Eurographics Symposium on Geometry Processing 2009
Marc Alexa and Michael Kazhdan
(Guest Editors)

Volume 28 (2009), Number 5

Shape Analysis Using the Auto Diffusion Function

K. Gębal[1], J. A. Bærentzen[1], H. Aanæs[1], and R. Larsen[1]

[1] DTU Informatics, Technical University of Denmark, Denmark

Abstract

Scalar functions defined on manifold triangle meshes is a starting point for many geometry processing algorithms such as mesh parametrization, skeletonization, and segmentation. In this paper, we propose the Auto Diffusion Function (ADF) which is a linear combination of the eigenfunctions of the Laplace-Beltrami operator in a way that has a simple physical interpretation. The ADF of a given 3D object has a number of further desirable properties: Its extrema are generally at the tips of features of a given object, its gradients and level sets follow or encircle features, respectively, it is controlled by a single parameter which can be interpreted as feature scale, and, finally, the ADF is invariant to rigid and isometric deformations.

We describe the ADF and its properties in detail and compare it to other choices of scalar functions on manifolds. As an example of an application, we present a pose invariant, hierarchical skeletonization and segmentation algorithm which makes direct use of the ADF.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

Many algorithms for processing or analysis of manifold shapes are aided by scalar functions defined on the surface. It turns out that functions which are smooth and aligned with the shape are excellent tools for parametrization and extraction of topological information. In particular, harmonic functions, piecewise harmonic functions or eigenfunctions of the Laplace Beltrami operator have been used a great deal. However, in order to define a harmonic function on a mesh, its boundary conditions need to be specified (sources and sinks), and while the eigenfunctions of the Laplace Beltrami operator carry a great deal of shape information, it is often hard to find out precisely what each eigenfunction represents. In particular, the picture gets more complicated when eigenfunctions corresponding to eigenvectors larger than the first non-zero eigenvalue are used.

Our goal is to find a smooth scalar function which attains its maxima on the tips of features (tips of fingers, noses, tentacles, hoofs, tails, tops of heads, etc.). Since the notion of a feature is scale dependent, we observe that the function must be parameterized by a scale parameter, but it would defeat the purpose if the user had to specify other information (e.g. feature points). Furthermore, we desire that the function be invariant to any rigid transformation, scaling and to isometric deformations of the shape. We believe such a function would be highly useful for a number of tasks such as feature point detection, skeletonization, segmentation, and parametrization. However, in this paper, we focus on showing the usefulness in relation to skeletonization and segmentation.

1.1. Contributions

Our main contributions are to describe the *auto diffusion function* (ADF) which is a tool for shape analysis and to investigate its applications to geometry processing.

Assume we assign some quantity, like for example heat or dye, subject to diffusion to a single point on a manifold shape. For a point on this shape, x, and a scale (or time) parameter t, the $ADF_t(x)$ is the fraction of the quantity that remains at x after time t. Intuitively, it is fairly clear that the quantity that remains will be bigger on or near features since for a family of metric disks centered at x, the ratio of area of metric disk to its perimeter will be bigger for a feature point than for a point on a flat part of the shape. Thus, more of the initial quantity escapes in the flat case. Detailed description of ADF comes in Sections 3 and 4, see also [Ros97, CRD84, Ber03].

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

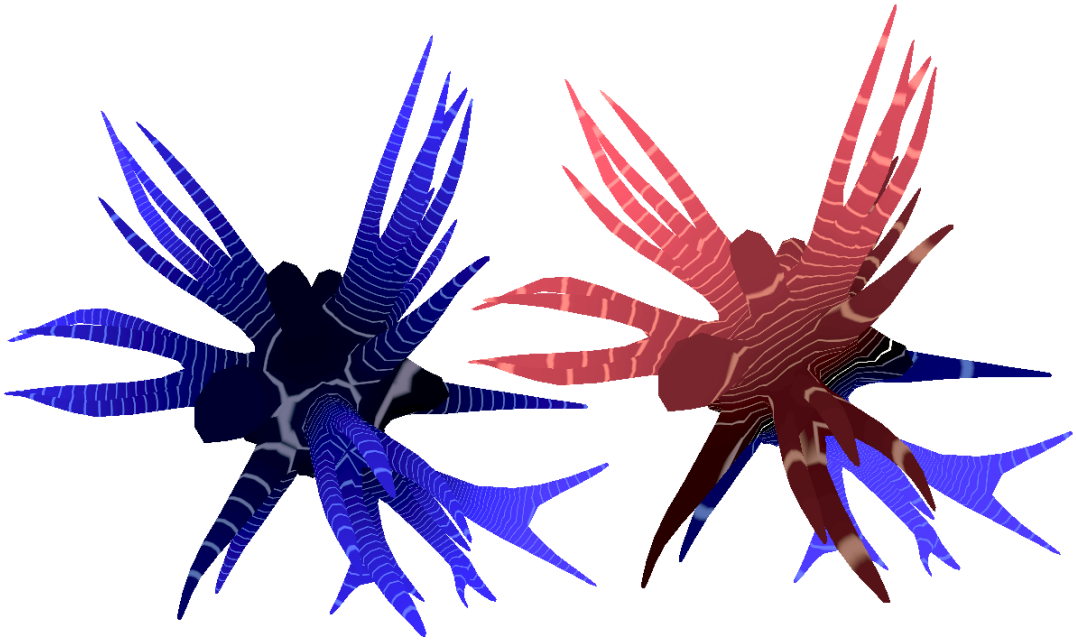

Figure 1: *ADF and Fielder functions for the tentacles shape. The rings are iso-curves and positive values map to intensity of blue while negative values map to intensity of red. Notice how the level sets of the ADF encircle the tentacles of the shape much more consistently than the Fiedler vector.*

In Figure 1 we show a complex shape with the ADF compared to the Fiedler vector (first eigenfunction corresponding to a non-zero eigenvalue). It is clear that all the tentacles of the shape are encircled by iso-curves of the ADF while that is not true of the Fiedler vector. We could have used a harmonic function, but compared to harmonic functions, the main advantage of the ADF is that it does not require the setting of any boundary conditions on feature points. Instead, its extrema prove to be the natural feature points.

We also propose an algorithm for skeletonization and segmentation based on the well known method of Reeb graphs applied to our ADF. Our contribution pertains to the fact that a single branch point often turns into multiple bifurcations on a Reeb graph, and each bifurcation tends to lie on or near the surface. We address this issue by averaging points on the Reeb graph within some small tolerance of a critical value. The final skeleton is combined from several Reeb Skeletons, which are extracted using the ADF function evaluated at different t-values.

2. Related work

2.1. Functions on surfaces

Ni et al. [NGH04] find a harmonic function by solving the Laplace equation with selected vertices as the constraints. This yields a smooth function that has maxima only at the given points, however it is necessary to provide the extrema in advance. Moreover, the values of the extrema need to be carefully assigned depending on the lengths of the protrusions. Otherwise, the saddle points will be misplaced from the natural branching area. The fast computation of harmonic functions in [XZCOX09] allows for an interactive approach where the user defines all of the constraint points and obtains a new harmonic function immediately. Dong et al. [DKG05] propose to use curve constraints as the boundary conditions. These constraints are imposed by requiring that all vertices which belong to the curve need to have the same value of the function. There exist some heuristic algorithms aimed at finding good constraints for harmonic functions [SF01, JWYG04, DKG05].

Given a single user provided source point, Aujay et al. [AHLD07] find the sinks as the clustered maxima of the function of shortest distance to the source. They create harmonic functions which have boundary conditions defined at the sources and sinks. Tierny [Tie08] finds feature points automatically as the intersection of two sets of points which, in turn, are the extrema of the geodesic distance to one of a pair of diametrically opposite points. The scalar function is then the geodesic distance to the nearest feature point. In [TVD07] the geodesic distance is modified by making areas separated by concavities, more distant. For more details see Section 4.1. In general, methods based on geodesic distance

are not always the best choice because they are very sensitive to the small topology changes as mentioned in [Rus07].

When using the ADF or eigenvectors of the LBO it is not necessary to provide extrema because they are defined by the function itself. The existing approaches based on eigensolutions of Laplace Beltrami operator usually involve choosing one of the eigenfunctions. Some of the papers [RBG*09, PSF08] suggest exactly that but without specifying which one to choose. [SLK*08] takes the first nontrivial eigenfunction and creates the skeleton based on that. However, this approach may overlook many shape details especially when the nontrivial eigenvalues are almost the same. [DBG*06] picks one of the Laplacian eigenfunctions according to a given number of critical points that the function produces. But in this way one imposes the complexity. For two shapes of very different complexity (say the tentacles shape in Figure 1) this would lead to very different levels of detail which is not likely to be desirable.

2.2. Skeletonization and segmentation

Many methods for producing a skeletonization require either a volumetric representation or a Voronoi diagram. Many of these are discussed in [ATC*08]. Most of them create a lot of tiny branches and are sensitive to noise, others have a big computational cost. Below, we restrict the discussion to those that directly work on the surface mesh since that is most pertinent to this paper. [PSF08, SLK*08] create a Reeb graph directly from the choosen LBO eigenfunctions and make a segmentation and shape skeleton from this graph. Au et al. [ATC*08] get the skeleton by shrinking the mesh using Laplacian smoothing. However it does not work for coarse meshes, and it does not seem possible to specify the desired level of detail.

Many segmentation algorithms are based on geometric properties of the parts, for example their convexity or the local curvatures. As we require pose invariance, we refer only to methods which meet this requirement. Katz et al [KT03] use geodesic distance and angular distance as a metric for the k-means clustering method. The number of clusters is obtained by taking the k that has the biggest derivative of minimal distance of the k-th added representative from other representatives. In a later paper [KLT05], Katz et al. obtain the pose invariance by using multi-dimensional scaling (MDS) to make the geodesic distances similar to the Euclidean distances in 3D space. The points that reside on the convex hull of this representation are chosen as the feature points, then spherical mirroring is used to extract the core component and the feature components. In the end, the boundaries of the segments are refined by finding the optimal cut. This method seems to work only with shapes that have a distinguishable core part.

Tierny et al [TVD07] use an approach based on Reeb graphs evaluated on geodesic and curvature based functions. The segmentation process is not only aided by the topological information but it is also enhanced by the placement of the constrictions – the curves at the bottlenecks of the shape which are good candidates for the boundaries of the segments. Some heuristic methods like the identification of the core part of the object is conducted in order to remove some of the constrictions and produce the final shape.

Spectral methods [ZvKD07], especially based on eigendecomposition of the Laplace Beltrami operator have pose invariance induced by the properties of the operator. Rustamov [Rus07] proposes k-means clustering based on inner products of points in GPS coordinates. The GPS coordinates of a vertex is the vector of values of the LBO eigenfunctions at that vertex where each value is divided by the square root of the corresponding eigenvalue. Reuter et al. [RBG*09] mention segmentation based on the nodal domains of some of the eigenfunctions. In both of those methods, the user needs to provide information. Liu et al. [LZ07] perform 2D contour analysis of the shape in the space of the first two eigenfunctions. The problem is that these may fail to capture even basic shape properties. In [HWAG09] modal analysis of the Hessian of the deformation energy is used to find the typical low energy deformations. Decomposition of the shape into parts is done by finding the optimal approximation of those deformations by deformations which are rigid for each segment.

[dGGV08] proposed to use the diffusion distance between points to make a shape segmentation. Like the ADF, the diffusion distance is defined in terms of the Gaussian kernel on the mesh, but the functions are otherwise very different: The diffusion distance only makes sense for two points since the diffusion distance from a point to itself is identically zero.

3. Theoretical background

3.1. The Laplace-Beltrami eigensolutions

The Laplace-Beltrami operator (LBO) is defined on a Riemannian manifold Ω as the divergence of the gradient of a scalar function $f : \Omega \to \mathbf{R}$.

$$\Delta f = \nabla \cdot (\nabla f)$$

The LBO can be found in equations describing physical phenomena such as wave propagation and heat distribution. This operator has also been used extensively by the computer graphics community in the last few years for many purposes such as: mesh smoothing [NISA06], parameterization [PP93, MTAD08], editing [SCOL*04], morphing and deformation [Ale03]. The decomposition [Lev06] of the LBO into eigenvalues ($0 \leq \lambda_0 \leq \lambda_1 \leq \dots$) and eigenfunctions ($\phi_0, \phi_1, \dots$), which can be expressed as the solution of the Helmholtz equation $\Delta f + \lambda f = 0$, has also lately been explored in a lot of applications for geometry processing including shape identification [RWP05], classifica-

tion [Rus07], segmentation [LZ07], registration [MHK*08] or identifying the shape symmetries [OSG08].

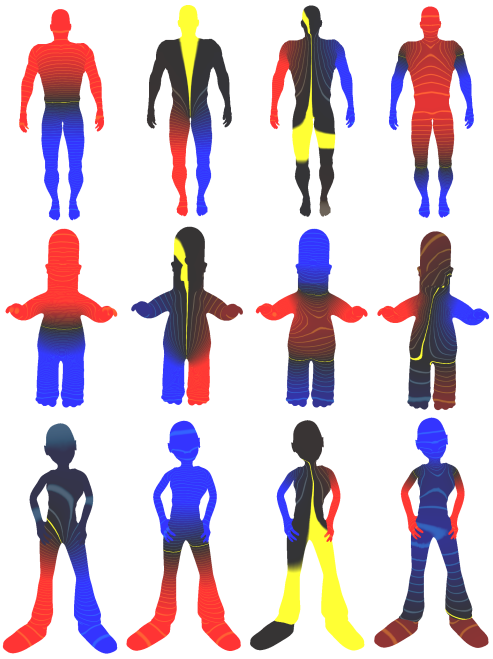

Figure 2: *First four nontrivial eigenfunctions of the LBO for three different human like shapes. Red color represent negative and blue positive values, yellow shows area close to the nodal set. The order of the eigenfunctions depends on the proportions of the specific parts. Note that the nodal sets do not always cut the object symmetrically.*

The eigenfunctions corresponding to the first few eigenvalues align surprisingly well with the protrusions and features of the object. This phenomenon can be explained by investigating the Rayleigh Quotient method which is used for calculating the eigensolutions to the symmetric operator, which in the case of the Helmholtz equation is equal to $-\Delta$. The problem is stated as a minimization problem:

$$\phi_i = \underset{\substack{0 \neq f_i \in C_0^2(\Omega) \\ \langle \phi_{j \in \{0..i-1\}}, f_i \rangle = 0}}{\arg\min} \frac{\langle f_i, -\Delta f_i \rangle}{\langle f_i, f_i \rangle}$$

The minimal value is equal to the corresponding eigenvalue λ_i. By applying the fact [CRD84] that for the compact manifold the divergence and minus gradient are formal adjoint operators $\langle \nabla f, \mathbf{X} \rangle_v = -\langle f, \nabla \cdot \mathbf{X} \rangle$ to the vector field $\mathbf{X}$ equal to ∇f [BN02], we transform this problem into the mimimization of the $\langle \nabla f_i, \nabla f_i \rangle_v$, which is equivalent to the Dirichlet Energy minimalization problem with the constraints $\langle f_i, f_j \rangle = \delta_{ij}$, where δ_{ij} is here the Kronecker delta. Note that we use here two different inner products: the first one $\langle f, g \rangle = \int_\Omega f(x) g(x) dx$ is the inner product of two scalar functions over our manifold Ω, and the second one $\langle \mathbf{X}, \mathbf{Y} \rangle_v = \int_\Omega \langle \mathbf{X}(x), \mathbf{Y}(x) \rangle_\Omega dx$ is the inner product of two vector fields, where $\langle , \rangle_\Omega$ is the inner product defined by the structure of our Riemmanian manifold Ω [Ros97].

Put more loosely, the eigenfunctions are mutually orthogonal, have as small as possible gradients everywhere while $\langle \phi_i, \phi_i \rangle = 1$. For a closed surface, a constant function will suffice as ϕ_0. However, ϕ_1, the Fiedler vector, must be orthogonal to ϕ_0 and consequently it is positive on half the shape and negative on the other half. The requirement that the gradients should be as small as possible translates into the well known fact that the direction of change of the Fiedler vector naturally follows the shape [Lev06] as illustrated in Figure 2 left column.

There is an analogy between the eigenfunctions of the LBO and harmonic functions mentioned in Section 2.1. Both minimize the Dirichlet energy, but the eigenfunctions are constrained by the orthogonality requirement rather than boundary conditions.

While using the LBO eigenfunctions, there are some important issues that need to be remembered: The signs of eigenvectors are undefined, two eigenvectors may be swapped, nodal sets can be unstable due to small metric changes. All of these are addressed by the Auto Diffusion Function presented below. In our implementation Ω is represented as a triangular mesh. We use the cotan weights [PP93, DHLM05] to calculate the entries for the LBO and we solve a generalized eigenproblem in a way similar to the one used by Rustamov [Rus07]. Because we are interested mostly in the first few hundreds of eigenvalues we use the sparse solver ARPACK together with SuperLU.

3.2. The diffusion kernel

The diffusion kernel $K(x,y,t)$, or heat kernel, is a fundamental solution (Green's function) to the heat equation:

$$(\Delta_x + \partial_t) u(x,t) = 0$$

where Δ_x denotes the Laplace Beltrami operator acting on the spatial variable x where $t \in [0, \infty)$ is the time variable. It solves the equation with the initial condition $u(0,x) = \delta(y)$, where $\delta(y)$ is Dirac delta function at the position y, which means that all heat is initially concentrated in one point at the position of y. The general solution can be obtained by convolution of the heat kernel with the initial condition $g(x) = u(0,x)$. The heat kernel can be expressed [Ros97, page 32] in the terms of LBO eigensolutions as:

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

4. The Auto Diffusion Function and its interpretation

If we inject a unit amount of some quantity like dye at a point x (this corresponds to the initial Dirac delta function),

the Auto Diffusion Function indicates how much of the dye that remains after time t. So the ADF describes the diffusion from the point to itself and can be written as:

$$K(x,x,t) = \sum_{i=0}^{\infty} e^{-\lambda_i t} \phi_i^2(x)$$

In this way, we obtain a function which is only dependent on the eigenvectors and eigenvalues of LBO, which, in turn, depend solely on the first fundamental form G, and is therefore independent of isometric deformations, translations, and rotations. To make the function scale invariant we can simply divide the exponential by the second eigenvalue [Reu06]. So we define the Auto Diffusion Function as:

$$ADF_t(x) = K(x,x,\frac{t}{\lambda_1}) = \sum_{i=0}^{\infty} e^{-t\frac{\lambda_i}{\lambda_1}} \phi_i^2(x) \qquad (1)$$

As the equation shows, the ADF is simply a linear combination of squared LBO eigenfunctions. If the parameter t is large, the eigenfunctions corresponding to big eigenvalues count less, and then only the main features 'detected' by the smaller eigenvalues influence the ADF. As we decrease t, more features can be seen (cf. Figure 3).

The physical interpretation of the ADF is given by the diffusion process: At the tips of protrusions, less of the dye will escape than from flat areas. But if t is big enough, and the protrusions are small, there will be enough time for the dye to spread evenly to neighboring areas. On the other hand, for small t there must be a connection between local Gaussian curvature and (1). This connection can be made explicit by observing the Minakshisundaram-Pleijel expansion of the heat kernel. For sufficiently close x and y

$$K(x,y,t) = (4\pi t)^{-n/2} \sum_{k=0}^{\infty} u_k(x,y)\, t^k$$

where n is the dimension of the Riemannian manifold and $u_k(x,y)$ are recursively defined. If $y = x$ then $u_0(x,x) = 1$ and $u_1(x,x) = \frac{1}{6}S(x)$, where $S(x)$ is the scalar curvature which is the Gaussian curvature in the case of a 2-manifold. Therefore, we have

$$ADF_{t\lambda_1}(x) = (4\pi)^{-1}\left(\frac{1}{t} + \frac{S(x)}{6}\right) + O(t) \qquad (2)$$

However, (2) is not a viable alternative to (1) since the truncated terms are insignificant only for small t. Conversely, (1) is faster to compute for bigger t since fewer high frequency eigenfunctions are needed.

In practice, we do not include the first eigenfunction since it is constant, and we only add eigenvectors corresponding to eigenvalues that fulfill $e^{-t\frac{\lambda_i}{\lambda_1}} < \delta$, where the threshold is $\delta = 0.01$.

4.1. Comparison to the existing functions

In this section, we compare the ADF to some other functions as illustrated in Figure 4. We concentrate on those functions

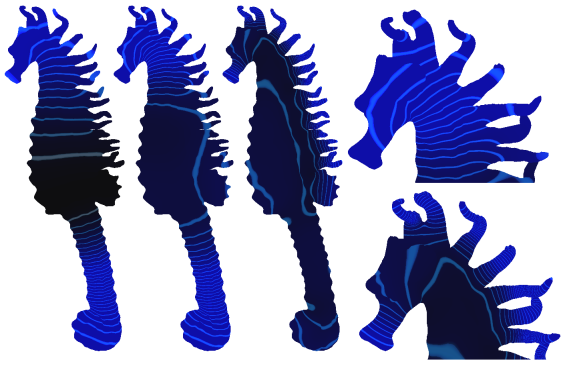

Figure 3: *ADF evaluated with t equal to respectively* $2, \frac{1}{2}$, *and* $\frac{1}{32}$ *(left to right). In the magnified images t is* $\frac{1}{2}$ *(top) and* $\frac{1}{32}$ *(bottom). The lines show the isocontours of the function; brighter color indicates bigger value of the ADF.*

that can be specified with no direct annotation of feature points and are pose invariant.

From our point of view, using the geodesic distance to the closest feature entails two problems. First, the function has first order discontinuities since it involves taking the minimum of several geodesic distances. The second issue occurs when some feature point is located at a much smaller protrusion than a neighboring protrusion. Then, most of the surface at the 'parent area' of the protrusion is geodesically closest to the feature point of the smallest protrusion. This can lead to a situation where the function at the 'parent area' is mostly defined as the distance to the point corresponding to the smallest detail. Observe what happens with the isocurves close to the little finger on the last two hands on the image 4 that represent Tierny's geodesic functions. Even with the geodesic distance enhanced by the curvature information the problem is still present.

The Fiedler vector aligns well with the overall shape, but some parts of the mesh close to the zero level set are poorly aligned with the protrusions. One general problem with using the Fiedler vector is that for highly symmetric objects, the smallest eigenvalues will be nearly identical, and the first few eigenvectors align with directions of roughly the same significance. The advantage of the ADF is that the weights of the eigenfunctions depend on the eigenvalues, so all of the first few eigenvectors may contribute. As noted previously, for small values of t, the ADF starts to resemble Gaussian curvature.

5. Feature Based Skeletonization and Segmentation

5.1. Reeb Graphs

Calculation of the Reeb graph is a natural application of functions defined over a manifold. Given a manifold Ω and a function $f : \Omega \to \mathbf{R}$, p is a critical point of f if $\nabla(f(p)) = 0$.

Figure 4: *Different functions calculated on the human hand. The first two top represent ADF for t=1 and t=0.1, the bottom two are the Fiedler vector and highly smoothed Gaussian curvature, and the last two are geodesic based (pure geodesic and enhanced by the curvature) from [Tie08] (pictures used with the author's permission).*

f is a Morse function when for each critical point p, the Hessian martix $H(f(p))$ of the second order derivatives of f is non-singular, which means that critical points of f are non-degenarate.

The Reeb graph [ABS03] of a Morse function f on a manifold Ω traces the evolution of the level sets of the function on the manifold and is the quotient space of $\Omega \times \mathbf{R}$ defined by the equivalence relation $(p, f(p)) \sim (q, f(q)) \Leftrightarrow f(p) = f(q)$ and p and q are in the same connected component of $f^{-1}(f(p))$.

The nodes of the Reeb graph correspond to the critical points of the function f and the arcs, which we call the Reeb edges, represent the connections between them. The Reeb graph itself is a topological construct. However, if the manifold Ω is embeded in Euclidean space, the point corresponding to each connected component of the level set function can be positioned at the center of mass of the corresponding component of the level set curve. This we denote the Reeb skeleton.

In the discrete case where the manifold is a triangular mesh, the function f is defined over the vertices. The points can be classified as regular or critical according only to the function values of the vertices that belong to the one ring of the vertex. The critical points are sinks if they are at the maxima, sources if there are at minima and saddles otherwise. The full description of the point classification according to the one-ring of the vertex can be found in [CMEH*03]. Our Reeb graph computation is done using a sweep algorithm that allows us to position the skeleton points in space. The algorithm is similar to the one described in [CMEH*03]. If the positioning of the vertices is not needed, a faster algorithm, that analyses the iso-contours only at the critical points [PSF08], can be used.

5.2. Algorithm Outline

Reeb skeletons tend to look nicest at points which are not close to branch points. Branch points have a tendency to be either on or very close to the surface rather than inside the shape where one would normally place a branch point. Also, where a designer would typically have placed a single branch point, Reeb skeletons tend to have multiple bifurcations. We address both problems by defining a small interval around the critical function value corresponding to a branch point. All points on the Reeb graph corresponding to values within this interval are averaged into a single point which we denote a *joint*. The remainders of the edges of the Reeb graph are denoted *bones*.

The method is somewhat similar to the extended Reeb graph method [ABS03, BFS00] which slices the shape into parts that have function values between probing points according to a given frequency. Both methods work on critical areas instead of the critical points which helps avoid degenerate situations. Critical areas for saddle points correspond to our *joint* areas, but in our case the critical points are placed in the middle of the sliced interval, and neighboring critical areas that have small differences in function values (for a given t parameter of the ADF function) are merged together in order to avoid topological noise.

The skeleton is combined from several Reeb skeletons which are extracted with ADF_t evaluated with different t-values. Usually, we refine a skeleton by computing a new one at $t \leftarrow \frac{t}{2}$ and then grafting details from the new fine skeleton onto the coarse skeleton. We could also just compute a familiy of skeletons using different t values for the ADF function, but this would make it hard to absolutely guarantee that we obtain a hierarchy, i.e. that there are no details in the coarse skeletons which later disappear in a fine skeleton.

5.3. Extracting *joints* and *bones* from the Reeb skeleton

We denote as the *Reeb edge area E* all the points that are transfered into the same arc of the Reeb graph. Each Reeb edge has two critical points at the endpoints, the smaller one p_s^E and the bigger one p_e^E. So for each point q that belongs to the given edge area $ADF_t(p_s^E) \leq ADF_t(q) \leq ADF_t(p_e^E)$. From each edge area, two *joint* areas are extracted at both ends and they are defined as:

$$J_s = \{q \in E : ADF_t(p_s^E) + \varepsilon \geq ADF_t(q)\}$$
$$J_e = \{q \in E : ADF_t(p_e^E) - \varepsilon \leq ADF_t(q)\}$$

The threshold epsilon used in those equation is:

$$\varepsilon = \kappa\mu = \frac{\kappa}{N}\sum_{i=0}^{\infty} e^{-t\frac{\lambda_i}{\lambda_1}}$$

where N is the number of vertices in the mesh and κ is a user defined threshold, we use $\kappa = 0.05$, which works fine, but can be changed according to the user preferences. μ is the hypothetical value of the ADF_t if all of the eigenfunctions were constant. Such an objective formulation helps in an intuitive threshold manipulation as κ can be the same for ADFs with different t values and different shapes.

Neighbouring *joint* areas are glued together into one *joint* area. If the edge is small $|ADF_t(p_s^E) - ADF_t(p_e^E)| < 2\varepsilon$ then the whole edge area is added into the *joint* area and the two critical points that are the endpoints of this edge are merged together. If a *joint* contains sinks, this means that some geometric details, such as fingers, are not fully captured, because of the threshold. Such joints are indicated as *improvement*. Edge areas from the edge not contained in the *joint* areas are called *bones*.

An edge area E which has a sink at the end p_e^E is treated in a special way since only the p_s^E is a joint endpoint. However, if the $|ADF_t(p_s^E) - ADF_t(p_e^E)| < 3\varepsilon$ the whole edge is added to the edge area.

All the sink edges are integrated into the joint. This kind of edge is defined recursively as an edge that has no brother edges and its parent is a sink edge or it has no parent. For the parent P and the brother B the critical points are shared in a way that: $p_e^P = p_s^E$ and $p_s^B = p_s^E$.

In this way we get a graph structure with the *bones* being the edges of the graph and the *joints* being vertices. An edge is incident with a vertex if its Reeb edge area have *joint* areas belonging to the *joint* areas corresponding to this vertex.

5.4. Integrating *joints* and *bones* from different Reeb graphs

As described, the algorithm produces a skeletal structure at a single LOD, but it is possible to go to a finer level of detail by refining the *improvement joints*: The structures from two Reeb graphs G_1, G_2 corresponding to different parameters $t_1 > t_2$ are merged together in the following way. If there is a new *bone* from G_2 that has both endpoints at the same *improvement joint* area from G_1 then this edge is grafted onto that *improvement joint*. The *bone* area is taken off from the *joint*. If the *bone* disconnects the joint then new joints are created and the edges are carefully connected to their incident joints. Refinement is illustrated in Figure 5.

The operation of merging new Reeb graphs with the existing structure can be repeated with gradually smaller values of t, until we have no *improvement joints* or up to the desired level of detail.

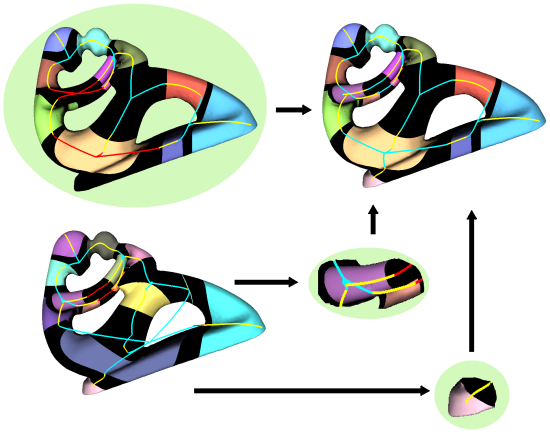

Figure 5: *Ilustration of the process of merging two different graphs. Joint areas on the mesh are marked black. Improvement joints are red, regular joints blue, and bones yellow. The top left coarse mesh is connected with the bottom left one. From the detailed mesh only those edges are taken which belong to the improvement joints area (middle and bottom right).*

5.5. Defining the final skeleton and segmentation

The geometry of the skeleton is computed as follows: For each *joint* we calculate the center of mass of the *joint* area, which we call the *joint* center, and then we connect to it the incident *bones*. If we are at a one-parent-many-children *joint* then it looks more pleasant if the *joint* center is moved towards the parent, so in that case we calculate the *joint* center as the center of parent *joint* area. The parts of the skeleton corresponding to the *bones* are the centers of the level sets. However, the bones need to be connected to the joint: The parts corresponding to the *joint* areas of the edges are a weighted average of the centers of level sets and the *joint* center. The closer we are to the critical point the more position of the *joint* center counts so finally for the critical point the position is equal to the position of the *joint* center.

When doing segmentation, we integrate some of the *joint* areas with the bone areas. If the *joint* connects only two *bones* then there is no branching at this level of detail. In that case those two *bone* areas and the *joint* area are connected together into one segment. The *joint* area is also integrated into a *bone* area if it has only one *bone* connection. In the one-parent-many-children case we connect the *joint* node to the parent *bone* area. In other cases we leave the *joint* area as a separate segment.

5.6. Parameters

The level of detail can be controlled by choosing the t and κ values. Especially setting the t for the basic skeleton is important as new edges can only be inserted into a joint that

Figure 6: *Segmented poses of the Armadillo. On the top, the segmentations have been created from a coarse skeleton ($t = 1, \kappa = 0.04$) and below we see the result after refining the improvement joints ($t = 0.25$). Note that there is some noise added to the last pose of the Armadillo to demonstrate insensitivity to noise.*

is marked as an *improvement area*. Those areas are detected if there are maxima inside. For bigger t, the ADF tends to have less maxima than for smaller values of t. κ controls whether features that are not very thin, like humps, can be transformed into bones. Such a feature would usually produce an edge of the Reeb graph with a small difference between minimal and maximal function values, compared to thin features.

The other measure of level of detail - the feature length - is controlled by the t for the smallest added skeleton.

The described parameters are not shape specific but rather general. For a defined set of parameters, all of the shapes can be processed by the skeletonization algorithm. Then the results can be the base for other geometric tasks - for example as a good similarity measure between the shapes.

6. Discussion and Future Work

We have proposed the ADF as an effective tool for shape analysis. It is governed by only one parameter, feature scale, which is arguably indispensable, and its construction requires nearly nothing more than a framework for computing eigenfunctions of the Laplace Beltrami Operator. Thus, it is also simple. We have also explored the application of the ADF to the related tasks of skeletonization which is shown in Figure 7 and segmentation which is shown in Figure 6. The

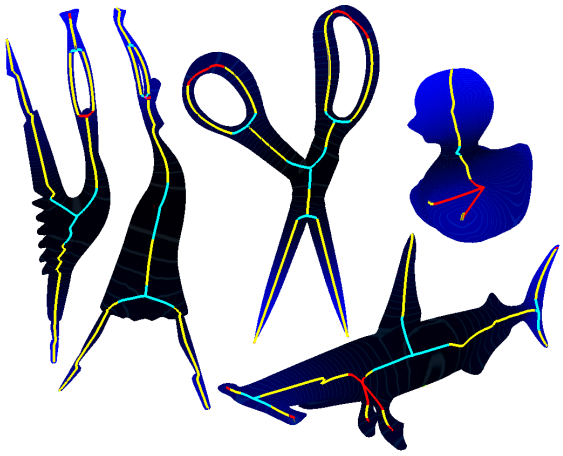

Figure 7: *Skeletons produced for different meshes with $t = 0.5$ and $\kappa = 0.04$ drawn on top of the ADF for the mesh. Improvement joints are red, regular joints blue, and bones yellow.*

result is an algorithm which has a controllable level of detail, insensitivity to noise and invariance to scale, rigid transformation and pose invariance. The last properties being inherited from the ADF.

In the future, we would like to explore other applications of the ADF. For instance matching features extracted as ADF maxima between shapes, and parameterization of shapes.

7. Acknowledgments

The 3D models used are provided courtesy of the Princeton and Aim@Shape repositories. We are very grateful to Steen Markvorsen for pointing out the connection to the Minakshisundaram-Pleijel expansion, to Julien Tierny for allowing us to use two of his images, and to the anonymous reviewers for insightful comments.

References

[ABS03] ATTENE M., BIASOTTI S., SPAGNUOLO M.: Shape understanding by contour-driven retiling. *The Visual Computer 19*, 2-3 (2003), 127–138.

[AHLD07] AUJAY G., HÉTROY F., LAZARUS F., DEPRAZ C.: Harmonic skeleton for realistic character animation. In *Symposium on Computer Animation, SCA 07, August, 2007* (San Diego, California, Etats-Unis, 2007), Metaxas D., Popović J., (Eds.), ACM-Siggraph/Eurographics, pp. 151–160.

[Ale03] ALEXA M.: Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2-3 (2003), 105–114.

[ATC*08] AU O. K.-C., TAI C.-L., CHU H.-K., COHEN-OR D., LEE T.-Y.: Skeleton extraction by mesh contraction. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–10.

[Ber03] BERGER M.: *A panoramic view of Riemannian geometry*. Springer, 2003.

[BFS00] BIASOTTI S., FALCIDIENO B., SPAGNUOLO M.: Extended reeb graphs for surface understanding and description. In *DGCI '00: Proceedings of the 9th International Conference on Discrete Geometry for Computer Imagery* (London, UK, 2000), Springer-Verlag, pp. 185–197.

[BN02] BELKIN M., NIYOGI P.: Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14* (2002), MIT Press, pp. 585–591.

[CMEH*03] COLE-MCLAUGHLIN K., EDELSBRUNNER H., HAREER J., NATARAJAN V., PASCUCCI V.: Loops in reeb graphs of 2-manifolds. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry* (New York, NY, USA, 2003), ACM, pp. 344–350.

[CRD84] CHAVEL I., RANDOL B., DODZIUK J.: *Eigenvalues in Riemannian geometry*. Acad. Press, 1984.

[DBG*06] DONG S., BREMER P.-T., GARLAND M., PASCUCCI V., HART J. C.: Spectral surface quadrangulation. *ACM Trans. Graph. 25*, 3 (2006), 1057–1066.

[dGGV08] DE GOES F., GOLDENSTEIN S., VELHO L.: A hierarchical segmentation of articulated bodies. *Comput. Graph. Forum 27*, 5 (2008), 1349–1356.

[DHLM05] DESBRUN M., HIRANI A. N., LEOK M., MARSDEN J. E.: Discrete exterior calculus. http://arxiv.org/abs/math?papernum=0508341, Aug 2005.

[DKG05] DONG S., KIRCHER S., GARLAND M.: Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Comput. Aided Geom. Des. 22*, 5 (2005), 392–423.

[HWAG09] HUANG Q., WICKE M., ADAMS B., GUIBAS L.: Shape decomposition using modal analysis. *Computer Graphics Forum 28*, 2 (2009). to appear.

[JWYG04] JIN M., WANG Y., YAU S.-T., GU X.: Optimal global conformal surface parameterization. In *VIS '04: Proceedings of the conference on Visualization '04* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 267–274.

[KLT05] KATZ S., LEIFMAN G., TAL A.: Mesh segmentation using feature point and core extraction. *The Visual Computer 21*, 8-10 (2005), 649–658.

[KT03] KATZ S., TAL A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics 22*, 3 (July 2003), 954–961.

[Lev06] LEVY B.: Laplace-beltrami eigenfunctions: Towards an algorithm that understands geometry. In *IEEE International Conference on Shape Modeling and Applications, invited talk* (2006).

[LZ07] LIU R., ZHANG H.: Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum (Special Issue of Eurographics 2007) 26*, 3 (2007), 385–394.

[MHK*08] MATEUS D., HORAUD R. P., KNOSSOW D., CUZZOLIN F., BOYER E.: Articulated shape matching using laplacian eigenfunctions and unsupervised point registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2008).

[MTAD08] MULLEN P., TONG Y., ALLIEZ P., DESBRUN M.: Spectral conformal parameterization. *Computer Graphics Forum 27*, 5 (July 2008), 1487–1494.

[NGH04] NI X., GARLAND M., HART J. C.: Fair morse functions for extracting the topological structure of a surface mesh. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 613–622.

[NISA06] NEALEN A., IGARASHI T., SORKINE O., ALEXA M.: Laplacian mesh optimization. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia* (New York, NY, USA, 2006), ACM, pp. 381–389.

[OSG08] OVSJANIKOV M., SUN J., GUIBAS L.: Global intrinsic symmetries of shapes. *Computer Graphics Forum 27*, 5 (July 2008), 1341–1348.

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2* (1993), 15–36.

[PSF08] PATANE G., SPAGNUOLO M., FALCIDIENO B.: Reeb graph computation based on a minimal contouring. *2008 IEEE International Conference on Shape Modeling and Applications* (2008), 73–82.

[RBG*09] REUTER M., BIASOTTI S., GIORGI D., PATANE G., SPAGNUOLO M.: Discrete laplace-beltrami operators for shape analysis and segmentation.

[Reu06] REUTER M.: *Laplace Spectra for Shape Recognition*. Books on Demand GmbH, 2006.

[Ros97] ROSENBERG S.: *The Laplacian on a Riemannian Manifold*. No. 31 in London Mathematical Society Student Texts. Cambridge University Press, 1997.

[Rus07] RUSTAMOV R. M.: Laplace-beltrami eigenfunctions for deformation invariant shape representation. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 225–233.

[RWP05] REUTER M., WOLTER F.-E., PEINECKE N.: Laplace-spectra as fingerprints for shape matching. In *SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling* (New York, NY, USA, 2005), ACM, pp. 101–106.

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (New York, NY, USA, 2004), ACM, pp. 175–184.

[SF01] STEINER D., FISCHER A.: Topology recognition of 3d closed freeform objects based on topological graphs. In *PG '01: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2001), IEEE Computer Society, p. 82.

[SLK*08] SHI Y., LAI R., KRISHNA S., SICOTTE N., DINOV I., TOGA A.: Anisotropic laplace-beltrami eigenmaps: Bridging reeb graphs and skeletons. pp. 1–7.

[Tie08] TIERNY J.: *Reeb graph based 3D shape modeling and applications*. PhD thesis, TELECOM Lille 1, Universite des Sciences et Technologies de Lille, 2008.

[TVD07] TIERNY J., VANDEBORRE J.-P., DAOUDI M.: Topology driven 3D mesh hierarchical segmentation. In *IEEE International Conference on Shape Modeling and Applications (SMI 2007)* (Lyon, France, 2007), pp. 215–220.

[XZCOX09] XU K., ZHANG H., COHEN-OR D., XIONG Y.: Dynamic harmonic fields for surface processing. *Computers and Graphics (Special Issue of Shape Modeling International) 33* (2009), 391–398.

[ZvKD07] ZHANG H., VAN KAICK O., DYER R.: Spectral methods for mesh processing and analysis. In *Proc. of Eurographics State-of-the-art Report* (2007), pp. 1–22.