SeamCut: Interactive Mesh Segmentation for Parameterization

Victor Lucquin
LTCI, Telecom Paristech, Paris Saclay University
Allegorithmic

Sébastien Deguy
Allegorithmic

Tamy Boubekeur
LTCI, Telecom Paristech, Paris Saclay University
Allegorithmic

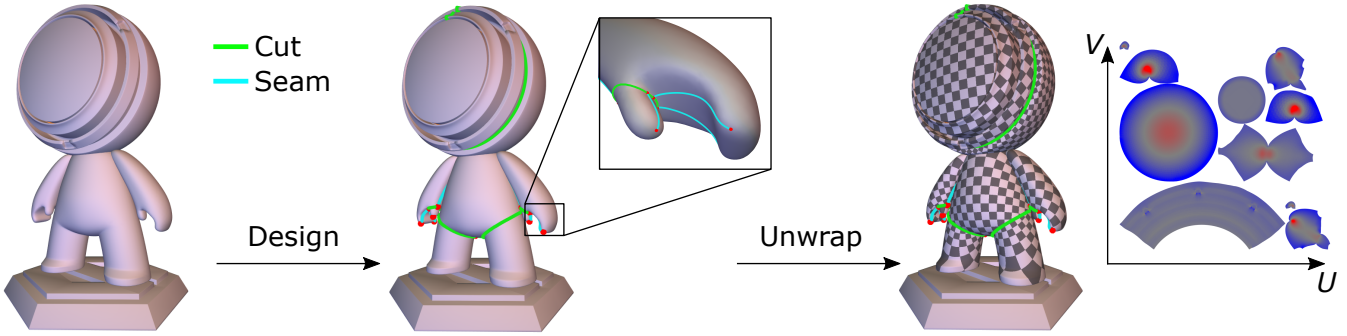

Figure 1: The SeamCut system allows designing a structured set of smooth curves on a surface (left), ultimately used to govern the segmentation used for computing a parameterization.

ABSTRACT

Mesh parameterization consists in unwrapping mesh regions having the topology of a disk onto the 2D plane. This geometry process is fundamental for 2D texture mapping and instrumental for a number of surface analysis primitives. Typically, users execute automatic unwrapping algorithms on handmade disk-like patches, whose design, often called "seaming" induces a massive amount of tedious manual actions to select the edges of the mesh that eventually form the regions boundaries i.e., the seams. We propose SeamCut, an analytic and interactive segmentation framework to build an organized set of curves, *cuts* and *seams*, prior to surface parameterization. While the *cuts* are in charge of dividing the mesh in semantic parts, the *seams* aim at minimizing parameterization distortion. To tailor them, our method analyzes the surface geometry using only sparse high level interactions on the surface, where we adopt a field-based approach to generate the curves independently of the actual connectivity of the mesh. Once stable, the curve set may be used to remesh the input or snapped to the mesh edges, giving rise to a consistent mesh segmentation ready for automatic parameterization. We evaluate our live surface analysis system on a variety of models and report interactive performances.

CCS CONCEPTS

•**Computing methodologies** →**Mesh models;** *Shape analysis;*
•**Human-centered computing** →*Interactive systems and tools;*

KEYWORDS

interactive mesh segmentation, shape parameterization

ACM Reference format:
Victor Lucquin, Sébastien Deguy, and Tamy Boubekeur. 2017. SeamCut: Interactive Mesh Segmentation for Parameterization. In *Proceedings of SIGGRAPH Asia 2017 Technical Briefs, Bangkok, Thailand, November 27–30, 2017 (SA '17 Technical Briefs),* 4 pages.
DOI: 10.1145/3145749.3149435

SA '17 Technical Briefs, Bangkok, Thailand

978-1-4503-5406-6/17/11…$15.00
DOI: 10.1145/3145749.3149435

1 INTRODUCTION

1.1 Motivation

Mesh parameterization consists in unfolding a 2-manifold embedded in the 3-dimensional space onto a 2D unit-square (known as UV space), offering a convenient layout to locate 2D maps (e.g., color texture) on the surface. The dominant workflow used in the game, VFX and feature animation industries [Hormann et al. 2007] consists in three steps: (i) segmenting the shape into disk-like charts, (ii) unwrapping these charts from 3D to the UV space and (iii) packing them by scaling and applying rigid transformations to the resulting 2D islands. In this workflow, users typically spend a great amount of effort on the first step, manually designing the mesh segmentation based on numerous constraints, such as semantics, planned signal in the map (texture), boundaries visibility or symmetry, for instance. Consequently, mesh segmentation for parameterization has to achieve two main objectives: (i) decompose the shape into disk-like components; (ii) minimize the related parameterization distortion. Since artists often use the UV space actively in the authoring applications and since regions boundaries create parameterization discontinuities – hence artifacts – they usually intend to: (i) hide and/or minimize region boundaries; (ii) keep the 2D layout visually understandable. These two sets of goals and constraints are contradictory, making mesh segmentation for parameterization an ill-posed problem, where one has to find a good trade-off between both. In particular, the former favors patch-like segmentation with a relatively high number of charts, while the latter motivates a minimal segmentation and/or a part-based approach.

Out of experience, we observe that artists tend to segment the mesh into meaningful semantic parts, where discontinuities are intrinsically less visible thanks to the minima rule [Hoffman and Singh 1997] or less important because of strong discontinuities of the mapped signal (e.g. color, material, etc). Therefore, they typically start by locating the boundaries of the charts at first, often referred as *cuts*, before adding additional *seams* that grow from inter-patch cuts in order to "relieve tension" in the unwrapping process, hence minimizing distortion. In practice, users do so by tediously marking edges, which leads to a segmentation that both lacks explicit organization and highly depends on the connectivity of the mesh, requiring going back and forth with remeshing iterations.

SeamCut aims at supporting this workflow while giving users high-level interactions instead of per-edge selection. More precisely, we propose a framework to design two types of on-surface curves, *cuts* and *seams*, that are then used to segment the surface and parameterize it. In order to easily place and edit the former, we define *cuts* as isolines of harmonic fields defined from a few sketched constraints only and inspired from state-of-the-art interactive part-based mesh segmentation [Guy et al. 2014; Zheng and Tai 2010]. Based on the current set of *cuts*, the user can then grow *seams* along anisotropic geodesic paths from a source point within a chart to an existing line, accounting for surface features. As a result, our system enables to segment and parameterize complex shapes with a very sparse input from the user, drastically reducing the amount of manual work typically required at this stage while maintaining interactive control over the dynamic UV layout.

In a nutshell, our main contributions are:

- a model of structured on-surface curves which captures explicit relationships between *seams* and *cuts* and exploits harmonic surface analysis and geodesic computations to generate and modify the curves interactively,
- a set of complementary control primitives to interactively edit these lines in a non-destructive manner, decorrelating them from the particular connectivity of the mesh,
- a set of export operators to interpret these lines into subsets of existing or new mesh edges, ready for parameterization.

1.2 Related work

1.2.1 Mesh segmentation. Mesh segmentation is a broad research domain in geometric modeling and computer graphics, for which several surveys are available [Shamir 2008; Theologou et al. 2015], with numerous applications such as shape parameterization, shape matching, multi-resolution modeling, mesh editing, animation and more. But mesh segmentation is also an intrinsically ill-defined problem, as many different and sometimes concurrent goals are sought. To that respect, interactive methods [Meng et al. 2011a] enable the user to guide the segmentation process with high-level input, while being able to modify and edit the current segmentation to suit the target application.

1.2.2 Segmentation for parameterization. The workflow of segmenting the surface into nearly-developable disk-like components motivates custom segmentation techniques for UV unwrapping. Some methods grow several patches, leading to low distortion but many visible discontinuities in the parameterization. To cope with this problem, Levy et al. [2002] grow charts so that they meet at feature lines, Zhou et al. [2004] use spectral analysis for clustering while Julius et al. [2005] use an alternative Lloyd clustering scheme guided by a metric favoring conic, hence developable charts.

Other methods transform a genus-0 closed surface into a disk by performing a single cut trying to minimize the induced distortion. Sheffer et al. [2002] join vertices with high gaussian curvature using an approximate minimal Steiner tree weighted by visibility. Gu et al. [2002] propose an iterative solution by joining the current distortion maximum to the boundary with a shortest path. Alternatively, Vallet and Levy [2009] propose a spectral over-segmentation that contains symmetry axis and where the user is only asked to remove excessive cuts until obtaining a satisfactory result, along with other editing tools.

1.2.3 Interactive harmonic field mesh segmentation. Mesh segmentation aiming at a semantic decomposition of shapes tends to use volumetric and/or global methods. In particular, several interactive methods use segmentation fields. All these approaches transform the sparse input of the user into constraints, then solve for the "most harmonic" field verifying them and provide segmentation candidates as isolines of this field. Methods differ in the way the user input is translated into field constraints and in the underlying Laplacian weighting scheme, using either cotangent weight [Guy et al. 2014; Zheng and Tai 2010] or custom weighted scheme based on normal variations [Meng et al. 2011b], curvature [Meng et al. 2008] or concavity [Zheng et al. 2012].

1.2.4 Anisotropic geodesic paths. The problem of finding the shortest path between two points on a polyhedron, as well as fast approximations, has motivated a number of methods. Most of them use a variation of Dijkstra's algorithm with window propagation [Surazhsky et al. 2005] or, more recently, approximate geodesic distance with a heat method [Crane et al. 2013]. In particular, finding geodesic paths according to a non-euclidian metric on the surface has recently been addressed. One such approach consists in embedding the polyhedron in a space such as its edges' euclidian lengths correspond to their length according to the given metric before searching for geodesics, but this leads to faces violating the triangle inequality. This can be fixed with quadratic programming [Campen et al. 2013] to find the least-square nearest viable set of edge lengths, or with a custom subdivision scheme [Zhuang et al. 2014] that subdivides invalid triangles until they are small enough and then treats them as holes. Note that the method by Campen et al. [2013] does not require the triangle inequality to hold and uses only a Short-Term Vector Dijkstra upgrade of the classic Dijkstra's algorithm to approximate anisotropic geodesics efficiently.

2 GENERATION

2.1 Cuts as harmonic field isolines

When starting the segmentation process, the user first creates *cuts* by sketching a few constraints on the surface. In our workflow, the user draws a possibly small and incomplete stroke along the desired *cut* which is then translated into constraints on the harmonic field. This harmonic field is computed on-the-fly, optimizing for the constraints in the least-square sense while accounting for the surface geometry. Finally, the isoline of the field is retrieved.

2.1.1 From strokes to constraints. This input handling is similar to the approach of Guy et al. [2014] and Meng et al. [2011b], with the noticeable difference that constraints are free to exist in mesh faces rather than being only at vertices, which frees the curves from the mesh resolution. First, the user draws a stroke S, with N_S points. For each point $\mathbf{S_i}$, we create two points $\mathbf{p_i} = \mathbf{S_i} + \epsilon_C \mathbf{n_i}$ and $\mathbf{p'_i} = \mathbf{S_i} - \epsilon_C \mathbf{n_i}$, where $\mathbf{n_i}$ is the local normal to the stroke projected on the surface. Finally $\mathbf{p_i}$ (resp. $\mathbf{p'_i}$) is projected again on

the mesh and given field value +1 (resp. −1). After this process, we have $N_C = 2 \times N_S$ constraints $C_i = (t_i, \alpha_i, \beta_i, \gamma_i, f_i)$ where t_i is the triangle, $(\alpha_i, \beta_i, \gamma_i)$ the barycentric coordinates of the point and f_i the constrained field value at this point. In our experiments, we typically set N_S to 4 and ϵ_i to 1 % of the diagonal of the mesh bounding box, but spatially varying values depending on the point of view or on the length of the stroke can be used as well.

2.1.2 Field solving. We then solve the Poisson equation $\Delta f = 0$, similarly to Guy et al. [2014] using a discretized Laplacian in matrix form L and solving the least-square equation of unknown u:

$$A^T A u = A^T b \tag{1}$$

where $A = (L|C)^T$ is a $(n + N_C) \times n$ matrix, and $b = (0, \dots, 0, wf_1, \dots, wf_{N_C})^T \in \mathbb{R}^{n+N_C}$. L is the Laplacian matrix of the mesh and C is such that if the vertices of the triangle of the i-th constraint t_i are a, b, and c, $C_{ia} = w\alpha_i$, $C_{ib} = w\beta_i$, $C_{ic} = w\gamma_i$ and $C_{ij} = 0$ for other j.

2.1.3 Contouring. Once the field value is computed for each vertex on the mesh, we extract the isoline $f = 0$ with linear interpolation in each triangle, using triangles similarly to cubes in the marching cube algorithm. In case of multiple disconnected curves, we keep the nearest to the projection of the user's stroke. Contrary to previous methods, we do not use a voting scheme for the isoline field value because we let the user tailor it interactively (see Sec. 3).

2.2 Seams as anisotropic geodesic paths

To create a *seam*, the user is simply asked to indicate a source curve and a target surface point, letting the system generate the *seam* as a geodesic path in a potentially anisotropic Riemannian metric expressed on the surface. This metric is used to guide *seams* along surface features which is a desirable behavior for segmentation, both for keeping the semantic meaning of lines and making them less visible. We opt for the metric proposed by Zhuang et al. [2014], since it is aligned with the mesh principal curvature directions and stable at umbilical points (locally planar regions). However, we adopt a different normalization strategy for the curvature difference s_x with $s_x = \frac{|k_1|-|k_2|}{|k_1|+|k_2|}$ for a dimension-less expression.

When a *seam* is created, we first propagate a distance field from the current source line – either an existing *cut* or *seam* – on the whole connected surface component it belongs to; then we estimate the gradient $\mathbf{v}$ of this distance field before computing the geodesic line γ^* by integrating:

$$\frac{d\gamma^*}{ds} = -D^{-1}\mathbf{v} \tag{2}$$

where D is the anisotropy tensor of our metric. Hence, *seams* are streamlines of the modified vector field $\mathbf{v}^* = -D^{-1}\mathbf{v}$. We integrate it with an Euler scheme in each triangle of the mesh, and by linearly interpolating the vector field in each face. We compute the distance field through a Short Term Vector Dijkstra [Campen et al. 2013] (or STVD), avoiding keeping the triangle inequality valid in each triangle of the mesh. This choice is motivated by the observation that subdivision schemes [Zhuang et al. 2014] lead to areas with a critical density of invalid triangles, even after subdividing up to very small perimeters. Note that the least square search for alternative edge lengths preserving triangle inequality proposed by Campen et al. [Campen et al. 2013] is too slow for interactive use. We typically use a depth value $k = 3$ in STVD. As a result, *seams* end up making orthogonal connections to their source lines, which is a desirable property for parameterization (e.g., symmetry, chart packing).

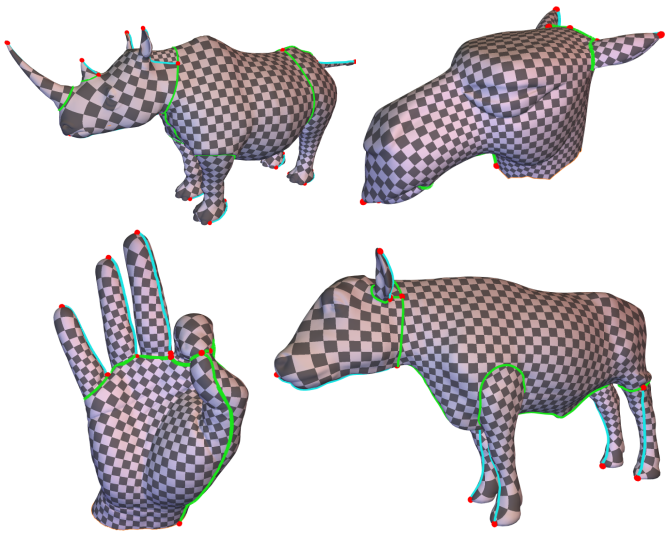

Figure 2: Examples of models segmented with our system. From top to bottom, left to right : *rhino, camel, hand, bull.*

3 EDITING

Once *cuts* and *seams* are generated, the user can edit them, using a small set of high-level control primitives. These interactions can alter lines on which depend other lines in a non-destructive way, as we maintain a dependency graph in memory, where a line update immediately implies the recomputation of the related ones.

Cut sliding allows to *slide* a *cut* by modifying its isovalue to the field value of the surface point currently selected. This translates into a parallel line motion which has a very low computational cost, enabling dynamic updates of *cuts* and *seams* in real time.

Cut refinement lets the user sketch additional strokes to edit a *cut*, which are then translated into a new set of constraints that are injected into the system, with the segmentation field being recomputed on-the-fly and a new *cut* being contoured. Since the field value of the isoline may change when *sliding*, the constraints are set to $I + 1$ and $I - 1$, where I is the current isoline field value.

Seam regrowth allows changing its target point. To do so, our system generates a new stream line based on the same vector field computed from the generation step, allowing to smoothly relocate the *seam* on the chart.

Seam anchoring allows prescribing the intersection between a given *seam* and its source line, therefore enforcing the intersection point. The *seam* is then automatically computed as a geodesic path from its original emitter to the prescribed point.

Anisotropy control modulates the metric, tailoring to which extent mesh features govern *seams*.

4 EXPORT

To pair line creation with UV unwrapping at interaction time, *cuts* and *seams* must be expressed w.r.t. the mesh connectivity. We propose 3 ways to export them as sets of mesh edges. Our lines are polylines made of segments whose extremities lie on the mesh edges. We refer to these segment extremities as polyline vertices.

First, the *Scissor* operator enables to keep the exact (smooth) geometry of the lines by cutting triangles to their exact location,

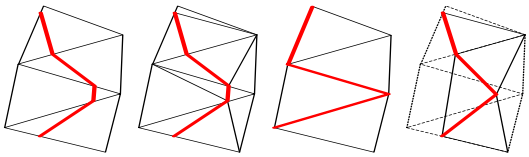

Figure 3: Export operators. Curve in red, original mesh in doted lines and resulting mesh in plain lines. From left to right: original mesh and polyline, scissor operator, snap operator, inverse snap operator.

Table 1: Performance measures: models are taken from Fig. 1 and 2, and from a small user experiment (*Dog* model). Timings are per user interaction. Stretch and shear are respectively $1 + |\sqrt{(\sigma_1^2 + \sigma_2^2)/2} - 1|$ and σ_1/σ_2, where σ_1 (resp. σ_2) is the larger (resp. smaller) singular value of the Jacobian of the mapping.

Model	$\|T\|$	$\|lines\|$	Distortion		Timing (ms)	
			stretch	shear	cut	seam
camel	22k	10	1.11	1.02	353	216
rhino	28k	32	1.93	1.04	285	165
bull	34.5k	18	1.56	1.02	517	367
mat	79k	19	1.41	1.03	1160	372
hand	100k	11	1.17	1.01	2820	1408
dog	11k	14	1.15	1.11	123	98

triangulating resulting faces whenever necessary. This remeshing process typically increases the polygon count but perfectly preserves the designed lines. Second, the *Snap* operator keeps the mesh geometry and topology invariant, and translate the polylines into edge chains made of the closest surface vertex to each polyline vertex. Last, the *Inverse Snap* operator preserves both the lines geometry and mesh connectivity by warping the mesh geometry to the polylines, moving the closest vertex of the mesh to each polyline vertex. Special care is taken when several line points share the same closest vertex (Fig. 3)

Finally, we provide the user with a live previsualization of the unwrapped surface by parameterizing each chart using LinABF [Zayer et al. 2007] and box-packing them in the unit square. Classic unwrapping energies (stretch and shear) as well as a checkerboard texture are rendered onto them to give an interactive feedback on the quality of the current UV unwrapping.

5 RESULTS

We implemented SeamCut in C++, using OpenGL and Qt for its visual component and Eigen for linear algebra. We report single threaded performances on an Intel i7 CPU at 3.7 GHz (Table 1). The execution time is dominated by the Poisson equation solving for *cut* creation, STVD propagation for new *seam* creation, and the two linear systems involved in the live LinABF parameterization. We evaluated SeamCut with a group of users having some experience with CG tools. They achieved segmentations of the simple *Dog* model in a few minutes each, including several iterations between line creation, line edition, and unfolding, and after only a few minutes of training. Our system remains interactive up to 100k polygons, while being limited by the *seam* generation for which convergence success diminishes with mesh resolution. Both *Cuts* and *Seams* use global methods that do not cope well with very large meshes and require manifold input, but this is also the case for state-of-the-art unwrapping algorithms. Finally, although flexible, our control primitives do not cover the whole space of possible segmentations. Ultimately, once the *seam/cut* set is initialized with our system, fine tuning shall be executed using edge selection.

6 CONCLUSION AND FUTURE WORK

We introduced SeamCut, an interactive framework to quickly design mesh segmentations for parameterization. By explicitly classifying segmentation curves into *cuts* or *seams*, our system allows designing UV layouts with a small number of high level interactions. They boil down to a few clicks and strokes that trigger automatic line generation through harmonic optimization and geodesic computations, providing an efficient way to easily design UV layouts, especially for novice users.

ACKNOWLEDGMENTS

This work was partially supported by the French National Research Agengy (ANR) under grant ANR 16-LCV2-0009-01 ALLEGORI and by BPI France, under grant PAPAYA.

REFERENCES

Marcel Campen, Martin Heistermann, and Leif Kobbelt. 2013. Practical Anisotropic Geodesy. *Comput. Graph. Forum* 32, 5 (2013), 63–71.
Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013. Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics* 32, 5 (2013), 152.
Xianfeng Gu, Steven J Gortler, and Hugues Hoppe. 2002. Geometry images. *ACM Transactions on Graphics* 21, 3 (2002), 355–361.
Emilie Guy, Jean Marc Thiery, and Tamy Boubekeur. 2014. SimSelect: Similarity-based selection for 3D surfaces. *Computer Graphics Forum* 33, 2 (2014), 165–173.
Donald D Hoffman and Manish Singh. 1997. Salience of visual parts. *Cognition* 63, 1 (1997), 29–78.
Kai Hormann, Bruno Lévy, and Alla Sheffer. 2007. Mesh Parameterization: Theory and Practice. In *ACM SIGGRAPH Courses*.
Dan Julius, Vladislav Kraevoy, and Alla Sheffer. 2005. D-Charts: Quasi-Developable Mesh Segmentation. In *Computer Graphics Forum*, Vol. 24. 581–590.
Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. In *Acm transactions on graphics*, Vol. 21. ACM, 362–371.
Min Meng, Lubin Fan, and Ligang Liu. 2011a. A comparative evaluation of foreground/background sketch-based mesh segmentation algorithms. *Computers & Graphics* 35, 3 (2011), 650–660.
Min Meng, Lubin Fan, and Ligang Liu. 2011b. iCutter: a direct cut-out tool for 3D shapes. *Computer Animation and Virtual Worlds* 22, 4 (2011), 335–342.
M Meng, Z Ji, and L Liu. 2008. Interactive Mesh Segmentation Based on Feature Preserving Harmonic Field. *Journal of CAD & CG* 20, 9 (2008), 1146–1152.
Ariel Shamir. 2008. A survey on mesh segmentation techniques. *Computer Graphics Forum* 27, 6 (2008), 1539–1556.
Alla Sheffer and John C. Hart. 2002. Seamster: Inconspicuous Low-distortion Texture Seam Layout. In *Proc. Viz*. 291–298.
Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J Gortler, and Hugues Hoppe. 2005. Fast exact and approximate geodesics on meshes. In *ACM transactions on graphics (TOG)*, Vol. 24. 553–560.
Panagiotis Theologou, Ioannis Pratikakis, and Theoharis Theoharis. 2015. *A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation*. Vol. 135. 49–82 pages.
Bruno Vallet and Bruno Lvy. 2009. *What you seam is what you get.* Technical Report. INRIA - ALICE Project Team.
Rhaleb Zayer, Bruno Lévy, and Hans-Peter Seidel. 2007. Linear angle based parameterization. In *Proc. SGP*. 135–141.
Youyi Zheng and Chiew Lan Tai. 2010. Mesh decomposition with cross-boundary brushes. *Computer Graphics Forum* 29, 2 (2010), 527–535.
Youyi Zheng, Chiew Lan Tai, and Oscar Kin Chung Au. 2012. Dot scissor: A single-click interface for mesh segmentation. *IEEE TVCG* 18, 8 (2012), 1304–1312.
Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-charts: stretch-driven mesh parameterization using spectral analysis. In *Proc.SGP*. ACM, 45–54.
Yixin Zhuang, Ming Zou, Nathan Carr, and Tao Ju. 2014. Anisotropic geodesics for live-wire mesh segmentation. *Comput. Graph. Forum* 33, 7 (2014), 111–120.