OptCuts: Joint Optimization of Surface Cuts and Parameterization

MINCHEN LI, University of British Columbia & Adobe Research
DANNY M. KAUFMAN, Adobe Research
VLADIMIR G. KIM, Adobe Research
JUSTIN SOLOMON, Massachusetts Institute of Technology
ALLA SHEFFER, University of British Columbia

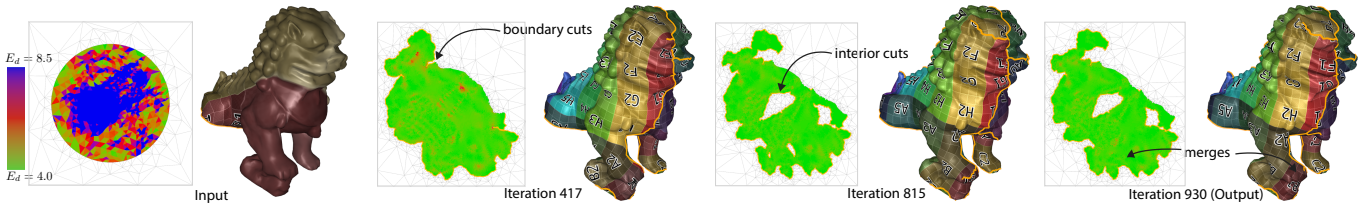


Fig. 1. Starting from an initial embedding (left), we show the iteration process of our OptCuts algorithm jointly optimizing surface cuts and mapping distortion while enforcing global bijectivity. OptCuts iteratively updates continuous changes in the embedding vertices and discrete topological changes in the UV mesh by propagating simple and local topology splitting and merging operations. In this example a distortion bound of 4.1 is enforced, measured by the symmetric Dirichlet distortion energy [Smith and Schaefer 2015]; we report total inner iterations at each rendered frame.

Low-distortion mapping of three-dimensional surfaces to the plane is a critical problem in geometry processing. The intrinsic distortion introduced by these UV mappings is highly dependent on the choice of surface cuts that form seamlines which break mapping continuity. Parameterization applications typically require UV maps with an application-specific upper bound on distortion to avoid mapping artifacts; at the same time they seek to reduce cut lengths to minimize discontinuity artifacts. We propose *OptCuts*, an algorithm that jointly optimizes the parameterization and cutting of a three-dimensional mesh. OptCuts starts from an arbitrary initial embedding and a user-requested distortion bound. It requires no parameter setting and automatically seeks to minimize seam lengths subject to satisfying the distortion bound of the mapping computed using these seams. OptCuts alternates between topology and geometry update steps that consistently decrease distortion and seam length, producing a UV map with compact boundaries that strictly satisfies the distortion bound. OptCuts automatically produces high-quality, globally bijective UV maps without user intervention. While OptCuts can thus be a highly effective tool to create new mappings from scratch, we also show how it can be employed to improve pre-existing embeddings. Additionally, when semantic or other priors on seam placement are desired, OptCuts can be extended to respect these user preferences as constraints during optimization of the parameterization. We demonstrate the scalable performance of OptCuts on a wide range of challenging benchmark parameterization examples, as well as in comparisons with state-of-the-art UV methods and commercial tools.

CCS Concepts: • **Computing methodologies → Mesh geometry models**;

Authors' addresses: Minchen Li, University of British Columbia & Adobe Research; Danny M. Kaufman, Adobe Research; Vladimir G. Kim, Adobe Research; Justin Solomon, Massachusetts Institute of Technology; Alla Sheffer, University of British Columbia.

0730-0301/2018/11-ART247 $15.00
https://doi.org/10.1145/3272127.3275042

Additional Key Words and Phrases: geometry processing, mesh parameterization, seam placement, numerical optimization

ACM Reference Format:
Minchen Li, Danny M. Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. 2018. OptCuts: Joint Optimization of Surface Cuts and Parameterization . *ACM Trans. Graph.* 37, 6, Article 247 (November 2018), 13 pages. https://doi.org/10.1145/3272127.3275042

1 INTRODUCTION

Mapping three-dimensional meshes to the plane is a fundamental task in computer graphics. The two-dimensional mesh embeddings produced by mapping methods are commonly used to store reflectance functions, normals, and displacements for the mesh, providing a domain for painting, synthesizing, and manipulating texture and geometric details. The usability of these embeddings is highly dependent on two interconnected factors: the surface distortion introduced by the mapping, and the quality of the surface cuts forming seams across which the mapping is discontinuous [Hormann et al. 2007]. Both high distortion and longer seams are detrimental to downstream applications. Yet, reducing distortion below a desired bound typically requires introducing longer seams.

Given its broad applicability, parameterization has long been a focus of research in geometry processing. Algorithms in this domain focus on these two key aspects of the problem [Sheffer et al. 2007]. Particularly well-studied are *geometric* techniques that assume a surface has already been cut into disk-topology segments, which then need to be mapped into the plane with minimal distortion while maintaining fixed connectivity; at this point, parameterization becomes a real-valued optimization problem that seeks to minimize changes in mesh angles and areas while maintaining local or global injectivity. Complementing these techniques, *topological* algorithms find reasonable seams, either keeping the surface in one piece or partitioning it into individual segments that can then be parameterized with low distortion.

In contrast, we propose a joint optimization algorithm *OptCuts* that simultaneously optimizes for both seams and the corresponding distortion of the embedding. Our algorithm is based on a minimization model problem that directly and automatically balances between seam length and parametric distortion measures. Manually balancing distortion and seam quality requires a choice of a relative scaling factor between these two objectives. From a practical perspective, it is difficult for users to choose this factor as the two terms measure very different quantities and no such setting can provide a guarantee on the quality of the generated map's distortion. On the other hand, users typically have a clear sense of the amount of distortion they consider acceptable for their application. Motivated by this observation we propose coupled seam and distortion optimization as a *constrained* problem to find charts with locally minimal seam lengths that strictly satisfy a user-set distortion bound. Treating the distortion bound as a hard inequality constraint guarantees a pre-specified level of mapping quality, while enabling us to explore optimal seams that satisfy this bound.

Prior methods coupling distortion reduction and cutting have generally required hand-tuning a number of user-exposed parameters and, as in the recently proposed AutoCuts [Poranne et al. 2017], even advocate manual intervention by interactively adjusting these parameters during the optimization process.

In contrast, OptCuts is a fully automatic optimization method: users provide their desired distortion bound and OptCuts then directly computes a parametrization satisfying this bound while minimizing seam lengths. While artists often also seek seam locations that will help hide cuts and/or preserve symmetries we focus here on seam length as a general-purpose quality measure. Maps provided are always locally injective and, as we will show, can additionally be constrained to be bijective and support additional, user-provided seam placement constraints and biases when desired. As demonstrated by our comparisons in Section 7, when compared to previous methods that do provide an automatic mode [Poranne et al. 2017; Sorkine et al. 2002], OptCuts produces much shorter seams when its bound is set for the same achieved distortion. Likewise, as we show in Section 7.5, e.g. Figure 13, OptCuts can also be used to polish pre-existing UV maps. OptCuts can take an arbitrary UV map as input and improve either seam length while preserving the current distortion bound or even improve upon distortion as well, by setting a lower distortion bound.

To achieve these gains we begin by casting global parameterization as a constrained minimization, formulated with seam length as our objective and a distortion bound as our inequality constraint. We then observe that the *Lagrangian* of this constrained minimization's saddle-point problem directly gives a multi-objective optimization formed by the weighted sum of our seam length measure and map distortion. However, the key observation here is that now there is a natural scaling implied between the two measures that is directly defined by the Lagrange multiplier of the distortion bound. Regularization of iterated updates to this multiplier then allow us to smoothly explore variations of the Lagrangian over the space of seam cuts.

Next, we observe that to solve this saddle-point problem we must optimize over both smooth vertex parameters and discrete changes in topology. Exhaustive search is clearly not an option. Instead, we propose a discrete-continuous optimization method that explores decrease of distortion and seam length over both classical, smooth descent directions and along propagations of topological merging and cutting operations on the UV mesh. When desired, we additionally enforce constraints to achieve globally bijective maps. Finally, we further allow UV artists the option to guide seam placement away from salient regions by enabling painting over the surface. OptCuts then avoids seam placement in these regions in proportion to the intensity of the painting.

Together, these components form the core of our OptCuts algorithm. Over a wide range of examples we show that OptCuts efficiently achieves all attempted distortion bounds while minimizing seam length for both locally injective and bijective mappings. In Section 7 we compare against both state of the art algorithms and industrial UV-parameterization tools and show that for the same achieved distortion bound, we consistently improve seam-length over prior automated methods, while our automated results closely match with the results of hand-tuned methods. We also evaluate OptCuts over a large benchmark of parametrization problems, demonstrating that across mesh scales and problem difficulties OptCuts successfully obtains user-specified distortion bounds while efficiently minimizing seam length.

Contributions. To our knowledge, OptCuts is the first fully automated global parameterization algorithm that obtains bijective maps satisfying prescribed distortion bounds while minimizing seam length. To do so we first propose a new, simple-to-state, constrained seam-length minimization model problem. We then solve our problem by our proposed discrete-continuous algorithm for the saddle-point problem using a combined discrete search over propagated mesh operations and smooth descent over vertex positions. We evaluate OptCuts to show efficient performance and scaling. Across a wide range of automated methods it improves over the state of the art, while automatically obtaining comparable quality results to hand-tuned parameterization methods.

2 RELATED WORK

Surface parameterization is a fundamental geometry processing problem that has been extensively researched [Hormann et al. 2007; Sheffer et al. 2007]. Much of the literature treats surface cutting and distortion minimizing parameterization as two separate, *sequential* tasks; only a handful of methods, discussed below, address these two goals in tandem.

Parameterization with fixed connectivity. A significant body of work takes three-dimensional surfaces with fixed connectivity and disk topology and embeds them in the plane. A primary distinction between these methods is often in the choice of distortion metrics they seek to minimize. While multiple methods focus on minimizing angular distortion [Aigerman and Lipman 2015; Floater 2003; Lévy et al. 2002; Sawhney and Crane 2017; Sheffer et al. 2005], others seek to produce more isometric parameterizations that account for triangle stretch [Claici et al. 2017; Hormann and Greiner 2000; Rabinovich et al. 2017; Sander et al. 2001; Shtengel et al. 2017; Smith and Schaefer 2015].

Many of these methods produce parameterizations that are not necessarily globally bijective. Recent methods obtain global bijectivity by initializing with a bijective map and then explicitly preventing both local and global overlaps during subsequent optimization steps [Jiang et al. 2017; Smith and Schaefer 2015]. OptCuts supports enforcement of global bijectivity by extending the scaffolding method of Jiang et al. [2017] with rapid meshing updates and additional distortion energies distributed on so-called air-meshes in void regions between UV-mesh boundaries. In concert with these scaffolds we optimize both mesh vertex positions and topology to jointly improve mapping distortion and seam quality.

A number of seamless parameterization approaches have also been recently proposed [Kharevych et al. 2006; Myles and Zorin 2013]. While these methods still can generate discontinuities in their embeddings they ensure that parameterization across seams are continuous up to a rigid transformation. They typically place seams by connecting cone singularities, discrete points on the surface where the mapping is discontinuous, to existing boundaries. While this is desirable for applications such as inter-surface mapping [Aigerman et al. 2015] and quad meshing [Ray et al. 2006], benefits are not obvious for storing surface signals, such as texture in atlases. Likewise, discontinuities still lead to artifacts thus spurring recent work [Liu et al. 2017; Ray et al. 2010] focused on manipulating textures to hide sampling artifacts produced by discontinuities. These texture-manipulating methods continue to appreciate shorter seams, further motivating our decision to focus on reducing seam lengths.

Separate Cut Computation. The purely geometric methods above rely on the multitude of existing methods that cut or segment meshes prior to parameterization [Julius et al. 2005; Lévy et al. 2002; Sheffer and Hart 2002; Snyder et al. 2003; Vallet and Lévy 2009]. Since the cutting is done before parameterization, these methods rely on proxy metrics as a predictor of anticipated mapping distortion. Consequently, achieving a desired distortion bound with these tools requires trial and error hand-tuning as users need to provide the right proxy parameter thresholds that will eventually result in the amount of distortion they ultimately wish to achieve. OptCuts combines the two tasks of cutting and parameterizing, enabling users to directly control the resulting tradeoff between mapping distortion and seam length.

Simultaneous Cutting and Parameterization. Motivated by the need for joint reasoning over distortion and seam placement, a few methods directly consider mapping distortion while making cutting choices. Sorkine et al. [2002] parameterize a surface triangle-by-triangle, introducing cuts when distortion exceeds user-prescribed bounds. Due to this locality, this generally introduces longer than necessary output seams to achieve a given bound [Hormann et al. 2007; Poranne et al. 2017]. Starting with an input parameterization Gu et al. [2002] repeatedly introduce cuts connecting the current boundary with distortion maxima in the current parameterization. This process terminates once distortion falls below a given bound. This aggressive approach works well in the presence of a few distortion extrema, but becomes less effective as the distortion becomes more evenly distributed (Figure 12, left). OptCuts performs equally well in both scenarios (Figure 12, right).

Most recently Poranne et al. [2017] proposed AutoCuts—a method that optimizes the weighted sum of a seam-penalty energy and the symmetric-Dirichlet distortion energy [Smith and Schaefer 2015] for parameterization. AutoCuts effectively treats the UV-mesh as a fixed topology triangle soup and uses its seam-penalty energy to pull triangles together. AutoCuts provides two usage settings: the first is interactively driven by direct user guidance throughout the optimization iteration process; and the second is fully automated. It is primarily targeted towards the user-assisted parameterization mode and provides multiple ways for users to interact with the system. OptCuts complements AutoCuts in its focus on efficiently serving settings where users want to obtain parameterizations automatically. In this automatic setting, OptCuts consistently outperforms AutoCuts in terms of distortion to seam-length trade-off as well as in timing and scalability; see Section 7.2 and Table 2.

In both automated and user-guided modes AutoCuts requires users to pre-select a balancing factor between the seam penalty and distortion terms in its multi-objective. Unfortunately, there is no intuitive, nor direct mapping between this balancing factor and the resulting distortion obtained per example. Similarly, although in the limit of stiffness the seam-penalty term would remove all cuts, there is no direct mapping between this penalty term and a meaningful measure of seam length. Consequently, AutoCuts requires trial and error, achieved via user interaction, to achieve the distortion versus seam-length tradeoffs users generally envision. Addressing these needs, OptCuts optimizes directly on the two quantities users typically want to control in parameterization—seam length and mapping distortion. OptCuts allows users to provide a hard bound on distortion and then automatically finds a mapping that satisfies this bound while keeping seam length small. This enables users to more easily communicate their intent and to generate the UV-maps they seek.

3 PROBLEM STATEMENT

Given an input triangle mesh $M = (V, F)$ of a three-dimensional surface with vertices V, and faces F, we seek its UV embedding with connectivity $T^* = (V_{T^*}, F_{T^*})$ and a corresponding two-dimensional embedding of vertex coordinates, $U^* \in \mathbb{R}^{2|V_{T^*}|}$, that locally optimizes the constrained parametrization problem

$$\min_{T,U} E_s(T) \quad s.t. \quad E_d(T,U) \leq b_d \text{ and } (T,U) \in \mathcal{I}. \tag{1}$$

Here V_{T^*} is a superset of V with possibly duplicated vertices, and F_{T^*} is the set of original faces indexed into this new set of vertices. We use $\mathcal{I}$ to define the set of *either* locally injective *or* globally bijective UV maps; in the following we will first initially focus on the locally injective set and then discuss the extension to globally bijective maps (Section 6.4). Energies E_s and E_d respectively measure seam quality (length) and map distortion while b_d is a user-specified *upper bound* on the acceptable distortion of the generated map. This optimization is always feasible as in the limit having all triangles separated would allow for zero distortion.

In general, distortion measures are smooth albeit *nonconvex*, while seam length measures are *nonsmooth* as they increase by discrete amounts when interior mesh edges are cut along or seam edges are merged. Hence, generic optimization techniques cannot be easily

applied to our setting; instead, we need to design a tailored optimization method that addresses both challenges.

3.1 Dual Objective

As a first step toward solving the problem at hand, we construct the Lagrangian for (1)

$$L(T, U, \lambda) = E_s(T) + \lambda(E_d(T, U) - b_d), \tag{2}$$

to form the equivalent saddle-point problem [Bertsekas 2016] defined over primal variables T, U and dual variable λ:

$$\min_{T,U} \max_{\lambda \geq 0} L(T, U, \lambda). \tag{3}$$

Here $\lambda \in \mathbb{R}_+$ is the Lagrange multiplier for our distortion bound. On examination the Lagrangian L can be seen as a multi-objective balancing between distortion and seam quality as dictated by λ. Here, however, λ effectively applies a local scaling between the seam and distortion terms that is implied by the user-specified distortion bound. As we iterate to solve (1), λ will grow when we threaten to violate our distortion bound prioritizing distortion minimization; similarly, λ will decrease toward 0 when our bound is strictly satisfied to prioritize seam quality.

3.2 Embedding Energy

Concretely we formulate our seam-quality energy as the normalized total seam length

$$E_s = \frac{1}{\sqrt{(\sum_{t \in \mathcal{F}} |A_t|)/\pi}} \sum_{i \in \mathcal{S}} |e_i| \tag{4}$$

where $\mathcal{S}$ is the set of all seam edges on the input surface and $|e_i|$ is the length of edge i in the input mesh. We measure distortion over the mapped domain using the symmetric Dirichlet energy [Smith and Schaefer 2015] normalized by surface area [1],

$$E_d = \frac{1}{\sum_{t \in \mathcal{F}} |A_t|} \sum_{t \in \mathcal{F}} |A_t|(\sigma_{t,1}^2 + \sigma_{t,2}^2 + \sigma_{t,1}^{-2} + \sigma_{t,2}^{-2}), \tag{5}$$

where $\mathcal{F}$ is the set of all triangles, $|A_t|$ is the area of triangle t on the input surface, and $\sigma_{t,i}$ is the i-th singular value of the deformation gradient of triangle t.

3.3 Adding Global Bijectivity

Most applications or UV embeddings require global bijectivity. Following the observations of Jiang et al.'s [2017] we realize this additional constraint on our mapping by first triangulating the outer, void regions of each iteration's updated UV map and then augmenting our distortion energy E_d with an additional term, *not included in the distortion bound constraint*, that prevents the added void-space triangles from collapsing during each optimization iteration. For details see Section 6.4.

[1]For simplicity we focus on symmetric Dirichlet here; alternate distortion energies follow similarly.

4 OPTIMIZATION FRAMEWORK FOR OPTCUTS

We solve our constrained optimization problem (1) by recasting it as a the saddle-point problem in (3). We start from an initial, valid UV map (T^0, U^0) and set our dual variable as $\lambda^0 = 0$. OptCuts then iteratively alternates between primal solves designed to improve geometry, (T, U), and dual solves updating our multiplier, λ, encoding the new balancing term between distortion and seam quality for the next primal solve we take. See Algorithm 1.

4.1 Primal Update

Our kth primal update is a joint discrete-continuous search over variations in geometry to minimize the weighted sum of seam length and distortion energies in our Lagrangian (2). We hold the current iterate's dual variable, λ^k, fixed and initialize with the last iterate's geometry (T^{k-1}, U^{k-1}).

We initially experimented with solving our primal problem by approximating topology change with non-smooth energies on duplicated vertices of the input mesh, in a manner similar to AutoCuts [Poranne et al. 2017]. Unfortunately, this quickly led to problems with strongly ill-conditioned real-valued optimization and prevented scaling to larger meshes.

To counter these challenges we directly optimize in alternating inner steps searching over changes in topology T and vertex positions U. These inner iterations loop until we reach a stationary point, giving us iterate k's newly updated geometry (T^k, U^k); see Algorithm 1.

Each vertex step of the primal update performs a single iteration of Newton-type, smooth descent with line-search towards minimizing our distortion energies over vertices U while holding topology, T, fixed. As our distortion energies, including symmetric Dirichlet, are generally nonconvex we employ the projected-Newton [Teran et al. 2005] approximation of the Hessian. This is then coupled with search of topology changes to form a customized discrete-continuous topology search method. We defer discussion of the details of this component to Section 5.

4.2 Dual Update

Our Lagrangian in (3) is nonsmooth in λ. When we exceed the distortion bound, i.e., $E_d(T, U) > b_d$, we have $\lambda = \infty$; when on boundary of the feasible set, $E_d(T, U) = b_d$, we have a finite $\lambda \in \mathbb{R}_+$; and finally, in the strict interior of the set of feasible distortions, we have $\lambda = 0$. While these conditions nicely characterize optimality, we need a way to iterate on λ towards the solution in a smooth and robust manner irrespective of whether we are locally exploring a feasible or infeasible distortion.

At iteration k we thus smoothly approximate the multiplier's behavior by adding a simple quadratic regularizer to the Lagrangian. This keeps the updated multiplier *proximal* to the previous iterate's estimate via Powell's extension [Powell 1973] of the Augmented Lagrangian to the inequality constrained setting,

$$\min_{T,V} \max_{\lambda \geq 0} E_s(T) + \lambda\Big(E_d(T, U) - b_d\Big) - \tfrac{1}{2}(\lambda - \lambda^{k-1})^2. \tag{6}$$

At each iteration, to find λ^k we simply fix vertices and topology to (T^{k-1}, U^{k-1}). Optimality of (6) then gives us our corresponding

dual update in closed form

$$\lambda^k \leftarrow \max\left(0, \left(E_d(T^{k-1}, U^{k-1}) - b_d\right) + \lambda^{k-1}\right). \quad (7)$$

5 COUPLED DISCRETE-CONTINUOUS DESCENT

To perform our primal update, we seek to minimize the Lagrangian over both continuous changes in vertex positions *and* discrete changes in topology. To optimize over topology we could potentially perform exhaustive search over the graph of all possible mesh changes. This approach, however, is intractable for any practical mesh size. Instead, we construct a local search algorithm for topological updates that is inspired by the standard descent process of optimizing a smooth energy over vertex positions.

For smooth descent methods it is standard to formulate a local approximation of the energy function, use it to estimate the direction for the gradient descent step, and then search along the proposed direction to find the step magnitude that ensures significant decrease in energy. We extend this process to include search over discrete variations in topology. In analogy to seeking a continuous search direction, at the start of each new primal solve we will build many localized energy approximations to search for a likely candidate mesh operation to repeatedly propagate topology change, i.e., cutting or merging, over our UV mesh. Each inner iterate of the primal solve will successively apply this operation in combination with standard smooth descent to explore discrete-continuous descent until no further progress is made.

In the remainder of this section we formulate the energy model that is used to evaluate the candidate topology operations (Section 5.1), enumerate the topology operations used in our algorithm (Section 5.2), explain how we find the search direction (Section 5.3), and iteratively explore the identified direction (Section 5.4).

5.1 Energy Model for Topology Updates

We next devise an approximate energy model to efficiently estimate the effect of a potential topological change. To keep our notation simplified, in the following we use i or j to index the inner loop of the primal update and k for indexing the outer loop. At each outer iterate k, the locally optimal Lagrangian for any proposed topology T^i is

$$\ell(T^i) = \min_U L(T^i, U) = E_s(T^i) + \lambda^k \min_U E_d(T^i, U). \quad (8)$$

Then, for any valid topology-changing operation, $o^{i,j} : T^i \rightarrow T^j$, the resultant change in Lagrangian is $\Delta\ell(o^{i,j}) = \ell(T^j) - \ell(T^i)$. We seek a valid operation that will produce large decrease in the Lagrangian.

Since minimizing over U for every potential topology change is impractical for large meshes, we start from a known (T^i, U^i), and approximate the change in Lagrangian by restricting the distortion update to the locally changed vertex stencil $U^{i,j}$ of the applied topological operation $o^{i,j}$. We construct this stencil to include only the vertices affected by the topological operation and their immediate neighbors. We hold all other vertices U_s, shared in common with T^i, fixed in the mesh to the same positions previously held in U^i. Thus vertex positions after the update are $U^j = (U^{i,j}, U_s)$. Our approximate change in Lagrangian is then

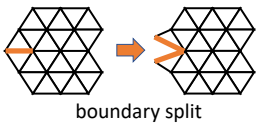
$$d(o^{i,j}) = E_s(T^i) + \lambda^k \min_{U^{i,j}} E_d\left(T^i, (U^{i,j}, U_s)\right) - L(T^i, U^i, \lambda^k). \quad (9)$$

Evaluating d requires solving continuous distortion optimization with respect to stencil vertices $U^{i,j}$. We use Newton-type smooth descent for this; see Section 6.3 for details.

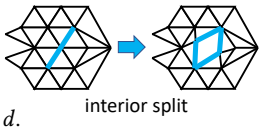
5.2 Local Topological Operations

We consider descent with topology changes induced by three mesh operations.

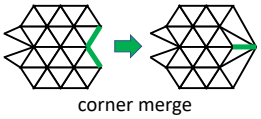
Boundary Vertex Split. Boundary vertices can be split along any interior incident edges. Each such candidate split generates two duplicate vertices forming the stencil to compute d. When splitting a boundary vertex along an edge connecting to another boundary vertex, we either remove a hole or else produce a new chart in our UV map. For the latter case, we generate four duplicate vertices forming the stencil to compute d.



Interior Vertex Split. Interior vertices can be split along any pair of incident edges. Each such candidate split generates two duplicate vertices forming the stencil to compute d.



Corner Merge. Corners are formed by three UV vertices corresponding to the tail edge of a cut seam on the input surface. Merging the end vertices generates a single new vertex forming the stencil to compute d. Merging requires extra care here. Unlike vertex splitting, an initial location for the newly merged vertex must be selected. Naïve merging can violate local injectivity and so prevent progress if we are working with barrier-type energies like symmetric Dirichlet. We initialize a merged vertex to the average of its parent vertices. If inversion is detected, we then apply Agmon's relaxation [1954] to project to an inversion-free position iteratively. On rare occasions this will not suffice and so we remove the proposed operation from our queue.



5.3 Topology Search Candidates

In analogy to computing a continuous search direction, at the start of each new primal solve we search for a candidate mesh operation to propagate descent. We first consider boundary vertex splits originating at one of m boundary vertices in the current topology T^k. To reduce unnecessary computational overhead we select a subset of $m_{\text{boundary}} = m^{0.8}$ boundary vertices that might have the largest effect on the energy. To estimate the priority, we compute the standard deviation over all the distortion energy gradients acting on the boundary vertex contributed from incident elements. We then pick boundary vertices with largest deviation, i.e. where distortion might best be alleviated by cutting. We use the selected vertices to initiate boundary vertex splits, and consider all current seams for a corner merge, building a set O^k of potential topological operations. We then find a minimizer for the first topological change:

$$o^{0,1} = \operatorname*{argmin}_{o \in O^k} d(o). \quad (10)$$

Since we use local support for vertex stencils, all queried d in this minimization can be efficiently evaluated in parallel.

It is possible for none of the boundary cuts to yield a decrease in energy (i.e., $d(o^{0,1}) \geq 0$). In this case, we consider initiating a new seam via an interior vertex split. In a similar process, we then pick $m_{\text{interior}} = (n-m)^{0.8}$ interior vertices with the largest deviation of distortion energy gradients, and build O^k as the set of all interior vertex splits on those vertices. We select the d-minimizing interior split operation $o^{0,1} \in O^k$ using (10). Once we identify the best topological change $o^{0,1}$, i.e. our search direction, we next expand it via iterative propagation.

5.4 Iterated Search, Propagation and Descent

We propagate the best seed topological operation $o^{0,1}$ by iteratively applying operations of the same type (a boundary or interior split, or a merge). During the primal update we alternate between propagating topological change, updating T^i, and a smooth coordinate update on U^i.

Each topology propagation step first generates a set of all possible mesh operations $\mathcal{E}(o^{i-1,i}, T^i)$ that could continue to propagate the previous operation $o^{i-1,i}$ on the current topology T^i; e.g., all valid edges to extend an existing seam at its tail. Figure 2 illustrates how we propagate each type of topological operation. Here we then pick a d-minimizing operation from this small set of candidate operations for our next operation to apply,

$$o^{i,i+1} = \underset{o \in \mathcal{E}(o^{i-1,i}, T^i)}{\operatorname{argmin}} \; d(o). \tag{11}$$

Since each change in topology alternates with smooth coordinate descent and we wish to come close to a local minimized of distortion in each primal solve, we work to avoid introducing too many topological changes as long as the coordinate update provides significant energy improvement. We denote the threshold for desired estimated decrease by δ^i. It is initialized to $\delta^0 = 0$ in the first iteration, and then set to the distortion energy improvement from each successive coordinate update. At any iteration, if $d(o^{i,i+1}) < \delta^i$ we update the topology based on the best operation $T^{i+1} \leftarrow o^{i,i+1}(T^i)$ and keep it unchanged otherwise, $T^{i+1} \leftarrow T^i$.

The subsequent coordinate update then simply applies a single step of projected Newton descent with line search to update the vertex coordinates U^i; see Section 6.3 for details. We then ask for the next topology update to gain similar or greater magnitude decrease by setting $\delta^i = E_d(U^i) - E_d(U^{i-1})$.

This process terminates at iteration $i+1$ when smooth iterations have converged (see Section 6.3 on varying conditions for this) *and* the propagation of the seed mesh operation produces no further descent. We then set $(T^k, U^k) \leftarrow (T^{i+1}, U^{i+1})$ and begin the next outer iterate $k+1$ with the dual variable update.

6 THE OPTCUTS ALGORITHM

With the key components now in place, Algorithm 1 summarizes our full OptCuts algorithm in pseudocode, including details on convergence detection and termination. Here, in this section, we next discuss key details including termination/convergence criteria, initialization, preconditioning, and inclusion of global bijectivity constraints.

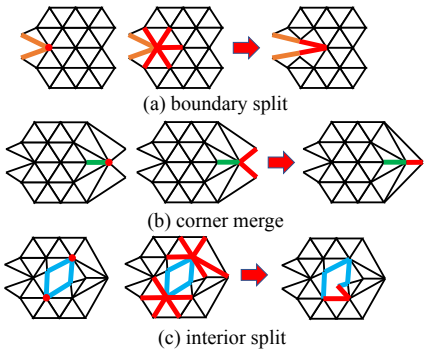


Fig. 2. Propagation of the boundary split (orange), corner merge (green), and interior split (blue) topology operations. When propagating a boundary split all edges that can continue the boundary splitting from the current cut's tail are set as propagation candidates (a). At corner merges we have a single candidate to query—the new corner at the current tail (b). Finally, for propagating an interior split, first we choose an incident edge from either of the two possible seam tails; once split, propagation then follows in the same fashion as the boundary split for following propagation steps (c).

6.1 Termination and Convergence

Our primal solution is *converged* whenever it is stationary with respect to variations in both topology *and* vertex position for a fixed λ^k multiplier value. Numerically, primal convergence is declared for any (T^i, U^i) pair when 1.) the topology descent is stationary—meaning there are no further available mesh operations at the current multiplier λ^k value that will give decrease to the Lagrangian below $L(T^i, U^i, \lambda^k)$; and 2.) the smooth distortion energy at topology T^i is sufficiently locally minimized so that $\|\nabla_U E_d(T^i, U^i)\| \leq 10^{-6}$. When both our primal solve is converged *and* our dual variable is likewise close to stationary so that $|\lambda^k - \lambda^{k-1}| < 10^{-3}$, OptCuts terminates with a solution that is numerically *converged* to a local minimizer of (1) with respect to all available variations in topology provided by our merge and cut operations.

Importantly, our OptCuts algorithm and its implementation *do not* apply a maximum iteration cap anywhere in any of our iteration loops. All presented results and corresponding timings are from solutions that have converged either as defined above *or* by a cyclic stationarity condition required for a robust implementation that warrants further discussion in text rather than in pseudocode—we cover this immediately below.

First, while we converge with respect to the mesh operations we make available, we of course do not cover all possible topological operations on a mesh. Our choice of propagated cutting and merging operations was selected to cover the basic topology changes that we find suitably expressive for our parameterization task. However, we expect many others to be useful; indeed, below in Section 7.5 we also consider and compare with another interesting mesh operation subset and remark that there is interesting future work to find improved mesh operation subsets for OptCuts. In the meantime, for *any* discrete subset of topological operations that do not fully cover all possible mesh changes, there can and will be discrete sets of points that are stationary with respect to variations of the available mesh operations. In turn, while some of these points will satisfy the imposed distortion bound, they may not match the bound. In such

ALGORITHM 1: OptCuts

Given: M, T^0, U^0, b_d
Initialize: $\lambda^0 \leftarrow 0$, converged $\leftarrow$ **false**, $k \leftarrow 1$
while ! converged **do**
  $\lambda^k \leftarrow \max\left(0, \left(E_d(T^{k-1}, U^{k-1}) - b_d\right) + \lambda^{k-1}\right)$ // Dual Update (§4.2)
  d_stationary $\leftarrow |\lambda^k - \lambda^{k-1}| < 10^{-3}$
  // Primal update (§4.1, details in §5):
  $i \leftarrow 0$, $(T^i, U^i) \leftarrow (T^k, U^k)$, $\delta^i \leftarrow 0$
  v_converged $\leftarrow$ **false**, t_stationary $\leftarrow$ **false**, t_stopped $\leftarrow$ **false**
  $o^{0,1} \leftarrow \text{argmin}_{o \in O^k}\, d(o)$ // Get seed topological operation (§5.3)
  if $d(o^{0,1}) \geq 0$
    t_stationary $\leftarrow$ **true**
  end if
  while ! (v_converged **and** t_stopped) **do**
    t_stopped $\leftarrow$ **true**
    if ! t_stationary
      $o^{i,i+1} \leftarrow \text{argmin}_{o \in \mathcal{E}(o^{i-1,i}, T^i)}\, d(o)$ // Get candidate op (§5.4)
      if $d(o^{i,i+1}) < \delta^i$ // Sufficient decrease
        $T^{i+1} \leftarrow o^{i,i+1}(T^i)$ // Update topology (§5.4)
        $U^{i+1}_{\text{init}} \leftarrow (U^{i,i+1}, U_s)$ // Initialize new geometry (§5.4)
        t_stopped $\leftarrow$ **false**
      else
        $T^{i+1} \leftarrow T^i$, $U^{i+1}_{\text{init}} \leftarrow U^i$
      end if
    end if
    $i \leftarrow i + 1$
    $g \leftarrow \nabla_U E_d(T^i, U^i_{\text{init}})$ // Distortion energy gradient
    if $\|g\| > 10^{-6}$
      $H \leftarrow \text{Project}(\nabla^2_U E_d(T^i, U^i_{\text{init}}))$ // Projected-Hessian (§6.3)
      Solve: $Hp = -g$ // Get smooth descent direction p (§6.3)
      $\alpha \leftarrow \text{LineSearch}(U^i_{\text{init}}, p, E_d)$ // Line-search (§6.3)
      $U^i \leftarrow U^i_{\text{init}} + \alpha p$
      $\delta^i \leftarrow E_d(T^i, U^i) - E_d(T^i, U^i_{\text{init}})$
      if $|\delta^i / E_d(T^i, U^i_{\text{init}})| < 10^{-6}\alpha$ **and !** t_stationary
        break // Safe to early exit
      end if
    else
      v_converged $\leftarrow$ **true**
    end if
  end while
  $(T^k, U^k) \leftarrow (T^i, U^i)$
  $k \leftarrow k + 1$
  converged $\leftarrow$ v_converged **and** t_stationary **and** d_stationary
end while
$(T^*, U^*) \leftarrow (T^{k-1}, U^{k-1})$

cases, without additional handling and preparation for cycling behavior, OptCuts would oscillate without stopping between a number of high-quality, close-to-optimal solutions. To handle this condition, we employ a simple cycle detection strategy, where after each primal solve we save hashes (E_s, E_d, λ) of solution triplets, and the best visited solution. When we detect that a duplicate solution is being revisited, we terminate with the best visited solution. Of the 254 examples computed with OptCuts in Section 7, 171 terminate with cycling about stationarity, while the remainder converge to a stationary point.

6.2 Initialization

To initialize our UV map, (T^0, U^0), for an input surface, we map its initial seam to a circle preserving edge lengths and parameterize the remaining vertices with Tutte's embedding [1963] using uniform weights to ensure bijectivity. For disk-topology surfaces, we simply pick the longest boundary as an initial seam while for genus-0 closed surfaces, we randomly pick two connected edges as an initial seam. For higher-genus surfaces we detect homology generators (using the `cut_to_disk` function in libigl [Jacobson et al. 2017]) and then connect them to form an initial seam. We then set (T^0, U^0) to this initial topology with vertex positions minimizing distortion, E_d, on the initial embedding, and start OptCuts by initializing $\lambda^0 = 0$. This sets the first iteration to initially ignore distortion and so begin by exploring a shortening of seam lengths.

6.3 Minimizing Distortion

Each smooth descent step takes as input a possibly updated UV map from the previous topology descent step and applies a single Newton-type iteration to reduce E_d over changes in vertex positions U while holding topology, T, fixed. As our distortion energies E_d are generally nonconvex, we apply Teran et al.'s [2005] projected-Newton (PN) method to gain a modified Hessian proxy that is guaranteed PSD. We parallelized PN's per-element Hessian construction, projection and assembly into the PN Hessian proxy, H, with Intel TBB [Reinders 2007]. We then apply Pardiso [Petra et al. 2014a,b] to solve the resulting linear system for the next descent direction. For line search we first use Smith and Schaefer's [2015] line-search filter to avoid element inversion followed by standard line search with Armijo conditions [1966] to ensure sufficient energy decrease. Finally, we employ one additional optimization that is specific to OptCuts: on smooth steps where we are *not* yet stationary with respect to topology operations, there is no need to expend extra iterations to gain tight convergence on distortion. In these cases we terminate iterations when we simply have the more relaxed condition of small change in energy. On the other hand, when a stationary topology is reached, we then always require and apply full convergence to small gradient norm of E_d; see Algorithm 1.

6.4 Global Bijectivity

Following Jiang et al. [2017], we apply global bijectivity constraints to our mapping by (re-)triangulating void regions each time we update our UV map. We use the Triangle library [Shewchuk 1996]. Void regions consist of all holes as well as a loose bounding box enclosing the UV map. We then add to our distortion energy E_d an additional term *not included in the distortion bound constraint* on these void triangles to form a collapse-preventing energy for the added negative-space triangles during each primal optimization iteration.

Our continuous descent steps on vertices then remains otherwise unchanged. However, for each topology step, we need to ensure that negative-space triangles are inserted correctly for each query of a candidate local topological operation. Here, there are two simple modifications we need to apply. First, for local queries of descent

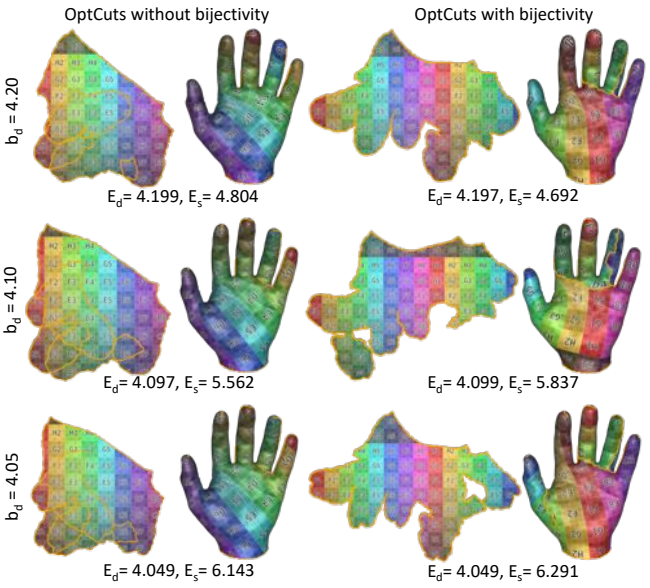


Fig. 3. UV maps generated for the hand model by OptCuts both without (left) and with (right) bijectivity enforcement enabled, as we vary the distortion bound b_d over 4.2, 4.1, and 4.05 respectively from top to bottom.

per candidate query minimizations evaluating d, we need only triangulate the void region about the union of one rings around the participating stencil vertices. This provides a sufficient scaffold to ensure bijectivity in the local solve without growing the problem size. Second, for each candidate edge split we need to carefully pull apart the new split vertices to form an initial void to build the scaffold triangulation. We pull the duplicated vertices along their curvature normals far enough to form a gap while ensuring no triangle elements are inverted.

7 EVALUATION

In the following sections we evaluate OptCuts and a range of other UV parameterization methods and commercial tools on meshes from a benchmark mesh dataset containing a diverse range of 71 surfaces commonly employed in geometry processing research. Meshes in this benchmark are organic shapes with intricate geometric details and varying resolutions ranging from 80 to 10K vertices (with 3.7K average vertex count); we also employ a set of refined Lucy models with resolutions increasing from 2.5K to 48K vertices.

For all examples reported in all of the following sections we note that OptCuts always successfully converges to a solution achieving the targeted symmetric Dirichlet energy distortion bound b_d (see our discussion in Section 6.1). Thus, in many experiments we will focus on evaluating the seam length measure, E_s, as an evaluation metric with the goal being to achieve any set target distortion bound with the shortest possible seam lengths. Recall that our seam-length measure, E_s, is normalized by square root of mesh area and so provides a consistent and comparable measure across examples and methods.

All our evaluations in the following sections employ our implementation of OptCuts using the libigl [Jacobson et al. 2017] library

Table 1. Seam length and performance statistics for our OptCuts algorithm applied to our benchmark of 71 surfaces as we decrease the distortion bound, b_d; each example is run with and without our additional enforcement of global bijectivity enabled. All examples converge satisfying the requested distortion bound. Note that here min E_s gives the minimum nonzero seam length.

b_d	bijectivity	E_s			time (s)		
		avg	min	max	avg	min	max
4.2	OFF	3.936	0.289	14.545	92.5	0.3	417.8
	ON	4.006	0.289	17.063	187.0	0.6	983.3
4.1	OFF	4.919	0.752	17.980	144.6	4.0	886.9
	ON	5.346	0.860	21.595	274.6	6.9	1767.8
4.05	OFF	6.416	1.035	21.566	223.3	3.9	1398.1
	ON	7.256	0.932	29.596	479.5	7.2	5141.2

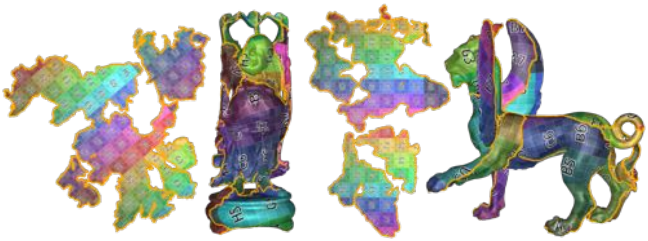


Fig. 4. UV maps generated by OptCuts on higher-genus input surfaces, with distortion bound $b_d = 4.1$, and bijectivity enforcement enabled.

for rapid prototyping and testing. Implementation details are reported above in Section 6. We are releasing our implementation publicly for future development and application. All experiments presented in this paper were executed on a Macbook Pro with 3.1 GHz quad-core Intel Core i7 CPU and 16 GB 2133 MHz LPDDR3 memory.

7.1 OptCuts Evaluation

We first execute OptCuts on our full set of benchmark examples over a range of decreasing target distortion bounds, $b_d \in \{4.2, 4.1, 4.05\}$. For each bound we run OptCuts twice—in each of its modes: once with the global bijectivity constraint enabled, and once with just local-injectivity, i.e. with bijection disabled. As discussed, for all examples in the benchmark, at all three distortion bounds, and both with and without bijectivity constraints, OptCuts always converges to a locally minimal seam length satisfying the prescribed distortion bound. See Figures 3 and 5 for examples of our results as we vary these terms and see our Supplemental materials for a detailed table and visualization of all our OptCuts results, as well as animations of the OptCuts iterations, on the benchmark set. In Table 1 we summarize statistics for OptCuts on the benchmark set and observe that as we increase constraint on the examples by tightening distortion bounds and/or enforcing additional bijectivity constraints, longer seams at convergence and comparably longer running times result as expected; see also Figure 3. In Figure 4 we additionally run OptCuts to convergence on two higher-genus surfaces (buddha with genus 6 and feline with genus 2), enforcing global bijectivity constraints and distortion constraint of $b_d = 4.1$.

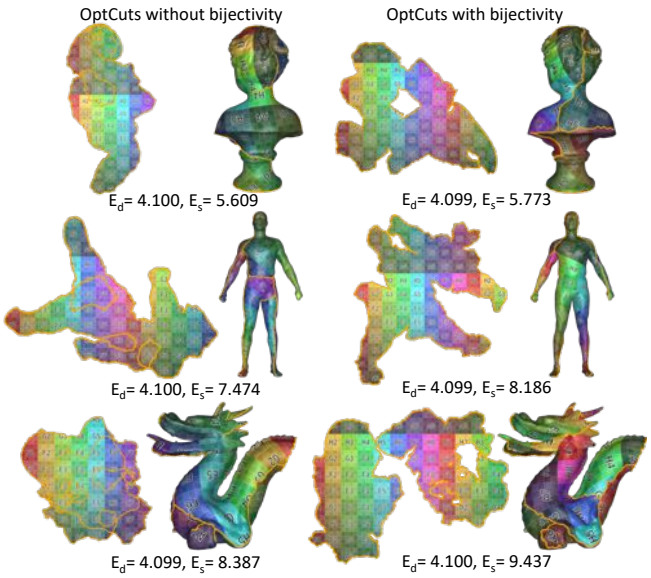


Fig. 5. UV maps generated by OptCuts both without (left) and with (right) bijectivity enforcement enabled with the distortion bound set to $b_d = 4.1$.

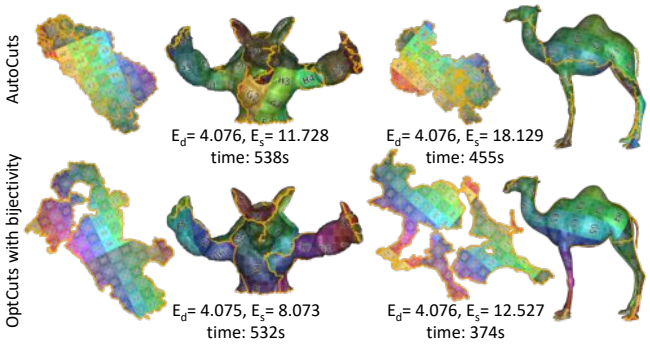


Fig. 6. Comparisons between parameterizations generated by AutoCuts (top) and OptCuts with bijectivity enabled (bottom) on Armadillo (left) and Camel (right) model. Here OptCuts obtains improved seam length measures for the same distortion bound while maintaining additional bijectivity constraints.

7.2 Comparisons to AutoCuts

As discussed in Section 2, AutoCuts [Poranne et al. 2017] couples the optimization of a weighted sum of a seam-penalty term that tries to pull triangles together, scaled by a scalar λ, and the symmetric-Dirichlet distortion energy for parameterization. AutoCuts runs in two modes: one interactively driven by direct user guidance throughout, and the second is fully automated. AutoCuts is generally most effective in interactive mode with a user in-the-loop and this is what is currently supported in the official AutoCuts implementation. To recreate the automated-mode version of AutoCuts we began with the official AutoCuts implementation and, with guidance from the authors of AutoCuts, set their parameters for recreating their fully automated method to those that performed best across the full set of benchmark examples. For details of this final automated-mode AutoCuts implementation we evaluate with see our Supplement.

Table 2. Distortion, seam length, and performance statistics comparing automated-mode AutoCuts (which does not provide bijectivity enforcement) and our OptCuts algorithm (both with and without bijectivity constraints enabled) on all input surfaces in our benchmark dataset. For each example we set the OptCuts target distortion (b_d) to the achieved (uncontrollable) AutoCuts output distortion on the same example (ranging from 4.016 to 4.187). OptCuts in both modes (without and *with* bijectivity) obtains shorter seam lengths and faster run times when compared to AutoCuts.

method	bijectivity	avg. E_d	avg. E_s	avg. time (s)
AutoCuts	N/A	4.077	7.9572	592.6
OptCuts	ON	4.075	5.9182	344.4
	OFF	4.075	5.4450	156.5

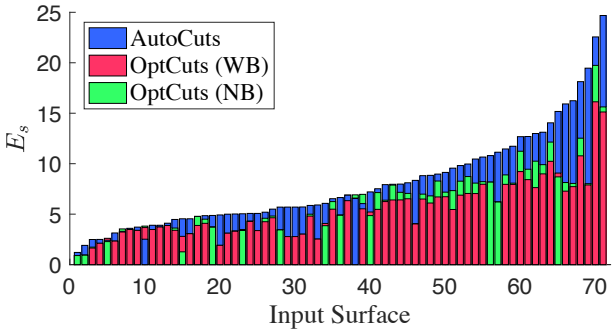


Fig. 7. A per-example comparison of seam lengths achieved by AutoCuts and OptCuts sorted by total seam length. OptCuts without bijectivity constraints in red achieves shorter seams than AutoCuts in blue for all except for two inputs (see text for details on these two examples) while OptCuts with additional bijectivity constraints in green (AutoCuts does not apply bijectvity constraints) still performs favorably in seam length despite the added bijectivity constraints.

Recall that while AutoCuts enables control of distortion indirectly, it can not and does not guarantee that it will achieve a particular distortion bound. Thus, we perform a re-analysis to gain an understanding of the comparable performance and quality of AutoCuts and OptCuts. We first run automated-mode AutoCuts on each example in the benchmark data set to termination. For each of the examples we then set the resulting distortion measure from the AutoCuts output as the target bound b_d for OptCuts. Here we run OptCuts both with and without global-bijectivity constraints enabled (recall that AutoCuts does not provide bijectivity enforcement in a fully automated mode).

As summarized in Table 2, both versions of OptCuts, both without bijectivity and *with* bijectivity require less time to on average to generate UV maps that have significantly shorter seam lengths than AutoCuts, while achieving the same targeted level of distortion. See Figure 6 for visual comparisons and our Supplemental for details of all comparison examples. In Figure 7 we provide a more detailed breakdown of the seam-length comparison for each model in the benchmark with bars sorted by total seam length. AutoCuts produced shorter seams than OptCuts without bijectivity for two meshes: cathead ($E_s = 2.52$ vs $E_s = 3.67$) and KingKong ($E_s = 6.55$ vs $E_s = 6.59$). Cathead mesh is very coarse containing only 131 vertices, thus starting with a triangle soup provides an effective strategy.

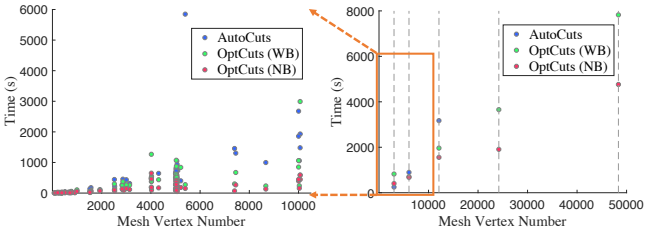


Fig. 8. Runtime scatterplots for AutoCuts (blue) and OptCuts with bijectivity enforcement (green) and OptCuts without bijectivity (red) across all examples in our benchmark set (left) and for increasing resolutions of the Lucy model (right). Here (left and right) we observe that as mesh resolution increases, the performance gap between AuotCuts and OptCuts increases while (right) for moderately large examples, e.g., the two higher resolution Lucy models, at 24k and 48k vertices respectively, there is no data points for AutoCuts as the official implementation is already out of memory; see our discussion below.

KingKong mesh (10K vertices) obtains comparable seams lengths from both methods, while AutoCuts is substantially slower than our approach ($t = 1858$s vs $t = 394$s).

Scalability. AutoCuts in both automated and user-guided modes duplicates all vertices in the UV mesh. We observe that this has a significant impact on scalability. Here we analyze how vertex count effects run-time (and implicitly) memory on our full benchmark data-set in Figure 8a, and over the Lucy meshes with increasing resolution in Figure 8b. This breakdown of the experiments largely repeats our observations from above that both modes of OptCuts generally outperform AutoCuts while providing better seam-length measures. Here there are two added observations. First, as mesh resolution increases, the performance gap between AuotCuts and OptCuts increases. Second, for moderately large examples, e.g., the two higher resolution Lucy models, at 24k and 48k vertices respectively, the official AutoCuts implementation is already out of memory for both automated and user-guided modes, with their system already 6× larger upon initialization. On the other hand OptCuts increases vertex counts slowly along cuts and actively seeks to reduce seam lengths, thus, reducing the number of duplicated vertices and overall system size.

7.3 Comparisons to Other Methods

Comparison to the Geometry Images Cutting Strategy. As discussed above our topology descent step leverages a small subset of local topological operations that we have so far found most expressive, efficient and effective. In choosing this subset we experimented with a number of options. In particular we found a subset of more aggressive topological operations taken from Geometry Images [Gu et al. 2002] very effective. Geometry Images leverages extrema-to-boundary (EB) cuts that connect the current boundary to the most distorted point, under current parameterization, using the shortest geodesic path. The advantage of this cutting strategy in the OptCuts setting is that it introduces more extreme topological updates at each iteration, potentially saving computational effort. We experimented with replacing our topological search in OptCuts with the EB cuts. As expected we find that the resulting optimization does indeed converge faster, but we likewise find that the more aggressive EB cutting converges to longer-seam solutions, especially when we seek tighter distortion bounds and when iterations treat nearly-isometric UV maps where extremities are less prominent. See Table 4 and Figure 12 for a comparison of OptCuts with these two cutting strategies.

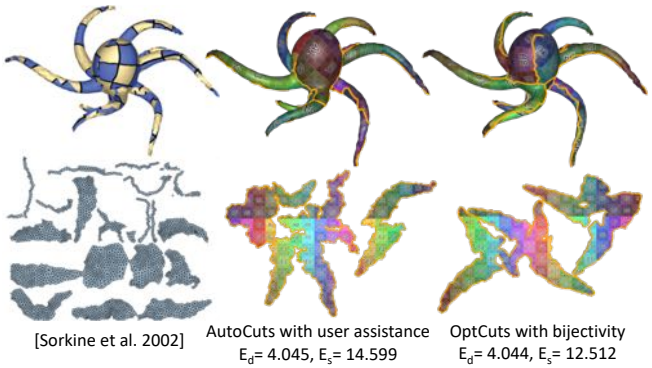


Fig. 9. Here we recreate a comparison with Sorkine et al.'s parameterization [2002] (left) and an interactive-mode AutoCuts example created to compare with it from Poranne et al. [2017] (middle). Right: we run OptCuts with bijectivity constraints enabled, setting its distortion bound (b_d) to match the distortion of the interactively user-assisted AutoCuts output. OptCuts satisfies the bound while improving seam-length over the interactively created example.

Comparison to interactive-mode AutoCuts and Sorkine et al. [2002]. Finally, in Figure 9 we repeat a comparison with Sorkine et al.'s parameterization [2002] and the interactive-mode AutoCuts example created to compare with it from Poranne et al. [2017], Figure 11 in that paper. Here we run OptCuts, with bijectivity constraints enabled, setting its distortion bound (b_d) to match the distortion achieved by the interactive user assisted AutoCuts output. We observe that OptCuts, Figure 9, right, then automatically achieves improved seam-length over the interactively created AutoCuts example, Figure 9d middle, as well as that of Sorkine et al. [2002], Figure 9 left, and the automated-mode AutoCuts (not shown here).

7.4 Comparison to Commercial Software Tools

Commercial software tools currently offer a range of automated approaches for UV-map creation. We picked five complex models from our dataset: Lucy, octopus, rgb_dragon, statue_5, and three_man, and asked two experienced artists to create UV maps using the fully automated modes of three commercial software tools: Unwrella[2], ZBrush[3], and Maya[4], with default parameter settings. ZBrush generates a single chart UV map and so generally produces higher distortion but lower seam lengths. On the other hand Maya and Unwrella cut the surface into multiple charts to achieve low distortion and efficient packing, generally at the cost of longer seam lengths. ZBrush, and Unwrella in organic mode generally produced best UV maps; the hard-surface mode of Unwrella and Maya both generally cut the surface into many small pieces with very long seams

[2]https://www.unwrella.com/
[3]http://pixologic.com/
[4]https://www.autodesk.com/products/maya/overview

Table 3. Summary of distortion and seam-length measures for comparison between five UV-maps created by artists with two commercial tools, Unwrella (organic mode) and ZBrush, and the corresponding results obtained by OptCuts (with bijectvity) when its distortion bound is set to the commercial results' map distortion.

Model	Unwrella Organic		OptCuts (Unwrella b_d)		ZBrush		OptCuts (ZBrush b_d)	
	E_d	E_s	E_d	E_s	E_d	E_s	E_d	E_s
Lucy	4.10	19.7	4.10	10.6	4.26	6.6	4.26	6.5
octopus	4.07	21.7	4.07	15.7	4.22	14.1	4.21	14.4
rgb_dragon	4.22	33.5	4.22	16.1	4.74	9.2	4.74	8.6
statue_5	4.08	14.3	4.08	5.1	4.16	4.0	4.15	3.0
three_man	4.03	19.2	4.03	13.5	4.15	9.0	4.15	9.3

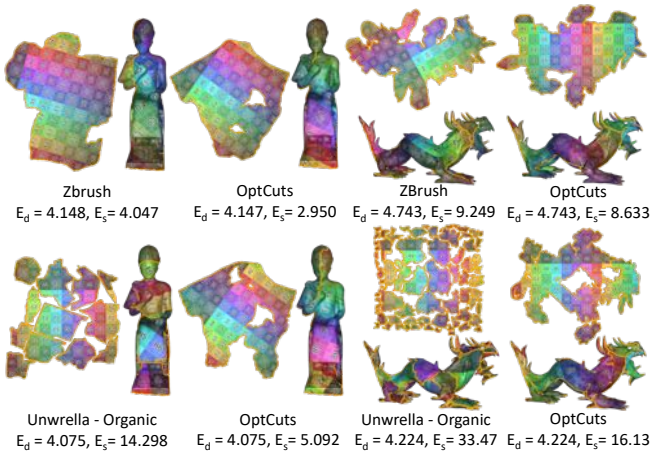


Fig. 10. Commercial software tools comparison. Here we compare the results generated by UV-mapping with commercial software tools ZBrush (top) and Unwrella (bottom) with those obtained by OptCuts with bijectivity constraints enabled. For each comparison example we set OptCuts to match the tool's obtained distortion bound, obtaining improved seam measures.

and occasionally produced UV meshes with local inversions and non-manifold vertices. We thus took just the output of ZBrush and Unwrella, in organic mode, for all five examples. Since the details of these methods are not public, and they might optimize for a slightly different distortion objective, we make comparisons more favorable to them and *improve* their output with respect to our evaluation metrics by minimizing the distortion energy E_d while keeping their seams. In Table 3 we compare this improved output to OptCuts for each mesh/tool pairing. In each row we respectively give the distortion and seam measure of each distortion-optimized commercial output, followed by the result obtained by OptCuts when we set the OptCuts distortion bound to match the commercial method's final, optimized distortion. Across all examples for the same level of distortion, we observe that OptCuts obtains shorter seams in eight out of the ten comparisons; see also Figure 10 for visual comparisons and our Supplemental for all results.

7.5 Variations

Regional Seam Placement. Discontinuities produced by seams make texture assignment challenging and can lead to unpleasant rendering artifacts. Thus, UV artists often place seams away from

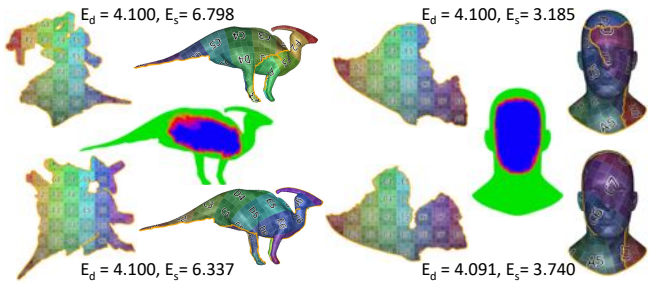


Fig. 11. OptCuts can also enforce additional constraints allowing users to bias or even fully prevent the resulting parameterization from placing cuts in salient regions by painting a map over the three-dimensional surface showing where seams are more or less desirable. Middle: a user-painted salience map goes from blue (no seams) to green (seams allowed). With these additional provided constraints OptCuts continues to yield low seam length and to satisfy the distortion bounds while also enabling additional controls on seam placement to avoid salient regions (bottom); compare with the results obtained from our undirected OptCuts (top). Note as well that here we happen to achieve shorter seams with additional user constraints for the dinosaur model as OptCuts computes local minimizers.

the salient surface regions, e.g., a face. We extend OptCuts to directly enable users to bias or even fully prevent the resulting parameterization from placing cuts in specified regions by painting an intensity weight, $w \in [1.0, +\infty]$ over the three-dimensional surface. As intensities grow we correspondingly penalize the cost of seam lengths in those regions. Correspondingly, in OptCuts we update the seam-length measure by looking up, per-edge i, the local painted weight, w_i, and then simply updating the edge measure to sum to

$$E_s = \frac{1}{\sqrt{(\sum_{t\in\mathcal{F}} |A_t|)/\pi}} \sum_{i\in\mathcal{S}} w_i |e_i|. \tag{12}$$

We demonstrate the results of OptCuts with these additional user-directed constraints in Figure 11 bottom, where the user-painted salience map goes from blue with $w = 100$ to green with $w = 1$ in Figure 11 middle, and compare with the results obtained from our undirected OptCuts results in Figure 11 top. Notice that with the addition of these constraints, OptCuts continues to yield low seam length and to satisfy the distortion bound, but now also enables additional controls on seam placement.

OptCuts Polishing. In many cases artists and users may have pre-existing UV-mappings, methods and/or pipelines that would benefit from further improvement. Here we observe that OptCuts can take *any* valid embedding as a starting point to improve seam length and/or distortion. To improve just seam length we can set the input map's distortion as an input bound for OptCuts and then optimize the input map for improved seam-length. Alternately, to also improve distortion, we can also apply a lower distortion bound. Likewise, as discussed earlier, constrained distortion optimization is highly nonconvex with many local minima; thus polishing multiple warm-started solutions can be an effective way to use OptCuts to explore many interesting locally optimal embeddings.

Here, we first explore taking UV-mappings that are quickly generated by Seamster [Sheffer and Hart 2002] as input. Seamster is a highly efficient seam cutting strategy that detects local curvature

Table 4. Here we summarize statistics for our benchmark set comparing performance and resulting seam-lengths obtained by our standard OptCuts topology operations as compared with an alternative version of OptCuts employing the more aggressive topology operation of cutting from extrema-to-boundary (EB) from Geometry Images [Gu et al. 2002]. We run all examples both with and without bijectivity enforcement. As the EB strategy is more aggressive, we observe faster runtimes at the cost of longer seam-lengths. Note that here min E_s gives the minimum nonzero seam length.

b_d	OptCuts	E_s			time (s)		
		avg	min	max	avg	min	max
4.2	Standard	3.936	0.289	14.545	92.5	0.3	417.8
	EB	3.971	0.754	14.929	13.6	0.1	72.1
4.1	Standard	4.919	0.752	17.980	144.6	4.0	886.9
	EB	5.026	1.207	16.895	17.9	0.1	87.2
4.05	Standard	6.416	1.035	21.566	223.3	3.9	1398.1
	EB	6.608	1.207	23.051	25.3	0.1	115.4

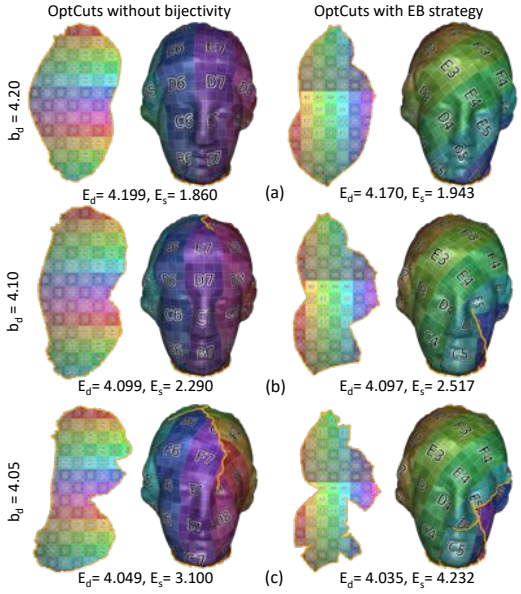


Fig. 12. Comparisons between OptCuts using our standard topology operations (left), and OptCuts using the extrema-to-boundary (EB) cutting strategy from Geometry Images [Gu et al. 2002] (right). Here we decrease distortion bounds (top to bottom). EB cuts are aggressive—leading to faster convergence but at the cost of larger seam-length measures at convergence.

extrema and connects them with a minimal spanning tree. This approach can be sensitive to the user-set parameters such as size of surface regions for computing local extrema, which is a shape-dependent parameter that can require parameter tuning. In this experiment, we pick two models, cow and a triceratops, that have been successfully cut by hand-tuning Seamster [Sheffer and Hart 2002]. We then apply OptCuts to these two Seamster-generated maps, setting the OptCuts distortion bound to the original distortion of the Seamster-generated maps. In Figure 13 we compare the original hand-tuned Seamster output in (a), with the resulting OptCuts-polished results in (b), and finally with the direct results of optimizing OptCuts from scratch [5] on these examples while satisfying the same distortion bound in (c).

[5]I.e., directly starting from the Tutte embedding of each mesh.

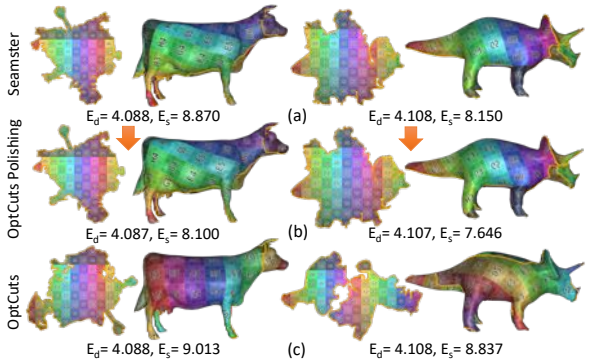


Fig. 13. UV-map Polishing with OptCuts starting with Seamster [Sheffer and Hart 2002]. OptCuts can take *any* valid embedding as a starting point to improve seam length and/or distortion. Here we improve seam length starting from an input map obtained from Seamster (a). We preserve distortion quality while improving seam length by taking the initial map's distortion as an input bound for OptCuts. Then we optimize the input map improving seam length while preserving distortion (b). As constrained distortion optimization is highly nonconvex, with many local minima, polishing warm-started solutions like these can be an effective way to use OptCuts to explore many interesting locally optimal embeddings aside from the default one we gain from initializing with a Tutte embedding (c).

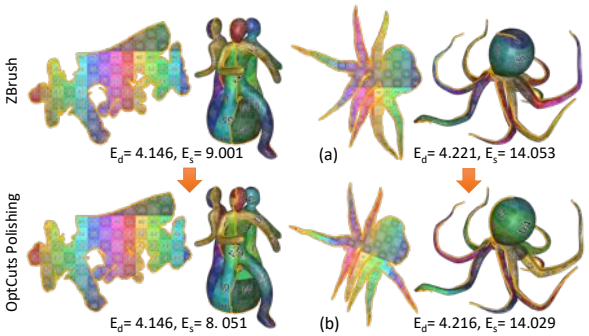


Fig. 14. UV-map Polishing with OptCuts starting from ZBrush results. Here OptCuts is initialized with UV maps produced by artists using the commercial software tool ZBrush. OptCuts shortens seams for both models, while respecting the original distortion bound and maintaining bijectivity constraints.

As a second example, we take the output of the best-performing commercial tool from our experiments, ZBrush. Starting with the two models in Table 3 where OptCuts generated slightly longer seams when starting from Tutte's embedding (three_man and octopus.) we use the ZBrush outputs as an input embeddings for OptCuts. OptCuts then produces new outputs that maintain the initial distortion bound while in both cases shortening the seams even further; see Figure 14. In all cases OptCuts-polished examples achieve the shortest seam length while maintaining both the distortion bound and bijectivity—highlighting both the utility of polishing and the value of exploiting warm-starts with OptCuts.

8 CONCLUSIONS

In this work we have proposed OptCuts, a new discrete-continuous optimization algorithm for mesh parameterization that jointly minimizes seam length while satisfying a target distortion bound. Here

our focus has been to automatically generate these high-quality parameterizations and likewise to add versatility by further supporting globally bijective maps and enabling user constraints on seam placement when desired. Along with generating high-quality maps from scratch, across our testing with state-of-the art methods and commercial tools we find that OptCuts typically yields shorter seams while achieving all targeted distortions.

8.1 Limitations and Future Work

Looking ahead we note that there are a number of promising directions to improve and explore. Numerically, in each inner iteration OptCuts is explicitly designed to decrease the Lagrangian. As a result we observe that OptCuts so far efficiently converges in all examples to either a fixed or cyclical stationary point satisfying a rough numerical optimality without any need to apply standard heuristic upper bounds on iteration counts. While OptCuts works well in practice, we have by no means proven its optimality nor convergence. Further analysis and understanding of both cyclical points and the overall convergence behavior are important future work here. Likewise, we also note that there can be locally minimizing seam-length solutions that are strictly on the interior of the distortion bound, i.e. with $E_d < b_d$. Here perturbation of vertex positions could find additional, nearby optimal points that offer further reduction of distortion. Such improved distortion points are seen as equivalent by problem statement (1) and so are not required for optimality. However, OptCuts always ends with a final smooth optimization step and so will obtain these locally improved points in practice. In terms of efficiency, OptCuts could support a range of additional parallelism, for example, many topological operations could be executed simultaneously in partitioned regions of the mesh, while linear system solves seem reasonable to decompose. Finally, in many cases we envision incorporating important additional priors to, for example, favor seamless parameterization, seam smoothness, and the creation of charts that efficiently use texture space.

ACKNOWLEDGEMENT

We thank Roi Poranne for guidance on automating AutoCuts, Justin Patton, Daichi Ito, and Jeanette Mathews for creating our commercial software examples, and Zhongshi Jiang for valuable discussions. This work has been supported in part by the NSERC, the NSF (grant IIS-1838071), the MIT Research Support Committee, the Army Research Office (grant W911NF-12-R-0011), an Amazon Research Award, the MIT–IBM Watson AI Laboratory, and the Skoltech–MIT Next Generation Program.

REFERENCES

Shmuel Agmon. 1954. The relaxation method for linear inequalities. *Canadian Journal of Mathematics* 6, 3 (1954), 382–392.

Noam Aigerman and Yaron Lipman. 2015. Orbifold Tutte Embeddings. *ACM Trans. Graph.* 34, 6, Article 190 (Oct. 2015), 12 pages.

Noam Aigerman, Roi Poranne, and Yaron Lipman. 2015. Seamless Surface Mappings. *ACM Trans. Graph.* 34, 4 (2015), 72:1–72:13.

Larry Armijo. 1966. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics* 16, 1 (1966), 1–3.

Dimitri P. Bertsekas. 2016. *Nonlinear Programming*. Athena Scientific.

S Claici, M Bessmeltsev, S Schaefer, and J Solomon. 2017. Isometry-Aware Preconditioning for Mesh Parameterization. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 37–47.

Michael S. Floater. 2003. Mean Value Coordinates. *Comput. Aided Geom. Des.* 20, 1 (March 2003), 19–27.

Xianfeng Gu, Steven J Gortler, and Hugues Hoppe. 2002. Geometry images. *ACM Transactions on Graphics (TOG)* 21, 3 (2002).

Kai Hormann and Günther Greiner. 2000. *MIPS: An efficient global parametrization method.* Technical Report.

Kai Hormann, Bruno Lévy, and Alla Sheffer. 2007. Mesh Parameterization: Theory and Practice. In *SIGGRAPH 2007 Course Notes*. ACM Press, San Diego, CA.

Alec Jacobson, Daniele Panozzo, and others. 2017. libigl: A simple C++ geometry processing library. (2017). http://libigl.github.io/libigl/.

Zhongshi Jiang, Scott Schaefer, and Daniele Panozzo. 2017. Simplicial complex augmentation framework for bijective maps. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 186.

Dan Julius, Vladislav Kraevoy, and Alla Sheffer. 2005. D-Charts: Quasi-Developable Mesh Segmentation. In *Computer Graphics Forum*, Vol. 24.

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006. Discrete Conformal Mappings via Circle Patterns. *ACM Trans. Graph.* 25, 2 (2006), 412–438.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Trans. Graph.* 21, 3 (July 2002), 362–371.

Songrun Liu, Zachary Ferguson, Alec Jacobson, and Yotam Gingold. 2017. Seamless: Seam erasure and seam-aware decoupling of shape from mesh resolution. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 216:1–216:15.

Ashish Myles and Denis Zorin. 2013. Controlled-distortion Constrained Global Parametrization. *ACM Trans. Graph.* 32, 4 (2013), 105:1–105:14.

Cosmin G. Petra, Olaf Schenk, and Mihai Anitescu. 2014a. Real-time stochastic optimization of complex energy systems on high-performance computers. *IEEE Computing in Science & Engineering* 16, 5 (2014), 32–42.

Cosmin G. Petra, Olaf Schenk, Miles Lubin, and Klaus Gärtner. 2014b. An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization. *SIAM Journal on Scientific Computing* 36, 2 (2014), C139–C162.

Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)* 36, 6 (2017).

M.J.D. Powell. 1973. On Search Directions for Minimization Algorithms. *Mathematical Programming* 4 (1973).

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Transactions on Graphics* 36, 2 (April 2017), 16:1–16:16.

Nicolas Ray, Wan Chiu Li, Bruno Lévy, Alla Sheffer, and Pierre Alliez. 2006. Periodic Global Parameterization. *ACM Trans. Graph.* 25, 4 (2006), 1460–1485.

Nicolas Ray, Vincent Nivoliers, Sylvain Lefebvre, and Bruno Lévy. 2010. Invisible seams. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 1489–1496.

James Reinders. 2007. *Intel threading building blocks: outfitting C++ for multi-core processor parallelism.* " O'Reilly Media, Inc.".

Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. 2001. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 409–416.

Rohan Sawhney and Keenan Crane. 2017. Boundary First Flattening. *ACM Trans. Graph.* 37, 1, Article 5 (Dec. 2017).

Alla Sheffer and John C Hart. 2002. Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of the conference on Visualization'02*.

Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. 2005. ABF++: fast and robust angle based flattening. *ACM Transactions on Graphics (TOG)* 24, 2 (2005), 311–330.

Alla Sheffer, Emil Praun, and Kenneth Rose. 2007. Mesh Parameterization Methods and Their Applications. *Foundations and TrendsÂő in Computer Graphics and Vision* 2, 2 (2007), 105–171.

Jonathan Richard Shewchuk. 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied computational geometry towards geometric engineering*. Springer, 203–222.

Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Kovalsky, and Yaron Lipman. 2017. Geometric Optimization via Composite Majorization. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH issue)* 36, 4 (2017), 38:1–38:11.

Jason Smith and Scott Schaefer. 2015. Bijective parameterization with free boundaries. *ACM Transactions on Graphics (TOG)* 34, 4 (2015).

John Snyder, Pedro V Sander, Zoe J Wood, Steven Gortler, and Hugues Hoppe. 2003. Multi-chart geometry images. (2003).

Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. 2002. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization*. IEEE Computer Society, 355–362.

Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005. Robust quasistatic finite elements and flesh simulation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*.

William Thomas Tutte. 1963. How to draw a graph. *Proceedings of the London Mathematical Society* 3, 1 (1963), 743–767.

Bruno Vallet and Bruno Lévy. 2009. *What you seam is what you get.* Technical Report.