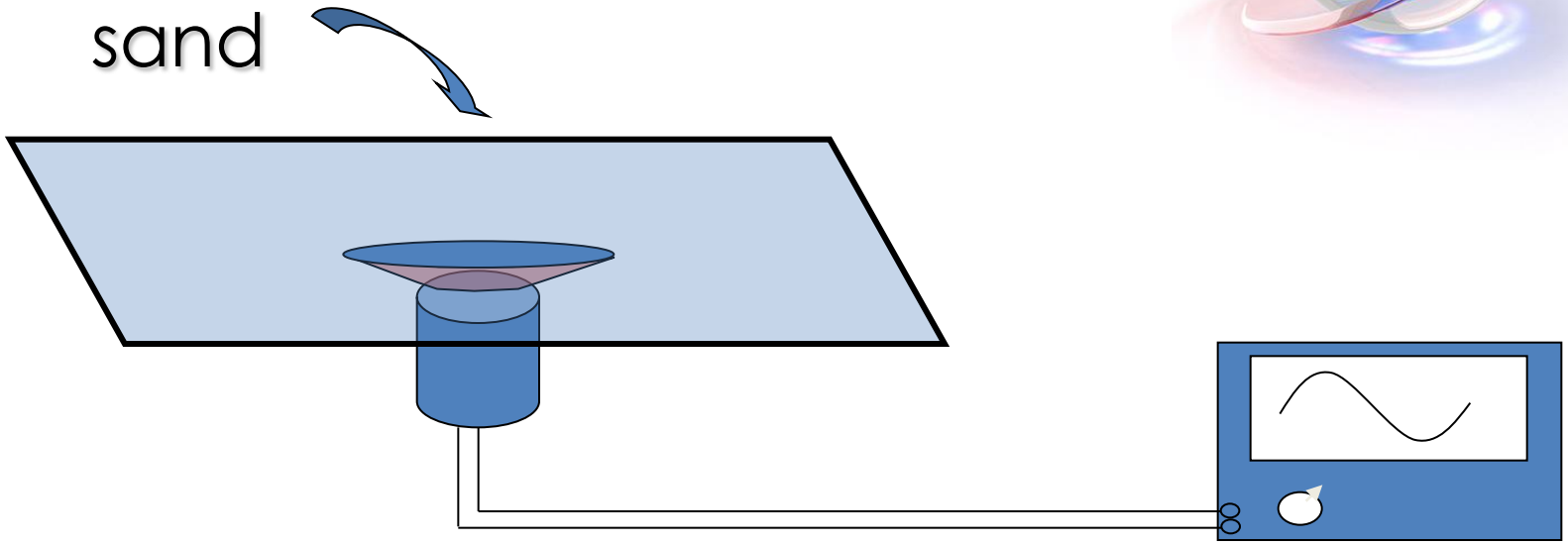


?



I Harmonics

Chladni plates

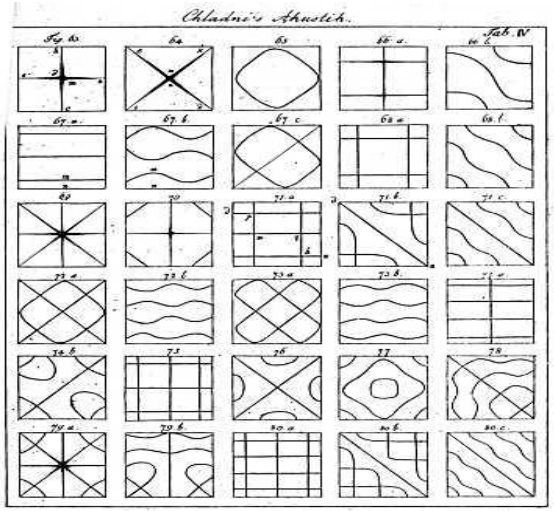


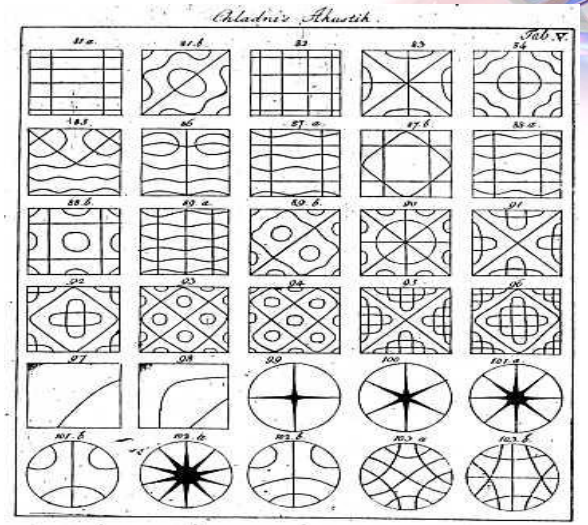
I Harmonics

Chladni plates

I Harmonics

Chladni plates

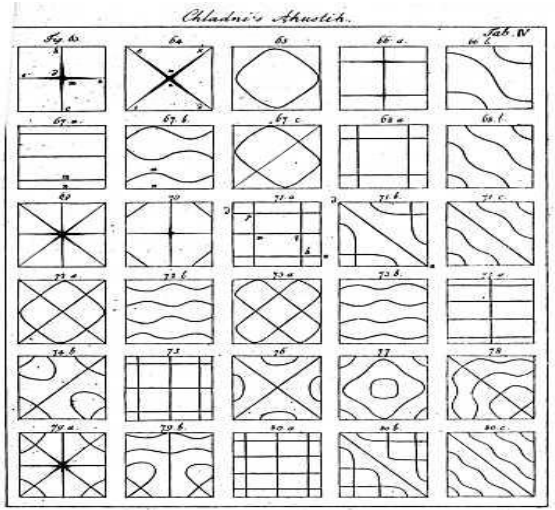




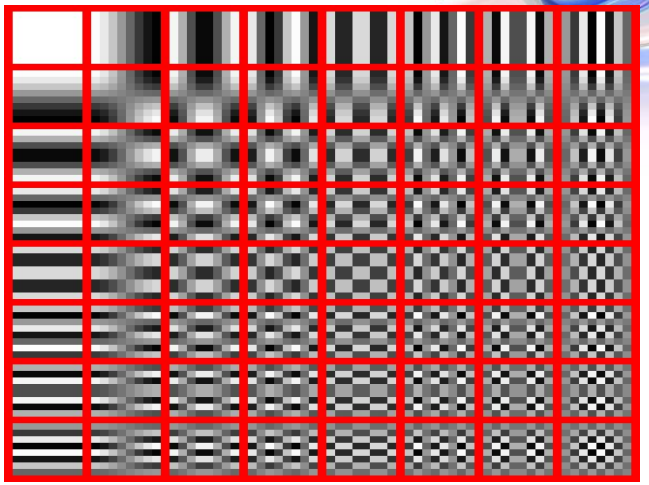
Discoveries concerning the theory of music, Chladni, 1787

I Harmonics

Chladni plates and jpeg



Chladni plates, 1787

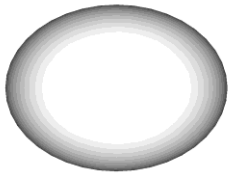


Discrete cosine transform (jpeg)

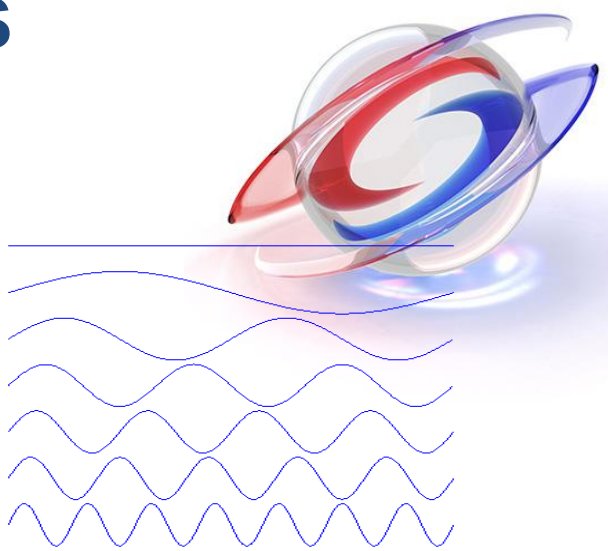
I Harmonics

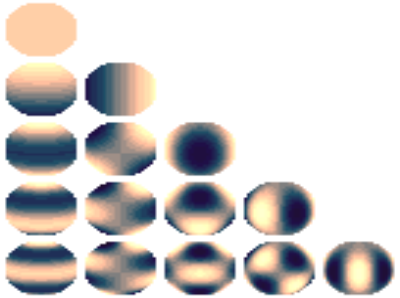
Spherical Harmonics

Harmonics



Harmonics



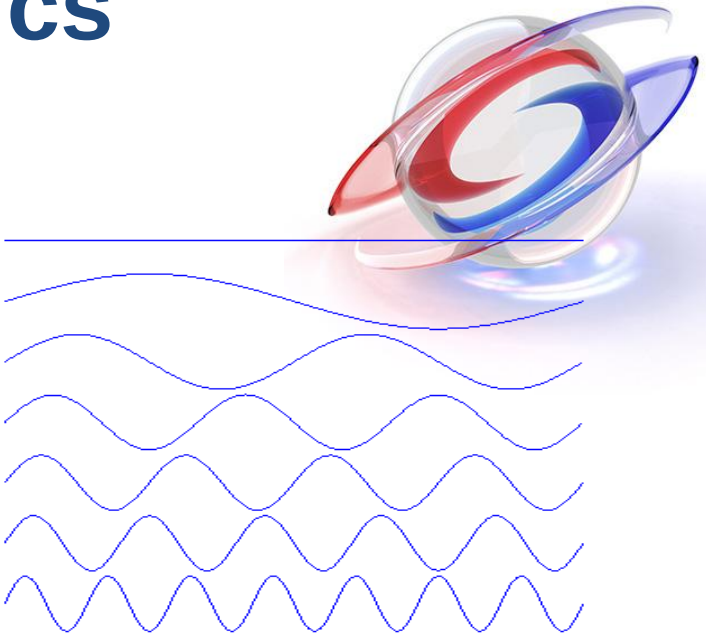


I Harmonics

Manifold Harmonics

Harmonics

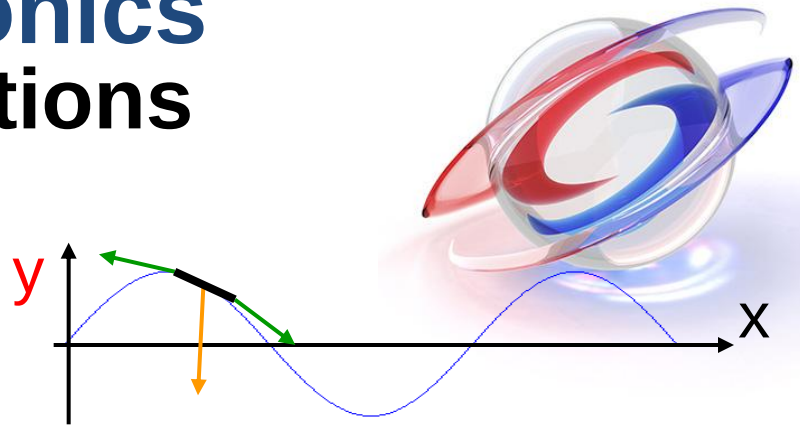
Harmonics



Stationary vibrating modes

I Harmonics

Harmonics and vibrations



- **Wave equation**:

$T\,\partial^2 y/\partial x^2 = \mu\,\partial^2 y/\partial t^2$

T: stiffness μ: mass

- **Stationary modes**:

$y(x,t) = y(x)\sin(\omega t)$

$\partial^2 y/\partial x^2 = -\mu\omega^2/T\ y$

eigenfunctions of $\partial^2/\partial x^2$

I Harmonics

I Harmonics : recap

- Harmonics are **eigenfunctions** of $\partial^2/\partial x^2$
- On a mesh, $\partial^2/\partial x^2$ is the Laplacian Δ
- We need the **eigenfunctions** of Δ
- Several possible discretizations:
  - Finite Element Modeling [Martin Reuter]
  - Discrete Exterior Calculus (this talk)

II DEC formulation

Introduction

I. Harmonics

II. DEC formulation

III. Filtering

IV. Numerics

Results and conclusion

II DEC formulation

Discrete Exterior Calculus

Discretize equations on a mesh

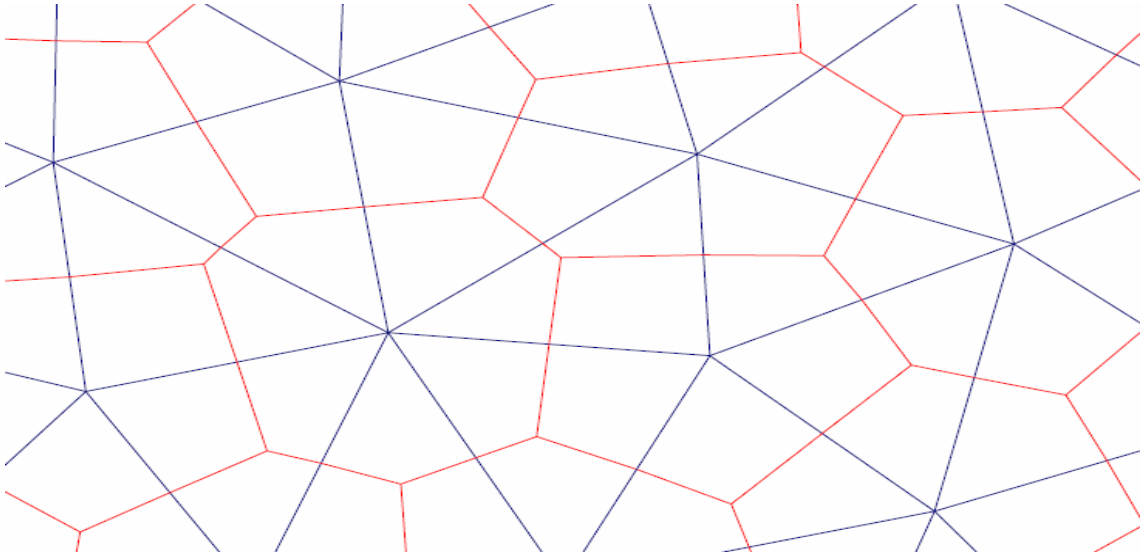
- Simple
- Rigorous

[Harrisson], [Mercat], [Hirani], [Arnold], [Desbrun]

Based on k-forms

II DEC formulation

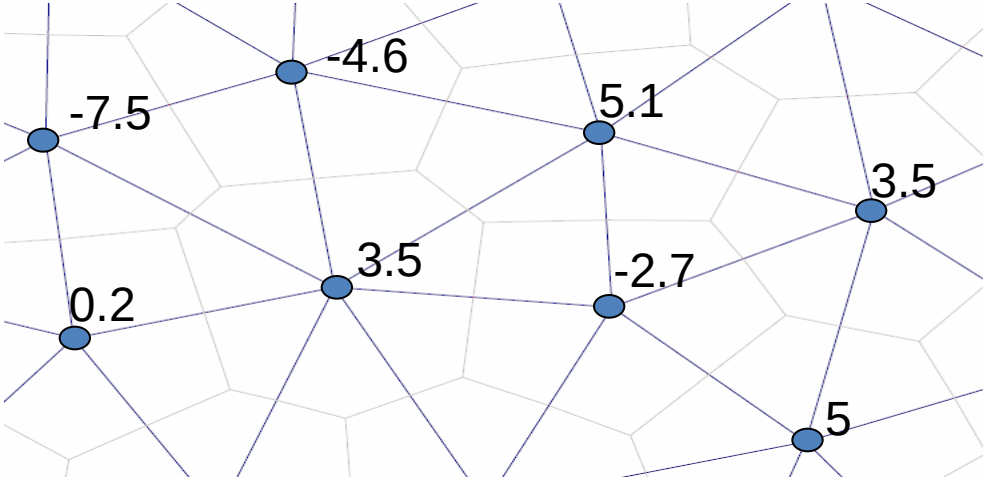
k-forms



mesh
dual mesh

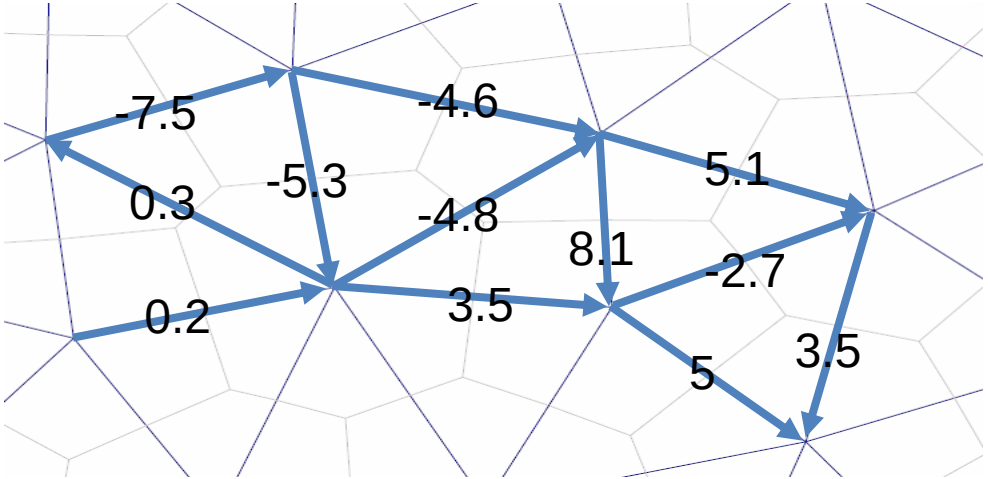
II DEC formulation

0-forms



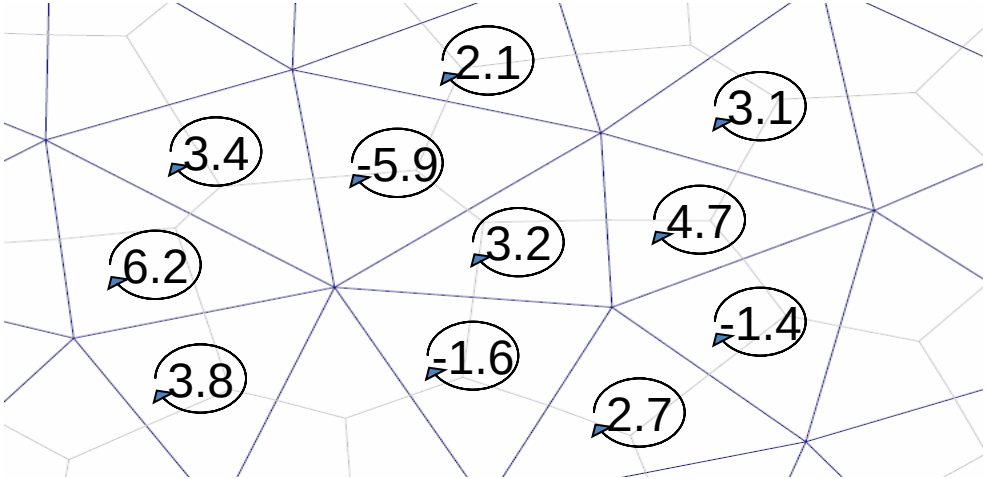
II DEC formulation

1-forms



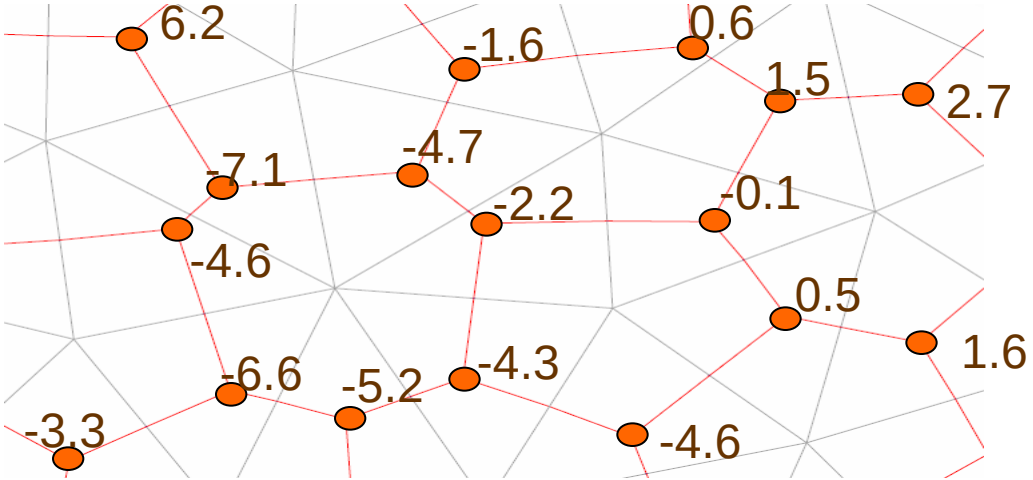
II DEC formulation

2-forms



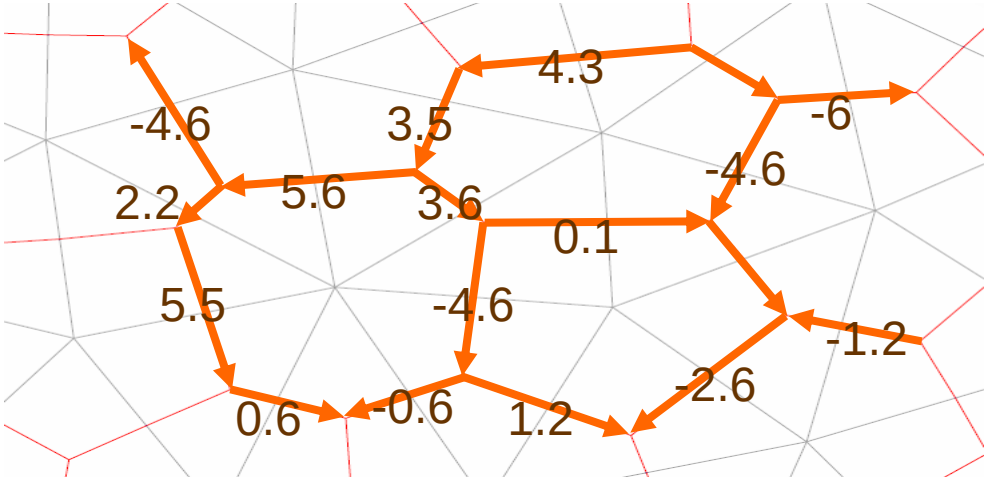
II DEC formulation

dual 0-forms



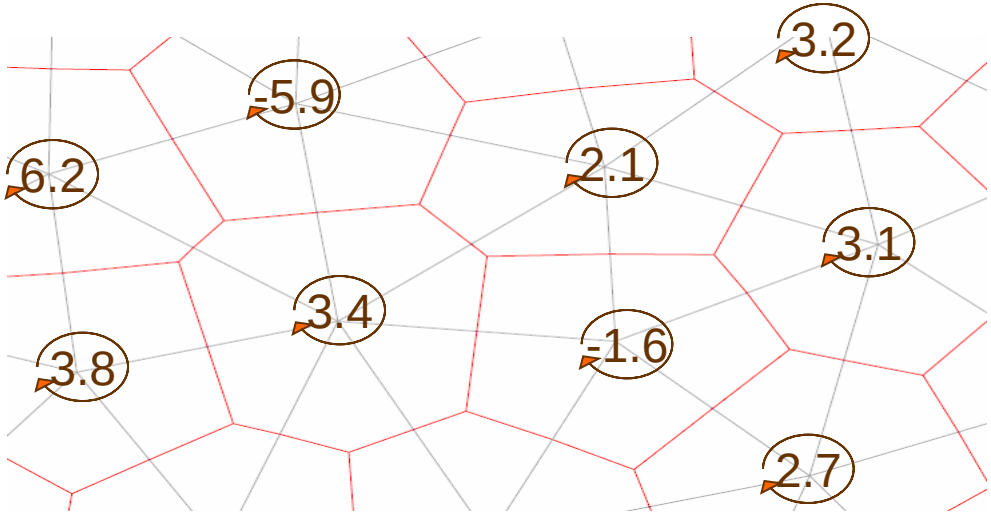
II DEC formulation

dual 1-forms



II DEC formulation

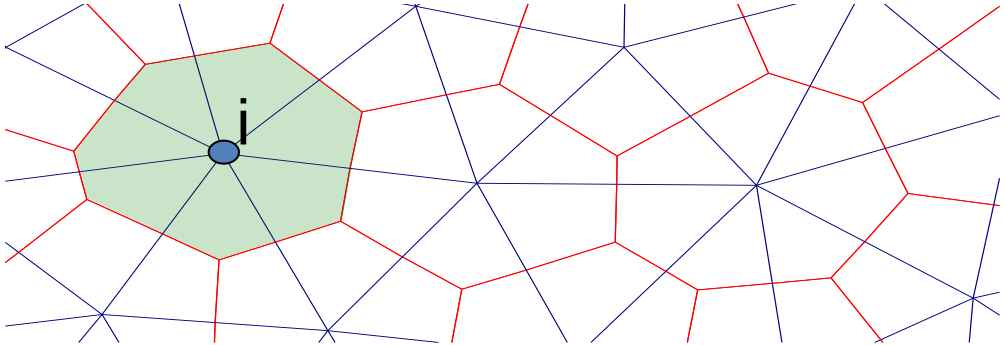
dual 2-forms



II DEC formulation

Hodge star $*_0$

from	to	term
0-forms	dual 2-forms	$\|*i\|$

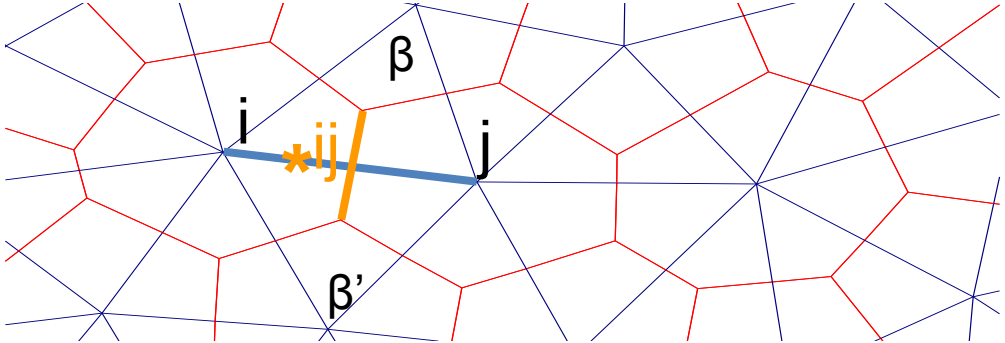


mesh
dual mesh

II DEC formulation

Hodge star $*_1$

from	to	term
1-forms	dual 1-forms	$\|*ij\|/\|ij\| = \cot(\beta)+\cot(\beta')$



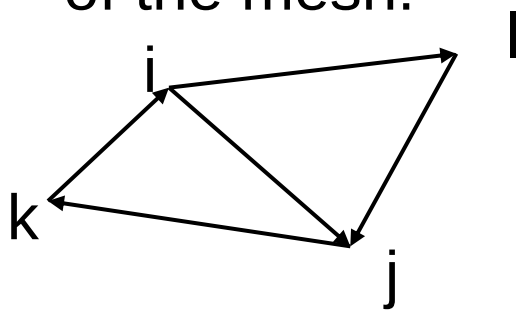
mesh
dual mesh

II DEC formulation

Exterior derivative d

from	to	term
0-forms	1-forms	$df(ij) = f_i - f_j$

Oriented connectivity of the mesh:



d	i	j	k	l
ij	-1	+1	0	0
jk	0	-1	+1	0
ki	+1	0	-1	0
il	-1	0	0	+1
lj	0	+1	0	-1

f
f_i
f_j
f_k
f_l

II DEC formulation

DEC Laplacian

In DEC the Laplacian is $\star_0^{-1}\, d^T\, \star_1\, d$

0-form (function) **f**

II DEC formulation

DEC Laplacian

In DEC the Laplacian is $*_0^{-1}\, d^T\, *_1\, d$

0-form (function) **f** $\xrightarrow{d}$ 1-form (gradient) d**f**

$(\mathbf{f}_j - \mathbf{f}_i)$

II DEC formulation

DEC Laplacian

In DEC the Laplacian is $*_0^{-1}\, d^T\, *_1\, d$

0-form (function) **f**

$\xrightarrow{d}$

1-form (gradient) d**f**

$(\cot(\beta)+\cot(\beta'))(f_j-f_i)$

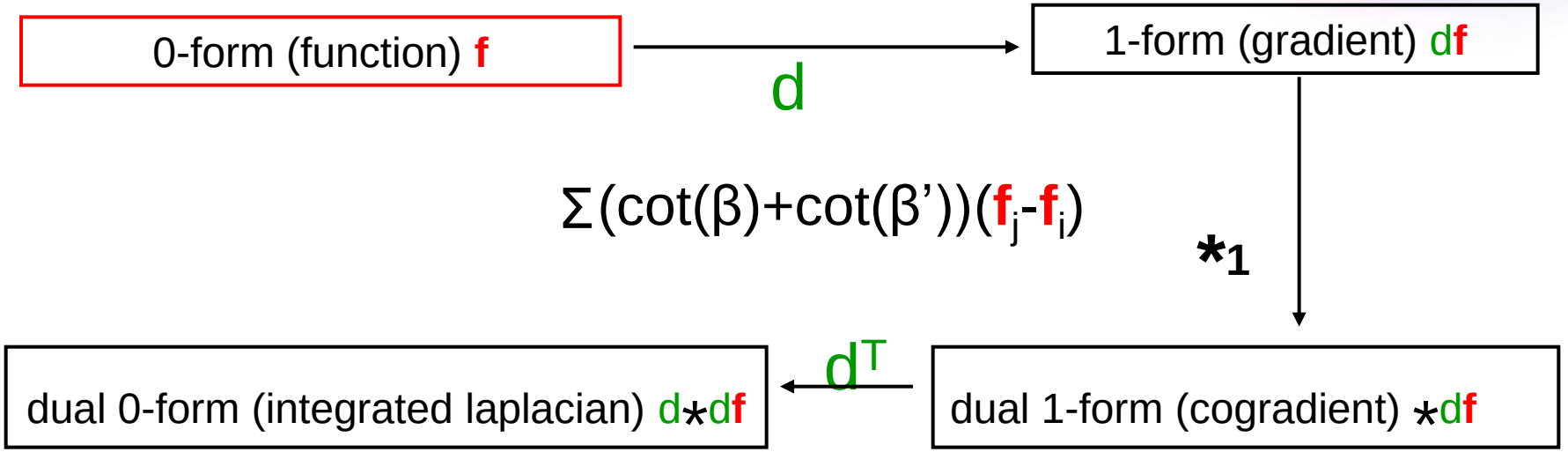
$\downarrow *_1$

dual 1-form (cogradient) $*$d**f**

II DEC formulation

DEC Laplacian

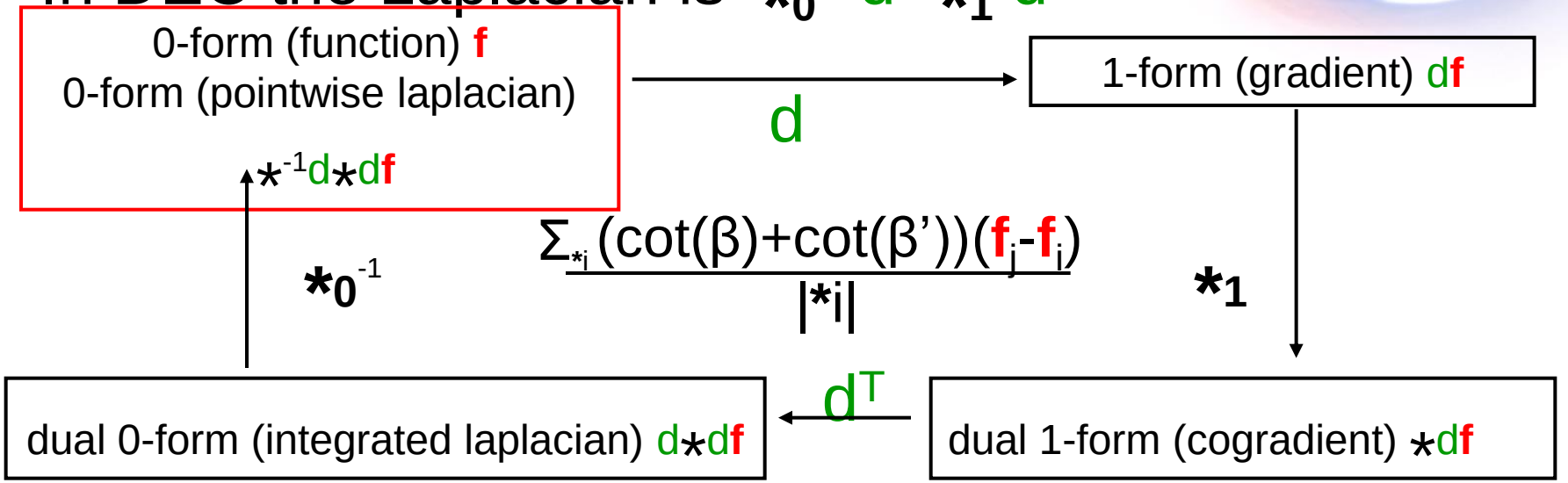
In DEC the Laplacian is $*_0^{-1}\, d^T\, *_1\, d$



II DEC formulation

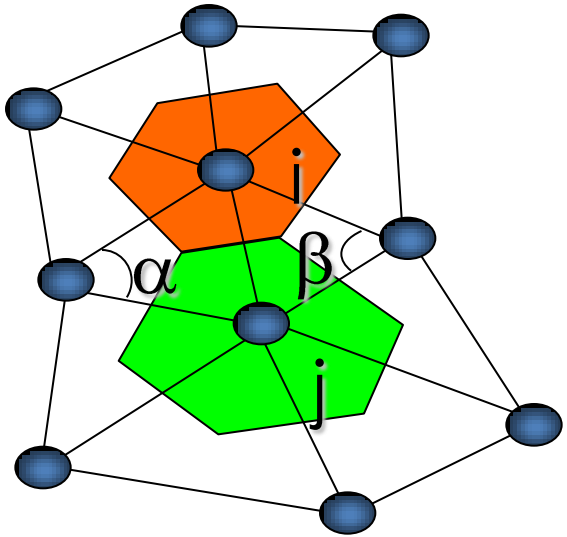
DEC Laplacian

In DEC the Laplacian is $\star_0^{-1}\, d^T \star_1 d$



II DEC formulation

DEC Laplacian



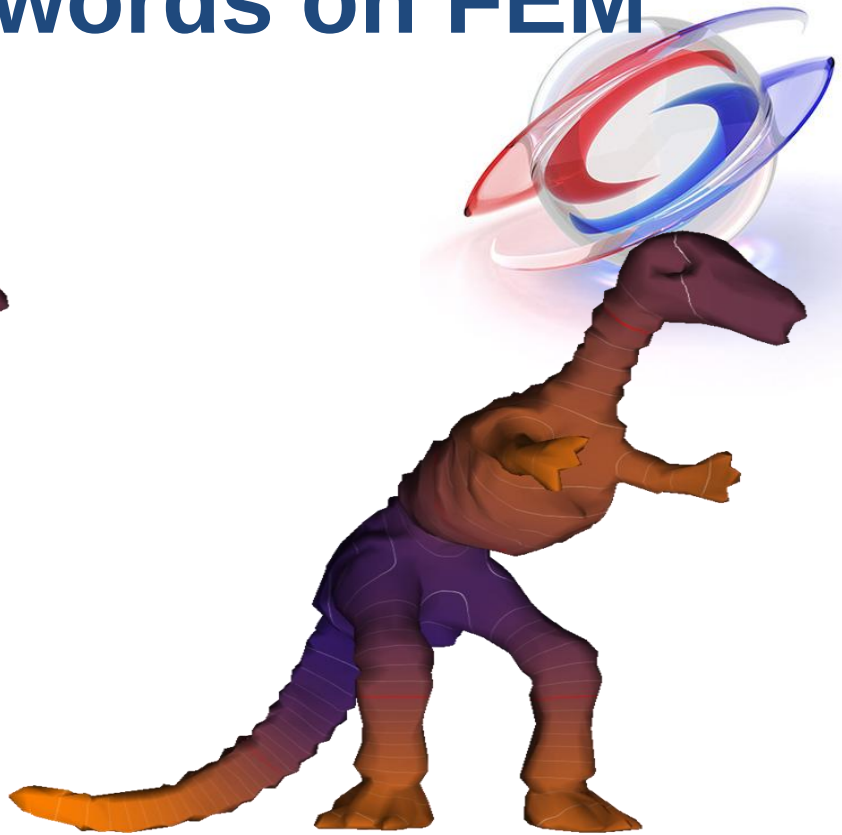
$$a_{ij} = 2\,(\cot\alpha + \cot\beta)\,/\sqrt{(A_i\ A_j)}$$

II Two words on FEM

FEM Laplacian

[Reuter & Wolver],
[Reuter & Spagnuolo],
[Vallet & L]
See course notes

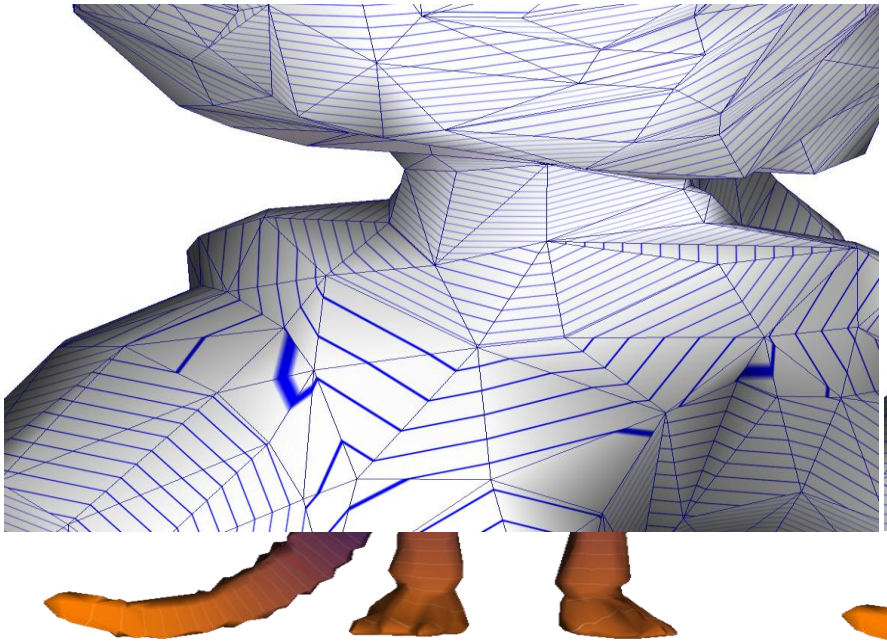
P1 function basis

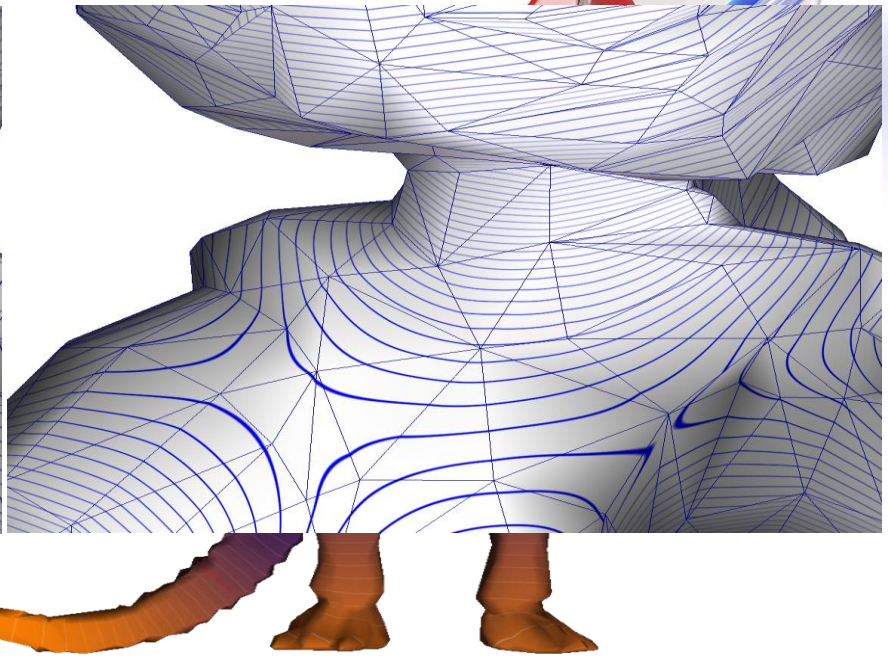


P3 function basis

II Two words on FEM

FEM Laplacian



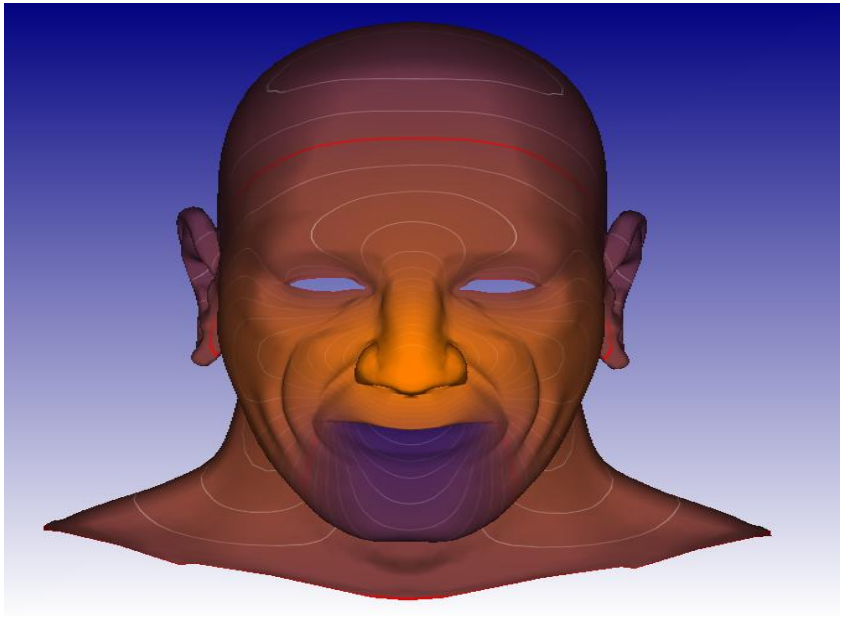


P1 function basis

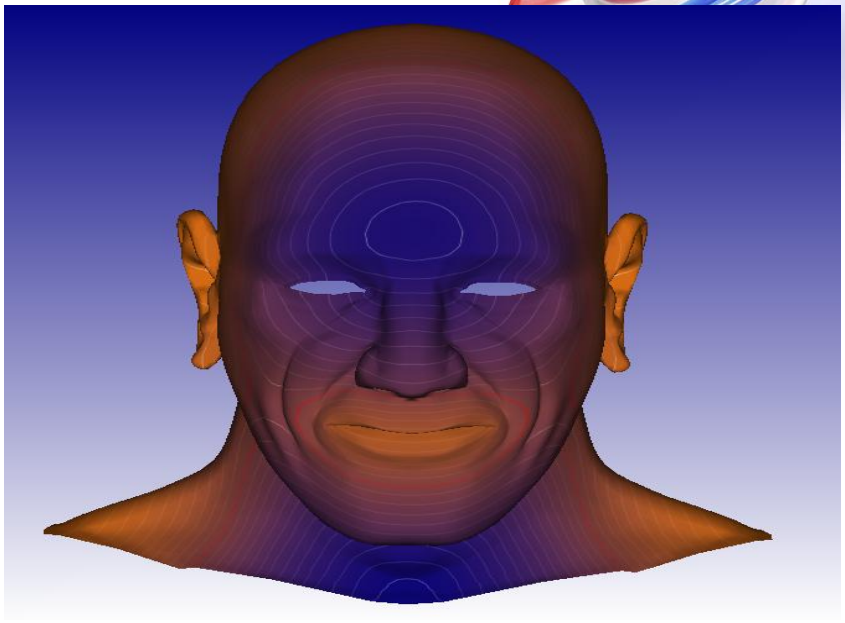
P3 function basis

II Two words on FEM

Boundary conditions



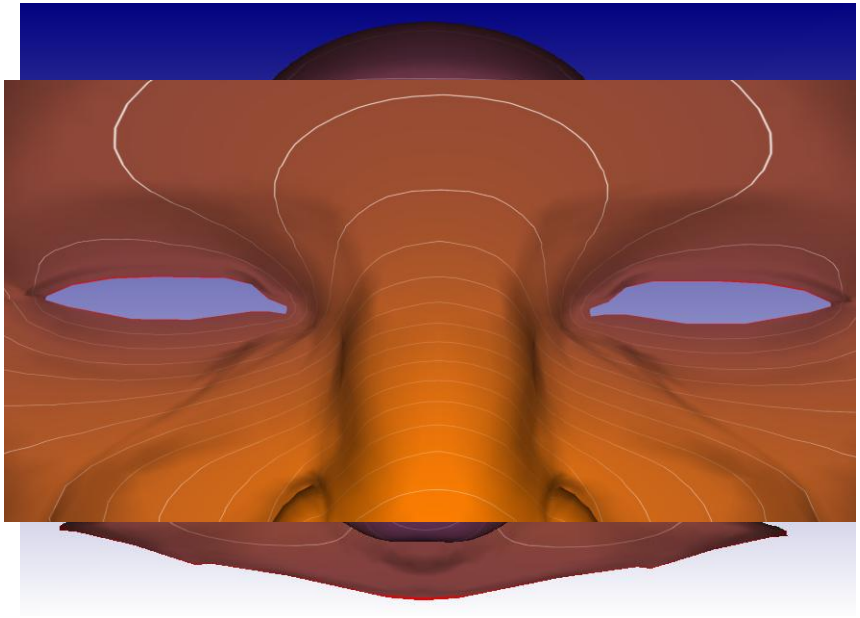
Dirichlet



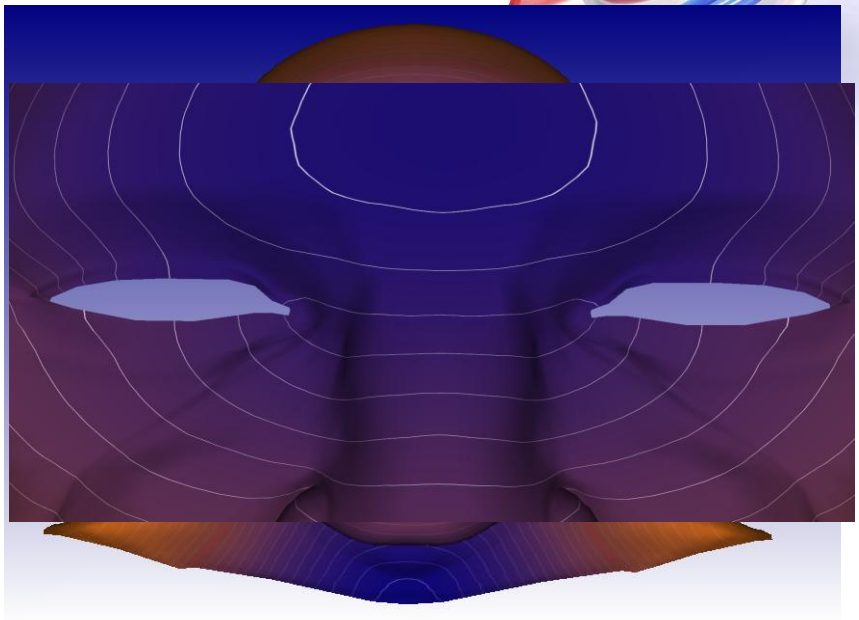
Neumann

II Two words on FEM

Boundary conditions



Dirichlet



Neumann

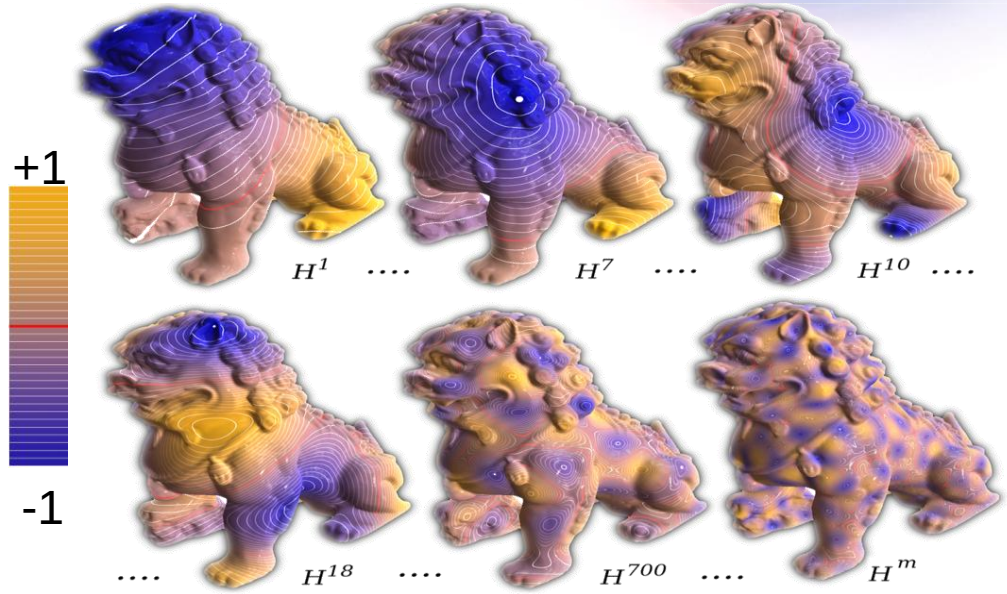
II DEC formulation

Manifold Harmonics Basis

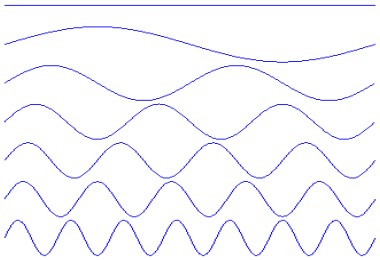
Eigenfunctions of operator Δ

↓ **FEM or DEC**

Eigenvectors of matrix L



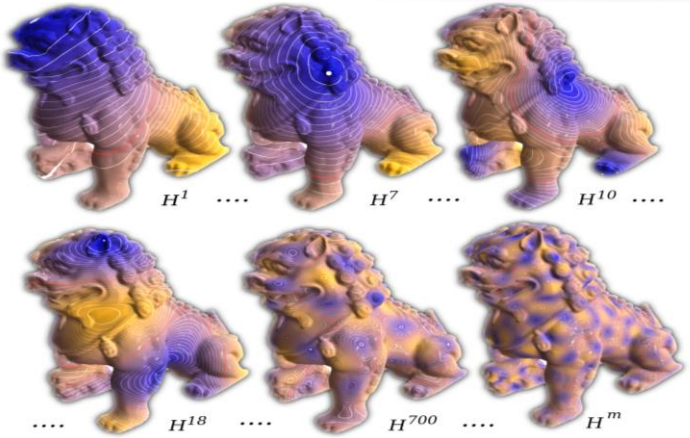
II DEC formulation : recap



sin(kx)

on

=



III Filtering

Introduction

I. Harmonics

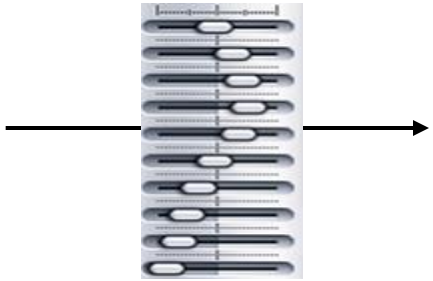
II. DEC formulation

III. Filtering

IV. Numerics

Results and conclusion

III Filtering

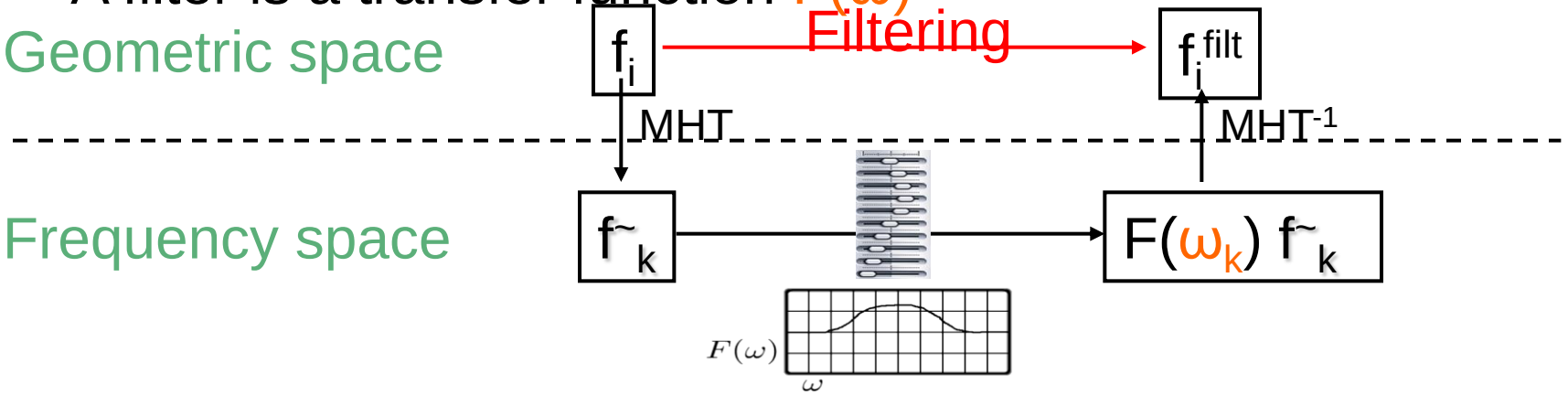


?

III Filtering

Spectral Filtering

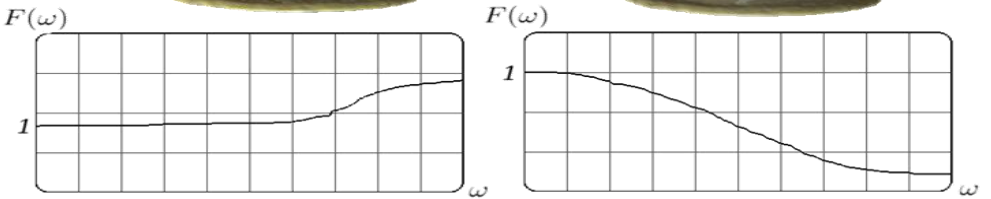
- The Manifold Harmonics H^k come with an eigenvalue λ_k
- The $\lambda_k = \omega_k^2$ is a squared spatial frequency
- A filter is a transfer function $F(\omega)$



III Filtering

Color Filtering

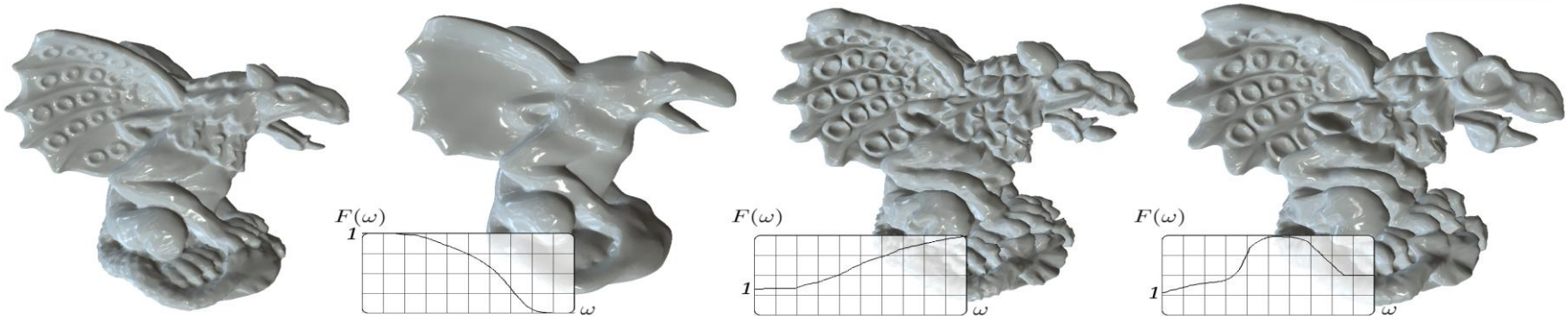
Take **f** =(r,g,b)



III Filtering

Geometry Filtering - DEMO

Take **f** =(x,y,z)



IV Numerics

Introduction

I. Harmonics

II. DEC formulation

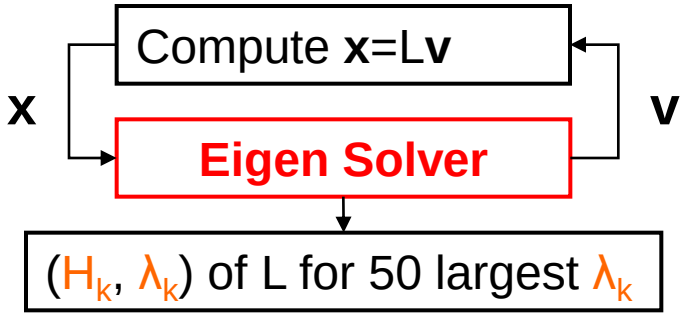
III. Filtering

IV. Numerics

IV Numerics

Eigenvalues

- Compute the eigenpairs (H_k, λ_k) of L = $\star_0^{-1} d^T \star_1 d$
- Solver returns eigenvectors of **highest** eigenvalue



Problems:

- (1) We want smallest λ_k
- (2) We want more than 50

IV Numerics

Eigenvalues

- Problem #1: Eigensolver returns **highest** eigenvalues, we want **smallest**

Use **invert** spectral transform

- $L\,H_k = \lambda_k\,H_k$
- $L^{-1}\,H_k = \lambda_k^{-1}\,H_k$

We do not need to invert L, factorizing it is sufficient. Each time the eigensolver queries $\mathbf{x} = L\mathbf{v}$, solve for $\mathbf{x}$ in $L\mathbf{x} = \mathbf{v}$ instead

IV Numerics

Eigenvalues

- Problem #2: We want more than 50 eigenpairs

  Use **shift** spectral transform

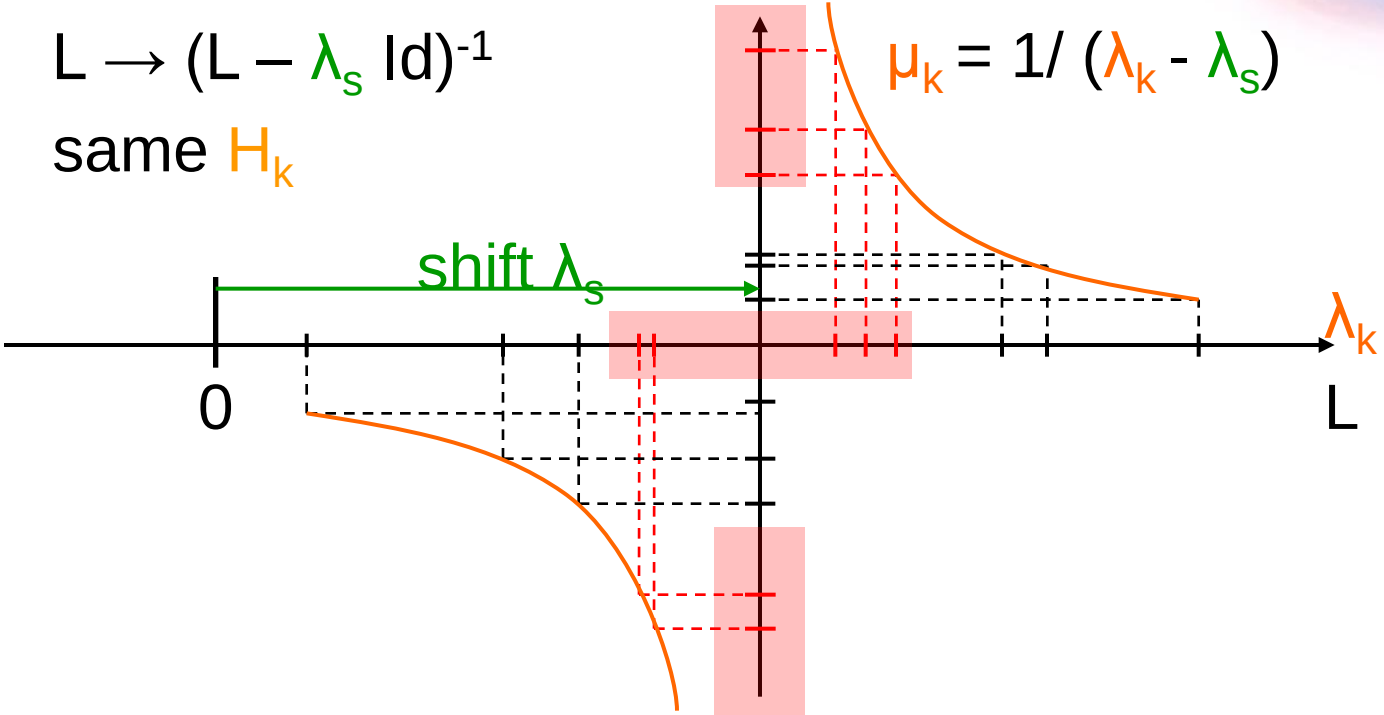
- $L\,H_k = \lambda_k\,H_k$
- $(L - \lambda_s\,Id)\,H_k = (\lambda_k - \lambda_s)\,H_k$

We do not need to invert L, factorizing it is sufficient. Each time the eigensolver queries **x** = L**v**, solve for **x** in L**x** = **v** instead

IV Numerics

Combining: Shift Invert

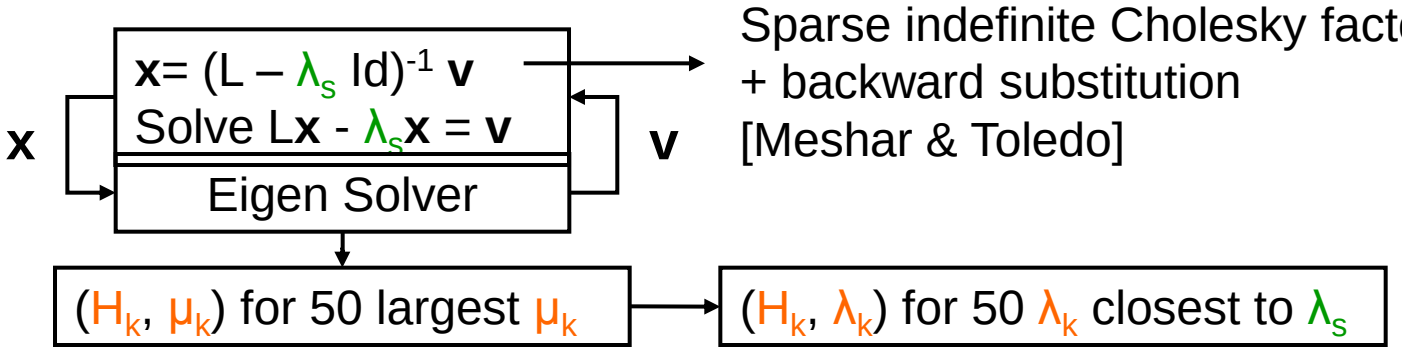
- $L \rightarrow (L - \lambda_s\ Id)^{-1}$
- same H_k



IV Numerics

Eigen solver

Compute a **band** of eigenpairs (H^k,λ^k) around λ_s

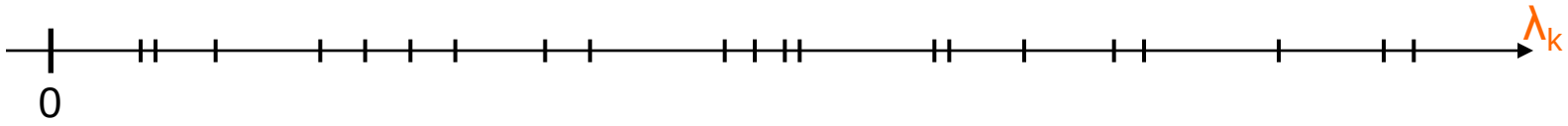


Sparse indefinite Cholesky factorization
+ backward substitution
[Meshar & Toledo]

IV Numerics

Band by band algorithm

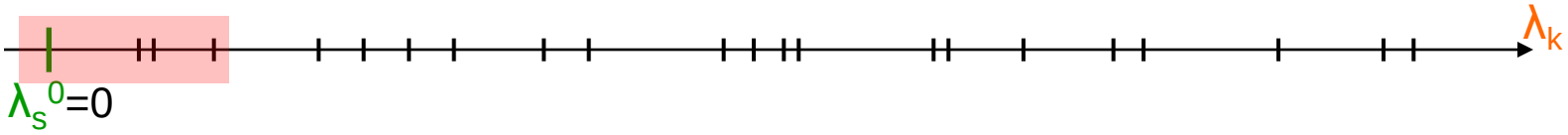
Compute the eigenpairs (h^k,λ^k) of L **band by band**



IV Numerics

Band by band algorithm

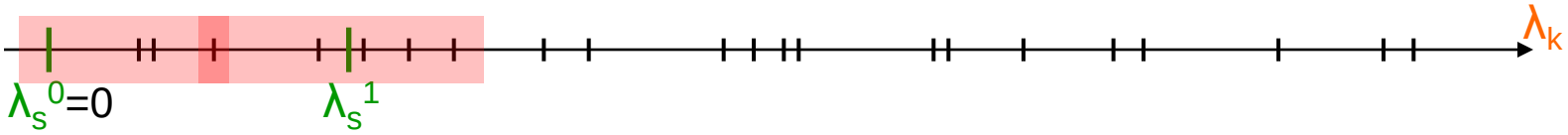
Compute the eigenpairs (h^k,λ^k) of L **band by band**



IV Numerics

Band by band algorithm

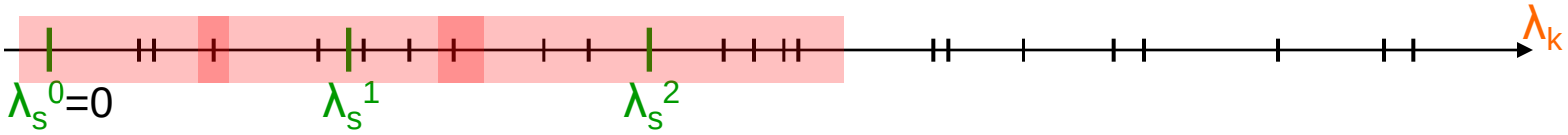
Compute the eigenpairs (h^k,λ^k) of L **band by band**



IV Numerics

Band by band algorithm

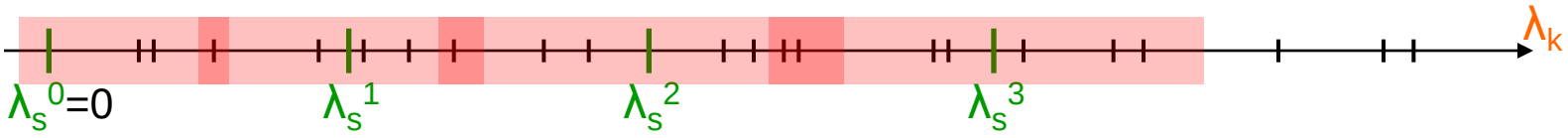
Compute the eigenpairs (h^k,λ^k) of L **band by band**



IV Numerics

Band by band algorithm

Compute the eigenpairs (h^k,λ^k) of L **band by band**



IV Numerics

Band by band algorithm

Compute the eigenpairs (h^k,λ^k) of L **band by band**

IV Numerics - Cookbook

Putting everything together

- (1) compute the discrete Laplacian L

$$a_{ij} = 2\,(\cot\alpha + \cot\beta) \,/\, (A_i \;\; A_j)$$

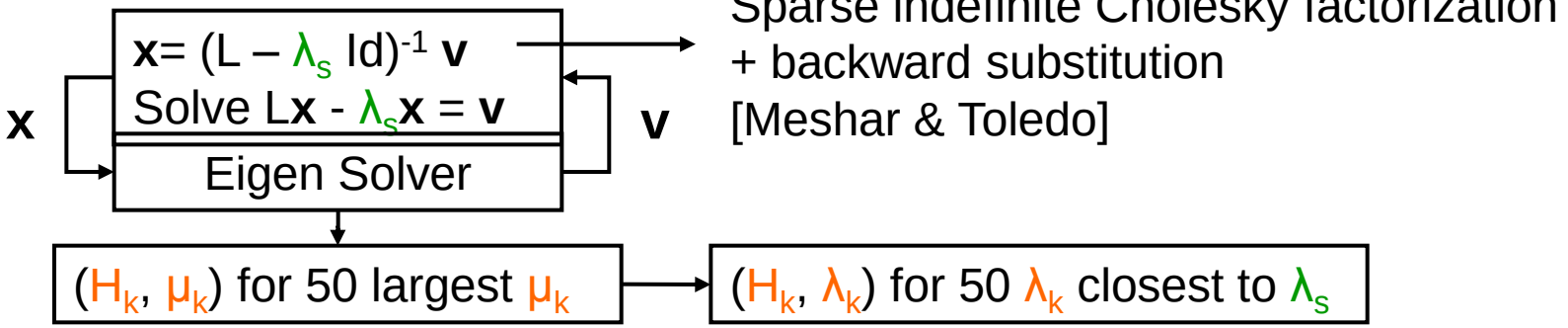
- (2) for each frequency band centered on λ_s
- Factorize ($L - \lambda_s$ Id) using a sparse direct solver (TAUCS, SuperLU …)
- Each time the eigensolver queries a matrix-vector product $\mathbf{x} = M\mathbf{v}$,

  solve $L\mathbf{x} - \lambda_s\mathbf{x} = \mathbf{v}$ instead

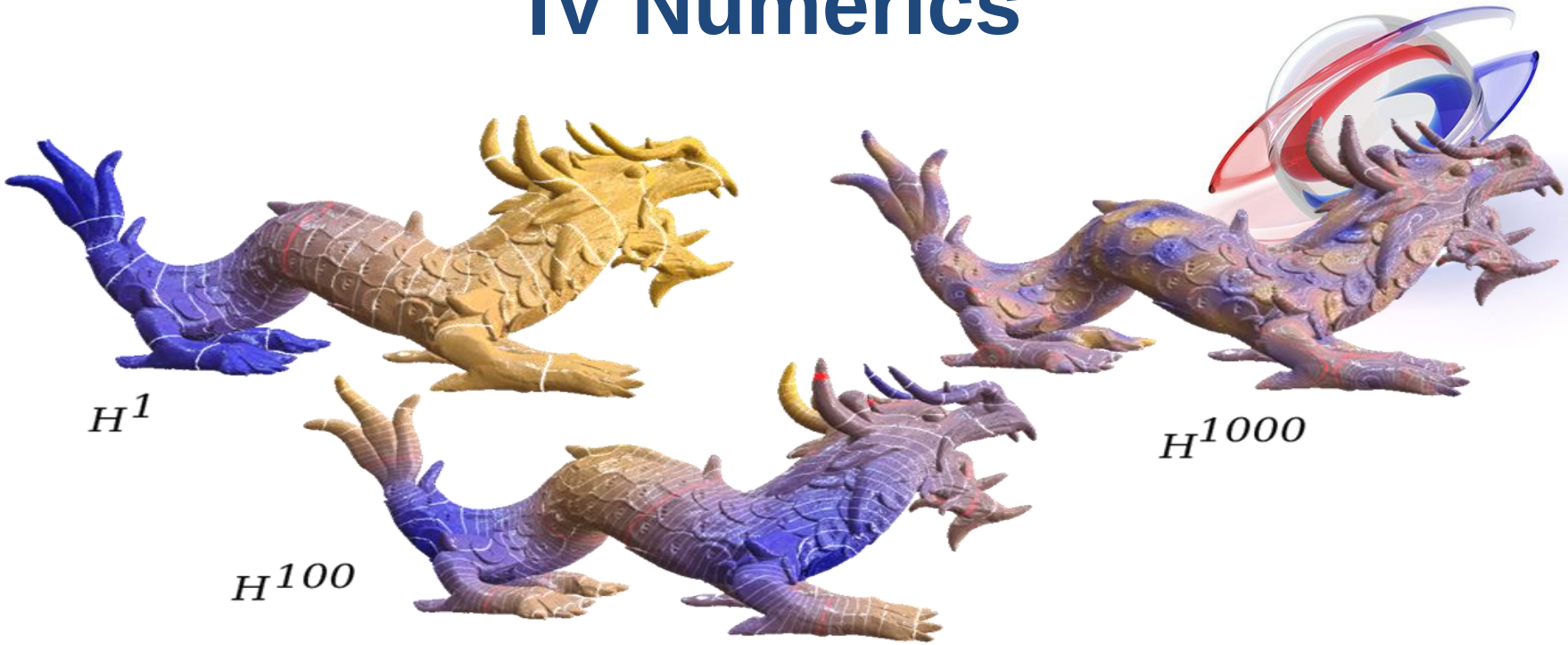
IV Numerics

Eigen solver

Main loop: ARPACK (Arnoldi solver)
Matrix factorization: SuperLU or TAUCS
Full Implementation: http://alice.loria.fr/software/graphite
http://alice.loria.fr/WIKI/index.php/Graphite/SpectralMeshProcessing



IV Numerics



1 million vertices, 1000 eigenfunctions
ARPACK + TAUCS + shift-invert

SIGGRAPH 2010

“*Spectral Mesh Processing*”

Bruno Lévy and Richard Hao Zhang

Applications I

Spectral Mesh Processing

Hao Zhang
School of Computing Science
Simon Fraser University, Canada

Recap

- Spectral mesh processing

Use eigenstructures of appropriately defined linear mesh operators for geometry analysis and processing

Solve a problem in a different domain via a transform — spectral transform

Fourier analysis on meshes

Captures global and intrinsic shape characteristics

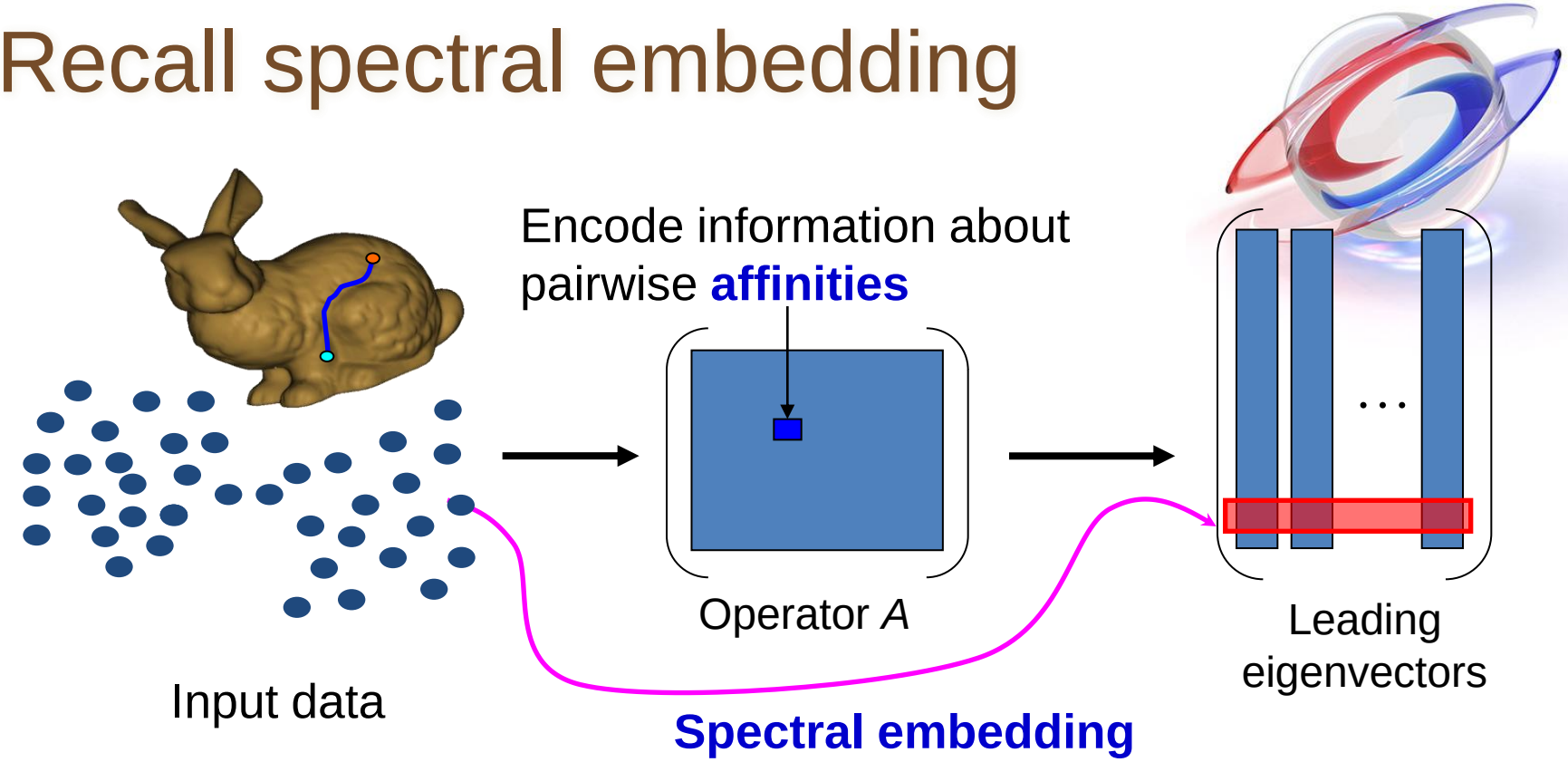
Dimensionality reduction: effective and simplifying

Applications covered

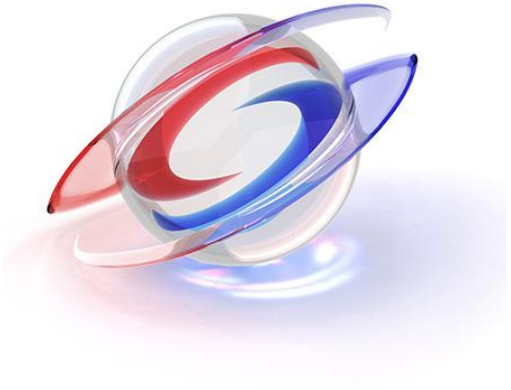
- Mesh segmentation
- Non-rigid correspondence
- Shape retrieval
- Global intrinsic symmetry detection

All utilize spectral embeddings in some way

Recall spectral embedding



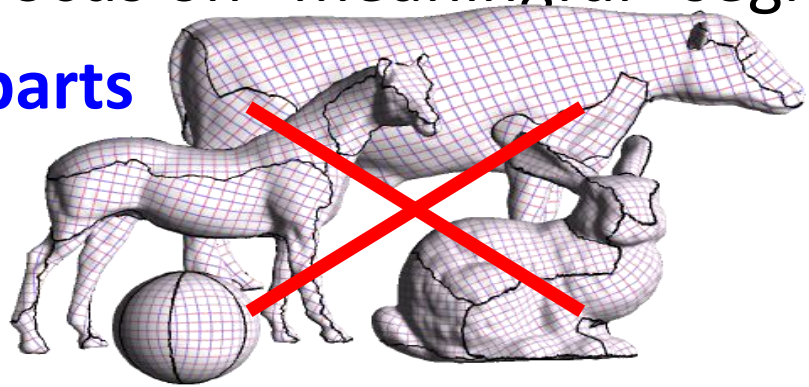
Applications covered

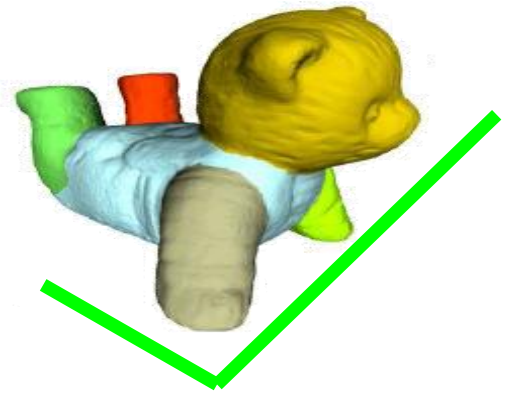


- Mesh segmentation
- Non-rigid correspondence
- Shape retrieval
- Global intrinsic symmetry detection

Mesh segmentation

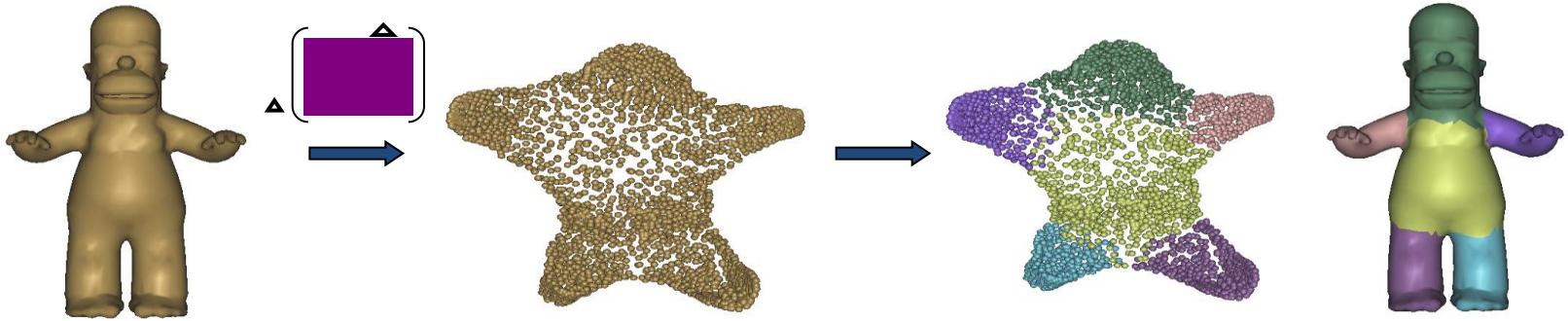
- Lots of recent work; survey by [Shamir 08]
- Focus on “meaningful” segmentation — notion of **parts**





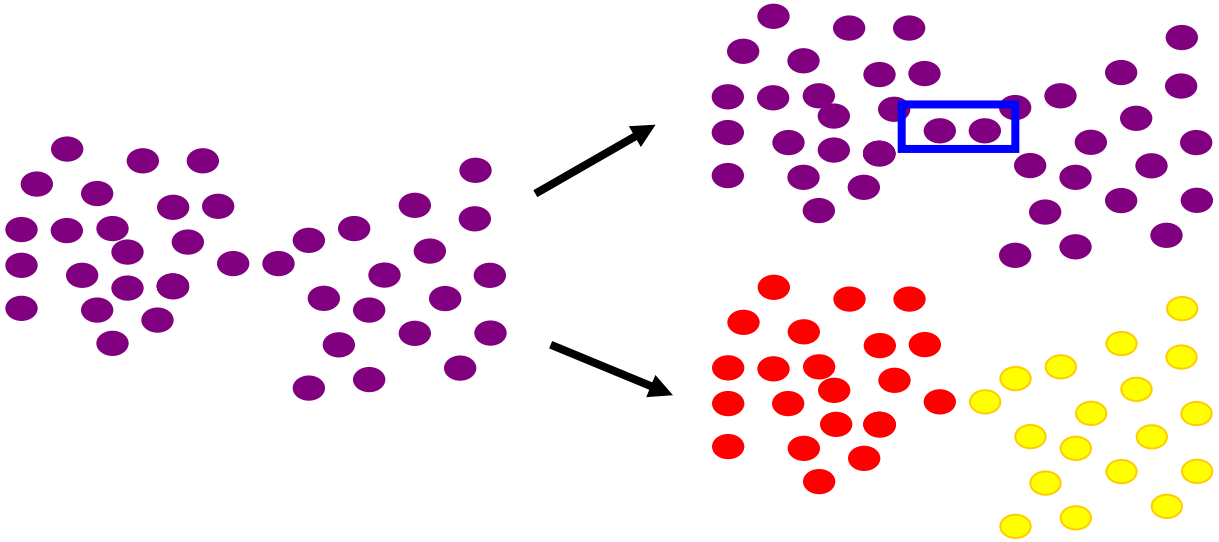
Framework for spectral approach

1. Define appropriate distance between mesh elements
2. Distances to affinities and form an affinity matrix
3. Eigen-decompose matrix to get spectral embedding
4. Cluster/segment in spectral domain, e.g., *k*-means



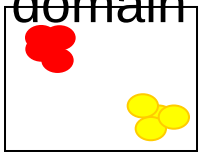
Why spectral?

- Captures global information; structures better revealed in spectral domain



Linkage-based
(local info)

spectral
domain



Spectral clustering

Key issues

- Distance definition
  - **Fundamental to most shape analysis tasks**
  - Ideal: short distance if elements in same part
  - But not clear what a part is …

Key issues

- Distance definition
  - **Fundamental to most shape analysis tasks**
  - Ideal: short distance if elements in same part
  - But not clear what a part is …
- Computation of spectral embeddings
  - Efficiency: Nyström approximation

Key issues

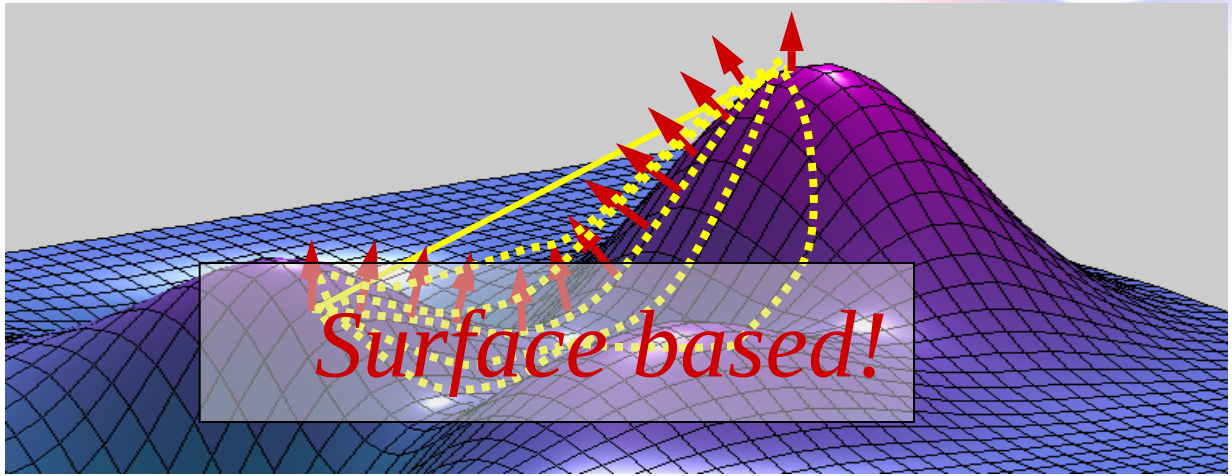
- Distance definition
  - **Fundamental to most shape analysis tasks**
  - Ideal: short distance if elements in same part
  - But not clear what a part is …
- Computation of spectral embeddings
  - Efficiency: Nyström approximation
- How to cluster, depending on DIM of embeddings
  - Higher-D: *k*-means most typical (easier in spectral)
  - 1D: linear search with hard-to-optimize energy

Key issues

- Distance definition
  - **Fundamental to most shape analysis tasks**
  - Ideal: short distance if elements in same part
  - But not clear what a part is …
- Computation of spectral embeddings
  - Efficiency: Nyström approximation
- How to cluster, depending on DIM of embeddings
  - Higher-D: *k*-means most typical (easier in spectral)
  - 1D: linear search with hard-to-optimize energy

Distance measure (a metric)

- Euclidean
- Geodesic
- Isophotic or angular [Pottmann 04, Katz 03]
- Diffusion distance [deGoes 08, Coifman 06]

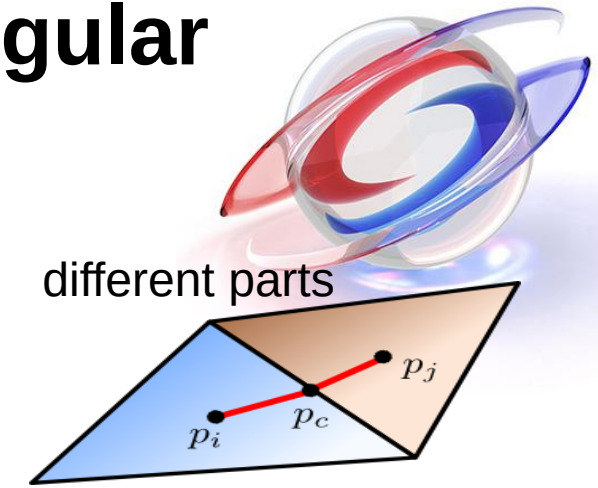


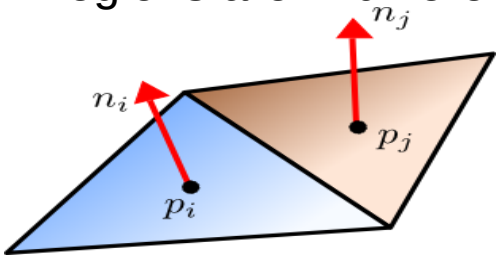
Diffusion and commute time distances

- Both model "**connected-ness**" between points
  - More **global** sense than geodesics
  - Consider more paths between two points
- Diffusion distance
  - Defined by a time/scale parameter *t*
  - Consider only paths of length *t* or less
- Commute time distance: consider paths of all lengths
- Both are still surface-based distances

Most typical: geodesic & angular

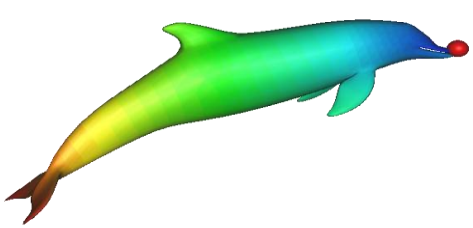
- Geodesic distance (approximate)
  - Distant faces tend to belong to different parts
  - Gestalt law of **proximity**
- Angular distance
  - Faces separated by concave regions are in different parts
  - The **Minima rule**



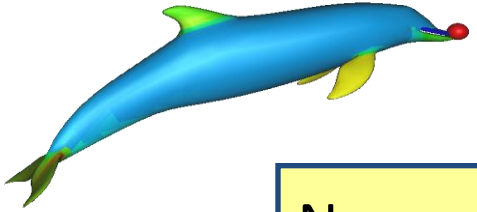


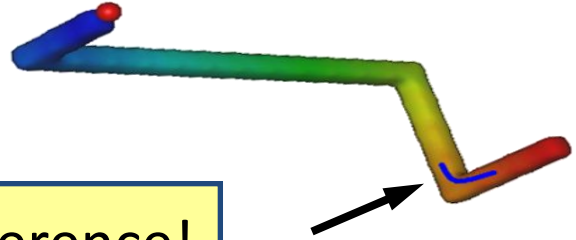
Geodesic & angular

- Geodesic distance: insensitive to parts



- Angular distance: subject to “leakage” problem





No angular difference!

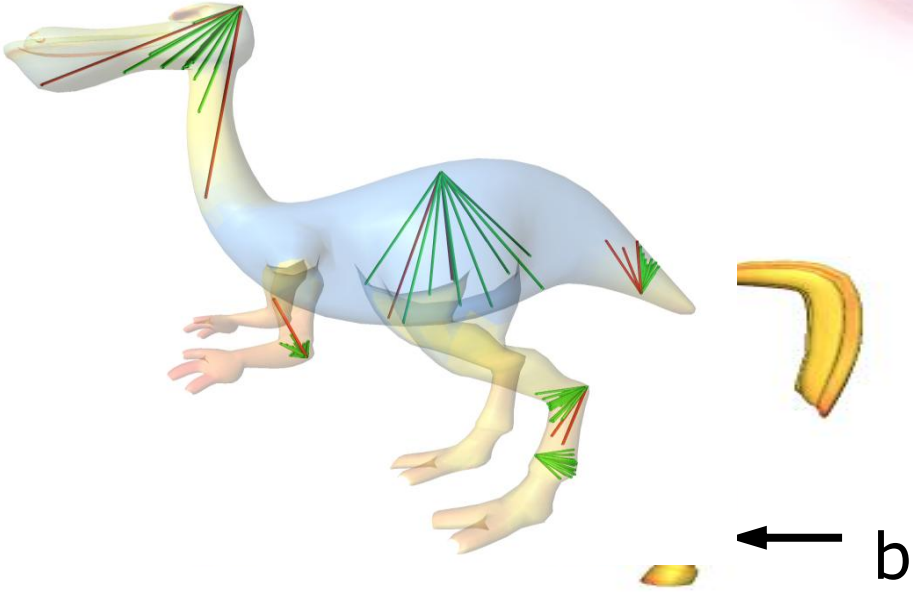
Missing: connect to volumes

Leakage problem not something detectable locally on surface.

Parts are volumetric.

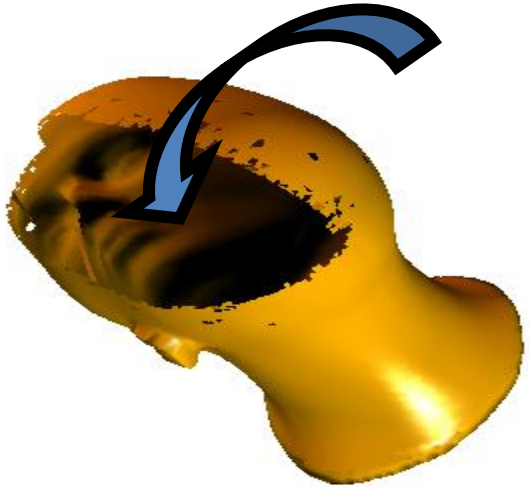
Shape Diameter Function

- Not a metric [Shapira et al. 08]
- Same value or
  - Distance(*a*,*b*



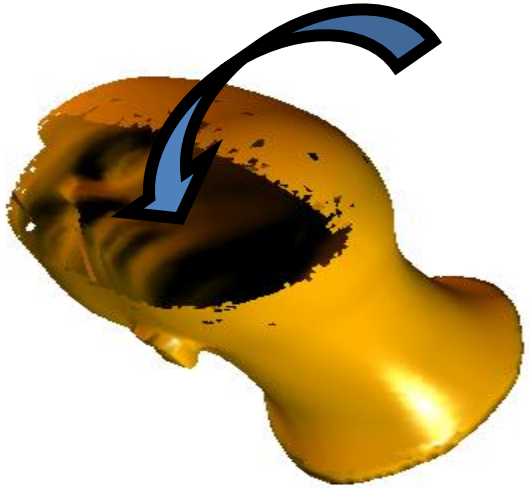
Volumetric view

- Look at the object from inside



Volumetric view

- Look at the object from inside and define a metric

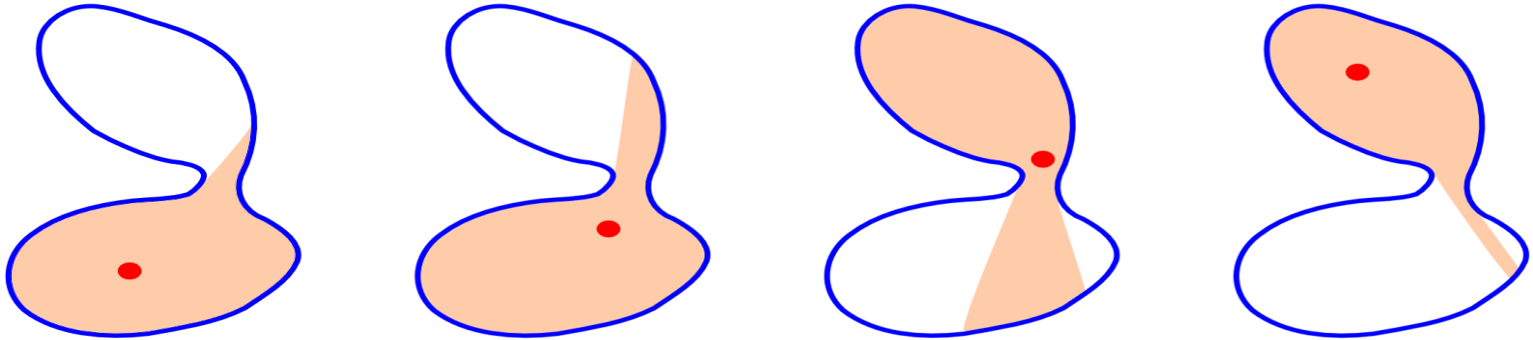


[Liu et al. 2009]

Use of visibility

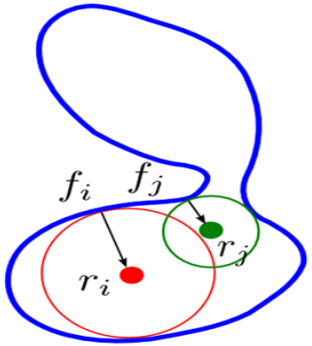
- Intuition: use of **visibility** information

Significant visibility changes across part boundaries



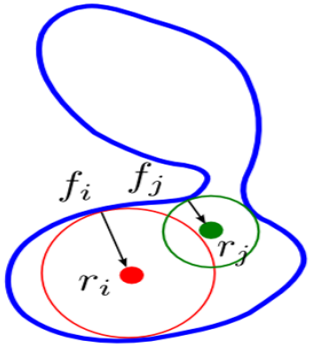
Reference points

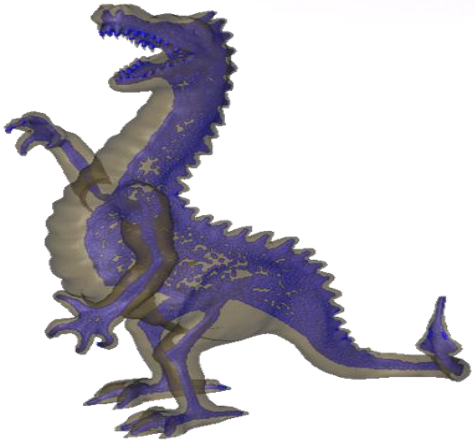
- Need a surface metric, but visibility from surface unstable
- Connect to surface: find reference points



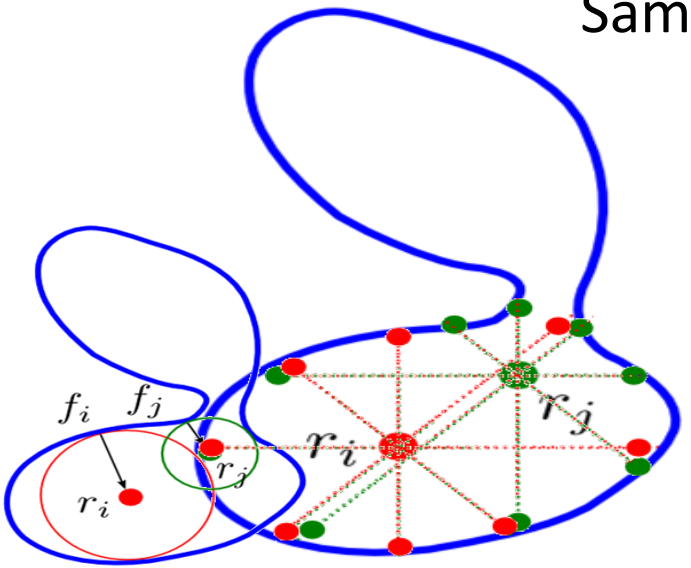
Reference points

- Need a surface metric, but visibility from surface unstable
- Connect to surface: find reference points





Volumetric Shape Images (VSI)



Sample visible regions from ref. points

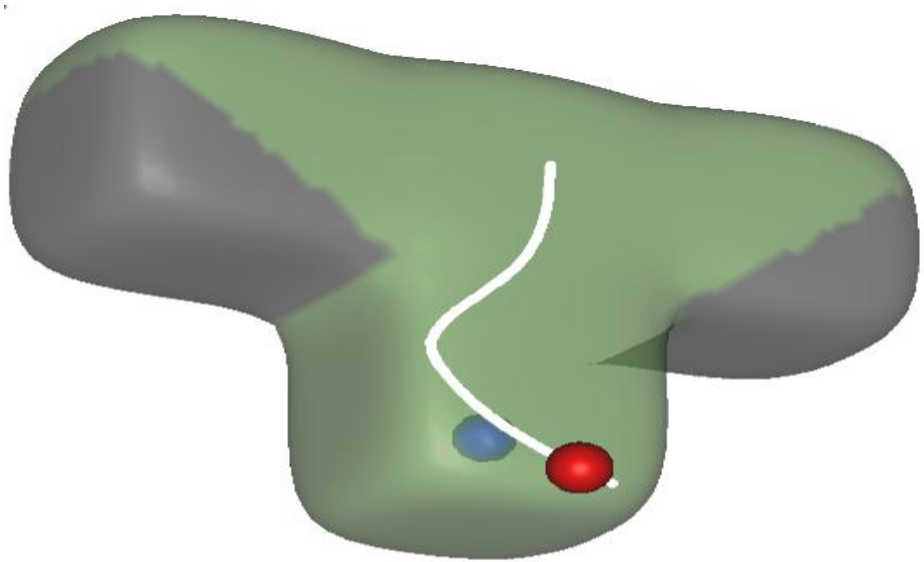
VSI difference

Compute VSI difference

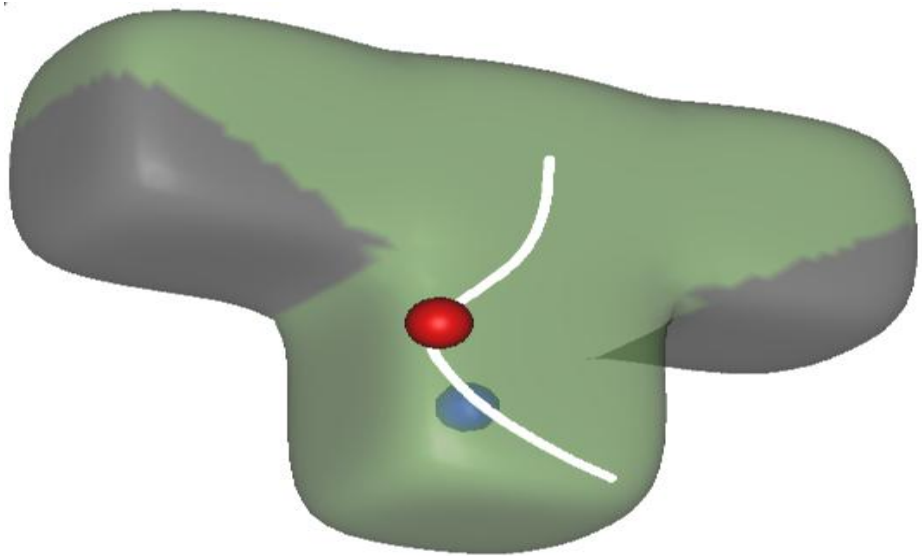
Difference is based on the reach of local volume along sampling direction

$$\mathrm{diff}(S_i, S_j) = \frac{1}{\sum_k w_k} \sum_{k=1}^{m/2} w_k \left(l_i^{(k)} - l_j^{(k)} \right)^2$$

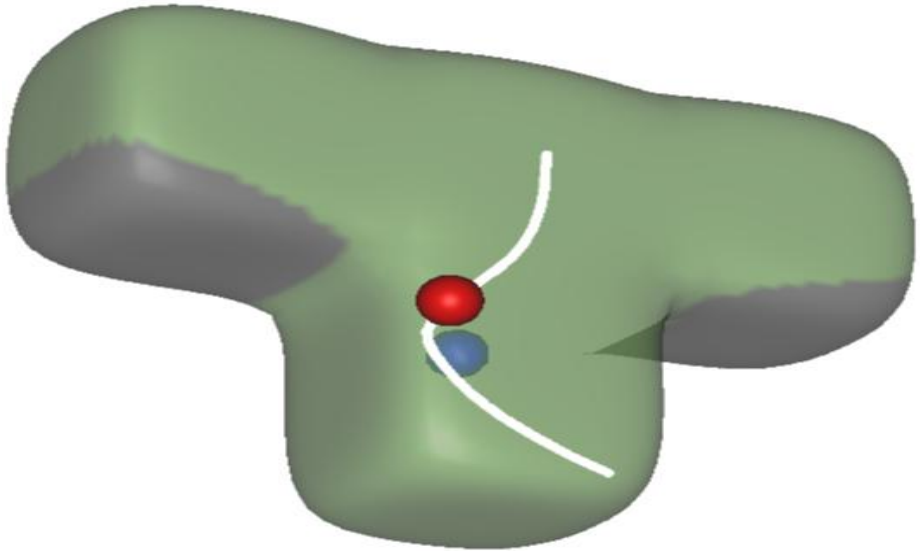
Moving along a path on the surface



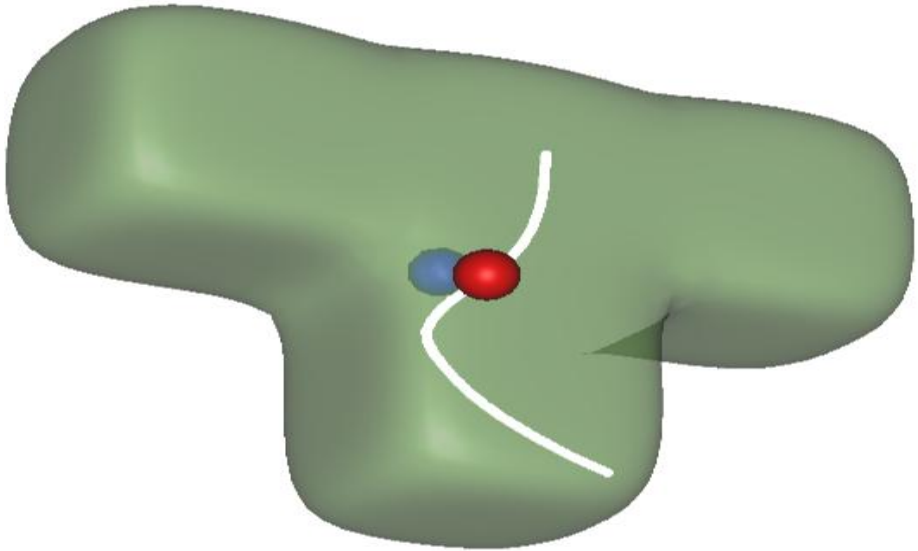
Moving along a path on the surface



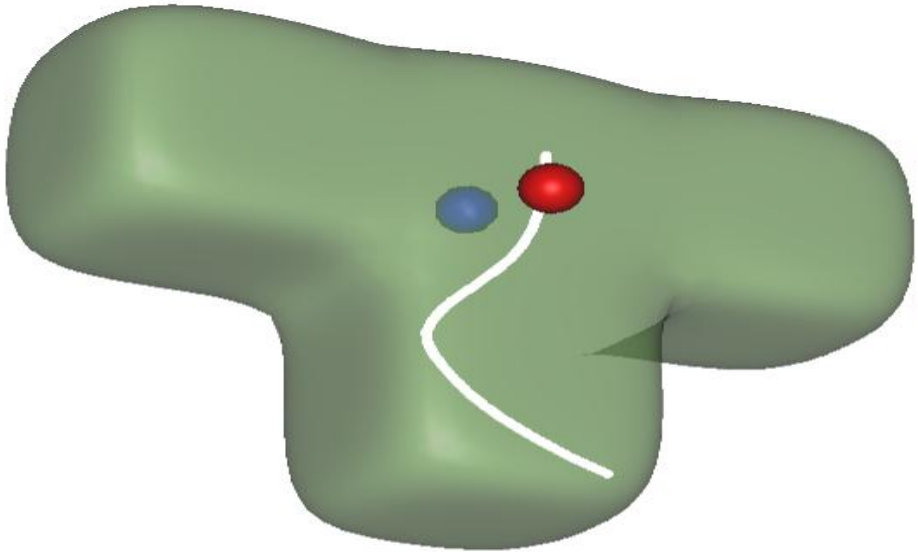
Moving along a path on the surface



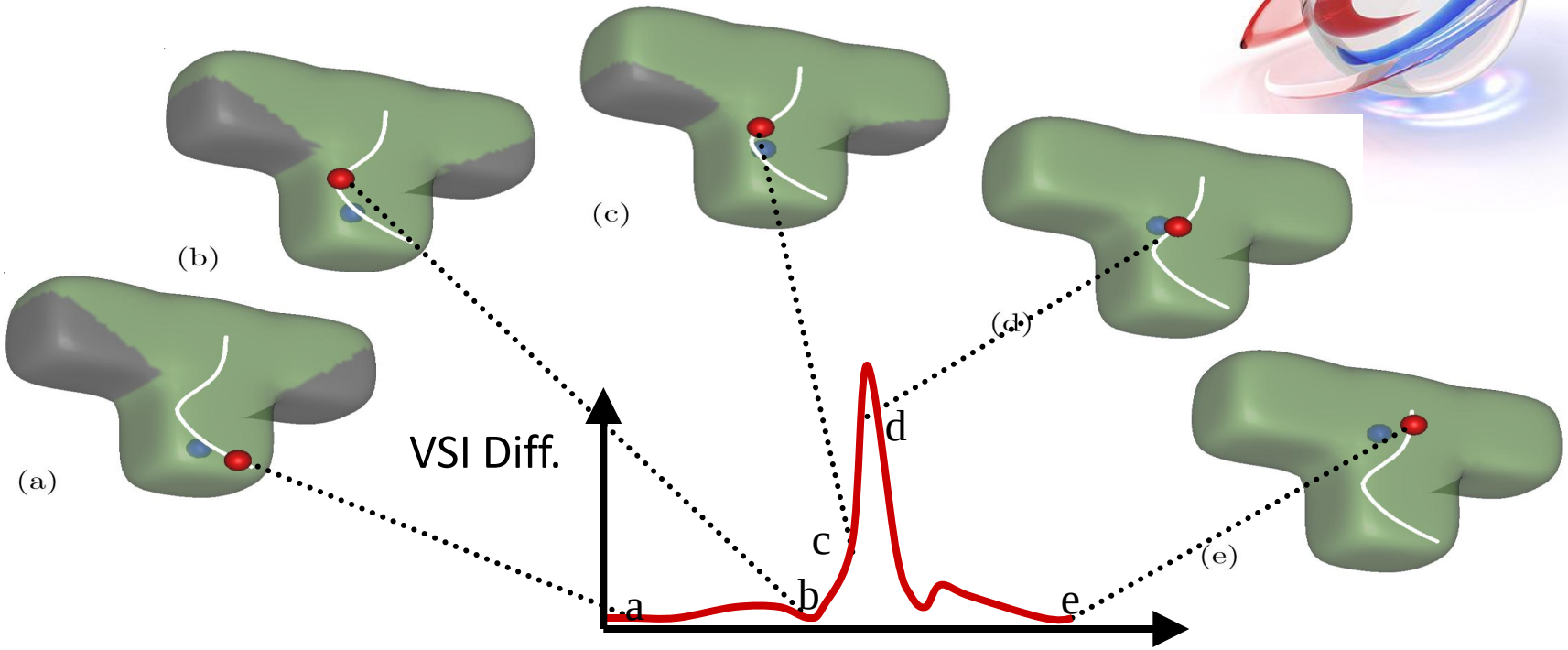
Moving along a path on the surface



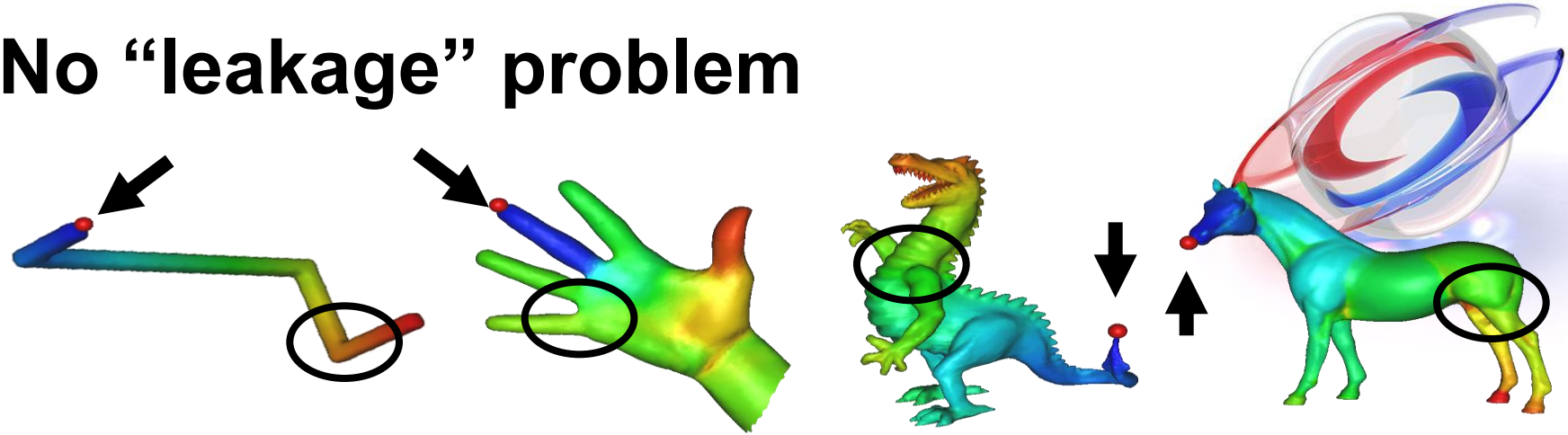
Moving along a path on the surface



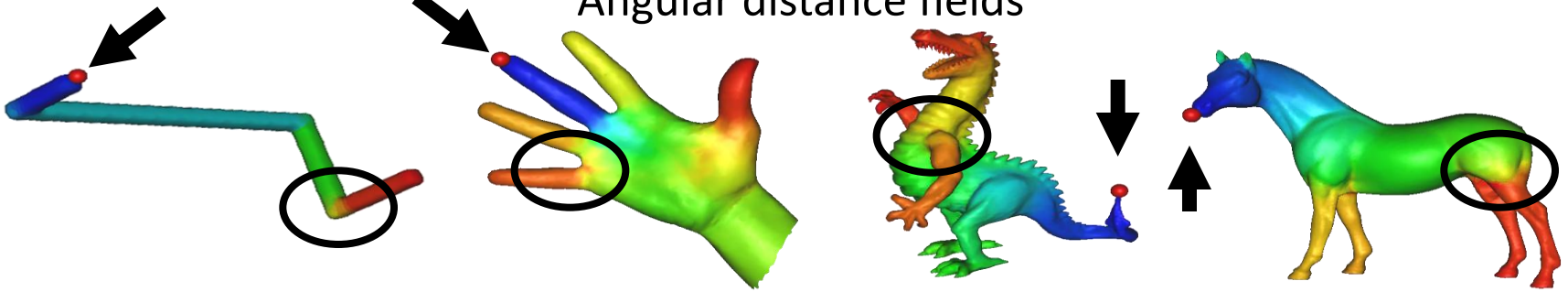
VSI differences along the path



No “leakage” problem



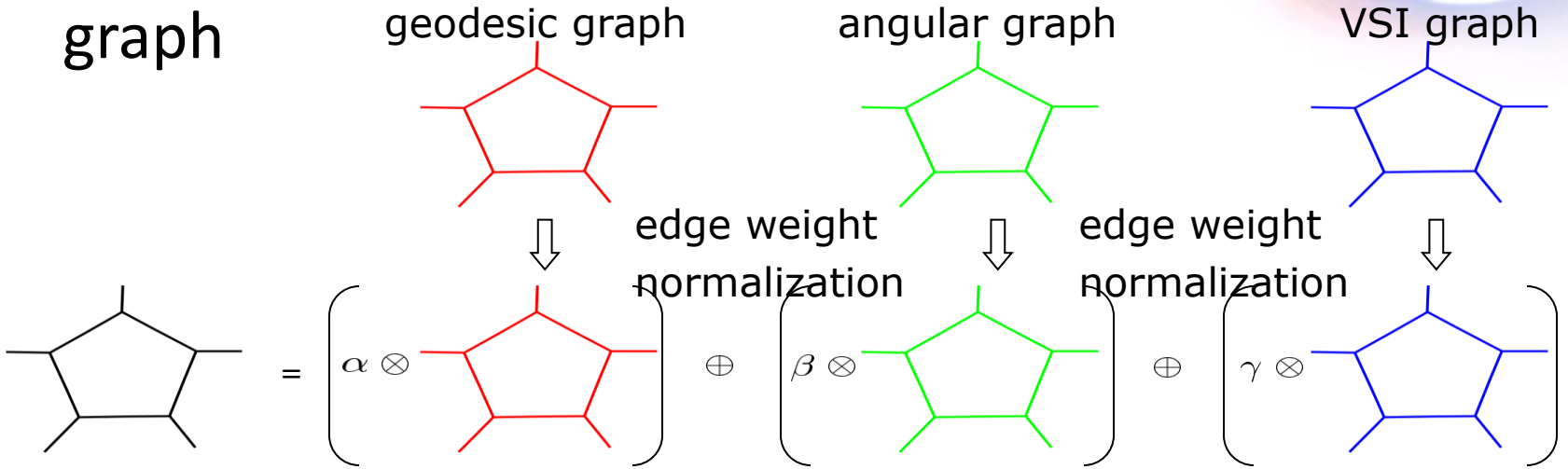
Angular distance fields



VSI distance fields

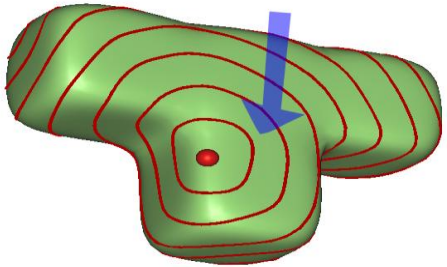
A part-aware metric

- Metric: graph distance on a combined weighted graph

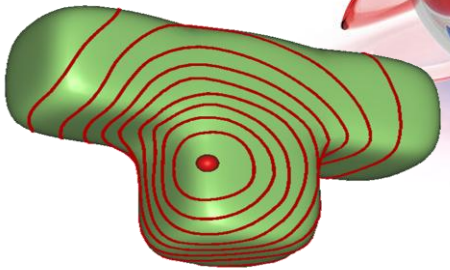


$$\alpha + \beta + \gamma = 1, \quad 0 \leq \alpha, \beta, \gamma \leq 1$$

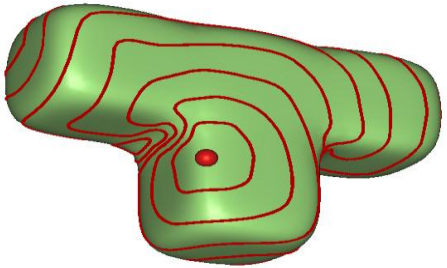
Comparison with other metrics



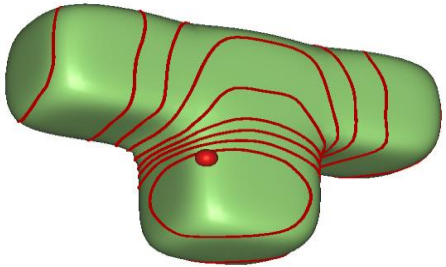
Geodesic



Diffusion [deGoes 08]



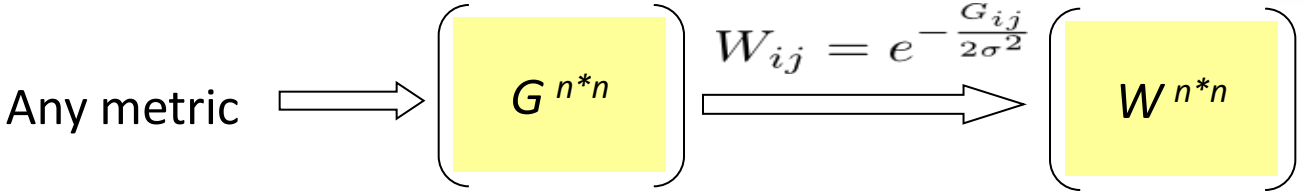
Angular [Katz 03]



Part-aware

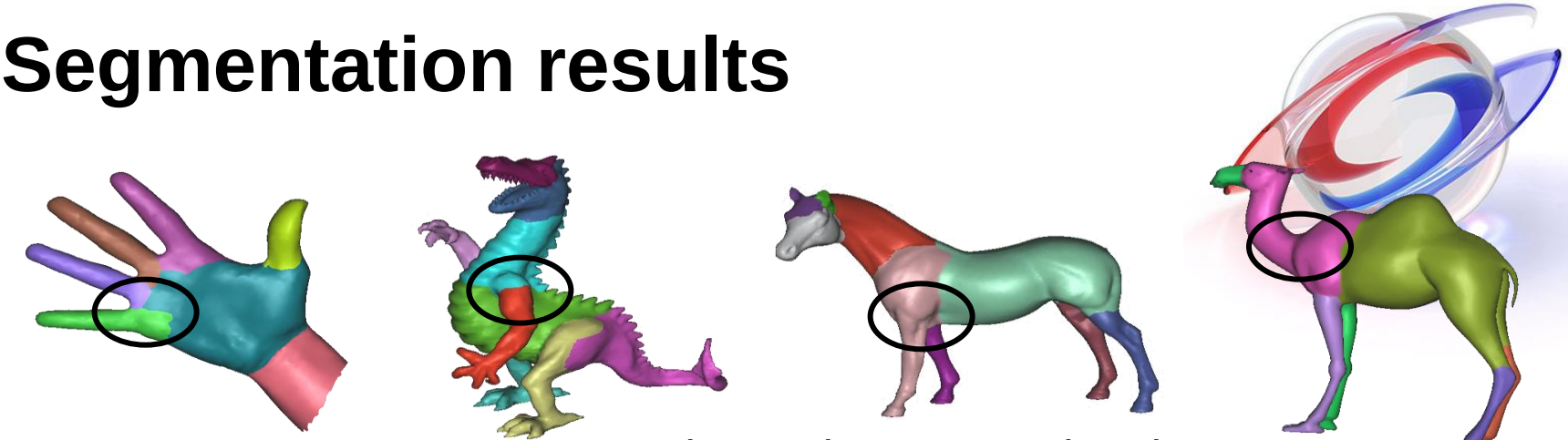
Placed into an algorithm

- Early method of [Liu & Zhang 04]
- Use a metric and exponential kernel to define affinities

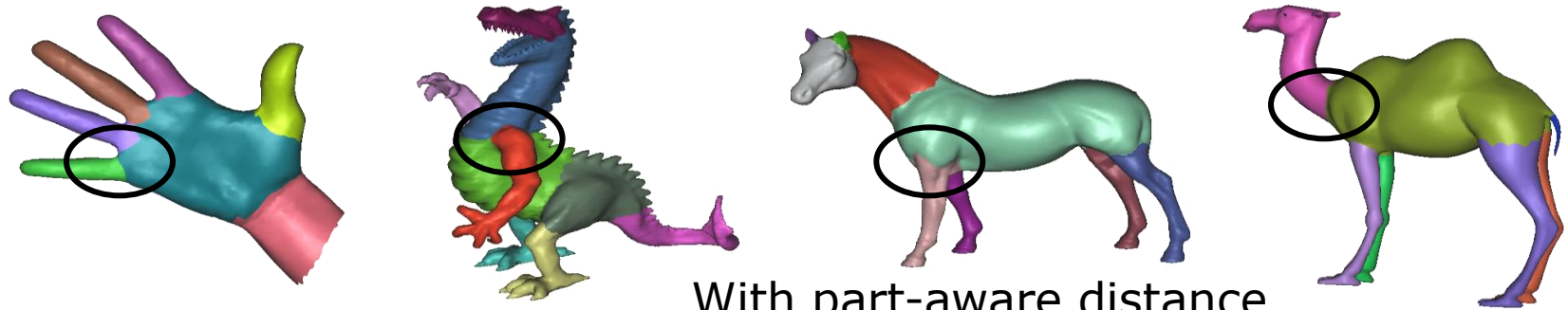


- Spectral embedding by eigen-decomposing the matrix
- Apply *k*-means in spectral domain

Segmentation results



With geodesic+angular distance



With part-aware distance

Computational issues

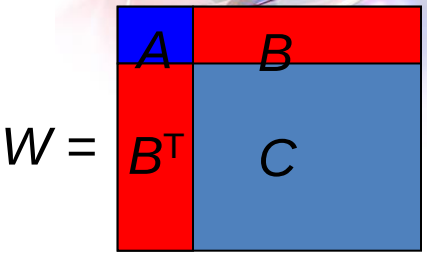
- Spectral embedding requires eigenvectors
- Expensive with large datasets (dense meshes)
- One possible solution: **Nyström approximation**
  - **Sub-sampling** of dense data
  - Solve small-scale spectral decomposition problem
  - **Extrapolate** results to full eigenvectors

Nyström approximation

- Basic steps (input matrix: $W \in \Re^{n \times n}$):
  - Select k samples, $k << n$
  - Compute submatrices A and B

    A: affinities between samples

    B: affinities between samples and non-samples
  - Eigendecompose $A = U \Lambda U^{\mathsf{T}}$
  - Extrapolate eigenvectors of W based on B, U, and Λ



Nyström approximation continued

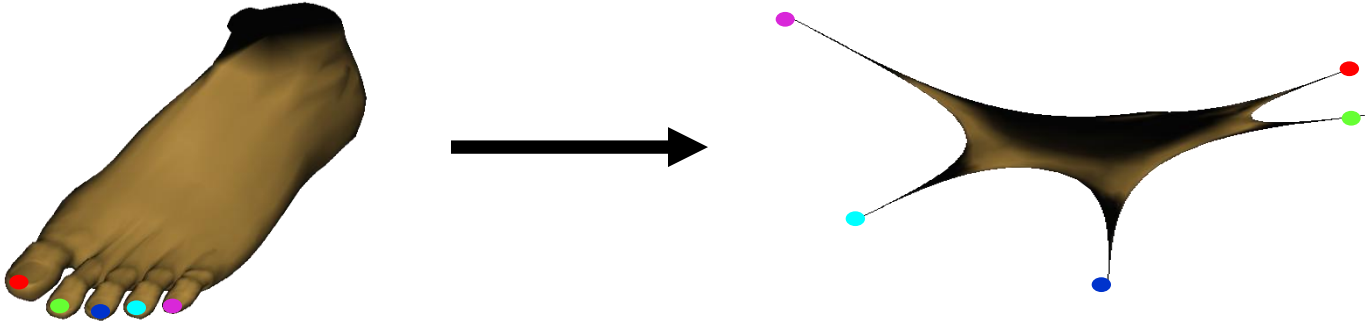
- Complexity: O(n^3) to O($mk^2 + k^3$), $k << n$ in practice
- Only approximations however, e.g.,

$$W = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

  not orthogonal in general
- Quality depends on sampling
  - Only limited studies on better approximation to C

Dimensionality of embeddings

- Typical *k*-means: a handful of dimensions
- Low dimensional embeddings, e.g., 1D or 2D
  - Large distortions, but possibly feature-exaggerating
  - Greatly simplifying certain analysis tasks, e.g., search



1D or 2D embeddings

- 1D embedding
  - Linear search possible
  - Can deal with hard-to-optimize energy, e.g., part salience [Zhang & Liu 05]
- 2D embedding
  - 3D analysis turned into contour analysis
  - Facilitate definition of segmentability measure and sampling for Nyström [Liu & Zhang 07]

Main references

R. Liu, H. Zhang, A. Shamir, and D. Cohen-Or, “A Part-Aware Surface Metric for Shape Processing”, *Eurographics* 2009

A. Shamir, “A Survey on Mesh Segmentation Techniques”, *Computer Graphics Forum* (*Eurographics STAR* 2006), 2008.

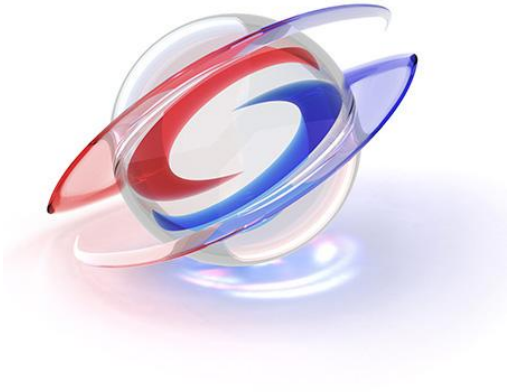
F. de Goes, Siome Goldenstein, and Luiz Velho, “A hierarchical Segmentation of Articulated Bodies”, *SGP* 2008.

R. Liu and H. Zhang, “Spectral Clustering for Mesh Segmentation”, *Pacific Graphics* 2004.

R. Liu and H. Zhang, “Mesh Segmentation via Spectral Embedding and Contour Analysis”, *Eurographics* 2007.

S. Katz and A. Tal, “Hierarchical Mesh Segmentation via Fuzzy Clustering and Cuts”, *SIGGRAPH* 2003.

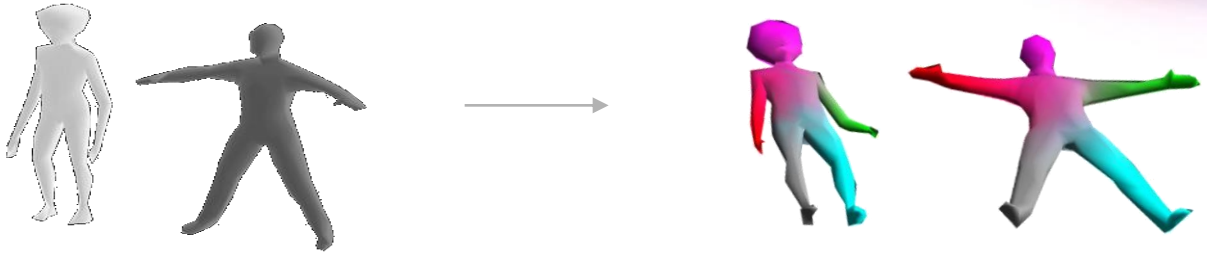
Applications covered



- Mesh segmentation
- Non-rigid correspondence
- Shape retrieval
- Global intrinsic symmetry detection

The correspondence problem

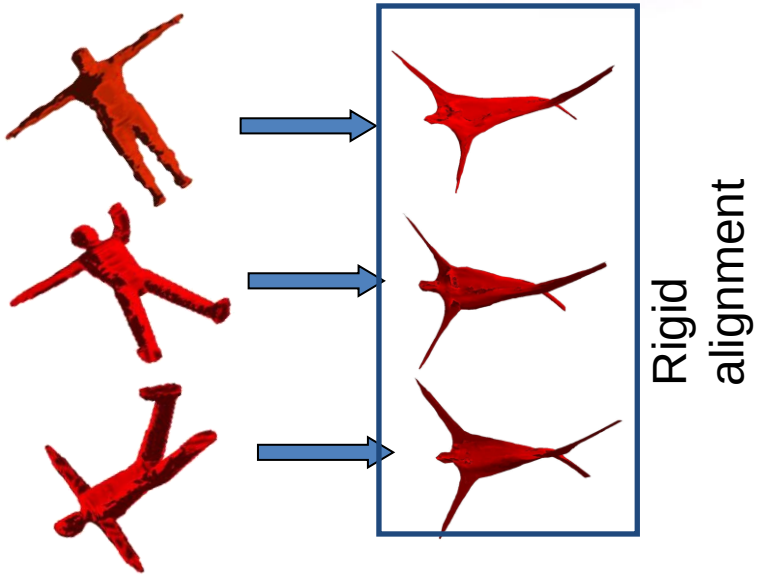
- Find meaningful correspondence between mesh vertices



  - Another fundamental problem in shape analysis
  - Handle rigid transformations and **poses** (not isometry)
  - Much harder is **non-uniform part scaling**

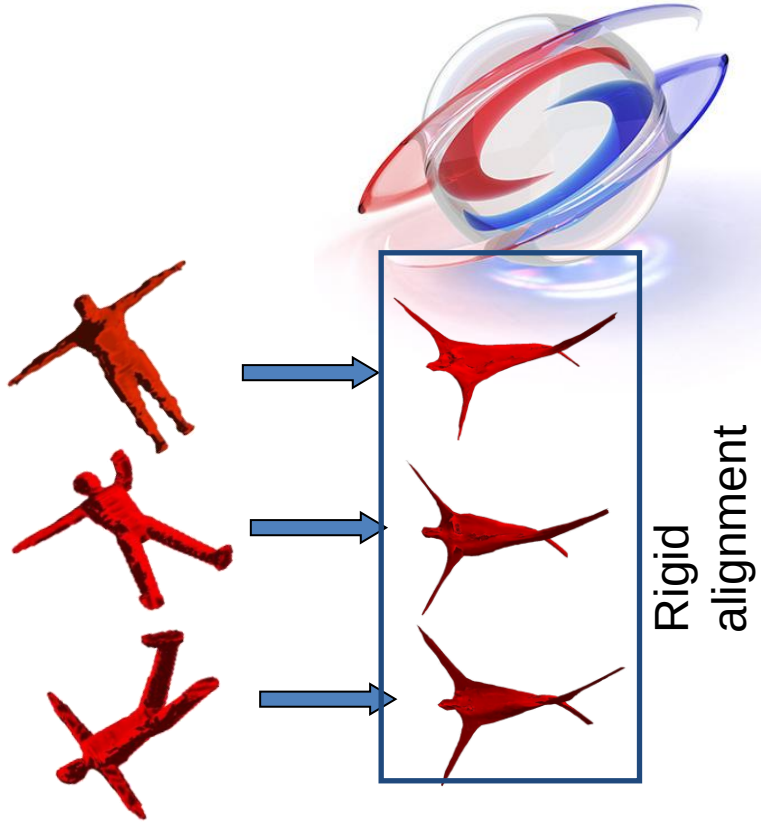
Framework for spectral approach

1. Define appropriate distance between mesh elements
2. Distances to affinities
3. Spectrally embed two shapes based on affinity matrix
4. Correspondence analysis in spectral domain
   - Perhaps rigid transforms would suffice
   - At least good initialization



Why spectral?

- **Intrinsic** affinities/distance
  - Whichever transformation, e.g., pose, correspondence is to be invariant of, build that invariance into affinity, e.g., geodesic distance
  - Serves as a **normalization**
- Affinities (e.g., Gaussian) give scale normalization

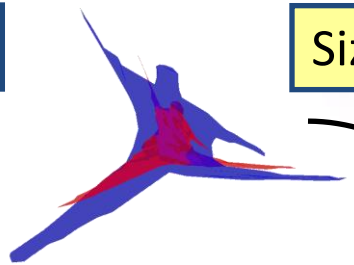


An algorithm

Pose normalization

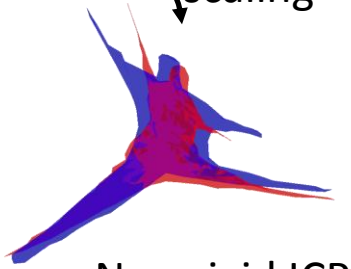
Geodesic-based

spectral embedding

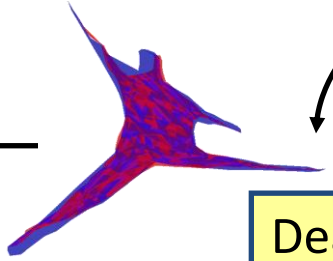


Size of data sets

Eigenvector scaling



Non-rigid ICP via thin-plate splines

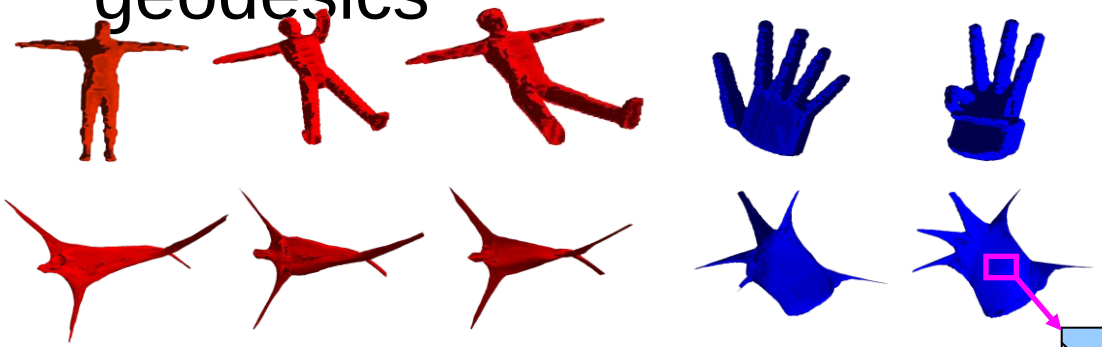


Deal with non-uniform deformation (to some extent)

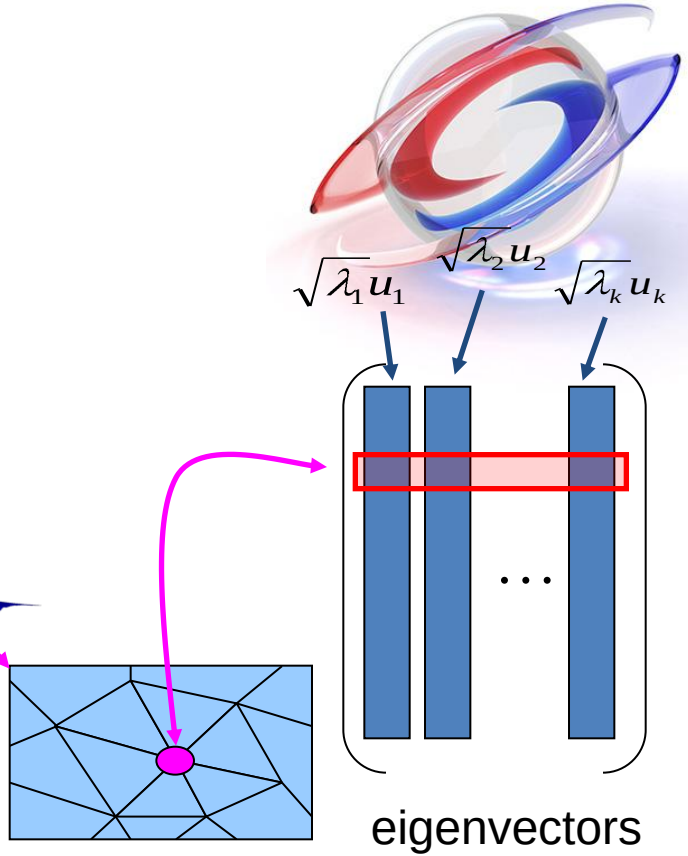
Best matching

Normalization

- Operator defined by geodesics

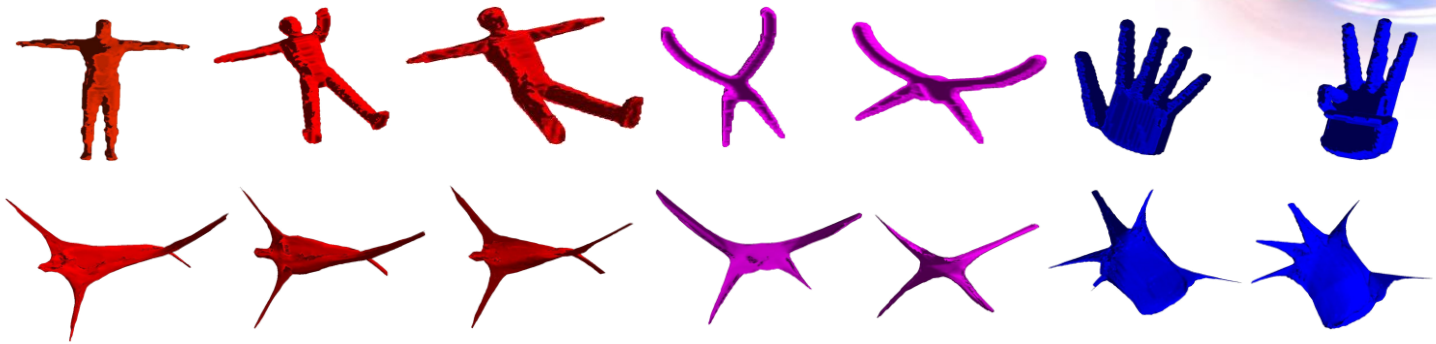


Spectral embeddings



eigenvectors

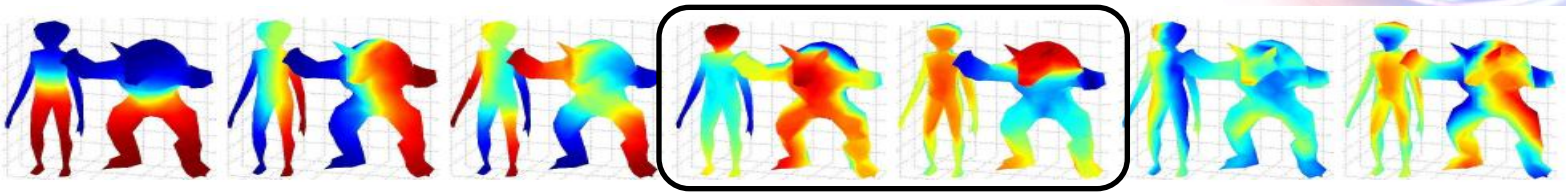
Examples: 3D spectral embeddings



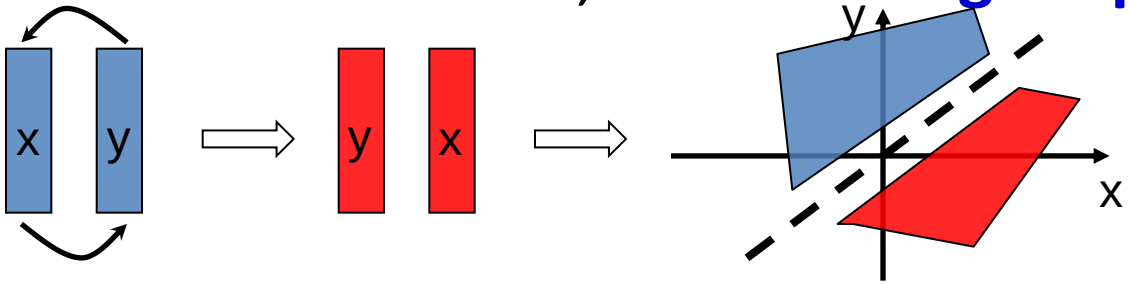
Use of 2nd, 3rd, and 4th scaled eigenvectors

Eigenvector switching

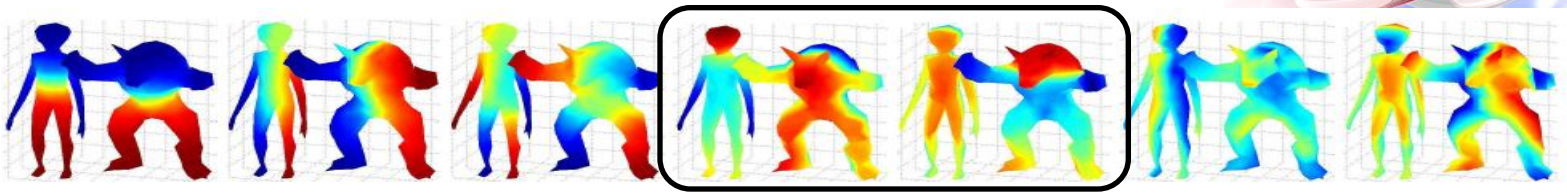
- Eigenvector plots reveal switching due to part variations



- Switching causes a **reflection**, as does a **sign flip**

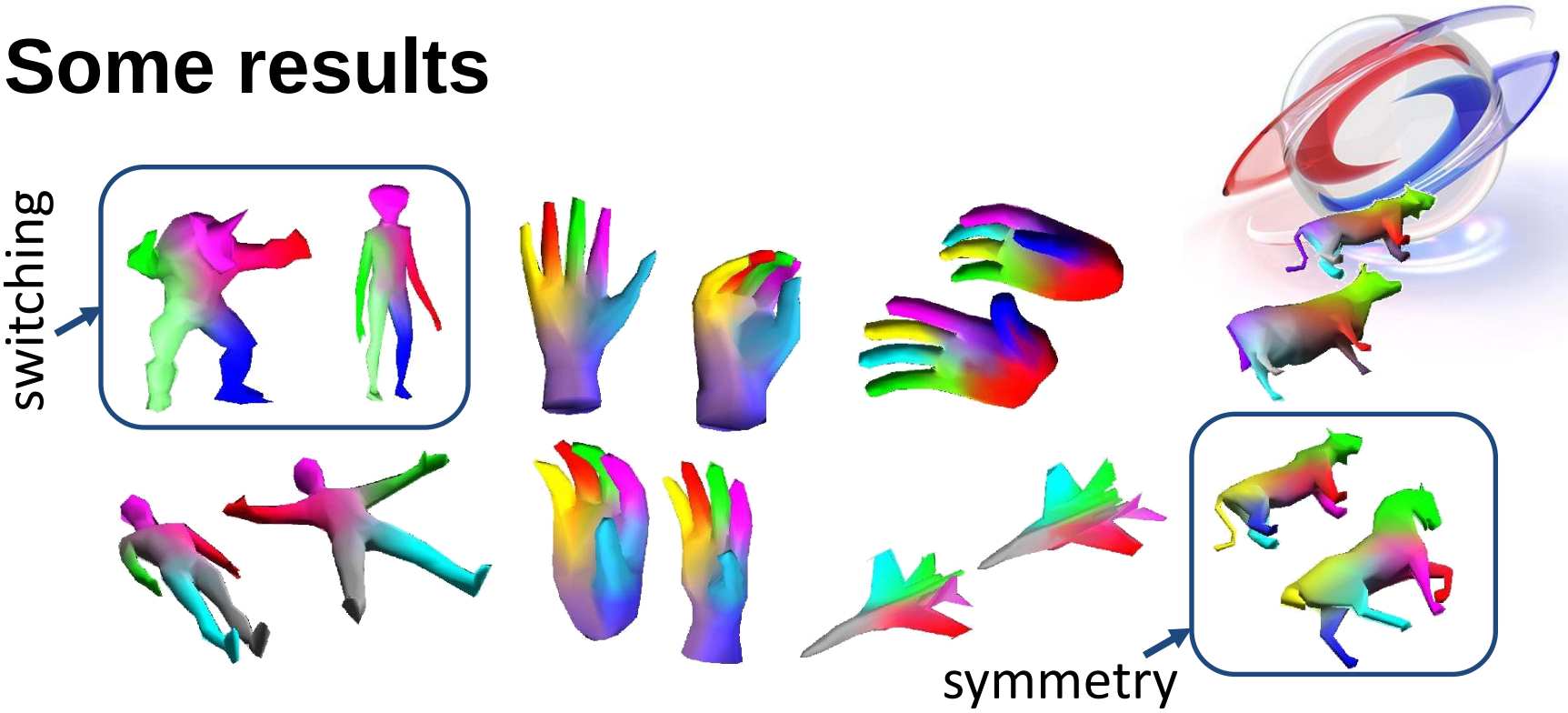


Eigenvector switching



- “Un-switching” and “un-flipping” by exhaustive search
- Search through all orthogonal transformations (rotation + reflection) in spectral domain [Mateus et al., 07]

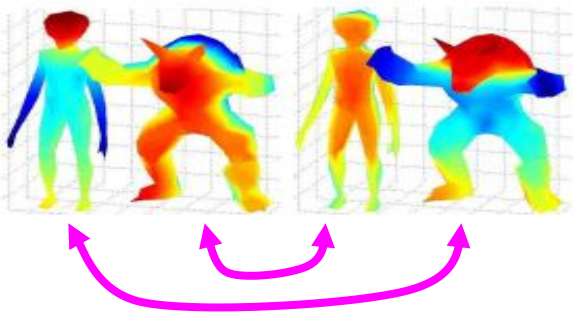
Some results

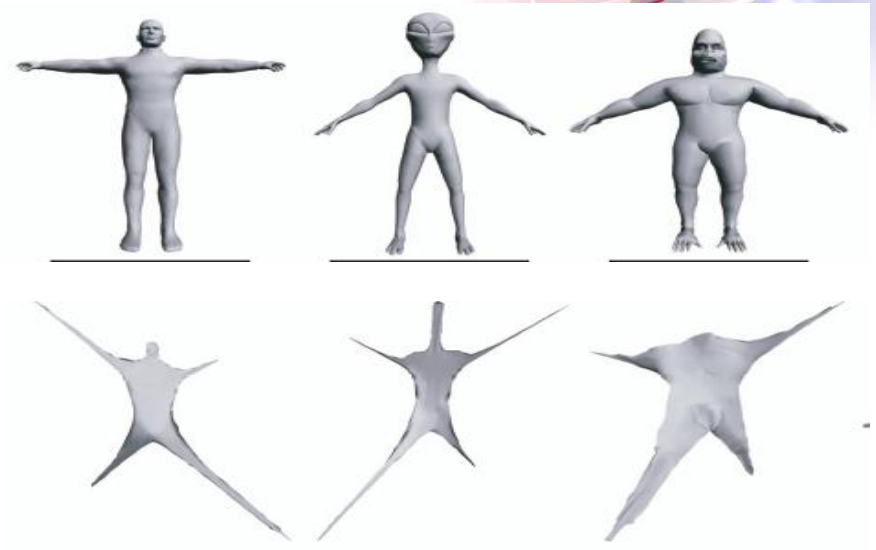


Models with articulation and moderate stretching

Thinking of the cause

- Stretching of parts

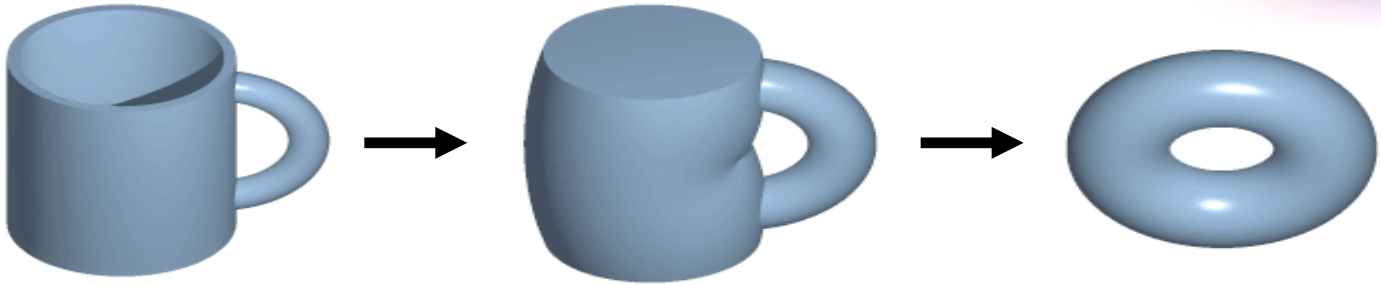




[Gal et al.2007]

Stretch-invariant correspondence?

- Cannot really allow arbitrary stretching

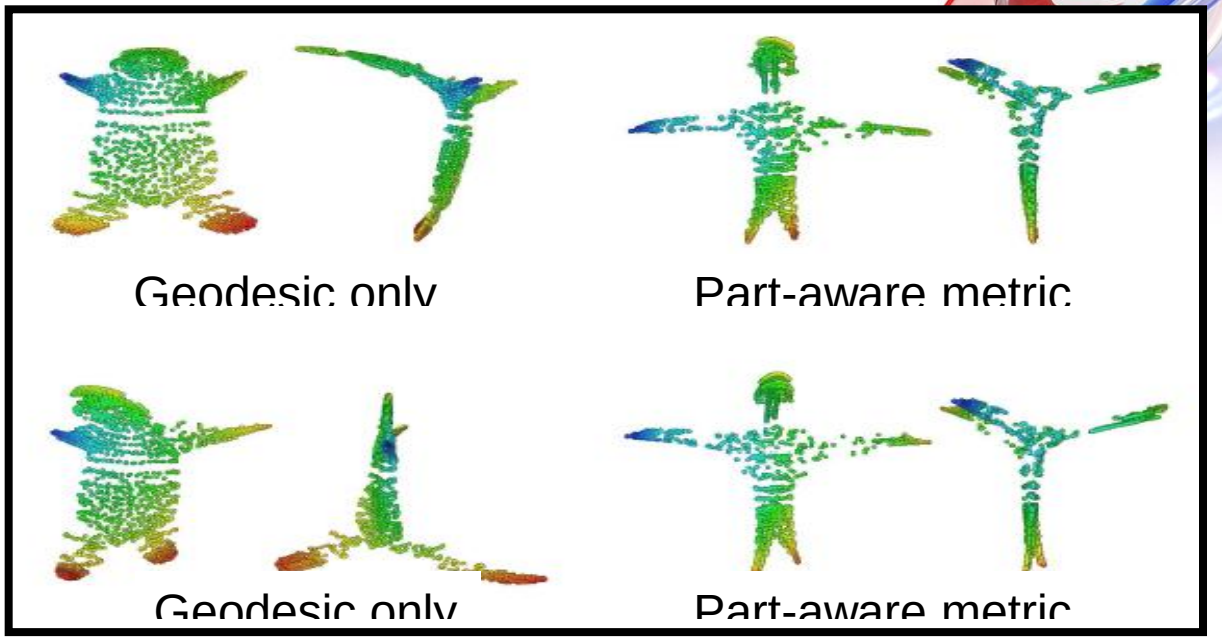


- But stretching that **preserves part structure**
  - Distance measure that captures part structure?

Use of part-aware metric

Homer

Stretched

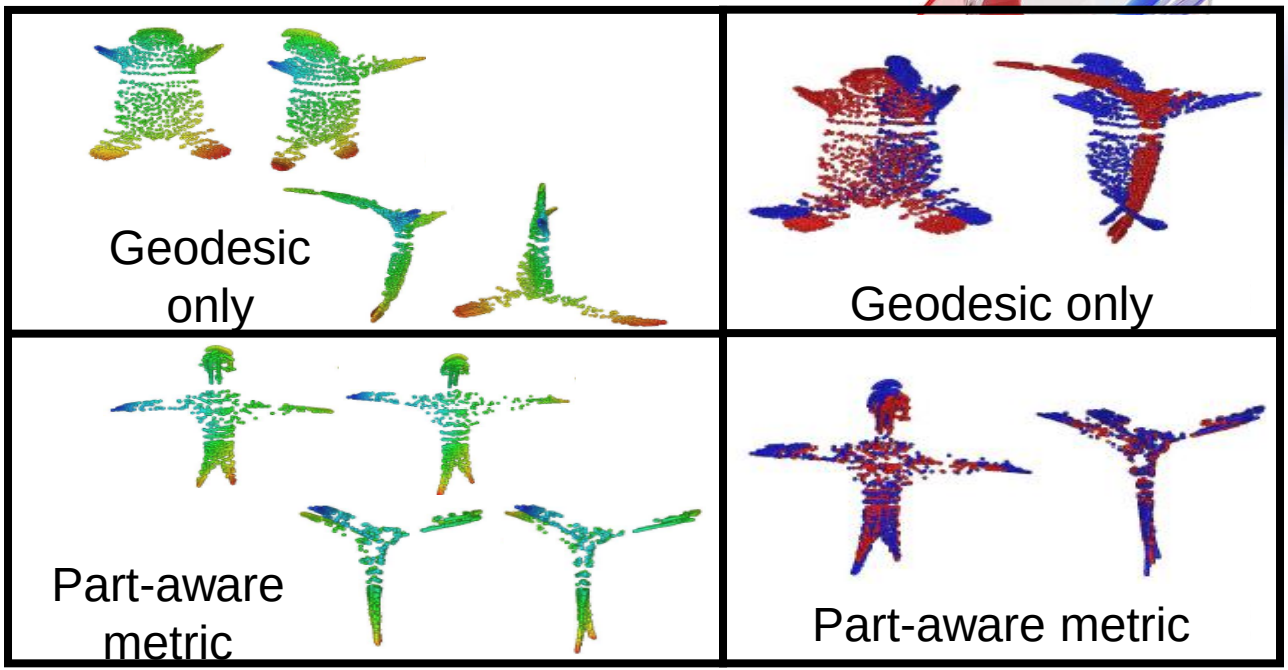


Spectral embeddings

Robustness against part stretching

Homer

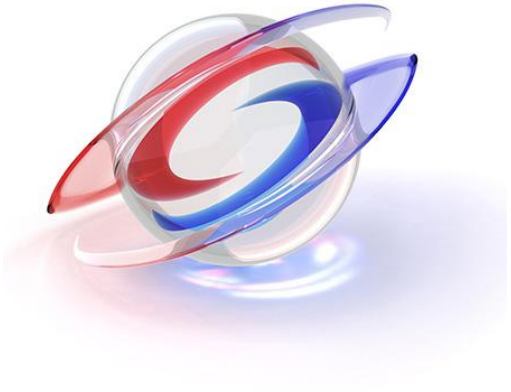
Stretched



Spectral embeddings

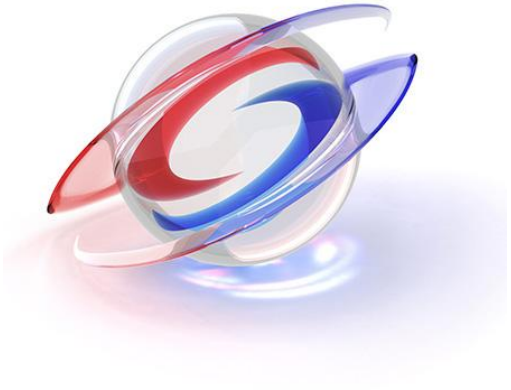
Rigid ICP registration

Limitations



- Intrinsic geodesic affinities
  - Symmetric switches: combine with extrinsic info
  - **Topology** sensitivity: use of Laplacian embeddings
- Primitive heuristic for resolving **eigenvector switching**
  - Shape characterization using heat kernels
- Being global, do not solve **partial matching** problem

Applications covered



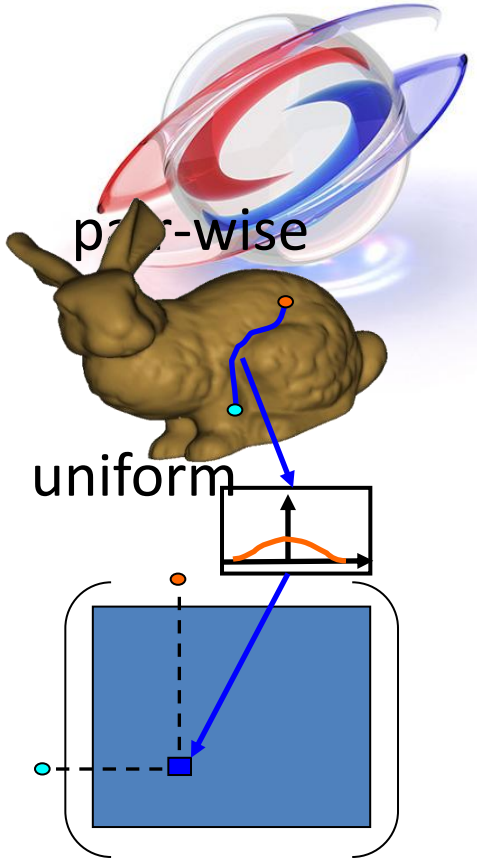
- Mesh segmentation
- Non-rigid correspondence
- Shape retrieval
- Global intrinsic symmetry detection

Shape retrieval

- Search and retrieval most similar shapes to a query from a shape database
- Geometry-based similarity, toeralating
  - Rigid transformations
  - Similarity transformation
  - Shape poses
  - Small topological changes
  - Small local structural variations
  - Non-uniform part stretching

A spectral approach

- Operator: Matrix of Gaussian-filtered geodesic distances
- Only take 20 samples — Nyström with sampling on dense mesh
- Apply conventional shape descriptors spectral embeddings



[Jian and Zhang 06]

Use of eigenvalues

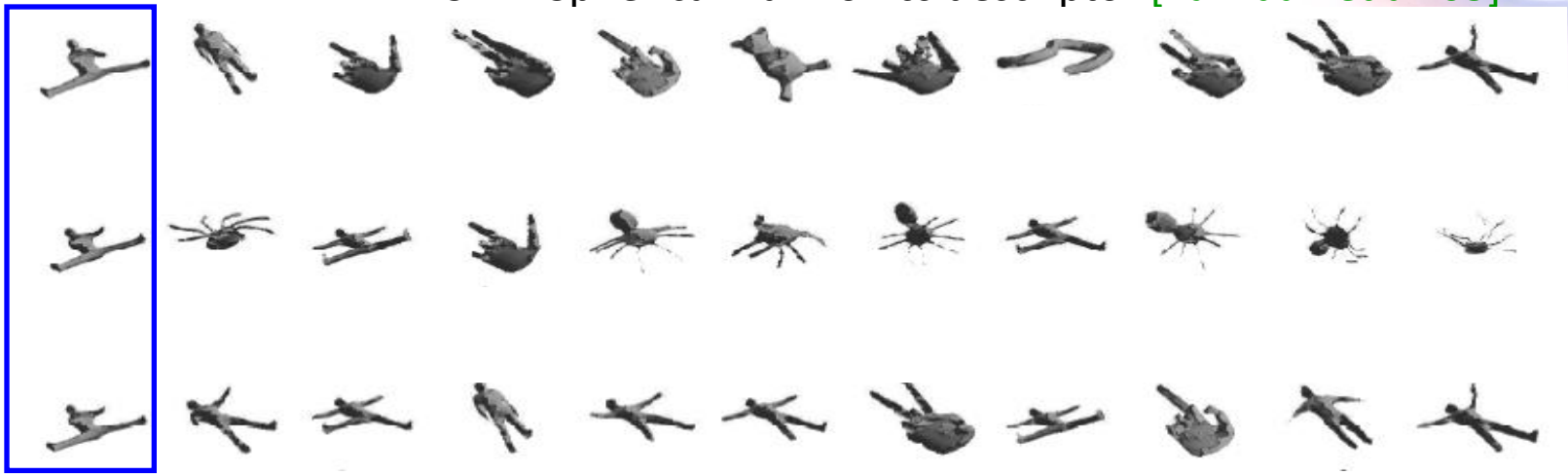
- Shape descriptor (EVD): eigenvalues λ_1, …, λ_{20} of 20×20 matrix — very **efficient to compute**
- Use χ^2 d

$$Dist_{EVD}(P,Q) = \frac{1}{2}\sum_{i=1}^{20} \frac{[\,|\lambda_i^P|^{\frac{1}{2}} - |\lambda_i^Q|^{\frac{1}{2}}]^2}{|\lambda_i^P|^{\frac{1}{2}} + |\lambda_i^Q|^{\frac{1}{2}}}.$$

Some results

LFD: Light-field descriptor [Chen et al. 03]

SHD: Spherical Harmonics descriptor [Kazhdan et al. 03]



LFD

SHD

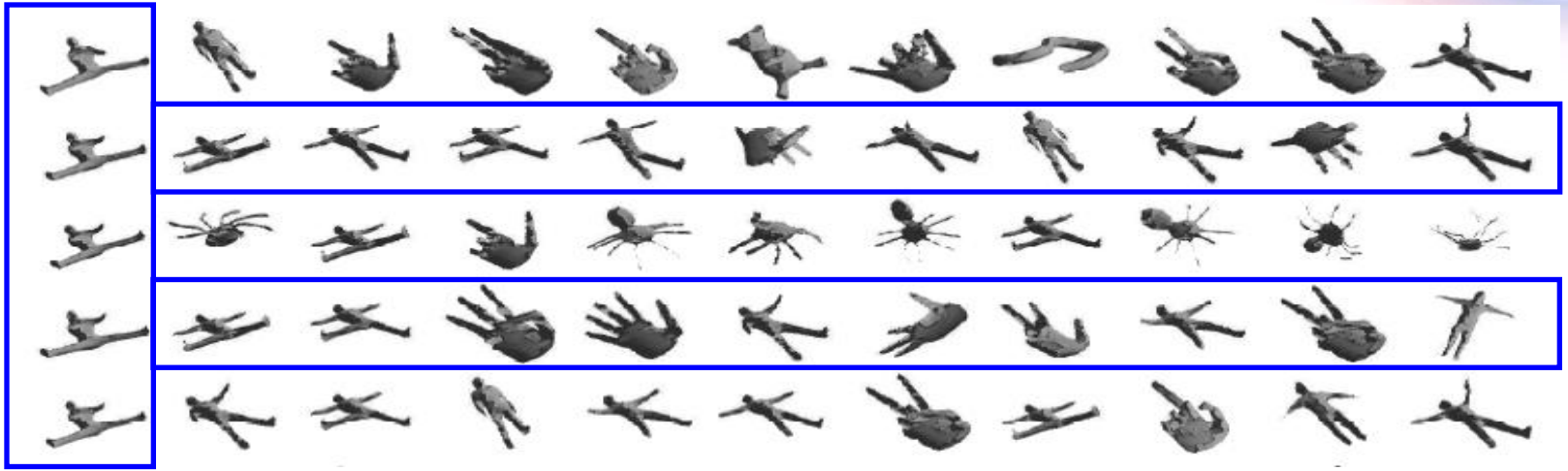
EVD

Retrieval on McGill Articulated Shape Database

Add in use of spectral embeddings

LFD-S: LFD on spectral embedding

SHD-S: SHD on spectral embedding



Retrieval on McGill Articulated Shape Database

Geodesics → Laplacian-Beltrami

- Deal with inherent limitation of geodesic distances — sensitivity to local topology change, e.g., a short “circuit”
- Use **Laplacian-Beltrami eigenfunctions** to construct spectral embeddings
  - Eigenfunctions have global nature
  - More stability to local changes
  - Isometry invariant — in sensitive to shape pose [Rustamov, *SGP* 07]

Global Point Signatures (GPS)

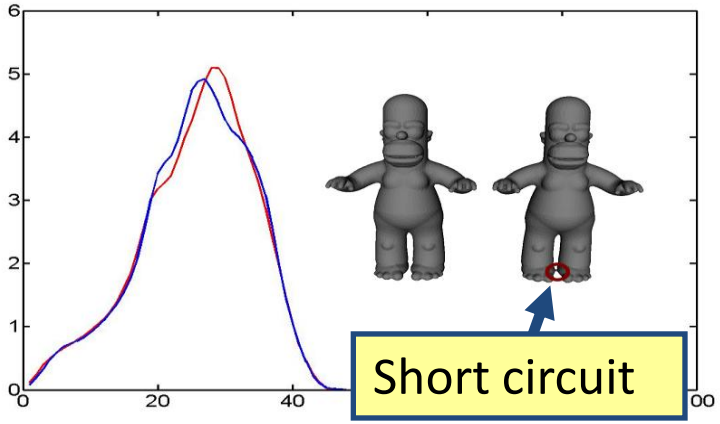
- Given a point **p** on the surface, define

$$GPS(\mathbf{p}) = \left(\frac{1}{\sqrt{\lambda_1}}\phi_1(\mathbf{p}), \frac{1}{\sqrt{\lambda_2}}\phi_2(\mathbf{p}), \frac{1}{\sqrt{\lambda_3}}\phi_3(\mathbf{p}), \ldots \right)$$

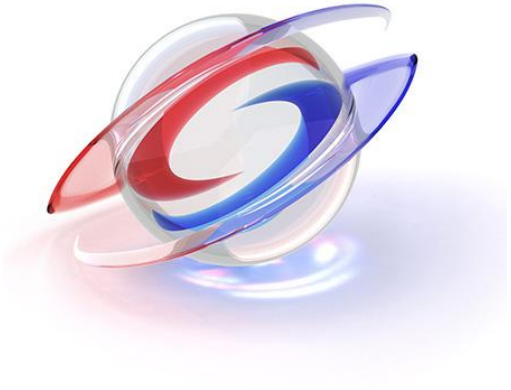
  - $\phi_i(\mathbf{p})$: value of the eigenfunction ϕ_i at the point **p**
  - λ_i's are the Laplacian-Beltrami eigenvalues
  - Scaling in light of commute time distances

GPS-based shape retrieval

- Use histogram of distances in the GPS embeddings
  - Invariance properties reflected in GPS embeddings
  - Less sensitive to topology changes with only low-frequency eigenfunctions
  - Sign flips and eigenvector still issues



Applications covered



- Mesh segmentation
- Non-rigid correspondence
- Shape retrieval
- Global intrinsic symmetry detection

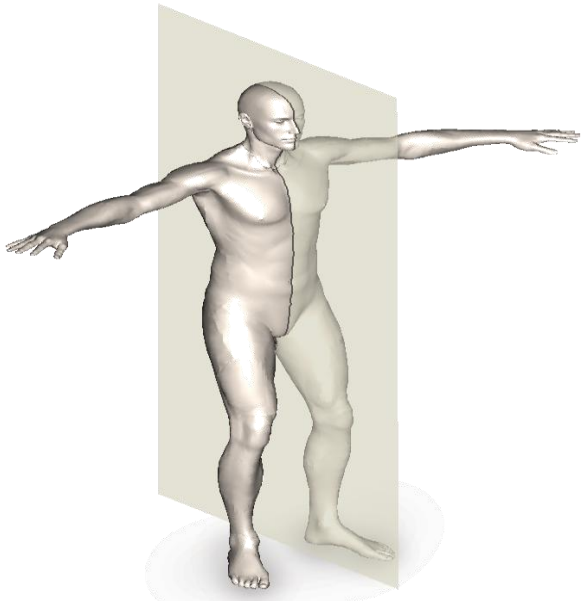
Symmetry detection

- Symmetry is ubiquitous in nature and man-made objects
- Crucial in shape understanding

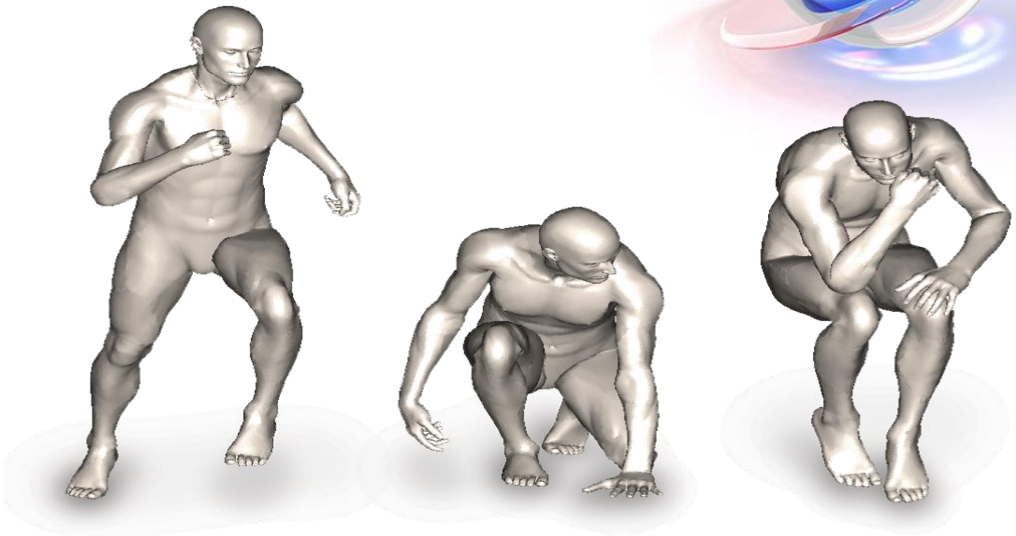
- Symmetric detection
  - Intrinsic vs. extrinsic, partial vs. global
- Symmetry-aware processing
  - Segmentation
  - Editing
  - Compression, etc.

Global intrinsic symmetries of 3D shapes

[Ovsjanikov et al. 08]



Extrinsic symmetry

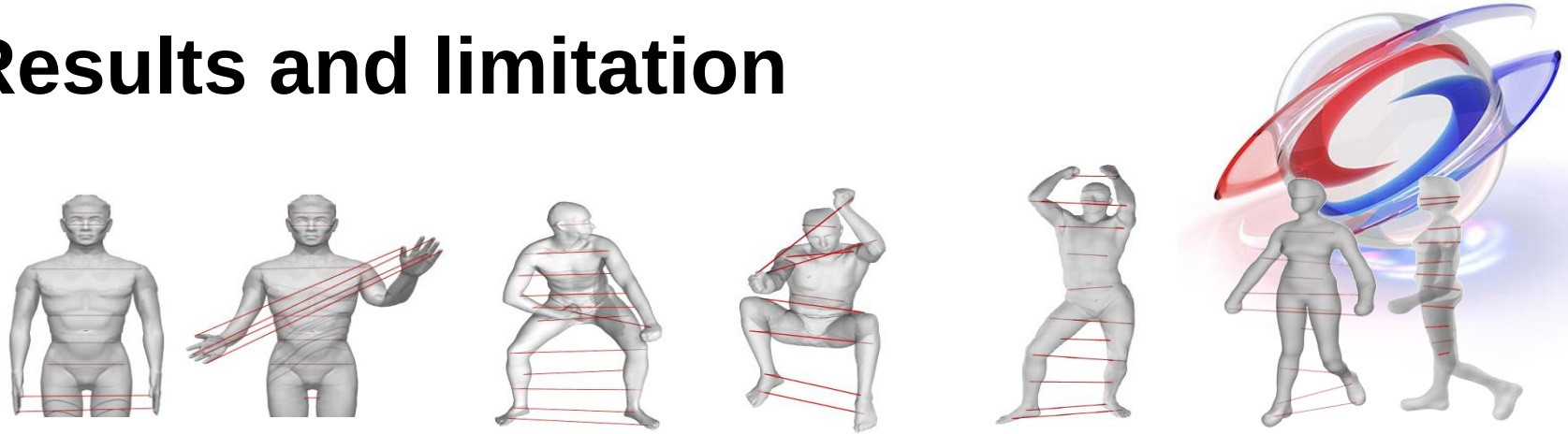


Global intrinsic symmetries

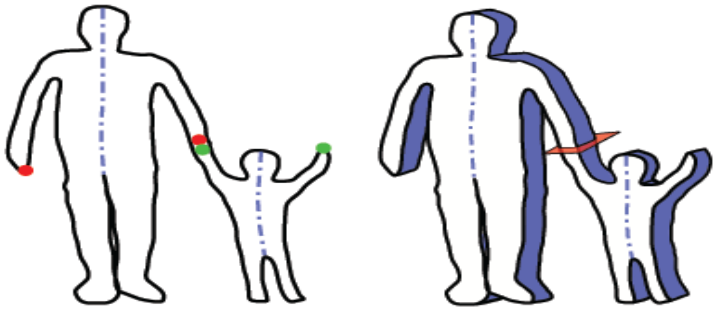
Application of GPS embeddings

- Reduce to Euclidean symmetry detection in GPS space
- **Restricted** GPS embeddings
  - Only consider **non-repeated eigenvalues**
  - Need to consider sign flips (canonical reflections)
  - Eigenfunctions associated with repeated eigenvalues
    - introduce rotational symmetries in GPS space
    - unstable under small non-isometric deformations

Results and limitation



- Limitation: **partial** intrinsic symmetry detection
  - Self-map discontinuous
  - Global symmetry unnatural
  - Distortion in embeddings



Main references

R. Rustomov, “Laplacian-Beltrami Eigenfunctions for Deformation Invariant Shape Representation”, *SGP* 2007.

V. Jain, H. Zhang, O. van Kaick, “Non-Rigid Spectral Correspondence of Triangle Meshes, *IJSM* 2007.

M. Ovsjanikov, J. Sun, and L. Guibas, “Global Intrinsic Symmetries of Shapes”, *SGP* 2008.

V. Jain and H. Zhang, “A Spectral Approach to Shape-Based Retrieval of Articulated 3D Models”, *CAD* 2007.

K. Xu, H. Zhang, A. Tagliasacchi, L. Liu, M. Meng, L. Guo, Y. Xiong, “Partial Intrinsic Reflectional Symmetry of 3D Shapes”, *SIGGRAPH Asia* 2009.

SIGGRAPH 2010

“*Spectral Mesh Processing*”

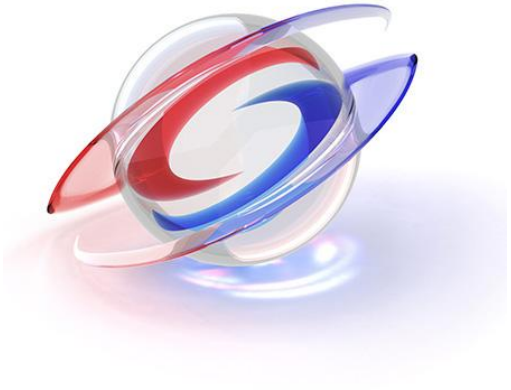
Bruno Lévy and Richard Hao Zhang

Spectral Mesh Processing Applications 2/2

Bruno Lévy

INRIA - ALICE

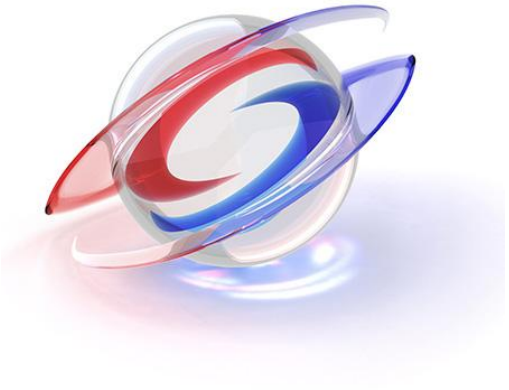
Overview



I. **1D parameterization**
II. Surface quadrangulation
III. Surface parameterization
IV. Surface characterization
    - Green function
    - Heat kernel

1D surface parameterization

Graph Laplacian



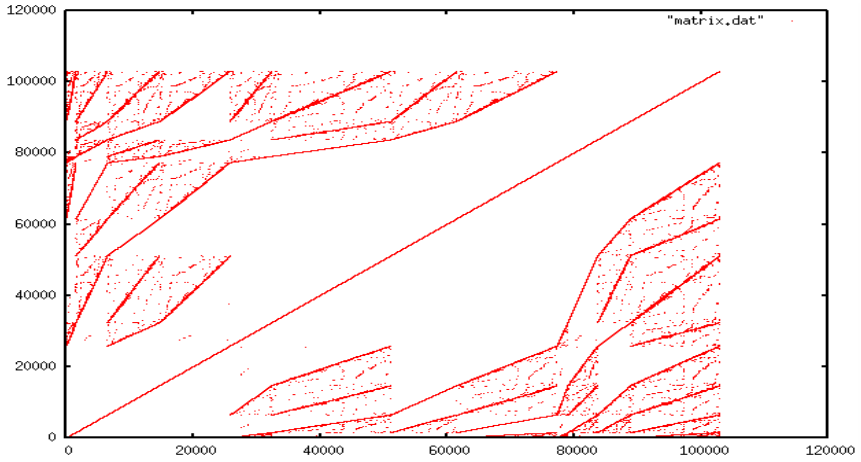
$a_{i,j} = w_{i,j} > 0$ if (i,j) is an edge

$a_{i,i} = -\sum a_{i,j}$

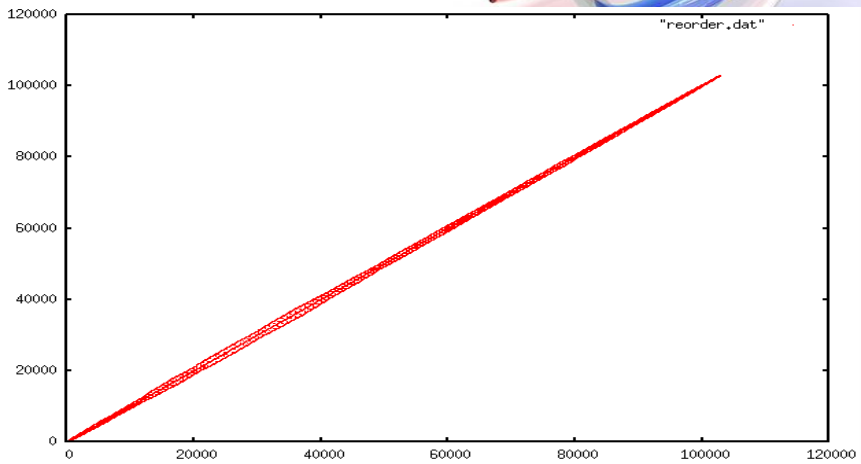
(1,1 … 1) is an eigenvector assoc. with 0

The second eigenvector is interresting
[Fiedler 73, 75]

1D surface parameterization
Fiedler vector

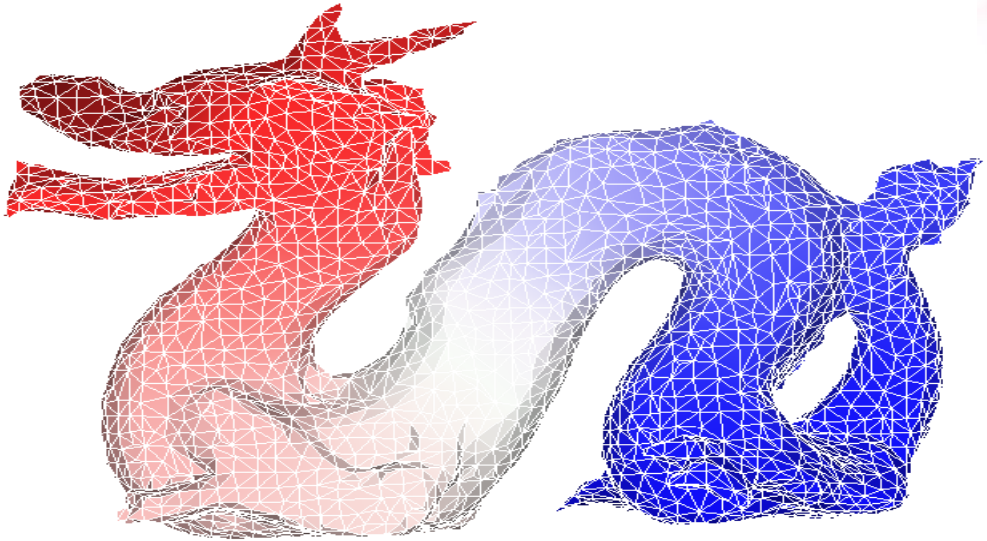


FEM matrix,
Non-zero entries



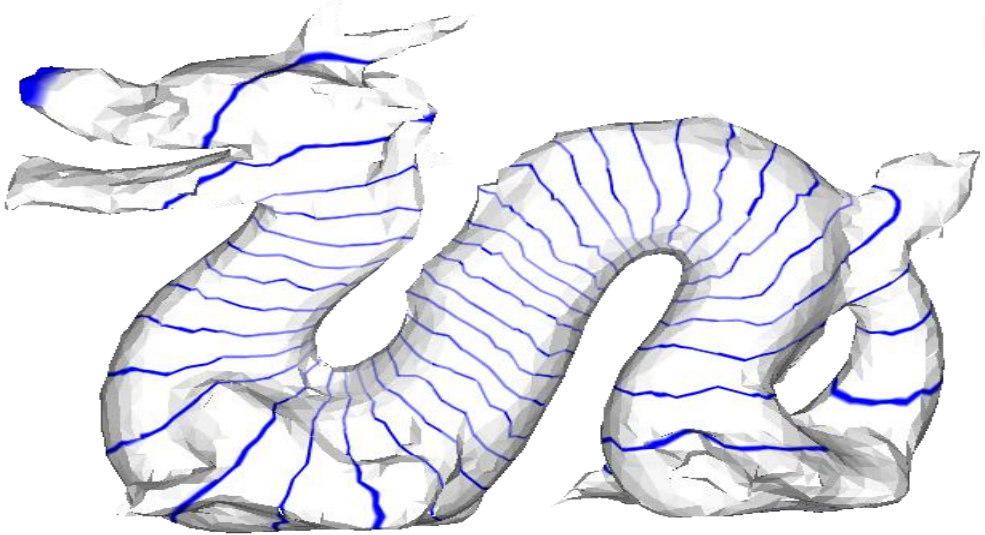
Reorder with
Fiedler vector

1D surface parameterization
Fiedler vector



Streaming meshes
[Isenburg & Lindstrom]

1D surface parameterization
Fiedler vector

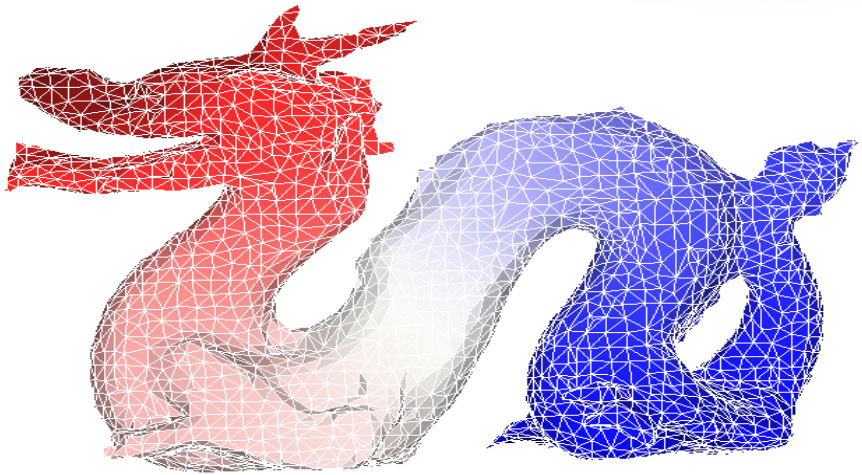


Streaming meshes
[Isenburg & Lindstrom]

1D surface parameterization
Fiedler vector

$$F(u) = \Sigma\, w_{ij}\, (u_i - u_j)^2$$

Minimize $F(u) = \frac{1}{2}\, u^t\, A\, u$



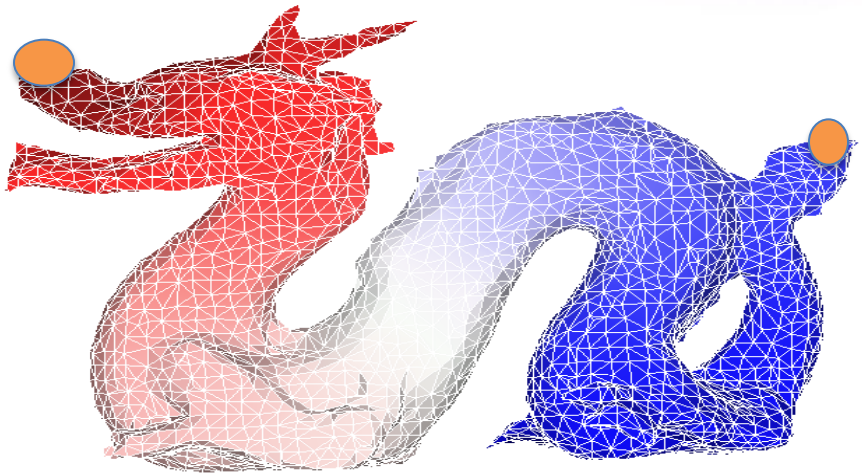
1D surface parameterization

Fiedler vector

$$F(u) = \Sigma\, w_{ij}\, (u_i - u_j)^2$$

Minimize $F(u) = \frac{1}{2}\, u^t\, A\, u$

How to avoid trivial solution ?
Constrained vertices ?



1D surface parameterization
Fiedler vector

$$F(u) = \Sigma\, w_{ij}\, (u_i - u_j)^2$$

Minimize $F(u) = \frac{1}{2}\, u^t\, A\, u$ subject to $\sum u_i = 0$

Global constraints are more elegant !

1D surface parameterization
Fiedler vector

$$F(u) = \Sigma\, w_{ij}\, (u_i - u_j)^2$$

Minimize $F(u) = \frac{1}{2}\, u^t\, A\, u$ subject to $\sum_i u_i = 0$

$\frac{1}{2} \sum_i u_i^2 = 1$

Global constraints are more elegant !

We need also to constrain the second mementum

1D surface parameterization
Fiedler vector

$$F(u) = \Sigma\, w_{ij}\, (u_i - u_j)^2$$

Minimize $F(u) = \frac{1}{2}\, u^t\, A\, u$ subject to $\sum u_i = 0$, $\frac{1}{2} \sum u_i^2 = 1$

$$L(u) = \frac{1}{2}\, u^t\, A\, u - \lambda_1\, u^t\, \mathbf{1} - \lambda_2\, \frac{1}{2}\, (u^t u - 1)$$

$$\nabla_u L = A\, u - \lambda_1\, \mathbf{1} - \lambda_2\, u$$
$$\nabla_{\lambda 1} L = u^t\, \mathbf{1}$$
$$\nabla_{\lambda 2} L = \frac{1}{2}(u^t\, u - 1)$$

u = eigenvector of A

$\lambda_1 = 0$

λ_2 = eigenvalue

1D surface parameterization
Fiedler vector

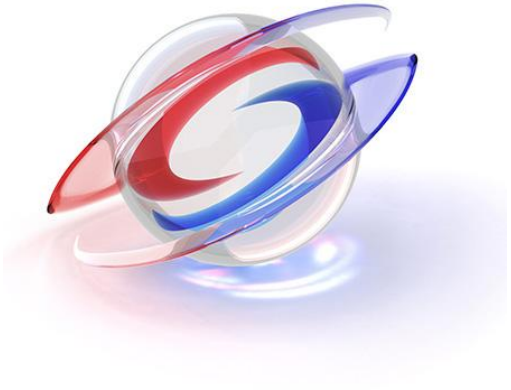
Rem: Fiedler vector is also a minimizer of the Rayleigh quotient

$$R(A,x) = \frac{x^t A\, x}{x^t\, x}$$

The other eigenvectors x_i are the solutions of :

minimize $R(A,x_i)$ subject to $x_i^t\, x_j = 0$ for $j < i$

Overview



I. 1D parameterization

II. Surface quadrangulation

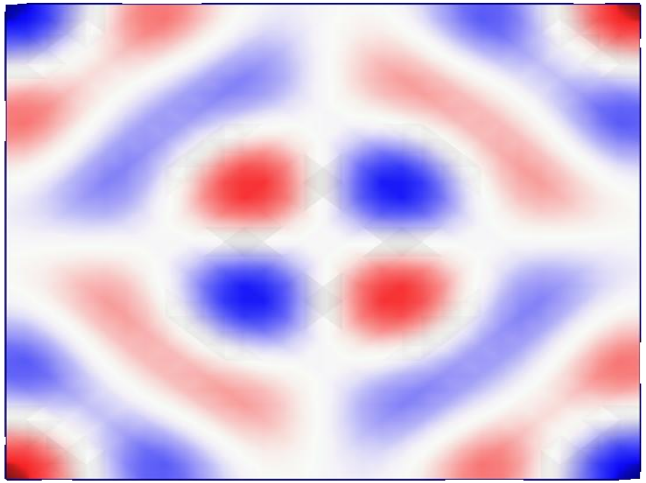
III. Surface parameterization

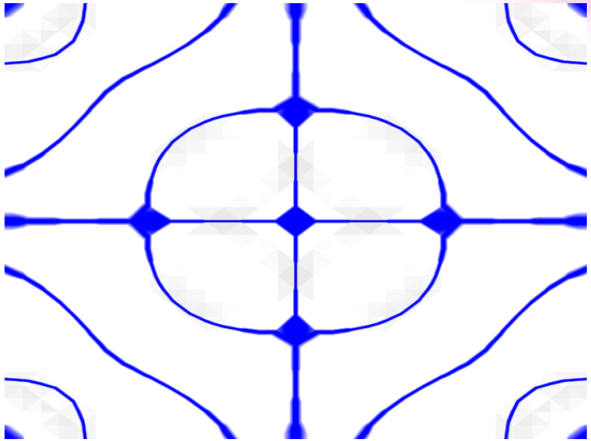
IV. Surface characterization

- Green function
- Heat kernel

Surface quadrangulation

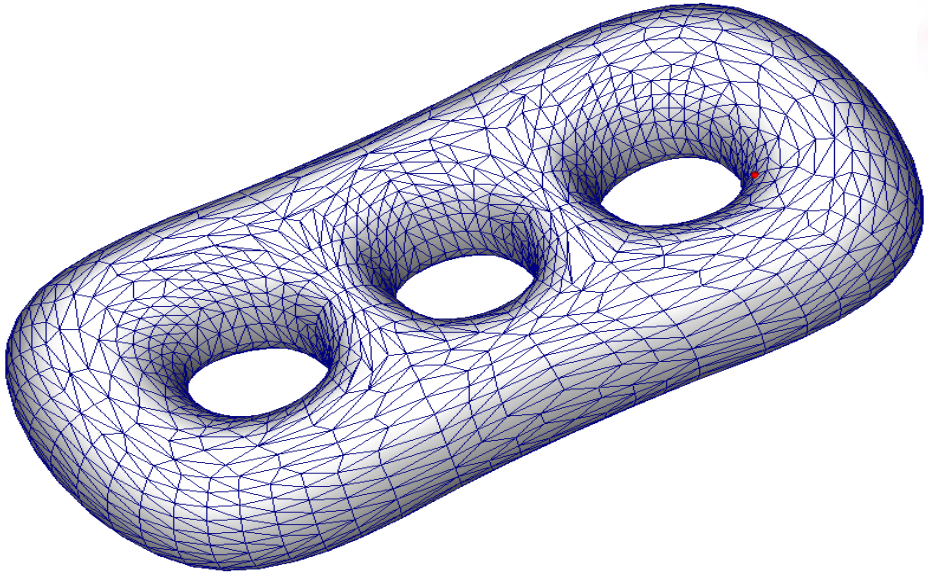
Nodal sets are sets of curves intersecting at constant angles



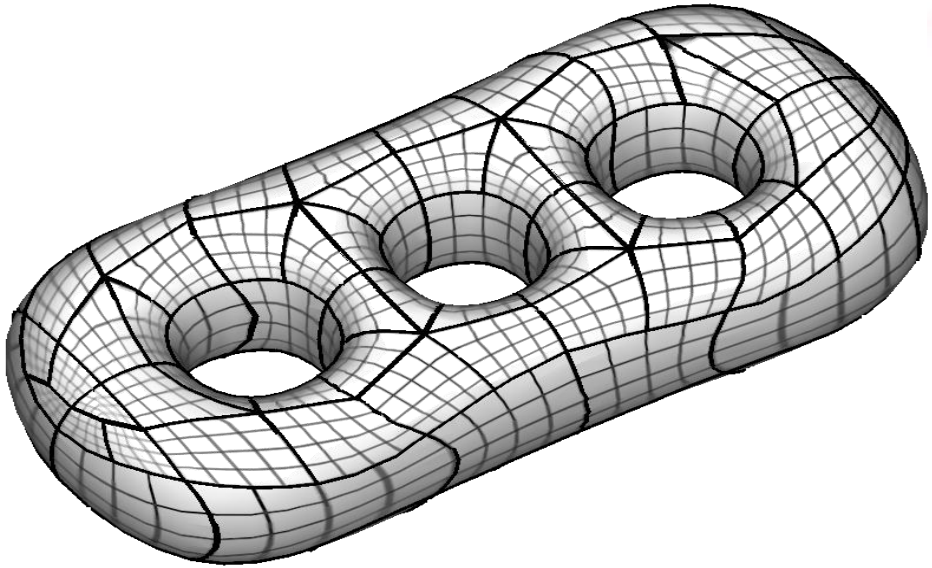


The N-th eigenfunction has at most N eigendomains

Surface quadrangulation

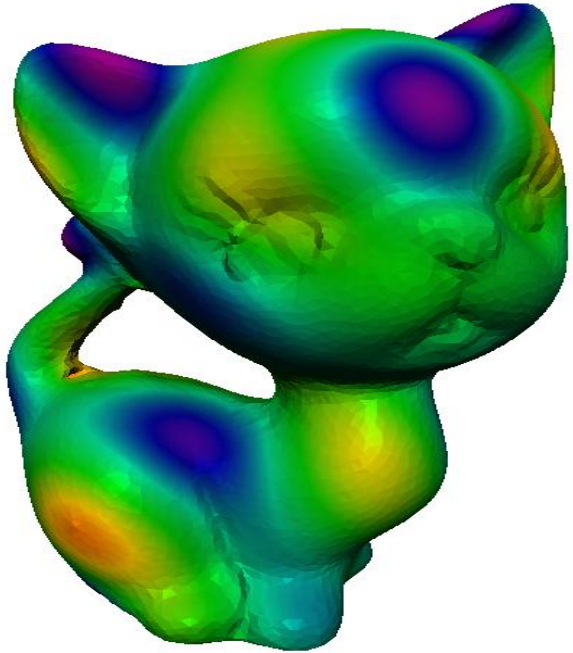


Surface quadrangulation

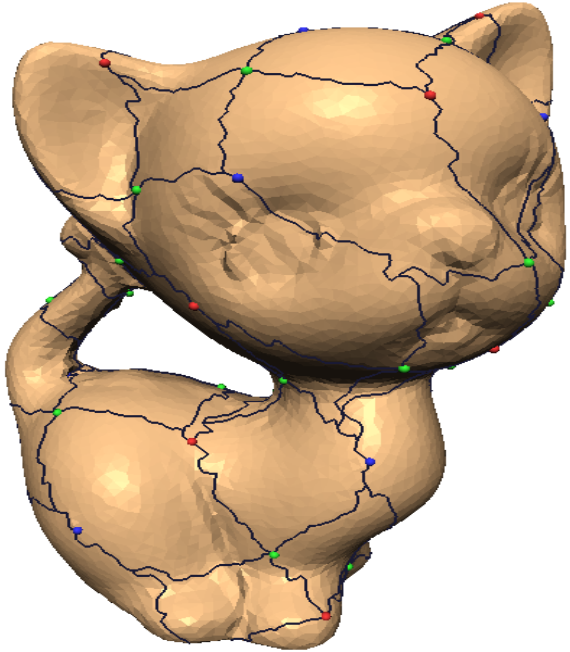


[L 2006], [Vallet & L 2006]

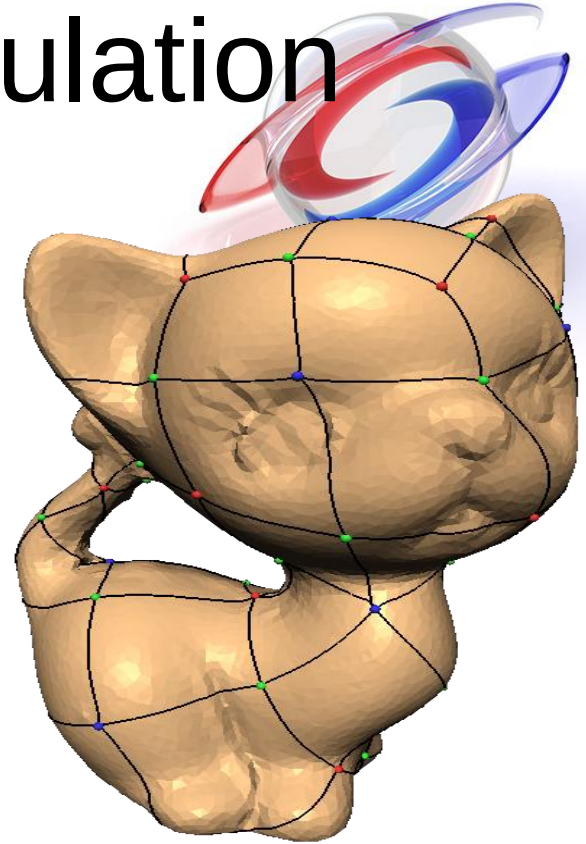
Surface quadrangulation



One eigenfunction



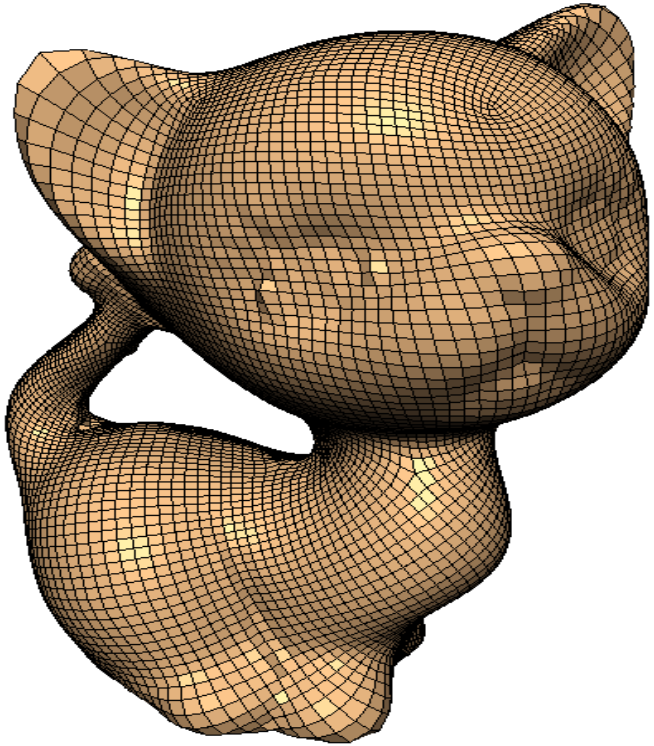
Morse complex

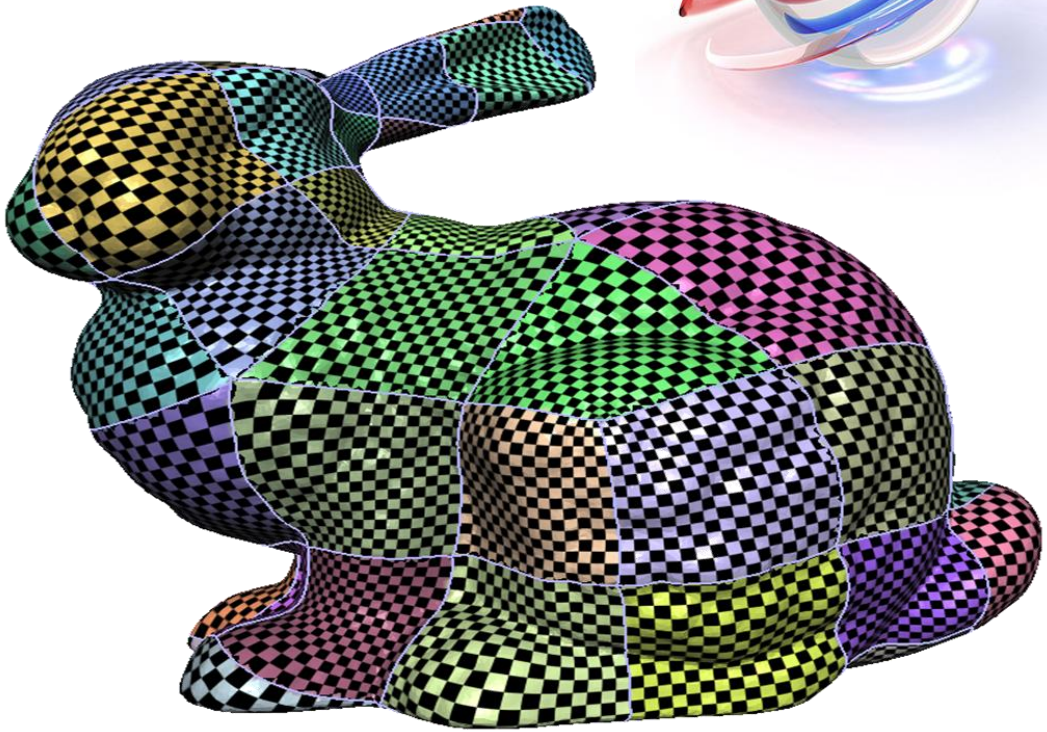


Filtered morse complex

[Dong and Garland 2006]

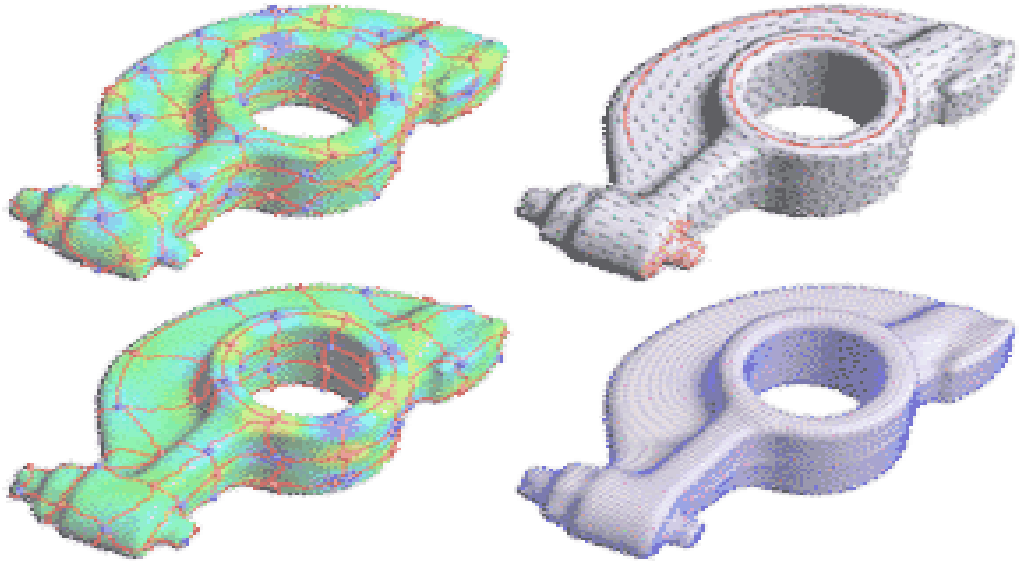
Surface quadrangulation





Reparameterization of the quads

Surface quadrangulation



Improvement in [Huang, Zhang, Ma, Liu, Kobbelt and Bao 2008], takes a guidance vector field into account.

Overview

I. 1D parameterization

II. Surface quadrangulation

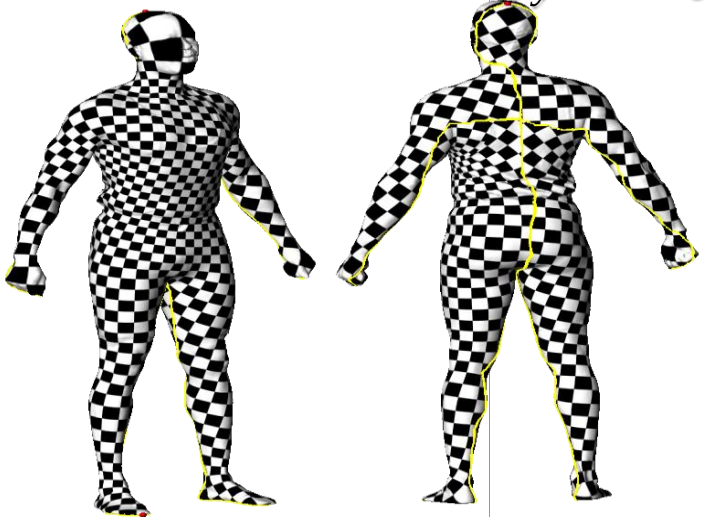
III. Surface parameterization

IV. Surface characterization

- Green function
- Heat kernel

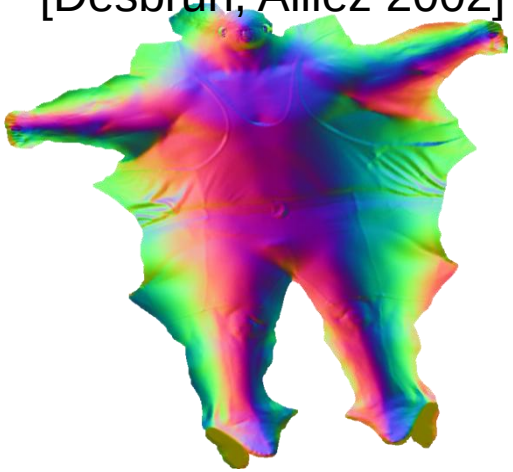
Surface parameterization

Minimize $\sum_{T} \left\| \begin{bmatrix} \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{bmatrix} - \begin{bmatrix} -\frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial x} \end{bmatrix} \right\|^2$



Discrete conformal mapping:

[L, Petitjean, Ray, Maillot 2002]
[Desbrun, Alliez 2002]

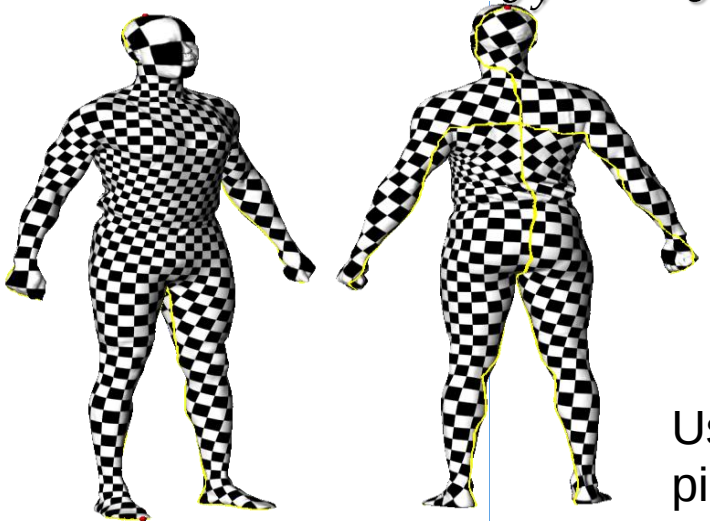


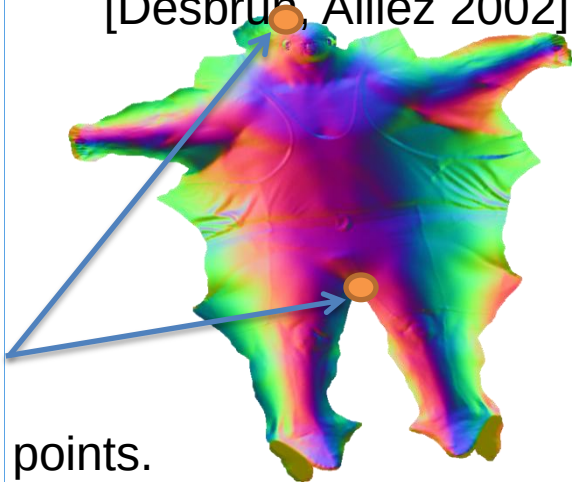
Surface parameterization

Minimize $\sum_{T} \left\| \begin{bmatrix} \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial y} \end{bmatrix} - \begin{bmatrix} -\frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial x} \end{bmatrix} \right\|^2$

Discrete conformal mapping:

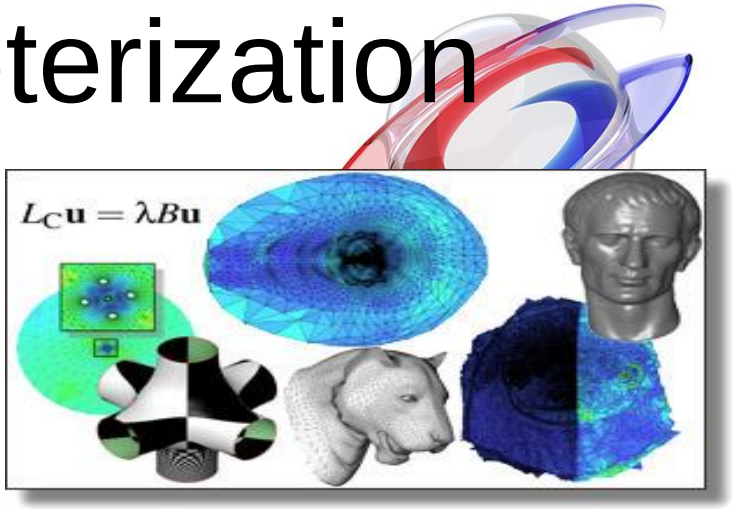
[L, Petitjean, Ray, Maillot 2002]
[Desbrun, Alliez 2002]





Uses pinned points.

Surface parameterization



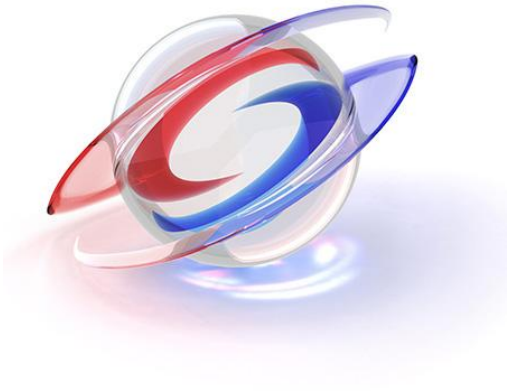
[Muellen, Tong, Alliez, Desbrun 2008]

Use Fiedler vector,
i.e. the minimizer of $R(A,x) = x^t A x / x^t x$
that is orthogonal to the trivial constant solution

Implementation:
(1) assemble the matrix of the discrete conformal parameterization
(2) compute its eigenvector associated with the first non-zero eigenvalue

See http://alice.loria.fr/WIKI/ Graphite tutorials – Manifold Harmonics

Overview



I. 1D parameterization

II. Surface quadrangulation

III. Surface parameterization

IV. Surface characterization

Green function

Heat kernel

Surface characterization

Green Function

Solving Poisson equation: $\Delta f = g$

$$f = \int G(x,y) f(y)\, dy$$

Where G: Green function is defined by: $\Delta G(x,y) = \delta(x-y)$

δ : dirac

Surface characterization

Green Function

Solving Poisson equation: $\Delta f = g$

$$f = \int G(x,y) f(y)\, dy$$

Where G: Green function is defined by: $\Delta G(x,y) = \delta(x-y)$

δ : dirac

Proof:

$$\int \Delta\, G(x,y) g(y)\, dy = \int \delta(x-y) g(y) dy = g(x) = \Delta\, f(x)$$

Surface characterization

Green Function

Solving Poisson equation: $\Delta f = g$

$$f = \int G(x,y) f(y)\, dy$$

Where G: Green function is defined by: $\Delta G(x,y) = \delta(x-y)$

δ : dirac

Proof:

$$\int \Delta\, G(x,y) g(y)\, dy = \int \delta(x-y) g(y) dy = g(x) = \Delta\, f(x)$$

$$\Delta\, f(x) = g(x) = \int \Delta\, G(x,y) g(y) dy = \Delta \left(\int G(x,y) g(y) dy \right)$$

Surface characterization

Green Function

Solving Poisson equation: $\Delta f = g$

$$f = \int G(x,y) f(y)\, dy$$

Where G: Green function is defined by: $\Delta G(x,y) = \delta(x-y)$

δ : dirac

Proof:

$$\int \Delta\, G(x,y) g(y)\, dy = \int \delta(x-y) g(y) dy = g(x) = \Delta\, f(x)$$

$$\Delta\, f(x) = g(x) = \int \Delta\, G(x,y) g(y) dy = \Delta \left(\int G(x,y) g(y) dy \right)$$

$$f(x) = \int G(x,y) g(y) dy$$

Surface characterization

Green Function

How to compute G ? G is defined by: $\Delta G(x,y) = \delta(x-y)$

δ : dirac

$\delta(x-y) = \sum \phi_i(x)\, \phi_i(y)$ (completeness of the eigen decomposition)

Surface characterization

Green Function

How to compute G ? G is defined by: $\Delta G(x,y) = \delta(x-y)$

δ : dirac

$\delta(x-y) = \sum \phi_i(x)\, \phi_i(y)$ (completeness of the eigen decomposition)

Using $G(x,y) = \dfrac{\sum \phi_i(x)\, \phi_i(y)}{\lambda_i}$ Works !

$$\left(\Delta G(x,y) = \sum \phi_i(x)\, \phi_i(y) = \delta(x-y) \right)$$

Note: Convergence of G series needs to be proved (complicated)

Surface characterization

Green Function

$$G(x,y) = \frac{\sum \phi_i(x)\,\phi_i(y)}{\lambda_i}$$

Summary:

Solution Poisson equation: $\Delta f = g$

$$f = \int G(x,y) f(y)\, dy$$

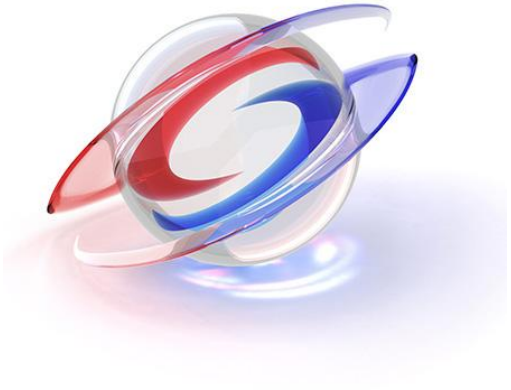
Pose-invariant embedding

Connection with GPS embedding [Rustamov 2007]

$$GPS(x) = [\phi_1(x)/\sqrt{\lambda_1}\ ,\ \phi_2(x)/\sqrt{\lambda_2}\ ,\ \ldots\ ,\ \phi_i(x)/\sqrt{\lambda_i}\ ,\ \ldots\ ]$$

$$G(x,y) = GPS(x) * GPS(y)$$

Overview



I. 1D parameterization

II. Surface quadrangulation

III. Surface parameterization

IV. Surface characterization

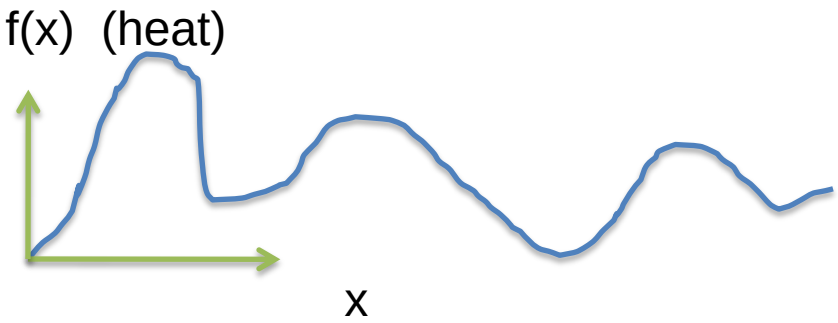
Green function

Heat kernel

Surface characterization

Heat equation

The heat equation: $\frac{\partial f}{\partial t} = -\Delta f$

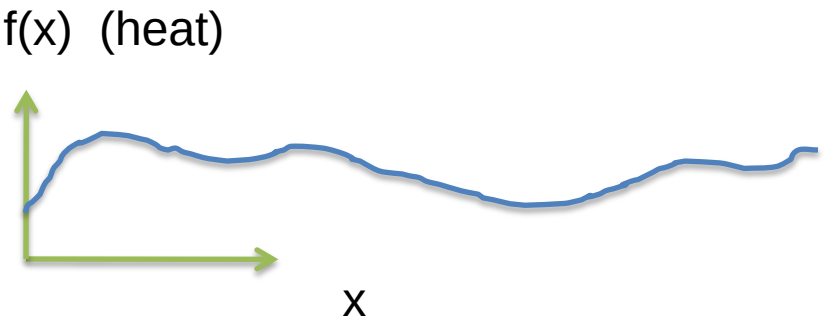


t = 0

Surface characterization

Heat equation

The heat equation: $\frac{\partial f}{\partial t} = -\Delta f$

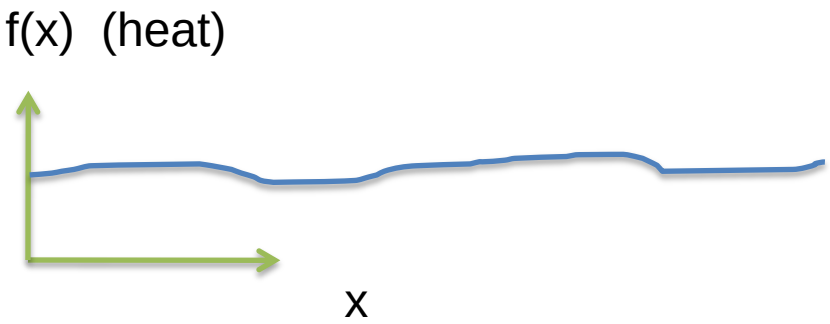


t = 100

Surface characterization

Heat equation

The heat equation: $\frac{\partial f}{\partial t} = -\Delta f$



t = 1000

Surface characterization

Heat equation

The heat equation: $\frac{\partial f}{\partial t} = -\Delta f$

Heat kernel: $K(t,x,y) = \sum e^{-\lambda_i t}\, \phi_i(x)\, \phi_i(y)$

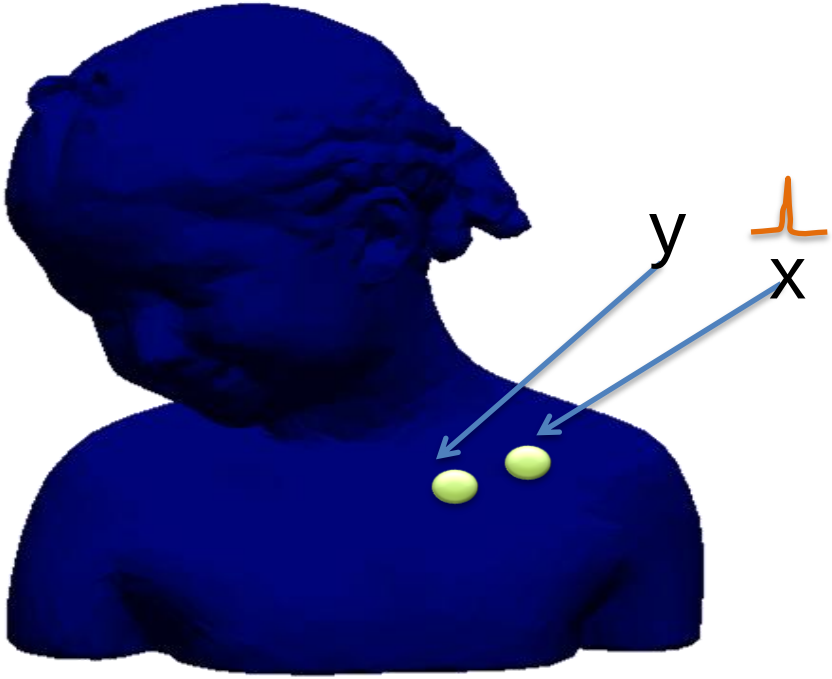
Solution of the heat equation: $f(t,x) = \int K(t,x,y)\, f(0,y)\, dy$

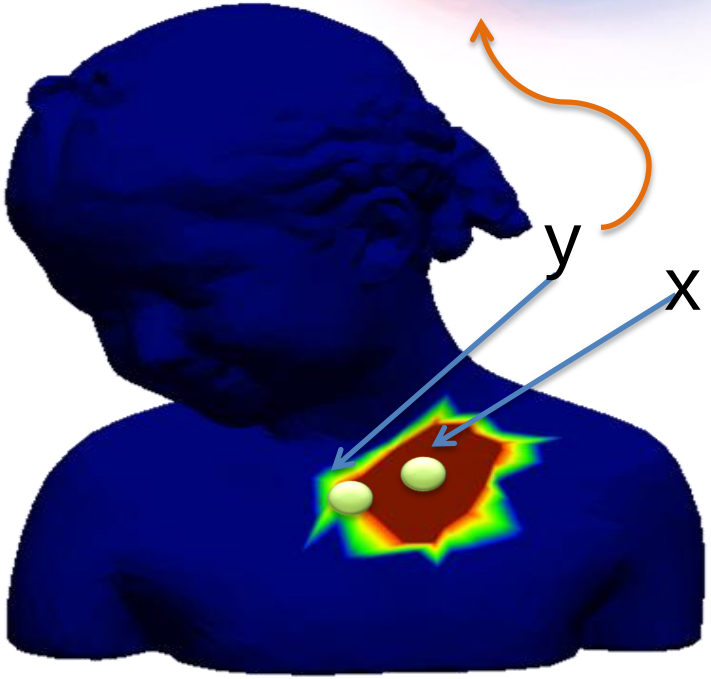
Surface characterization

Heat equation - What is the meaning of K(t,x,y) ?

How much heat do we get at y after t seconds ? K(t,x,y)

Initial time: we inject a Dirac of heat at x





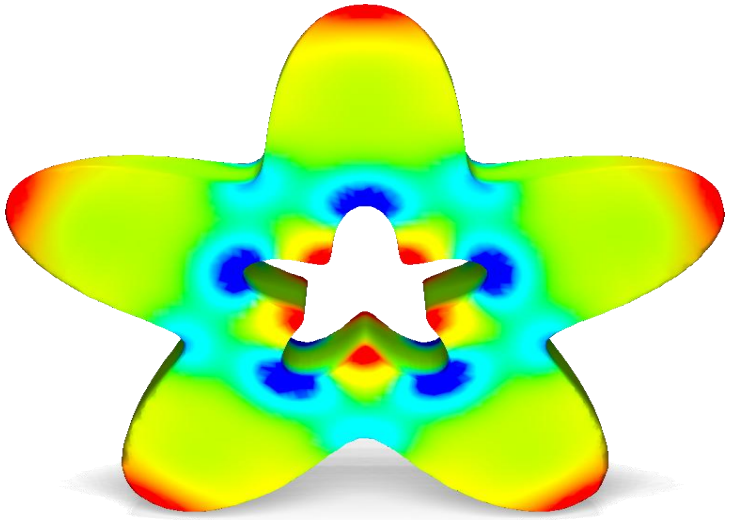
Surface characterization

Heat equation

Heat Kernel Signature [Sun, Ovsjanikov and Gubas 09]
Auto-diffusion [Gebal, Baerentzen, Anaes and Larsen 09]

How much heat remains at x after t seconds ?

$$ADF(t,x) = HKS(t,x) = K(t,x,x) = \sum e^{-\lambda_i t} \phi_i^2(x)$$



Applications:
- shape signature,
- segmentation using Morse decomposition,
- …

Summary

•Minimizing Rayleigh quotient instead of using « pinning »
enforces global constraints (moments) that avoid the trivial solution
- *•fieldler vector for streaming meshes [Isenburg et.al]*
- *•Spectral conformal parameterization [Muellen et.al]*

•The notion of fundamental solution plays a … fundamental role.
Strong connections with spectral analysis (and this is what Fourier invented Fourier analysis for !)
- *•Green function / Poisson equation - GPS coordinates [Rustamov]*
- *•Heat kernel signature ,[Sun et.al] / auto-diffusion function [Gebal et.al]*

•More to explore: the Variational Principle (see Wikipedia)