STAG: Smart Tools and Applications in Graphics (2017)
A. Giachetti, P. Pingi and F. Stanco (Editors)

Heat Flow Based Relaxation of n Dimensional Discrete Hyper Surfaces

Marco Livesu

CNR IMATI (Genoa, Italy)

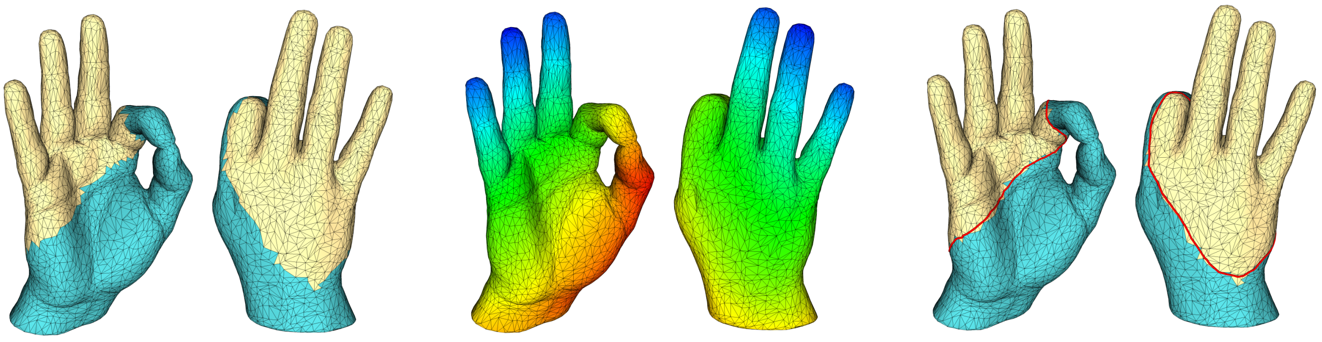

Figure 1: *Boundaries between adjacent regions in discrete segmentations can be geometrically poor, exposing a typical jagged behaviour (left). We relax discrete boundaries by approximating them with level sets (right, red curves) of continuous functions (middle).*

Abstract

We consider the problem of relaxing a discrete $(n-1)$ dimensional hyper surface defining the boundary between two adjacent n dimensional regions in a discrete segmentation. This problem often occurs in computer graphics and vision, where objects are represented by discrete entities such as pixel/voxel grids or polygonal/polyhedral meshes. A common approach consists in assigning to each element of the domain a value (or label). Elements sharing the same label belong to the same region, whereas elements with different labels belong to different regions. Segmentation boundaries are therefore only intrinsically defined, and amount to the union of the interfaces between adjacent elements having different label, which tend to be geometrically poor and expose a typical jagged behavior. We propose a relaxation scheme that replaces the original boundary with a smoother version of it, defined as the level set of a continuous function. The problem has already been considered in recent years, but current methods are specifically designed to relax curves on discrete 2-manifolds embedded in $\mathbb{R}^3$, and do not clearly scale to multiple discrete representations or to higher dimensions. Our biggest contribution is a smoothing operator that is based only on three canonical differential operators: namely the Laplacian, gradient and divergence. These operators are ubiquitous in applied mathematics, are available for a variety of discretization choices, and exist in any dimension. To the best of the author's knowledge, this is the first intrinsically dimension-independent method, and can be used to relax curves on 2-manifolds, surfaces in $\mathbb{R}^3$, or even hyper-surfaces in $\mathbb{R}^n$. As such, not only it is useful to refine the boundaries of discrete segmentations, but also for applications like data mining, where clustering in high dimensional spaces often occur, and the refinement of the clusters' boundaries may be beneficial for classification algorithms.

1. Introduction

Labeling (or *segmenting*) an object is a fundamental operation in computer graphics and vision, widely used in applications such as analysis of medical images, object recognition and detection.

The majority of segmentation algorithms work on discrete domains, such as regular grids [BVZ01], polygonal [LVS*13, RBG*09, MPS*04] and polyhedral [LAPS17, AMSF08] meshes. A common approach consists in assigning to each element of the domain a value (or *label*). Elements sharing the same label belong

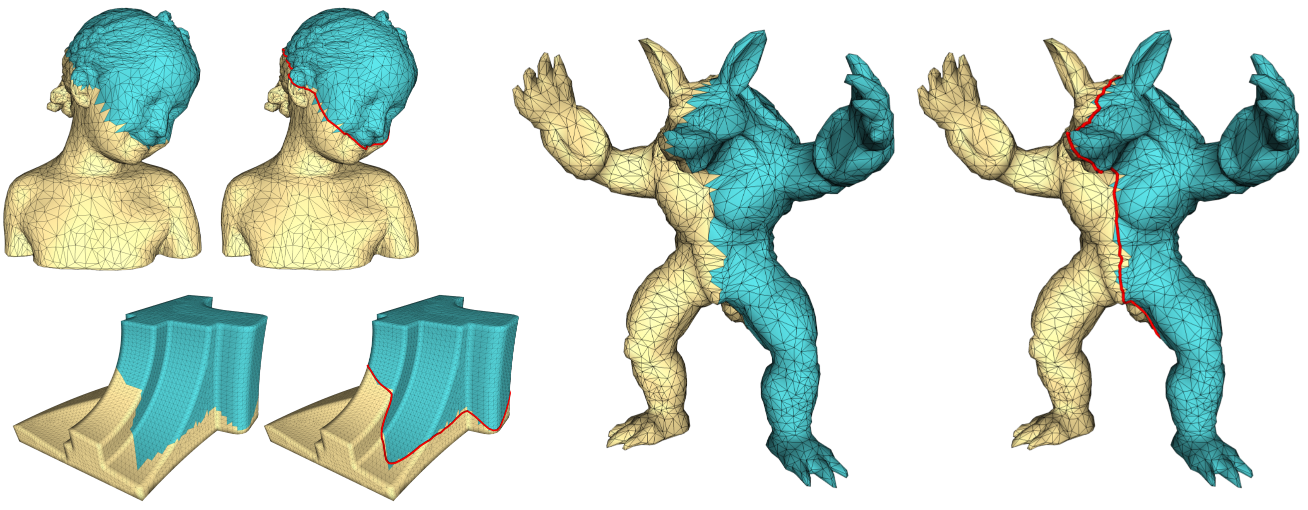

Figure 2: *A gallery of results produced using our method.*

to the same region, whereas elements with different labels belong to separate regions. As a consequence, boundaries between adjacent regions are only intrinsically defined, and amount to the union of the interfaces between adjacent elements.

Given a n dimensional object and a labeling defined on it, the boundary between adjacent regions is a $(n-1)$ dimensional hyper surface. As a concrete example one may consider a binary partition of a discrete two dimensional surface (e.g. a triangle mesh): the boundary between the two regions is the chain of edges having polygons with opposite labels at its sides. The same goes for three-dimensional objects (e.g. a tetrahedral mesh): the boundary is the set of faces having polyhedra with opposite labels at its two sides. Indeed, the boundary is one dimensional if the object is two dimensional, and is two dimensional if the object is three dimensional.

Depending on the quality of the discretization, both in terms of number, regularity, and shape of each element in the domain, the boundaries between adjacent regions can be geometrically poor, showing a typical jagged behaviour (Figure 1). In this article we focus on this very specific problem, and propose a neat method to relax jagged discrete boundaries, replacing them with smooth, yet geometrically faithful, versions of them.

Our most important contribution is a method which is completely agnostic both on the discrete representation used for the domain, and on the dimension of the space in which it operates. To do so, we define our smooth hyper surfaces as level sets of continuous functions. We take inspiration from [CWW13], and generate such functions relying on three classical discrete differential operators: namely the laplacian, the gradient and the divergence. These operators are ubiquitous in applied mathematics, have been implemented for a variety of discretization choices, and exist in any dimension.

Our method is efficient and easy to implement. We believe it has great potential not only for classical applications, like refining the boundaries of a discrete segmentation, but also for applications like data mining, where clustering problems in high dimensional spaces often occur, and the refinement of the clusters' boundaries may be beneficial for classification algorithms.

2. Related Works

The problem of having jagged boundaries in discrete segmentations is well known in literature. Most of the segmentation algorithms are not capable of producing smooth boundaries [Sha08]. An exception are the concavity-aware segmentations [AZC*12], which exploit harmonic functions to natively generate smooth cuts. Some other methods achieve boundary smoothness in post processing, implementing additional steps in their pipeline [JLCW06,LLS*04]. Whenever the segmentation algorithm of choice is not capable of producing smooth boundaries by itself, smoothness between adjacent regions can be achieved using third party algorithms. The method discussed in [KT09] is very similar in spirit to our approach, as it is based on the level sets of continuous functions. However, it produces smooth curves that tend to *escape* from their original position (see Figures 1a and 4b from the original paper). Our method produces smooth boundaries that faithfully follow their discrete counterparts, improving geometric fidelity. Panozzo *et al.* [PBDSH13] introduced a method to project B-Spline curves on discrete surfaces using the Phong projection. This tool could in principle be used to define smooth segmentation curves, but the user should define and place the control points that define the smooth curve. Iawonn and colleagues [LGRP14] proposed an iterative Laplacian smoothing algorithm that at each iteration reduces the local boundary curvature. In [LL02] a local parameterization method for defining geometric features in triangle meshes is proposed. The method is based on the concept of *snakes*, pioneered by Kass and colleagues in [KWT88]. All these methods specifically address discrete 2-manifolds embedded in $\mathbb{R}^3$, and do not directly extend to higher dimensions. To the best of our knowledge ours

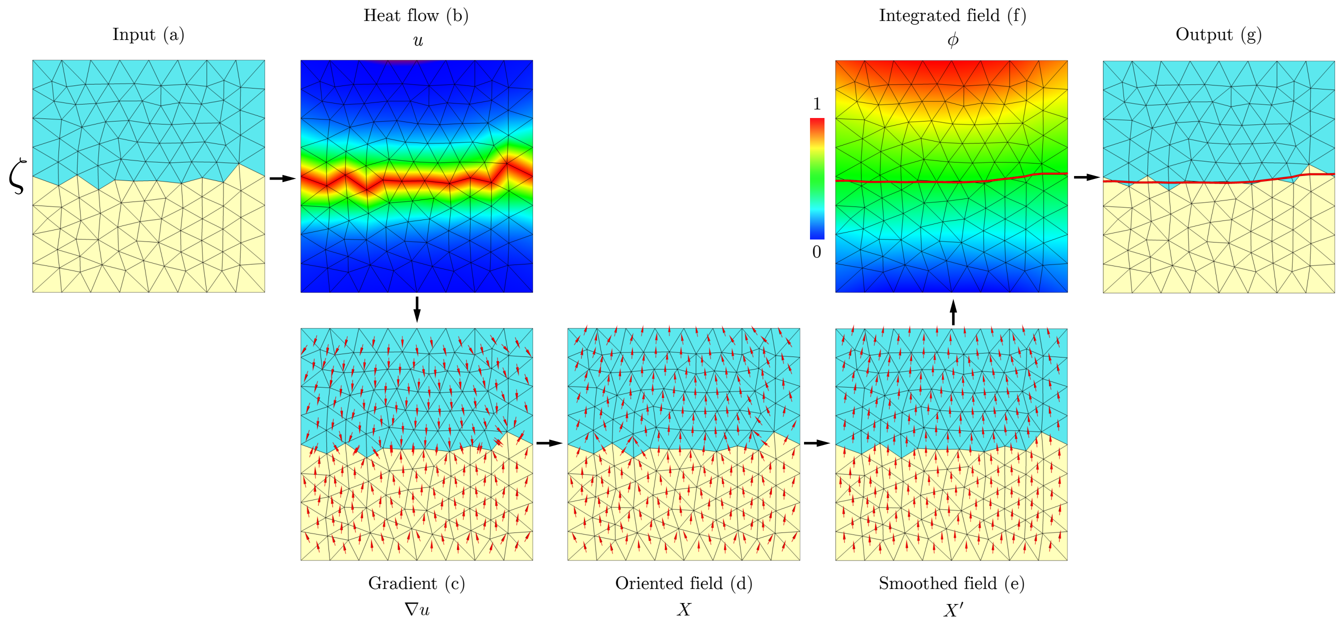

Figure 3: *Overview of our method: (a) we input a discrete segmentation with a jagged boundary ζ; (b) heat diffuses from ζ, producing a function u; (c,d) the vector field X is defined as $\nabla u/\|\nabla u\|$ on the yellow region, and $-\nabla u/\|\nabla u\|$ on the blue region; (e) the field X is smoothed, producing a new vector field X'; (f) the field X' is integrated, producing a function ϕ that evaluates as consistently as possible along ζ; (g) the output smoothed boundary is defined as a level set of ϕ.*

is the first intrinsically dimension-independent method, and can be used to relax discrete curves on 2-manifolds, surfaces in $\mathbb{R}^3$, or even hyper-surfaces in $\mathbb{R}^n$.

Diffusion. Solutions to the heat equation have been extensively used in computer graphics and vision to compute segmentations [BPVR11], geodesic distances [CWW13,BF15], Voronoi diagrams [HHA17] and compare shapes [BBK*10]. A in depth review of the applications and computational methods used to compute diffusion distances is beyond the scope of this paper. We point the reader to [Pat16] for a recent survey on this topic.

3. Method

For the sake of clarity and ease of illustration we introduce our system considering as input a triangle mesh M, representing our discrete domain, and a chain of edges representing a discrete curve ζ defined on it (Figure 3). We also assume that ζ is either a closed curve, or its endpoints are both at the boundary of M, if any. We remind the reader that, as long as the laplacian, gradient and divergence operators are available, the same algorithmic steps apply to any other dimension or discretization choice. The hyper-surface smoothing method can be decomposed in the following four distinct algorithmic steps:

1. Diffuse the heat flow $\dot{u} = \Delta u$ starting from ζ for some time t (Figure 3b);
2. Initialize the vector field X as $\nabla u/\|\nabla u\|$ in one region, and $-\nabla u/\|\nabla u\|$ in the other region (Figure 3d);
3. Smooth X, producing a new vector field X' (Figure 3e);
4. Integrate X' in the least squares sense, producing a function ϕ that aligns to X' and at the same time evaluates consistently along ζ (Figure 3f).

The smoother contour will then be a level set of ϕ (Figure 3g).

In the remainder of the section we provide more information on each step, also discussing implementation details on the discretization of differential operators for triangle meshes, and proposing a convenient unified representation for the gradient and divergence operators, which also applies to general polygonal and polyhedral meshes.

Heat Flow. The first step of the algorithm consists in placing heat charges along the boundary ζ and let them diffuse over time, according to the flow

$$\frac{\partial u}{\partial t} = \Delta u. \tag{1}$$

We perform implicit time integration using the backward Euler method, as described in [DMSB99]. This amounts to solving the following equation

$$(I - t\Delta)u = 0_{|\zeta=1}\,. \tag{2}$$

In our discrete setting we substitute the identity (I) with a diagonal mass matrix that associates to each vertex in the mesh one third of the sum of the areas of its incident triangles, and the Laplacian

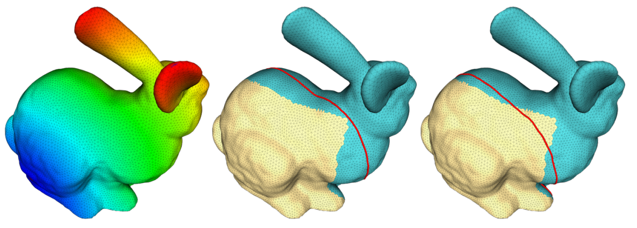

Figure 4: *The function ϕ computed solving Equation 5 (left) and two level sets of it (middle, right). As can be noticed the function does not align with the segmentation boundary. This depends from the fact that smoothing the field X' does not give any guarantee that ϕ will evaluate consistently throughout the whole boundary. In general, it does not.*

operator (Δ) with the cotangent operator, defined on each vertex v_i as

$$\Delta(v_i) = \sum_{j \in N(i)} (\cot\alpha_{ij} + \cot\beta_{ij})(v_j - v_i) \tag{3}$$

Here $N(i)$ represents the vertices v_j adjacent to vertex v_i, and α_{ij}, β_{ij} the angles opposite to edge (v_i, v_j) [MDSB03]. For the time step t, we used the squared average edge length of M, as suggested in [CWW13].

Discrete Gradient and Divergence. Differently from [CWW13], which uses different discretizations for the gradient and divergence operators, we use a convenient unified representation. Let $\widehat{ijk}$ be the triangle connecting the vertices v_i, v_j, v_k. We define the piece-wise constant gradient ∇u as follows

$$\nabla\widehat{ijk} = \frac{1}{2A}\left[(u_j - u_i)\cdot(v_i - v_k)^{\perp} + (u_k - u_i)\cdot(v_j - v_i)^{\perp}\right], \tag{4}$$

where u_i is the value of u evaluated at vertex i, $(v_i - v_k)^{\perp}$ is the oriented edge (v_i, v_k) rotated by 90° counter clockwise (i.e. using the triangle normal as rotation axis), and A is the triangle area.

Assuming M contains $\#F$ triangles and $\#V$ vertices, the gradient operator can be efficiently packed into a $3\#F \times \#V$ matrix G. This representation has a twofold advantage: the first is that by multiplying G for a column vector containing the function values at each vertex in the mesh, a $3\#F$ long column vector containing the serialized gradient can be efficiently computed by matrix vector multiplication; the second is that the transposed matrix $G^{\top}$ implements the divergence operator, meaning that multiplying $G^{\top}$ with a vector containing a serialized vector field, gives the divergence of the field. This translates to an extremely compact implementation, easier to code and debug. Notice that by substituting the gradient computation in Equation 4 with a more general strategy (e.g. the Green-Gauss gradient method [SBK14]) this twofold use of the matrix G extends also to general polygonal and polyhedral meshes.

Field Processing. The gradient of the heat flow ∇u is a vector field that points towards the boundary ζ from any point in the domain (Figure 3c). Our goal is to generate a field that *traverses* ζ in a smooth way. To do so, we first initialize a vector field X, considering $\nabla u/\|\nabla u\|$ on one region and $-\nabla u/\|\nabla u\|$ on the other region (Figure 3d); then, we smooth the field, in order to alleviate the sharp turns induced by the discrete jagged boundary. We simply perform a few iterations of laplacian smoothing on the dual mesh, meaning that each triangle takes as vector the average between itself and the vectors associated to its edge-adjacent triangles. Typically a few iterations (three in all our examples) are enough to produce a sufficiently smooth field X' that crosses the boundary without having sharp turns (Figure 3e). More aggressive smoothing strategies (e.g., more iterations) can be performed in order to further relax the boundary and allow it to deviate from ζ.

Field Integration. The last step consists in generating the function ϕ that aligns to the smoothed vector field X'. The output segmentation boundary will then be a level set of ϕ (Figure 3g). At this point one may think that, similarly to [CWW13], ϕ corresponds to the function that has X' as gradient, which can be computed by solving the Poisson problem

$$\Delta\phi = \nabla X'. \tag{5}$$

However, even though the function traverses ζ in a smooth way, there can be no level set which approximates the discrete boundary well. This is because smoothing the gradient of the function does not give any guarantee on the fact that the function will evaluate consistently throughout the whole boundary ζ. In the general case, it does not; especially if ζ is non smooth and low curvature everywhere, but rather has some curvature peak and is pretty flat elsewhere (Figure 4).

We instead solve for a function ϕ that aligns to X' only in the least squares sense, while at the same time tries to evaluate as consistently as possible along ζ. Specifically, we look for the function ϕ that minimizes

$$\arg\min_{\phi} \left\|\Delta\phi - \nabla X'\right\|^2 + \lambda\left\|\phi(\zeta) - \frac{1}{2}\right\|^2, \tag{6}$$

where $\frac{1}{2}$ is the function value that we want to replicate along ζ. Notice that $\frac{1}{2}$ provides a reference to the *best fitting* level set, but it could potentially be substituted with any other value. Since we are dealing with the differential properties of the field, the scalar function will shift accordingly. Minimizing 6 corresponds to solving a linear system $A\phi = b$, with

$$A = \begin{pmatrix} \Delta \\ M_{\zeta} \end{pmatrix} \quad b = \begin{pmatrix} \nabla X' \\ \frac{1}{2}^{T} \end{pmatrix}. \tag{7}$$

Here M_{ζ} is a sub-matrix having as many lines as the number of vertices in ζ. Each line is null everywhere, and has a single 1 entry corresponding to a vertex in ζ. On the right hand side $1/2^T$ is a column vector containing as many $\frac{1}{2}$ as the number of vertices in ζ. We solve the system using weighted least squares, according to the normal equations $(A^T W A)\phi = (A^T W)b$. The matrix $W = (1\ \lambda)^T$ is diagonal and associates weight 1 for each row corresponding to $\|\Delta\phi - \nabla X'\|^2$ and weight λ for each row corresponding to $\|\phi(\zeta) - 1/2\|^2$. Figure 5 shows how the value of λ influences the level sets of ϕ. As can be noticed, for $\lambda = 0$ there is still some rough approximation error, whereas for higher values the function

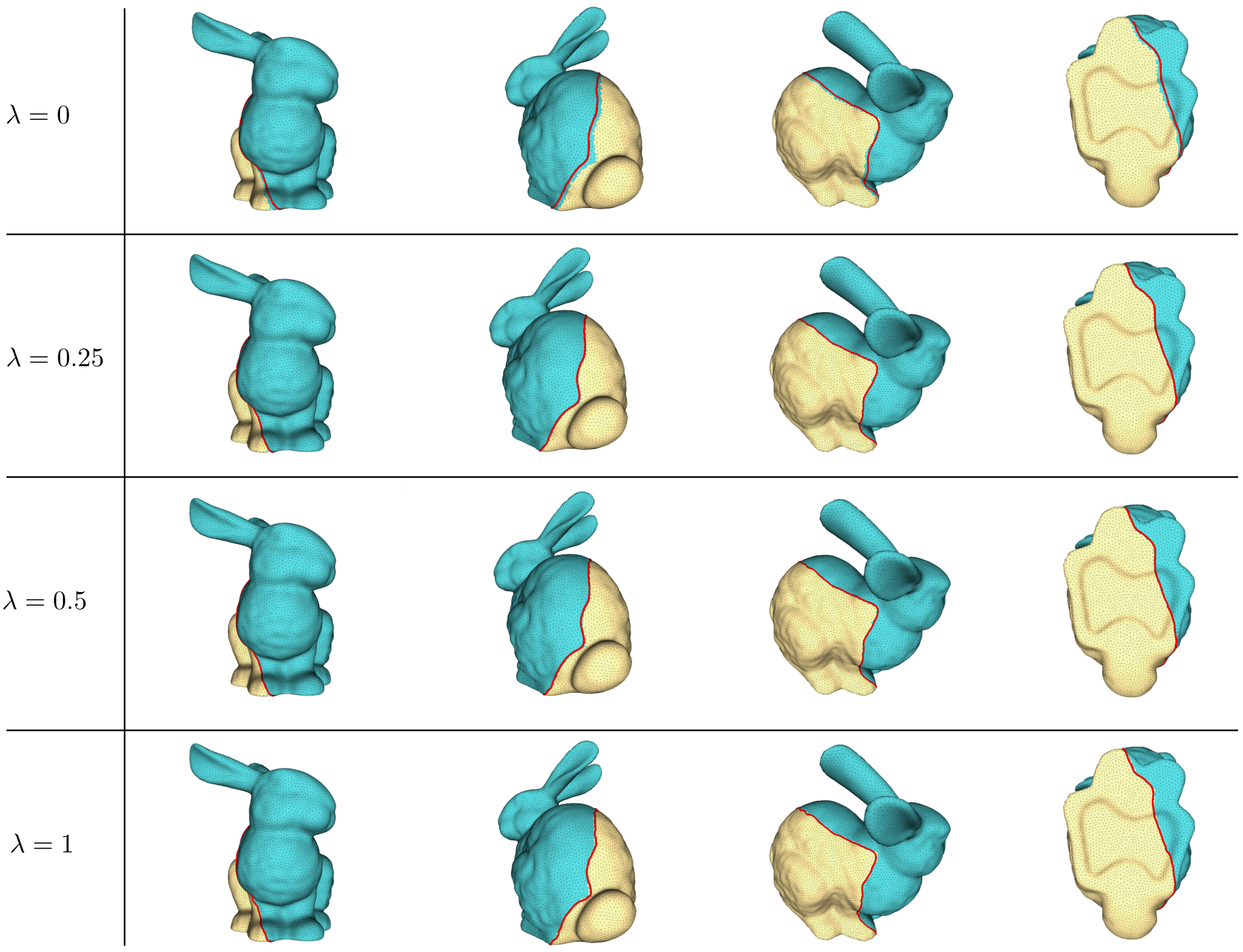

Figure 5: *Computing a function ϕ starting from a smoothed vector field X' (Equation 5) does not give any guarantee that it will evaluate consistently throughout the whole segmentation boundary. In Equation 6 we weigh the alignment with the guiding field X' with the requirement of evaluating as consistently as possible along the boundary. The parameter λ weights the importance of this latter term in the energy. Here we show the level sets of different functions produced considering growing values of λ. As can be noticed, for $\lambda \geq 0.25$ a level-set capable of interpolating the discrete boundary well always exists.*

is well aligned with with the segmentation boundary while still being smooth. In a sense, this is the only parameter exposed to the user, who can decide to relax the boundary as preferred by reducing the value of λ.

4. Results

We implemented our hyper surface smoothing algorithm as a single threaded C++ application on a MacBook Pro equipped with a 2,9 GHz Intel Core i5 and 16GB of RAM, using CinoLib [Liv17] for geometry processing and Eigen [GJ*10] as linear solver.

From a computational point of view the algorithm amounts to solving two linear systems, one for the heat flow, and the other for the field integration. All the other steps (gradient computation and smoothing) introduce negligible delays. Timings are satisfactory: for medium sized meshes (less than 50K faces) the whole computation happens in a fraction of a second, and is therefore compatible with interactive use.

In Figures 1, 2, 3 and 5 we show several results obtained with our implementation. At the moment we implemented only a version for triangle meshes. We plan to improve our prototype in order to be able to work with other discrete entities.

Being based on widespread discrete differential operators included in many freely available geometry processing libraries, the algorithm is quite easy to implement. From a usability point of view the method is pretty intuitive, meaning that everything works as expected without requiring complex parameter tuning. Nevertheless, advanced users can fine tune the results by acting on two parameters: the scalar λ that balances between field alignment and consistent function evaluation along the boundary (Figure 5), and the amount of smoothing performed on the gradient space, which may produce functions that tend to escape from the original boundary, if needed.

5. Conclusions and Future Works

We introduced a novel smoothing operator to relax the boundaries of discrete segmentations. The operator is inspired by the heat flow geodesic algorithm described in [CWW13], from which it inherits a number of remarkably good properties, such as the ability to scale on different discrete domain representations, and to generalize to n dimensional spaces.

The prototype we implemented to generate the results shown throughout the paper can process only triangle meshes. We are currently working to extend the code base to be able to process also pixel images, as well as general polygonal and polyhedral meshes.

For future works, at the moment this method is capable of smoothing discrete boundaries shared between *pairs* of regions. Indeed, one interesting question that we plan to investigate further is how to extend it to more complex scenarios, where more than two regions are involved at the same time.

Acknowledgements

This work is supported by the EU ERC Advanced Grant CHANGE, grant agreement No 694515. Thanks are due to all the members of the Shapes and Semantics Modeling Group at IMATI-CNR.

References

[AMSF08] Attene M., Mortara M., Spagnuolo M., Falcidieno B.: Hierarchical convex approximation of 3d shapes for fast region selection. In *Computer graphics forum* (2008), vol. 27, Wiley Online Library, pp. 1323–1332. 1

[AZC*12] Au O. K.-C., Zheng Y., Chen M., Xu P., Tai C.-L.: Mesh segmentation with concavity-aware fields. *IEEE Transactions on Visualization and Computer Graphics 18*, 7 (2012), 1125–1134. 2

[BBK*10] Bronstein A. M., Bronstein M. M., Kimmel R., Mahmoudi M., Sapiro G.: A gromov-hausdorff framework with diffusion geometry for topologically-robust non-rigid shape matching. *International Journal of Computer Vision 89*, 2 (2010), 266–286. 3

[BF15] Belyaev A. G., Fayolle P.-A.: On variational and pde-based distance function approximations. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 104–118. 3

[BPVR11] Benjamin W., Polk A. W., Vishwanathan S., Ramani K.: Heat walk: Robust salient segmentation of non-rigid shapes. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 2097–2106. 3

[BVZ01] Boykov Y., Veksler O., Zabih R.: Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence 23*, 11 (2001), 1222–1239. 1

[CWW13] Crane K., Weischedel C., Wardetzky M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG) 32*, 5 (2013), 152. 2, 3, 4, 6

[DMSB99] Desbrun M., Meyer M., Schröder P., Barr A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 317–324. 3

[GJ*10] Guennebaud G., Jacob B., et al.: Eigen v3. http://eigen.tuxfamily.org, 2010. 5

[HHA17] Herholz P., Haase F., Alexa M.: Diffusion diagrams: Voronoi cells and centroids from diffusion. In *Computer Graphics Forum* (2017), vol. 36, Wiley Online Library, pp. 163–175. 3

[JLCW06] Ji Z., Liu L., Chen Z., Wang G.: Easy mesh cutting. In *Computer Graphics Forum* (2006), vol. 25, Wiley Online Library, pp. 283–291. 2

[KT09] Kaplansky L., Tal A.: Mesh segmentation refinement. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 1995–2003. 2

[KWT88] Kass M., Witkin A., Terzopoulos D.: Snakes: Active contour models. *International journal of computer vision 1*, 4 (1988), 321–331. 2

[LAPS17] Livesu M., Attene M., Patané G., Spagnuolo M.: Explicit cylindrical maps for general tubular shapes. *Computer-Aided Design* (2017). doi:10.1016/j.cad.2017.05.002. 1

[LGRP14] Lawonn K., Gasteiger R., Rössl C., Preim B.: Adaptive and robust curve smoothing on surface meshes. *Computers & Graphics 40* (2014), 22–35. 2

[Liv17] Livesu M.: cinolib: A generic programming C++ library for processing polygonal and polyhedral meshes, 2017. https://github.com/maxicino/cinolib/. 5

[LL02] Lee Y., Lee S.: Geometric snakes for triangular meshes. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 229–238. 2

[LLS*04] Lee Y., Lee S., Shamir A., Cohen-Or D., Seidel H.-P.: Intelligent mesh scissoring using 3d snakes. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on* (2004), IEEE, pp. 279–287. 2

[LVS*13] Livesu M., Vining N., Sheffer A., Gregson J., Scateni R.: Polycut: Monotone graph-cuts for polycube base-complex construction. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA 2013) 32*, 6 (2013). doi:10.1145/2508363.2508388. 1

[MDSB03] Meyer M., Desbrun M., Schröder P., Barr A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*. Springer, 2003, pp. 35–57. 4

[MPS*04] Mortara M., Patanè G., Spagnuolo M., Falcidieno B., Rossignac J.: Plumber: a method for a multi-scale decomposition of 3d shapes into tubular primitives and bodies. In *Proceedings of the ninth ACM symposium on Solid modeling and applications* (2004), Eurographics Association, pp. 339–344. 1

[Pat16] Patané G.: Star-laplacian spectral kernels and distances for geometry processing and shape analysis. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 599–624. 3

[PBDSH13] Panozzo D., Baran I., Diamanti O., Sorkine-Hornung O.: Weighted averages on surfaces. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 60. 2

[RBG*09] Reuter M., Biasotti S., Giorgi D., Patanè G., Spagnuolo M.: Discrete laplace–beltrami operators for shape analysis and segmentation. *Computers & Graphics 33*, 3 (2009), 381–390. 1

[SBK14] Sozer E., Brehm C., Kiris C. C.: Gradient calculation methods on arbitrary polyhedral unstructured meshes for cell-centered cfd solvers. In *Proceedings of the 52nd Aerospace Sciences Meeting, National Harbor, MD, USA* (2014), vol. 1317. 4

[Sha08] Shamir A.: A survey on mesh segmentation techniques. In *Computer graphics forum* (2008), vol. 27, Wiley Online Library, pp. 1539–1556. 2