DOI: 10.1111/cgf.13923
EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)

Volume 39 (2020), Number 2

Greedy Cut Construction for Parameterizations

Tianyu Zhu[1] Chunyang Ye[1] Shuangming Chai[2] Xiao-Ming Fu[†1]

[1]University of Science and Technology of China [2]Shining 3D Tech Co., Ltd.

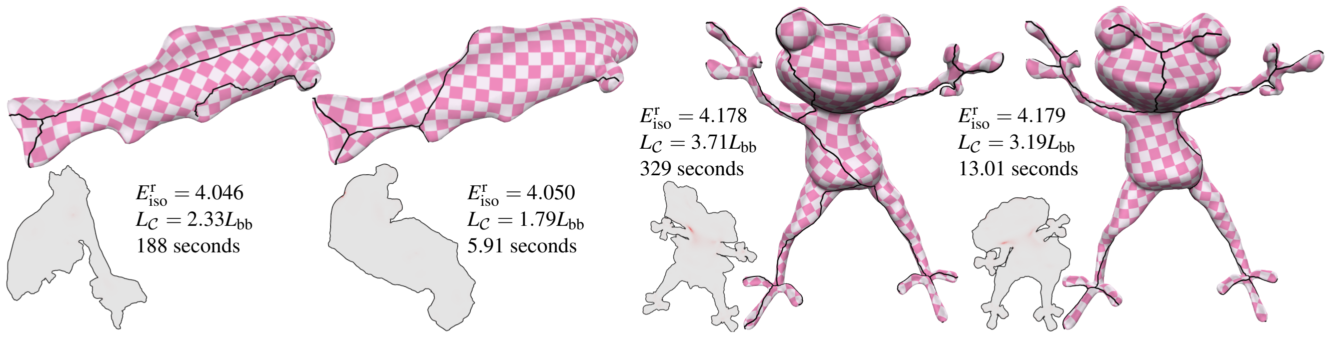


Figure 1: *Cut construction for two examples. Compared to OptCut [LKK*18], our method generates shorter cuts and similar isometric distortions in much less time. We generate bijective parameterizations for both methods and annotate the resulting symmetric Dirichlet energy E^{r}_{iso}, the cut length $L_{\mathcal{C}}$, and the running time next to each figure. Here, L_{bb} refers to the diagonal length of the bounding box of an input mesh.*

Abstract

We present a novel method to construct short cuts for parameterizations with low isometric distortion. The algorithm contains two steps: (i) detect feature points, where the distortion is usually concentrated; and (ii) construct a cut by connecting the detected feature points. Central to each step is a greedy method. After generating a redundant feature point set, a greedy filtering process is performed to identify the feature points required for low isometric distortion parameterizations. This filtering process discards the feature points that are useless for distortion reduction while still enabling us to obtain low isometric distortion. Next, we formulate the process of connecting the detected feature points as a Steiner tree problem. To find an approximate solution, we first successively and greedily produce a collection of auxiliary points. Then, a cut is constructed by connecting the feature points and auxiliary points. In the 26,299 test cases in which an exact solution to the Steiner tree problem is available, the length of the cut obtained by our method is on average 0.17% longer than optimal. Compared to state-of-the-art cut construction methods, our method is one order of magnitude faster and generates shorter cuts while achieving similar isometric distortion.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Shape modeling;***

1. Introduction

The task of parameterizing 3D meshes to a plane is fundamental in computer graphics. Parameterized 2D meshes are commonly used to store surface signals, such as colors, normals, and displacements. Two major factors affecting the feasibility and practicality of parameterizations are distortion and cut length. Short cuts and low isometric distortion are both required for high-quality parameterizations.

Solving this problem is very challenging, and the reason is twofold. First, since a cut is discretely represented as mesh edges, it is highly complex to reduce the length using combinatorial optimization techniques. Second, cut construction and parameterization generation are coupled. Parameterizations are usually computed after cuts are determined, and the distortion heavily depends on the cut location. To handle these challenges, simultaneous optimization of both cuts and the corresponding distortions have recently been developed [LKK*18,PTH*17]. However, since these two goals are highly coupled, the optimization problem is nonlinear and non-convex, rendering it exceedingly costly. Consequently, it is easy to be trapped by local minima, thereby resulting in long cuts (Figure 1).

† The corresponding author

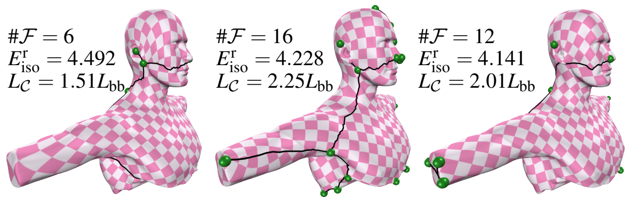

$\#\mathcal{F} = 6$
$E^{\mathrm{r}}_{\mathrm{iso}} = 4.492$
$L_{\mathcal{C}} = 1.51L_{\mathrm{bb}}$
$\#\mathcal{F} = 16$
$E^{\mathrm{r}}_{\mathrm{iso}} = 4.228$
$L_{\mathcal{C}} = 2.25L_{\mathrm{bb}}$
$\#\mathcal{F} = 12$
$E^{\mathrm{r}}_{\mathrm{iso}} = 4.141$
$L_{\mathcal{C}} = 2.01L_{\mathrm{bb}}$
(a) [GGH02]
(b) [CFH*18]
(c) Ours

Figure 2: *Different methods for detecting feature points (colored in green). Our feature point connection method is used to connect the feature points. $\#\mathcal{F}$ indicates the number of feature points.*

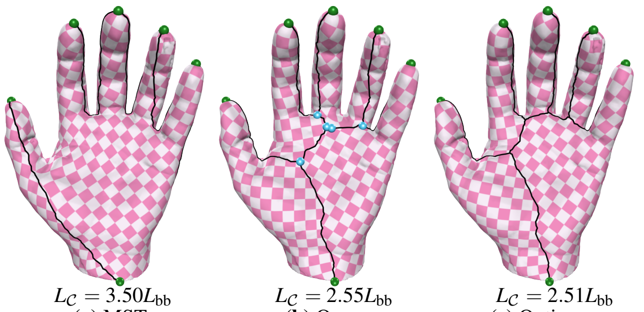

$L_{\mathcal{C}} = 3.50L_{\mathrm{bb}}$
(a) MST
$L_{\mathcal{C}} = 2.55L_{\mathrm{bb}}$
(b) Ours
$L_{\mathcal{C}} = 2.51L_{\mathrm{bb}}$
(c) Optimum

Figure 3: *Different methods for connecting the green points. (a) The MST-based method [KMB81]. (b) Our method. (c) The exact solution to the Steiner tree problem. The cyan points in (b) are the auxiliary points generated by our method. The symmetric Dirichlet energies are similar (4.098 in (a), 4.095 in (b), and 4.074 in (c)).*

A possible strategy to overcome these difficulties is to first detect feature points where the distortion is usually concentrated and then connect these feature points to construct cuts [SH02, CFH*18]. We call this a *point-to-cut* strategy. Since parameterizations are not determined during the feature point detection process, proxy metrics, such as the Gaussian curvature [She02, SH02] and distortion from spherical parameterizations [CFH*18], are used as predictors of anticipated parameterization distortion. However, the relationship between proxy metrics and parameterization distortion is not clear and direct, and the configuration (number and locations) of generated feature points is not always appropriate. For example, large distortions can occur if feature points are missing (Figure 2 (a)), whereas too many points produce long cuts (Figure 2 (b)). To make the constructed cuts as short as possible, connecting detected feature points is usually formulated as a Steiner tree problem; however, it is an NP-hard problem. One approximate solution used in [SH02, She02, CFH*18] is based on the minimum spanning tree (MST) [KMB81]. However, sometimes it is far from an exact solution, so much longer cuts may be generated (Figure 3). It is nontrivial to develop an effective and efficient method for computing approximate solutions.

We propose a novel method to compute short cuts for low isometric distortion parameterizations. We use the point-to-cut strategy with two key additions: (1) a filtering-based process for detecting feature points and (2) an approximate solution to the Steiner tree problem driven by auxiliary points. The process of feature point detection first computes a set of candidate feature points, which are the local maximizers for a distortion metric function. These feature points are usually redundant, so we then greedily filter out some points so that the isometric parameterization distortion is similar before and after filtering. Thanks to this filtering process, we simultaneously prevent the occurrence of too many feature points and achieve low isometric distortion. To approximately solve the Steiner tree problem, we first generate auxiliary points one by one and then construct a cut by connecting the detected auxiliary points and feature points using the algorithm in [KMB81]. Our intensive experiments and comparisons indicate that our approximate solution approaches the exact solution to the Steiner tree problem. Our approximate solution is parallelizable, and the implementation is simple and efficient. Since feature point filtering and auxiliary point generation are greedy processes, we call our algorithm *GreedyCuts*.

Our method is able to generate short cuts and low isometric distortion parameterizations. We demonstrate the feasibility and efficacy of our method on a data set with 3,757 complex models. Compared to state-of-the-art methods for cut construction, our method achieves shorter cuts while maintaining similar parameterization distortion and is one order of magnitude faster.

2. Related Work

Parameterizations. Given a 3D mesh that has been cut into disk-topology charts, many parameterization methods have been presented [SPR06, FH05, HLS07]. The flip-free property is a basic requirement in many computer graphics tasks. Tutte's embedding method [Tut63, Flo03] guarantees bijective parameterizations; however, the parameterizations often contain high distortions. Progressive embedding [SJZP19] has similar theoretical guarantees to Tutte's embedding and more reliable with regard to the rounding error of floating point arithmetic. Maintenance-based methods initialize flip-free parameterizations using the Tutte's embedding method and then try to reduce parameterization distortion while ensuring no flips [FLG15, RPPSH17, SPSH*17, GSC18, HG00, LYNF18]. There are also some recent methods that guarantee a final bijective parameterization by maintaining bijection during the optimization [SS15, JSP17]. Since computing flip-free planar parameterizations is theoretically guaranteed, we use proxy metrics from planar parameterizations for our feature point detection.

Mesh cutting. Simultaneous optimization of the parameterization distortion and the cut length is introduced in [LKK*18, PTH*17]; however, since the nonlinear and non-convex optimization problem is very complicated, these methods are time-consuming and usually generate long cuts. Some methods adopt the point-to-cut strategy containing two steps: (1) detect the feature points and then (2) connect the feature points [CFH*18, She02, SH02]. We follow this strategy. Segmenting an input mesh into multiple charts is another way to construct cuts [JKS05, LPRM02, SGSH02, ZSGS04], but these methods do not explicitly minimize cut lengths. Gu *et al*. [GGH02] alternately parameterize the mesh and connect the maximum distortion vertex to the existing cut via the shortest path. As observed by [CFH*18], this algorithm often terminates early, resulting in large isometric distortion. Triangles are parameterized one by one in [SCOGL02] without violating the user-provided distortion bound. In general, the one-by-one way is too local to produce a shorter cut than the cut required to achieve a given bound, as observed by [PTH*17, HLS07]. An analytic and interactive segmentation framework is developed to construct cuts for minimizing parameterization distortion [LDB17]. A global variational approach is presented to generate cuts [SC18]. Several extra cuts are introduced for atlas refinement [LVS18, LFY*19]. Liu *et al.* [LZF*19] construct cuts for peeling art to meet various design constraints.

Feature point detection. Numerous methods for detecting feature points have been proposed. Some of them are used for semantic tasks, e.g., segmentation and shape correspondences, and there is no clear, direct relationship between these feature points and low distortion parameterizations. For example, the Heat Kernel Signature [SOG09] is a function of time, and it is challenging to determine an appropriate time to detect local maxima for all models so that low distortion is always achieved. Here, we review the most relevant prior works. For the goal of low distortion, there are various proxy metrics used to detect feature points, such as Gaussian curvatures [She02, SH02] and distortion metrics [GGH02, CFH*18]. High curvature vertices have a high probability of producing high isometric distortion. However as observed by [SSC18], the relationship between curvatures and distortions is not direct or clear. A hierarchical clustering method uses distortion metrics from spherical parameterizations [CFH*18]. Since the spherical parameterization method [HFL18] used in [CFH*18] may fail to generate bijective spherical parameterizations, distortion metrics from planar parameterizations are used [CFL19]. Similar to [CFL19], we also use planar parameterizations to generate proxy metrics. However, different from [CFL19], we develop a greedy filtering process to detect necessary feature points to achieve low isometric distortion and prevent too many feature points. Conformal cone singularities [SSC18, MZ12, BCGB08, SSP08] and singularities of regular fields [KCPS13, BZK09, VCD*16] can also be treated as feature points. As observed in Figure 15 in [SSC18], some important feature points are not captured with default parameters, leading to high distortion. As shown in Figure 17 in [SSC18], a low isometric distortion parameterization is not achieved when using the singularities from a regular field [KCPS13] as feature points.

Steiner tree problem. Given a graph and a set of terminal vertices in the graph, the Steiner tree problem seeks to find the minimum cost tree connecting all the terminal vertices. This is an NP-hard problem [HRW92]. Some algorithms for computing an exact solution to the Steiner tree problem have been proposed [FGK08, Bea89, Hak71]. However, they cannot generate the exact solution in a reasonable amount of time for large-scale graphs or when there are many terminal vertices. On this account, some approximation methods have been proposed [BR94, BGRS13, RZ05, PUW18]. Two commonly used approaches are based on the minimal spanning tree [KMB81] and the shortest paths heuristic [TA80]. The algorithm in [KMB81] is used by [She02, SH02, CFH*18]. We propose a greedy algorithm to compute an approximate solution driven by auxiliary points. In practice, our results approximate the optimal solution better in the sense of relative error.

3. Method

Input. We study cut construction for parameterizations. The input is a closed triangular mesh $\mathcal{M}$ that consists of a set of vertices $\mathcal{V} = \{\mathbf{v}_i\}$, edges $\mathcal{E} = \{\mathbf{e}_i\}$, and triangles $\mathcal{T} = \{\mathbf{t}_i\}$. The mesh structure naturally forms a weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The weight of each edge is its length.

Goal. Our goal is to construct a cut $\mathcal{C}$ consisting of mesh edges that satisfies the following three requirements:

1. After being cut along $\mathcal{C}$, the mesh should only contain disk-topology charts.
2. The isometric parameterization distortion in each chart should be as low as possible.
3. The length of the cut $\mathcal{C}$, i.e. the total length of the mesh edges it contains, should be as short as possible.

Overview. Our method follows the point-to-cut strategy but proposes two key techniques:

- A set of feature points $\mathcal{F}$ is detected using two steps: (i) generate redundant candidates guided by proxy metrics from a planar parameterization (Section 3.1.1), and (ii) filter out some candidates while keeping the parameterization distortion low (Section 3.1.2).
- A cut $\mathcal{C}$ is constructed by connecting $\mathcal{F}$ via a two-step procedure: (1) generate auxiliary points $\mathcal{A}$ using a greedy technique, and (2) connect $\mathcal{A}$ and $\mathcal{F}$ using [KMB81] (Section 3.2).

Figure 4 illustrates the workflow of our method. In Section 3.1 and 3.2, we assume that $\mathcal{M}$ is a closed, orientable surface with genus zero, while the high-genus surfaces are discussed in Section 3.3.

3.1. Detecting feature points

Distortion metrics. There are two kinds of distortion metrics being used as optimization energies or proxy metrics. After being cut into a disk topology, the input mesh is able to be parameterized to the plane. We denote the parameterized triangle of $\mathbf{t}_i$ as $\mathbf{t}_i^p$, and the mapping from $\mathbf{t}_i$ to $\mathbf{t}_i^p$ is an affine transformation whose Jacobian matrix is denoted as J_i. The MIPS energy [HG00] on $\mathbf{t}_i$ is used to measure the conformal distortion:

$$\delta_i^{\text{con}} = \frac{1}{2}\frac{\|J_i\|_F^2}{\det J_i}, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm. The symmetric Dirichlet energy [SS15] is computed as the isometric distortion of $\mathbf{t}_i$:

$$\delta_i^{\text{iso}} = \|J_i\|_F^2 + \|J_i^{-1}\|_F^2. \tag{2}$$

3.1.1. Generating redundant feature points

Overview. We generate the initial feature points by cutting the mesh $\mathcal{M}$ and collecting all local maximizers for an isometric distortion function defined by an as-conformal-as-possible (ACAP) planar parameterization. In this section, we first describe isometric distortions as proxy metrics and the generation process for candidate feature points. Then we introduce a dual cutting strategy to avoid excluding feature points.

Proxy metrics. As observed by [CFH*18], the distortion concentration in an ACAP parameterization is more significant compared to an as-isometric-as-possible parameterization (See Figure 3 in [CFH*18]). Therefore, we use the isometric distortion metrics of an ACAP planar parameterization as proxy metrics. The isometric

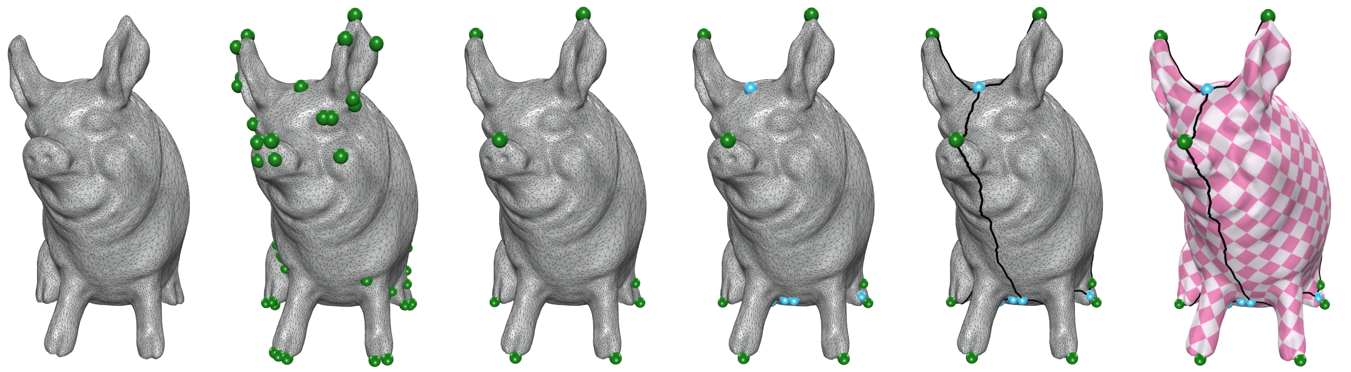


Figure 4: *Workflow of our method. Given a closed mesh $\mathcal{M}$ (a), we first detect feature points $\mathcal{F}$ (b, c) and then connect them to construct the cut $\mathcal{C}$ (d, e). The feature points and the auxiliary points are colored in green and cyan, respectively. The cut is rendered as a black line.*

distortion is considered to be a single-valued function δ^{iso} defined on $\mathcal{M}$, where each triangle $\mathbf{t}_i$ has a constant distortion value δ_i^{iso}. We also define the distortion value for each vertex $\mathbf{v}_i$ as follows:

$$\delta_{\mathbf{v}_i}^{\text{iso}} = \frac{1}{\text{Area}(\Omega(\mathbf{v}_i))} \sum_{\mathbf{t}_j \in \Omega(\mathbf{v}_i)} \text{Area}(\mathbf{t}_j)\delta_j^{\text{iso}}, \tag{3}$$

where $\Omega(\mathbf{v}_i)$ is a triangle set from the one-ring neighborhood of $\mathbf{v}_i$. We compute an ACAP planar parameterization by optimizing $\sum_i \text{Area}(\mathbf{t}_i)\delta_i^{\text{con}}$ using the method in [GSC18].

Candidate feature points generation. We observe that feature points are usually included amongst the local maximizers of the proxy metric δ^{iso}. We denote $\Theta_n(\mathbf{v}_i)$ as the n-ring vertices of $\mathbf{v}_i$. Since the distortion metric $\delta_{\mathbf{v}}^{\text{iso}}$ of an ACAP planar parameterization is a discrete function defined in the vertices, the definition of a local maximizer is: a vertex $\mathbf{v}_i$ is a local maximizer, if $\delta_{\mathbf{v}_i}^{\text{iso}} \geq \delta_{\mathbf{v}_j}^{\text{iso}}, \forall \mathbf{v}_j \in \Theta_1(\mathbf{v}_i)$. Then, the feature point set $\mathcal{F}$ is initialized in the same manner as all of the local maximizers. Note that this initial set $\mathcal{F}$ is usually redundant and requires further filtering. For each candidate feature point $\mathbf{v}_i \in \mathcal{F}$, we also define the scope of influence s_i as the geodesic distance to the closest vertex with a distortion level higher than $\delta_{\mathbf{v}_i}^{\text{iso}}$, i.e. $s_i = \min_{\delta_{\mathbf{v}_j}^{\text{iso}} > \delta_{\mathbf{v}_i}^{\text{iso}}} d(\mathbf{v}_i, \mathbf{v}_j)$, where $d(\cdot,\cdot)$ denotes the geodesic distance between two vertices. In particular, the vertex with the greatest level of distortion has an infinite scope of influence. In our implementation, we use the Dijkstra's distance to approximate the geodesic distance due to its computational efficiency. This distance is computed only once and reused in other parts of our method.

A dual cutting strategy. If a cut, denoted as $\mathcal{C}^p$, passes through points that should be detected as feature points, their isometric distortions are usually not local maxima, thereby resulting in the omission of certain feature points. Therefore, cuts should stay far away from feature points to avoid leaving any out. Since we have no prior knowledge regarding the location of feature points, we devise a dual cutting strategy to generate candidate feature points accurately. Briefly, we generate a first cut $\mathcal{C}_0^p$ and parameterize the mesh to establish a forbidden region $\mathcal{R}$. In order to refine the feature point detection process, we then construct a second cut $\mathcal{C}^p$ that is prohibited from passing through this forbidden region. The detailed algorithm for the construction of $\mathcal{R}$ and $\mathcal{C}^p$ is as follows:

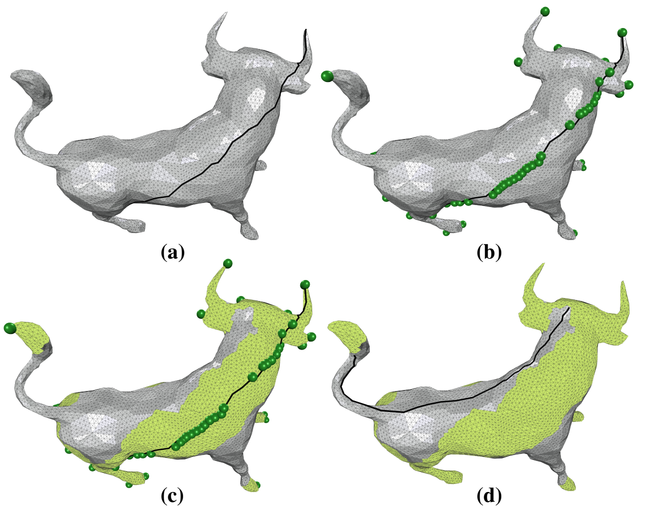


Figure 5: *Constructing $\mathcal{C}^p$ to generate redundant feature points. (a) The first cut $\mathcal{C}_0^p$. (b) We generate $\mathcal{F}_0$ (vertices in green) using $\mathcal{C}_0^p$. (c) The forbidden region $\mathcal{R}$ is colored in lime. (d) The resulting $\mathcal{C}^p$ is generated in $\mathcal{M} \setminus \mathcal{R}$. In this example, $\overline{\mathcal{R}} = \mathcal{M} \setminus \mathcal{R}$.*

1. Construct the first cut $\mathcal{C}_0^p$ via three steps: (i) randomly select one vertex $\mathbf{v}_i$, (ii) for each vertex $\mathbf{v}_j \in \mathcal{V}, j \neq i$, compute the shortest distance l_{ij} from $\mathbf{v}_i$ to $\mathbf{v}_j$ using the Dijkstra's algorithm, and (iii) find the farthest vertex $\mathbf{v}_k$ with the largest l_{ik} and the shortest path between $\mathbf{v}_i$ and $\mathbf{v}_k$ to form the first cut $\mathcal{C}_0^p$.
2. Cut the mesh $\mathcal{M}$ along $\mathcal{C}_0^p$ into a disk topology and compute an ACAP parameterization. Then, compute δ^{iso} for each vertex and collect the local maximizers into a set $\mathcal{F}_0$.
3. Initialize a positive number d_n.
4. Define the forbidden region $\mathcal{R}$ as the union of triangles whose three vertices have geodesic distances to $\mathcal{F}_0$ or $\mathcal{C}_0^p$ less than d_n.
5. Find the largest connected component $\overline{\mathcal{R}}$ of $\mathcal{M} \setminus \mathcal{R}$.
6. If $\text{Area}(\overline{\mathcal{R}}) < \alpha\text{Area}(\mathcal{M})$, where α is a positive threshold, set $d_n \leftarrow 0.9 d_n$ and revert to Step 4; otherwise, we generate $\mathcal{C}^p$ in $\overline{\mathcal{R}}$ by using the approach in Step 1.

Note that we detect the second cut in the largest connected component $\overline{\mathcal{R}}$. If the area of $\overline{\mathcal{R}}$ is small, then the cut $\mathcal{C}^p$ is short, thereby affecting the detection accuracy of feature points in next steps. Thus, our iterative shrinking of d_n ensures that the area of $\overline{\mathcal{R}}$ is not too small. In our experiments, the initial d_n is set as $10l_{\text{avg}}$, where l_{avg} is the average edge length, and the threshold α is set as 0.1. Figure 5 shows an example to illustrate the process for constructing $\mathcal{C}^p$. After the cut $\mathcal{C}^p$ is computed, we repeat Step 2 in which the first cut $\mathcal{C}_0^p$ is replaced by the second cut $\mathcal{C}^p$. Then, we collect all the local maximizers as a candidate feature point set $\mathcal{F}$ for the next filtering process. Note that the dual cutting strategy is only performed once in order to preserve all feature points in our experiments. It is possible to perform the strategy multiple times, but the important feature points remain unchanged.

3.1.2. Filtering redundant feature points

Overview. Since connecting all the candidate feature points leads to a long cut, the purpose of the filtering process is to eliminate redundant feature points while shortening the cut and slightly increasing distortion. Therefore, we determine whether each feature point should be filtered out based on its contribution to the parameterization distortion. Specifically, if the distortion value significantly decreases after a point is connected to the cut, then the contribution of this point is considered to be large. Based on this criterion, we propose an algorithm that calculates the contribution of candidate feature points and filters out the points that contribute less.

Initialization. Since a cut must connect at least two feature points, we initialize the set of feature points with more than two points. Recall that we have already computed the scope of influence for each feature point. In general, we observe that candidate points with a large scope of influence are more likely to be feature points. Based on this key observation, we initialize the feature point set by dividing the candidate set $\mathcal{F}$ into two subsets, $\mathcal{F}_s$ and $\mathcal{F}_l$, representing the points whose scope of influence s_i is smaller and larger than a specific threshold β, respectively. Next, we initialize a cut by connecting the points in $\mathcal{F}_l$ using an MST-based method [KMB81]. Without a loss of generality, we reuse the notation $\mathcal{C}^p$ to denote the cut during the filtering process. After being cut along $\mathcal{C}^p$, the mesh is parameterized to the plane using the progressive parameterization method [LYNF18], which minimizes the isometric distortion energy $E_{\text{iso}} = \sum_i \text{Area}(\mathbf{t}_i)\delta_i^{\text{iso}}$. We denote the optimal energy as E_{iso}^p.

Modified cut initialization. There is a drawback to the above simple initialization. If a candidate point in $\mathcal{F}_s$ is located on or near $\mathcal{C}^p$, our filtering process does not correctly evaluate this point's contribution. To counteract this drawback, we introduce forbidden regions for the candidate points in $\mathcal{F}_s$. The forbidden region is initialized as the triangles within a geodesic disk with a radius of $5l_{\text{avg}}$. If the forbidden regions of two candidate points $\mathbf{v}_i$ and $\mathbf{v}_j$ overlap, we reset the forbidden region of $\mathbf{v}_i$ and $\mathbf{v}_j$ as the triangles within a geodesic disk with a radius of $d(\mathbf{v}_i, \mathbf{v}_j)/2$, where $d(\mathbf{v}_i, \mathbf{v}_j)$ is the geodesic distance between $\mathbf{v}_i$ and $\mathbf{v}_j$. Without a loss of generality, we also reuse the notation $\mathcal{R}$ to denote the union of these forbidden regions. The cut $\mathcal{C}^p$ is initialized as follows:

1. Divide the candidate feature points into two sets $\mathcal{F}_s$ and $\mathcal{F}_l$ so that the influence scope is smaller or larger than a threshold β.
2. Compute the forbidden region $\mathcal{R}$ for the feature points in $\mathcal{F}_s$.
3. Compute all the connected components of $\mathcal{M} \setminus \mathcal{R}$.

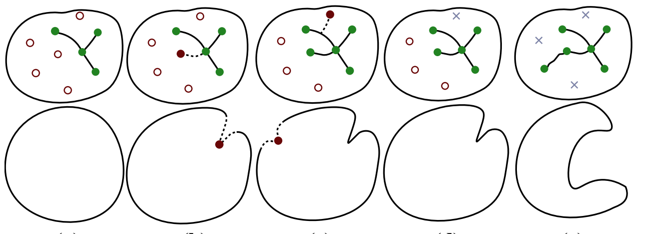


Figure 6: *The schematic diagram of the greedy filtering process. We render the candidate feature points, the selected feature points, and the discarded feature points in red circles, green dots, and gray crosses, respectively. The first row shows cuts on the input mesh and the second row shows the parameterized meshes. (a) The mesh is cut and parameterized using the initial cut. (b) A candidate point is considered as a feature point if the distortion decrease significantly when it is connected to the cut. (c) If the distortion decrease slightly when a candidate point is connected to the cut, then (d) we discard this point and (e) search for the next.*

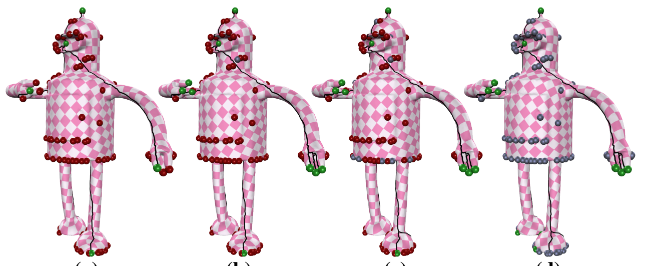


Figure 7: *Progressive filtering results. The candidate feature points ($\mathcal{Q}$), the selected feature points ($\mathcal{F}_l$), and the discarded feature points are in red, green, and gray, respectively. $\mathcal{C}^p$ is shown as a black line. (a) Initialization ($\#\mathcal{F}_l = 6$, $E_{iso}^p = 4.302$), where $\#\mathcal{F}_l$ indicates the number of points in $\mathcal{F}_l$. (b) After the fifth iteration ($\#\mathcal{F}_l = 10$, $E_{iso}^p = 4.183$). (c) After the nineteenth iteration ($\#\mathcal{F}_l = 11$, $E_{iso}^p = 4.144$). (d) Final result ($\#\mathcal{F}_l = 15$, $E_{iso}^p = 4.091$) after 115 iterations.*

4. Count the number of candidate points in $\mathcal{F}_l$ for each connected component.
5. Find the connected component $\overline{\mathcal{R}}$ with the largest number of candidates $N_{l,\overline{\mathcal{R}}}$.
6. If $N_{l,\overline{\mathcal{R}}} < 2$, we halve the threshold $\beta \leftarrow \beta/2$ and revert to Step 1 to update $\mathcal{F}_l$ and $\mathcal{F}_s$; otherwise, we connect the candidate feature points in $\mathcal{F}_l$ within $\overline{\mathcal{R}}$ to construct the initial cut $\mathcal{C}^p$ using the MST-based method [KMB81].

The initial β is set as $20l_{\text{avg}}$.

Greedy filtering. We filter the candidate points in $\mathcal{F}_s$ one by one according to their contributions to the distortion decrease. Here, we introduce a threshold ε_{iso}, and if the distortion decreases less than ε_{iso} after connecting a point to the cut, we consider this point to be inconsequential, and it is discarded.

The detailed greedy filtration process is illustrated in Figure 6 and is described as follows:

1. Sort the points in $\mathcal{F}_s$ according to their scope of influence s_i from large to small and construct a queue Q by pushing back these points in turn.

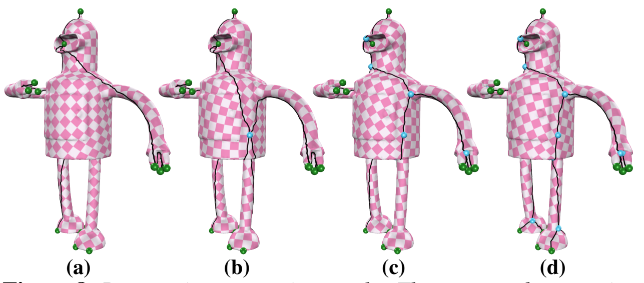


Figure 8: *Progressive connection results. The green and cyan points represent the detected feature vertices and auxiliary vertices, respectively. (a) Initial cut ($L_{\mathcal{C}} = 3.32L_{bb}$, $E^r_{iso} = 4.091$). (b) After the first iteration ($L_{\mathcal{C}} = 3.23L_{bb}$, $E^r_{iso} = 4.096$). (c) After the seventh iteration ($L_{\mathcal{C}} = 2.82L_{bb}$, $E^r_{iso} = 4.098$). (d) Final result ($L_{\mathcal{C}} = 2.80L_{bb}$, $E^r_{iso} = 4.098$) after 12 iterations.*

2. If $\mathcal{Q} \neq \emptyset$, pop a candidate point $\mathbf{v}_q$ from $\mathcal{Q}$; otherwise, set $\mathcal{F}$ $\mathcal{F}_l$ and stop the algorithm.
3. Compute the shortest path $\mathcal{C}^p_q$ from $\mathbf{v}_q$ to $\mathcal{C}^p$ within $\mathcal{M} \setminus \mathcal{R}$ using the Dijkstra's algorithm. If such a path does not exist due to the occlusion of forbidden regions, we push back $\mathbf{v}_q$ into $\mathcal{Q}$ and return to Step 2.
4. We cut $\mathcal{M}$ along $\mathcal{C}^p \bigcup \mathcal{C}^p_q$ and parameterize it to the plane using [LYNF18] to obtain an optimized distortion E^q_{iso}. If $E^p_{\text{iso}} - E^q_{\text{iso}} > \varepsilon_{\text{iso}}$, we move $\mathbf{v}_q$ from $\mathcal{F}_s$ to $\mathcal{F}_l$ and update $E^p_{\text{iso}} \leftarrow E^q_{\text{iso}}$ and $\mathcal{C}^p \leftarrow \mathcal{C}^p \bigcup \mathcal{C}^p_q$; otherwise, we keep $\mathbf{v}_q$ in $\mathcal{F}_s$.
5. Update the forbidden region by $\mathcal{R} \leftarrow \mathcal{R} \setminus \mathcal{R}_q$, where $\mathcal{R}_q$ is the forbidden region corresponding to $\mathbf{v}_q$, and then revert to Step 2.

Figure 7 shows the process of the greedy filter algorithm. In each iteration, the parameterization is initialized to the parameterization of the last iteration, so our greedy filtration process is efficient. Since there is at least one 'nearest' $\mathbf{v}_q \in \mathcal{Q}$ so that the shortest path $\mathcal{C}^p_q$ exists, Step 3 will never cause an infinite loop, and our filtering process always terminates.

3.2. Connecting feature points

Minimum spanning tree (MST) based method. The method in [KMB81] efficiently finds an approximate solution for the Steiner tree problem. Given a set of terminal points $\mathcal{P}$, it proceeds as follows:

1. For each pair of terminal points $\mathbf{v}_a \in \mathcal{P}$ and $\mathbf{v}_b \in \mathcal{P}$, compute the shortest path $\mathcal{C}_{a,b}$ between $\mathbf{v}_a$ and $\mathbf{v}_b$ on $\mathcal{G}$.
2. Construct a complete graph with the node set $\mathcal{P}$. The weight of each edge $\overline{\mathbf{v}_a\mathbf{v}_b}$ is equal to the length of $\mathcal{C}_{a,b}$.
3. Construct an MST for this shortest distance graph.
4. All mesh edges in the shortest paths that correspond to edges in the MST form an approximate Steiner tree for $\mathcal{G}$.

This method generates cuts in the same manner as MST, so we call it the MST-based method.

Greedy connection. Our greedy connection process computes the auxiliary points $\mathcal{A}$ and constructs the final cut $\mathcal{C}$ as follows:

1. We initialize $\mathcal{A} \leftarrow \emptyset$ and use the MST-based method to construct a tree connecting all feature points $\mathcal{F}$. The cost L of the initial tree cost is set as the total edge length of this tree.

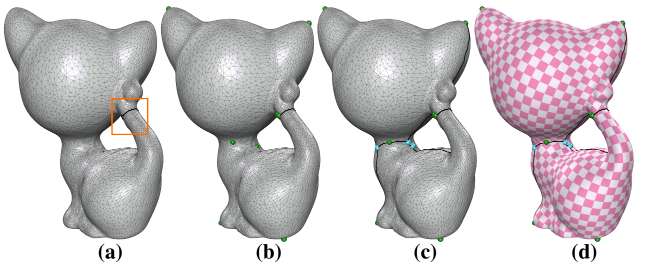


Figure 9: *Cut construction in high genus meshes. (a) Convert $\mathcal{M}$ to a genus-zero mesh $\mathcal{M}_{one}$. The used handle indicated by an orange box. (b) The detected feature points in $\mathcal{M}_{one}$. (c) The constructed cut $\mathcal{C}_{one}$ on $\mathcal{M}_{one}$. (d) The final $\mathcal{C}$ is generated by mapping $\mathcal{C}_{one}$ back onto $\mathcal{M}$.*

2. For each vertex $\mathbf{v} \in \mathcal{V} \setminus (\mathcal{A} \bigcup \mathcal{F})$, we treat $\mathcal{P} = \{\mathbf{v}\} \bigcup \mathcal{A} \bigcup \mathcal{F}$ as the terminal points and use the MST-based method to compute a tree, whose length is $L_{\mathbf{v}}$.
3. Compute the minimal length $L_{\min} = \min_{\mathbf{v} \in \mathcal{V} \setminus (\mathcal{A} \bigcup \mathcal{F})} L_{\mathbf{v}}$, and the corresponding vertex is $\mathbf{v}_{\min}$.
4. If $L - L_{\min} > \varepsilon_{\text{len}}$, where ε_{len} is a small positive number, we first add the vertex $\mathbf{v}_{\min}$ into $\mathcal{A}$. Then we update the tree cost $L \leftarrow L_{\min}$, and finally go back to Step 2.
5. We connect all the points in $\mathcal{A} \bigcup \mathcal{F}$ to construct the final cut $\mathcal{C}$ using the MST-based method.

Figure 8 shows the process of greedy connection. As the connection algorithm iterates, the cut length monotonously decreases and the isometric distortion is maintained at a low level. In Step 2, the shortest paths from a vertex in $\mathcal{A} \bigcup \mathcal{F}$ to each vertex in $\mathcal{V} \setminus (\mathcal{A} \bigcup \mathcal{F})$ are pre-computed, so the shortest paths required for the MST-based method do not need repeated calculation. In addition, Step 2 is parallelizable. Therefore, our greedy connection is very fast.

3.3. High-genus meshes

Similar to [CFH*18], we first convert an input high-genus mesh $\mathcal{M}$ to a genus-zero mesh $\mathcal{M}_{\text{one}}$ according to the following procedure: (1) compute handles of the mesh $\mathcal{M}$ using [DFW13], (2) cut $\mathcal{M}$ along the handles, and (3) fill the holes without any extra vertices. Each handle vertex of $\mathcal{M}$ has two copies in $\mathcal{M}_{\text{one}}$, and both of them must be connected to the final cut. We denote the set of all the vertices at the handles in $\mathcal{M}_{\text{one}}$ as $\mathcal{F}_h$. Then, we construct the cut $\mathcal{C}_{\text{one}}$ for $\mathcal{M}_{\text{one}}$ accordingly: (i) apply our feature detection algorithm to $\mathcal{M}_{\text{one}}$ to detect the feature points $\mathcal{F}_{\text{one}}$, (ii) set $\mathcal{F}_{\text{total}} = \mathcal{F}_{\text{one}} \bigcup \mathcal{F}_h$, and (iii) connect the vertices in $\mathcal{F}_{\text{total}}$ on $\mathcal{M}_{\text{one}}$ by using our greedy connection technique. Finally, we map $\mathcal{C}_{\text{one}}$ back onto $\mathcal{M}$ to obtain the resulting cut $\mathcal{C}$. Figure 9 shows an example.

3.4. Discussions

ACAP parameterizations. Since ACAP parameterizations of a mesh are not unique, different ACAP parameterizations result in different feature points and cuts. However, as observed in our experiments, the ACAP parameterization method we employ helps us to accurately detect the feature points required for reducing isometric distortion. We implement it as follows: starting from the Tutte's embedding that parameterizes an open mesh to a unit disk, the ACAP

Parameter	Description
d_n	Radius of forbidden region, initial: $10l_{\text{avg}}$
α	Threshold in the dual cut stategy, default: 0.1
β	Scope of influence threshold, initial: $20l_{\text{avg}}$
ε_{iso}	Distortion threshold, default: $0.01L_{\text{bb}}$
ε_{len}	Cut length threshold, default: $0.01L_{\text{bb}}$

Table 1: *List of parameters*

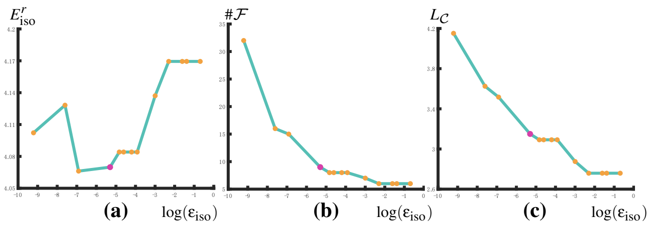


Figure 10: *Different ε_{iso}s. Thirteen values for ε_{iso} are tested on a Hand model, as shown in Figure 3. The deep pink points represent our selected ε_{iso}.*

parameterizations are generated by optimizing $\sum_i \text{Area}(\mathbf{t}_i)\delta_i^{\text{con}}$ with free boundaries using the KP-Newton method [GSC18].

4. Experiments

Our method generates short cuts for parameterizations, and we apply it to various models. We report the timings and the cut quality statistics, as shown in Table 3. Our experiments were performed on a desktop PC with a 4.0 GHz Intel Core i7-4790K CPU and a 16 GB RAM.

Implementation details. We implement the KP-Newton method independently, and the implementation for the progressive parameterization method is kindly provided by the authors. The Pardiso solver is used to solve the linear systems in these two methods. Note that no early adaptive termination criteria is used for them. We list the used parameters in Table 1.

Quality metrics. The cut length and parameterization distortion are used to measure the quality of a cut. For the cut length (denoted as $L_{\mathcal{C}}$), we measure the sum of the length of each edge in $\mathcal{C}$. To mitigate the impact of the model size, we report $L_{\mathcal{C}}$ w.r.t. L_{bb}, which is the diagonal length of the bounding box of $\mathcal{M}$. Here, we use flip-free parameterizations as the resulting parameterizations. After cutting the input mesh $\mathcal{M}$ with a cut $\mathcal{C}$, we optimize E_{iso} using the method in [LYNF18] to generate a resulting parameterization whose isometric distortion is denoted as E^r_{iso}.

4.1. Evaluations

The filtering threshold ε_{iso}. In the filtering process, we use a threshold ε_{iso} to determine the contribution of a candidate point. We test thirteen different thresholds using a Hand model, and show the relations to the distortion E^r_{iso}, the number of feature points $\#\mathcal{F}$, and the cut length $L_{\mathcal{C}}$ in Figure 10. These tests show that a larger ε_{iso} value filters out more candidate feature points and results in higher isometric distortions, while a smaller ε_{iso} value preserves

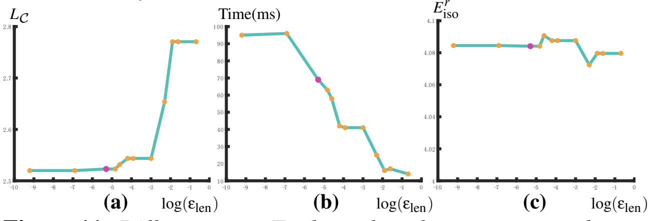


Figure 11: *Different ε_{len}s. Twelve values for ε_{len} are tested on a Hand model, as shown in Figure 3. Our selected ε_{len} is shown in deep pink.*

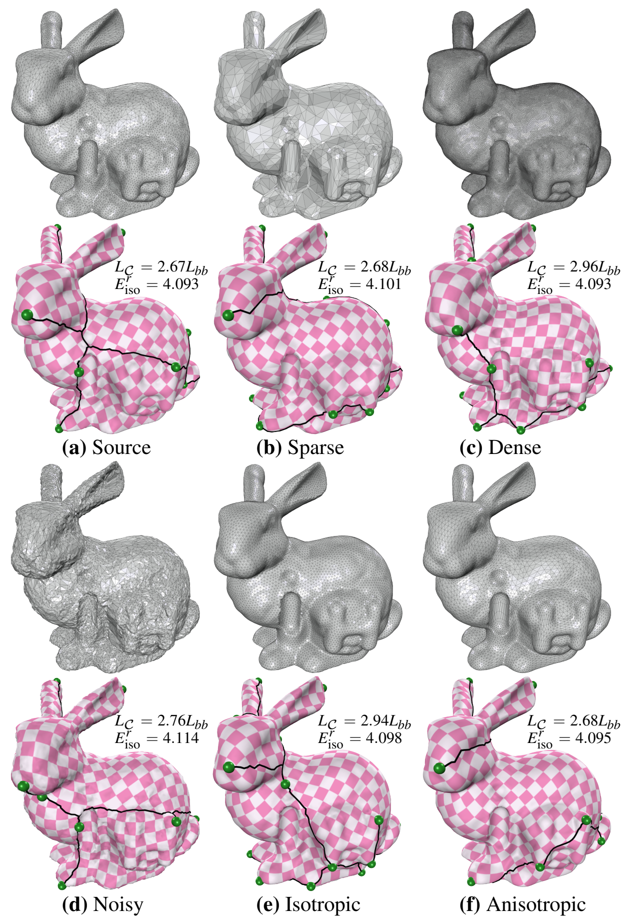

$L_{\mathcal{C}} = 2.67L_{bb}$
$E^r_{\text{iso}} = 4.093$
$L_{\mathcal{C}} = 2.68L_{bb}$
$E^r_{\text{iso}} = 4.101$
$L_{\mathcal{C}} = 2.96L_{bb}$
$E^r_{\text{iso}} = 4.093$
(a) Source
(b) Sparse
(c) Dense
$L_{\mathcal{C}} = 2.76L_{bb}$
$E^r_{\text{iso}} = 4.114$
$L_{\mathcal{C}} = 2.94L_{bb}$
$E^r_{\text{iso}} = 4.098$
$L_{\mathcal{C}} = 2.68L_{bb}$
$E^r_{\text{iso}} = 4.095$
(d) Noisy
(e) Isotropic
(f) Anisotropic

Figure 12: *Different triangulations. We include six types of triangulations: (a) source; (b) sparse; (c) dense; (d) noisy; (e) isotropic; and (f) anisotropic. The anisotropic mesh is generated by the LCT method [FLSG14].*

more candidate feature points, thereby resulting in longer cuts. This parameter can be used to control the tradeoff between cut length and parameterization distortion. We select $\varepsilon_{\text{iso}} = 0.01$ as a default for all other experiments to achieve a balance.

The threshold ε_{len} in the greedy connection. In the greedy connection algorithm, we use another threshold ε_{len} to determine if an auxiliary vertex makes a significant contribution to cut length shortening. In Figure 11, we test twelve different ε_{len} values for a Hand model (Figure 3). From the line charts, we observe that a smaller ε_{len} value induces more auxiliary points and results in

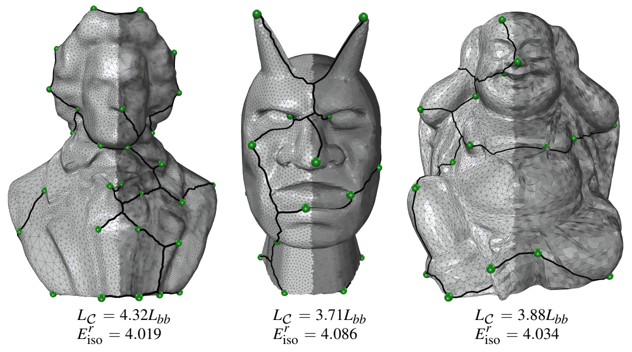

$L_{\mathcal{C}} = 4.32L_{bb}$
$E^r_{\text{iso}} = 4.019$
$L_{\mathcal{C}} = 3.71L_{bb}$
$E^r_{\text{iso}} = 4.086$
$L_{\mathcal{C}} = 3.88L_{bb}$
$E^r_{\text{iso}} = 4.034$

Figure 13: *Results of uneven densities in triangulations.*

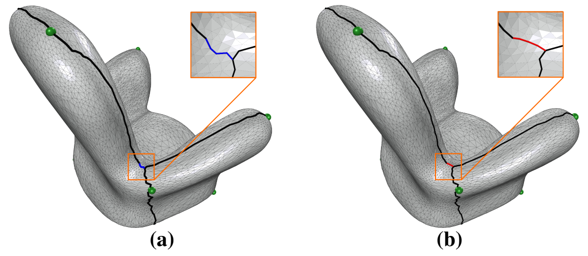


Figure 14: *Different cut construction methods. In our greedy feature point connections, the MST-based method (a) and the SPH method (b) are used, respectively. The zoom-out figures show the differences (rendered in different colors) between the two methods for cut construction.*

shorter cuts but higher computational costs, while a larger ε_{len} value locates fewer auxiliary points, leading to longer cuts. To achieve a tradeoff between efficiency and quality, we set $\varepsilon_{\text{len}} = 0.01L_{\text{bb}}$.

Different triangulations. In Figure 12, six types of triangulations representing a common shape are tested as input meshes. Although only some of the detected feature points are found in similar locations, such as the ears, mouth, and feet, $L_{\mathcal{C}}$ and E^r_{iso} are similar in all triangulations. In addition, we show some examples of triangulations with uneven densities in Figure 13. It is observed that these models have some symmetric features, and the detected feature points are almost symmetric. Although our results are slightly different for various triangulations for a single shape, the isometric distortions are all at a low level. This indicates that our method reliably generates high-quality results when using different triangulations as inputs.

Cut construction. A cut $\mathcal{C} = (\mathcal{V}_{\mathcal{C}}, \mathcal{E}_{\mathcal{C}})$ is a subgraph of the mesh with a vertex set $\mathcal{V}_{\mathcal{C}}$ and an edge set $\mathcal{E}_{\mathcal{C}}$. Another commonly used approximate solution to the Steiner tree problem is proposed in [TA80]. Given a set of feature points $\mathcal{P}$, it constructs $\mathcal{C}$ as follows:

1. Compute the shortest path $\mathcal{C}_{a,b}$ on $\mathcal{G}$ between all pairs of the feature points $\mathbf{v}_a$ and $\mathbf{v}_b$, and find the nearest pair $(\mathbf{v}_i, \mathbf{v}_j)$. Initialize the cut $\mathcal{C} \leftarrow \mathcal{C}_{i,j}$.
2. Find the nearest pair of vertices $\mathbf{v}_k \in \mathcal{V}_{\mathcal{C}}$ and $\mathbf{v}_l \in \mathcal{P} \setminus \mathcal{V}_{\mathcal{C}}$ in terms of graph distance, and the shortest path between them is $\mathcal{C}_{k,l}$.
3. Add the vertices and edges of $\mathcal{C}_{k,l}$ to $\mathcal{C}$.
4. If $\mathcal{P} \setminus \mathcal{V}_{\mathcal{C}} = \emptyset$, stop the algorithm; otherwise, return to Step 2.

We call this method the shortest paths heuristic (SPH) method. For comparison, the MST-based method is replaced with the SPH method in our greedy connections for feature points. Figure 14 shows an example, where almost the same cuts are obtained by the two methods. In all of the testing examples, the intermediate differences are negligible.

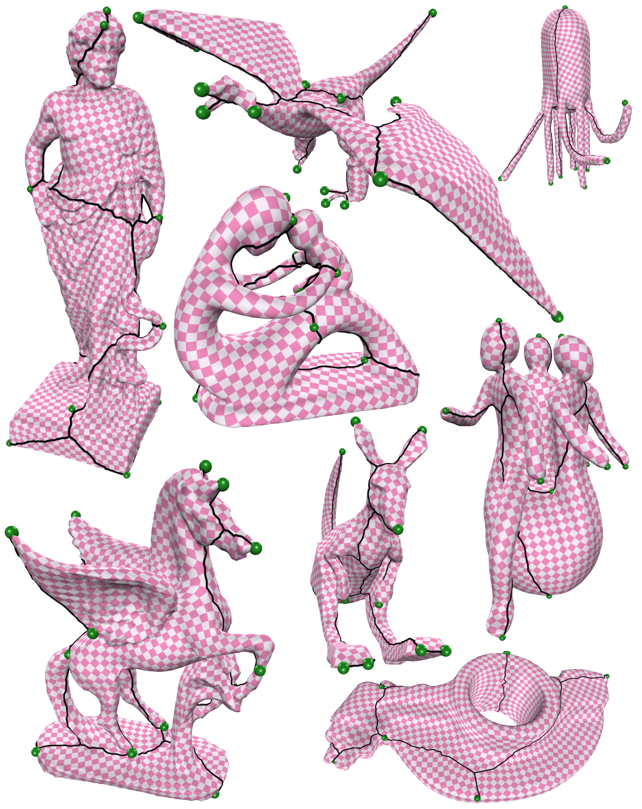
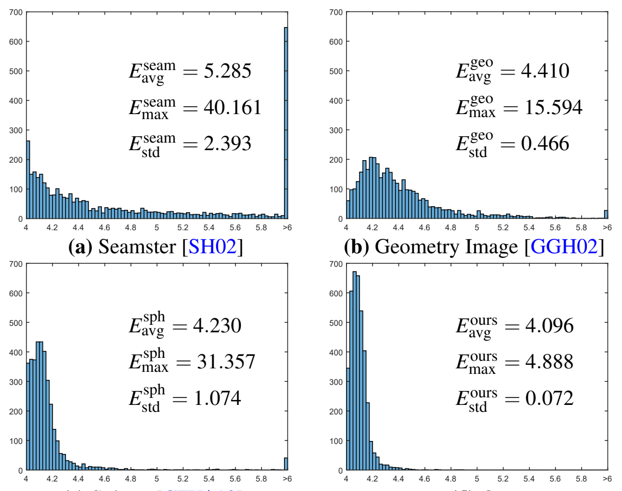


Figure 15: *Gallery. Our method succeeds in generating short cuts for parameterizations with low isometric distortion.*

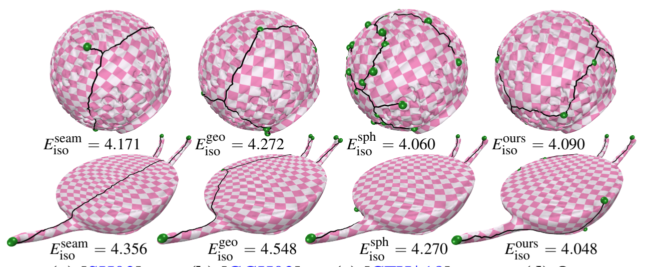
Experiments on a data set. We test our method on a data set consisting of 3,757 models with an average of 13,000 vertices, of which two-thirds are CAD models and one-third are organic models. Our data set also contains some simple models. Our method succeeds in constructing cuts that result in flip-free parameterizations with low isometric distortion in all models. We show eight models in Figure 15. All of the models and our C++ implementation are publicly accessible at `http://staff.ustc.edu.cn/~fuxm/`.

Timings. For the Bunny model in Figure 12(c) with 120,000 triangles, it takes 22.75 seconds, 175.91 seconds, and 3.63 seconds to generate redundant feature points, filter redundant feature points, and connect the feature points, respectively. Redundant feature point generation, redundant feature point filtering, and feature point connection takes 1.61 seconds, 3.74 seconds, and 0.08 seconds for the Teddy model in Figure 22 with 11,596 triangles. The larger the size of the input mesh, the more time it takes to compute planar parameterizations. Since planar parameterizations are used to generate and filter redundant feature points, the time spent to carry out these two processes is much longer than the time it takes to connect feature points, especially in the greedy feature point filtering. Although we have used the parameterization of the previous iteration to initialize



(a) Seamster [SH02] **(b)** Geometry Image [GGH02] **(c)** Sphere [CFH*18] **(d)** Ours

Figure 16: *Distributions of E^{seam}, E^{geo}, E^{sph}, and E^{ours}. The subscripts* max, avg, *and* std *indicate the average, the maximum, and the standard deviation of the resulting distortion for each method across all test models.*


$E^{\text{seam}}_{\text{iso}} = 4.171$
$E^{\text{geo}}_{\text{iso}} = 4.272$
$E^{\text{sph}}_{\text{iso}} = 4.060$
$E^{\text{ours}}_{\text{iso}} = 4.090$
$E^{\text{seam}}_{\text{iso}} = 4.356$
$E^{\text{geo}}_{\text{iso}} = 4.548$
$E^{\text{sph}}_{\text{iso}} = 4.270$
$E^{\text{ours}}_{\text{iso}} = 4.048$

(a) [SH02] **(b)** [GGH02] **(c)** [CFH*18] **(d)** Ours

Figure 17: *Comparisons to the Seamster method, the Geometry Image method, and the sphere-based method using the Red Box and Vase for feature point detection.*

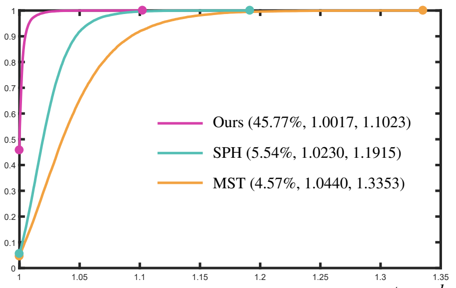


Figure 18: *Cumulative distribution functions of θ^{mst}, θ^{sph}, and θ^{ours}. The text in the bracket indicates the percentage of examples with $\theta = 1$, the average θ, and the maximum θ of the corresponding method over all examples, respectively.*

the next iteration, the total number of iterations for solving the linear system is still so large that it accounts for the majority of the total time.

4.2. Comparisons with other methods for feature point detection

First, we conduct comparisons with other methods for feature point detection. We select the Seamster method [SH02], the Geometry Image method [GGH02], and the sphere-based method [CFH*18] as competitors. The implementation of the sphere-based method is kindly provided by the authors, and we implement the other two competitors by ourselves. We run the three competitors and our method on our data set. Our feature point connection method is used in all methods to construct the final cuts.

We denote the resulting isometric distortion of the Seamster method, the Geometry Image method, the sphere-based method, and ours as E^{seam}, E^{geo}, E^{sph}, and E^{ours}, respectively. We show the distortion distributions via histograms in Figure 16. Our method outperforms the other methods judging from the statistics. Compared to the other three methods, the feature points required to reduce the isometric distortion are detected with higher accuracy.

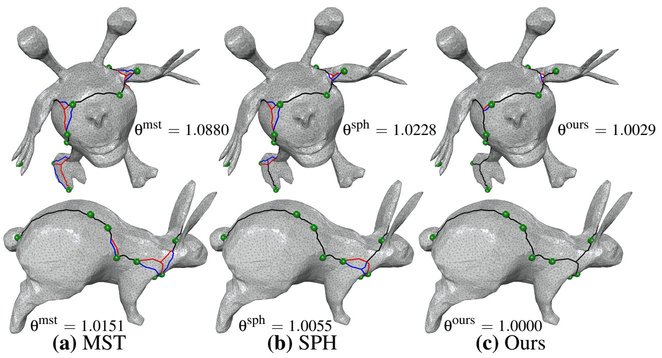
We observe that the Seamster method and the Geometry Image method usually leave out some necessary feature points for reducing isometric distortion. Thus, their results often exhibit higher distortion than ours (see the two examples in Figure 17). For the Red Box model in the first row of Figure 17, the sphere-based method achieves slightly smaller distortion than ours; however, its cut is much longer than ours ($4.94L_{\text{bb}}$ versus $2.66L_{\text{bb}}$). The hierarchical clustering method developed by the sphere-based method may omit some feature points, resulting in large distortion (see the Vase model in the second row of Figure 17). Maintaining small distortion while reducing the number of feature points is a characteristic of our greedy filtering process. In addition, the spherical parameterization method used by the sphere-based method cannot always generate bijective parameterizations. In fact, there are 203 failed cases in which bijective spherical parameterizations are not achieved.

4.3. Comparisons with other methods for feature point connection

Our greedy connection of feature points is an approximate solution to the Steiner tree problem. We then compare our technique to three competitors, including the MST-based method [KMB81], the SPH method [TA80], and the exact solution. The exact solution is achieved by a dynamic programming algorithm [FKM*07]. Since the exact solution cannot be generated in a reasonable time if the number of feature points is too large, the number of feature points for testing is between four and ten. For each model in our data set, we randomly generate seven collections of feature points, whose numbers range from four to ten. We run the four methods on 26,229 test models.

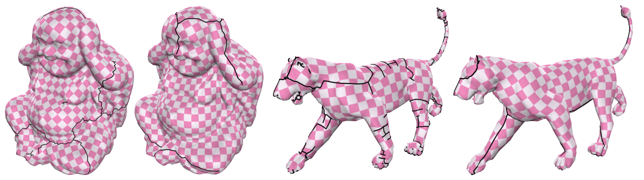
We denote the cut length of the MST-based method, the SPH method, ours, and the exact solution as L^{mst}, L^{sph}, L^{ours}, and L^{exact}, respectively. We report the ratios: $\theta^{\text{mst}} = L^{\text{mst}}/L^{\text{exact}}$, $\theta^{\text{sph}} = L^{\text{sph}}/L^{\text{exact}}$, and $\theta^{\text{ours}} = L^{\text{ours}}/L^{\text{exact}}$, of which the cumulative distribution functions are illustrated in Figure 18. For 45.77% of the examples, our results are the same as the exact solutions. According to the statistics, we outperform the MST-based method and the SPH method and approach the exact solutions.

Comparisons on the Alien and Rabbit models are shown in Figure 19. The cuts generated by our method are shorter than the


$\theta^{\text{mst}} = 1.0880$
$\theta^{\text{sph}} = 1.0228$
$\theta^{\text{ours}} = 1.0029$
$\theta^{\text{mst}} = 1.0151$
$\theta^{\text{sph}} = 1.0055$
$\theta^{\text{ours}} = 1.0000$

(a) MST **(b)** SPH **(c)** Ours

Figure 19: *Comparisons to the MST-based method, the SPH method, and the exact solution on the Alien and Rabbit for feature point connection. On each model, the union of the black and red lines represents the exact solution, while the black and blue lines form the approximate solution.*



(a) AutoCuts $E^{\text{r}}_{\text{iso}} = 4.068$ $L_{\mathcal{C}} = 3.42L_{\text{bb}}$ **(b)** Ours $E^{\text{r}}_{\text{iso}} = 4.062$ $L_{\mathcal{C}} = 2.87L_{\text{bb}}$ **(c)** AutoCuts $E^{\text{r}}_{\text{iso}} = 4.168$ $L_{\mathcal{C}} = 8.06L_{\text{bb}}$ **(d)** Ours $E^{\text{r}}_{\text{iso}} = 4.131$ $L_{\mathcal{C}} = 2.92L_{\text{bb}}$

Figure 20: *Comparisons to the AutoCuts method for cut construction on the Buddha and Tiger models.*

MST-based method and the SPH method, and for the Rabbit model, the cut is even the same as the exact solution.

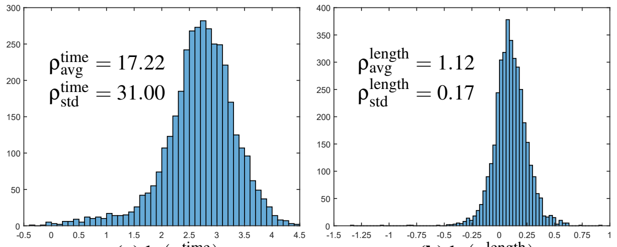
4.4. Comparisons with other methods for cut construction

Finally, we perform comparisons with other methods for cut construction. We select the AutoCuts method [PTH*17] and the OptCuts method [LKK*18] as competitors. Besides, the cut construction methods in [SH02, CFH*18] use MST-based methods to connect the detected feature points. From the comparisons in Section 4.2 and 4.3, it is apparent that our method outperforms them in producing low distortion and short cuts.

We conduct a comparison with the AutoCuts method using the results provided by [LKK*18]. Comparisons using the Buddha model and the Tiger model are shown in Figure 20. Our results have much shorter cuts while the distortions are almost the same.

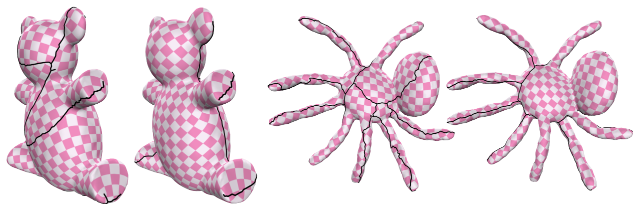
The implementation for the OptCuts method is kindly provided by the authors. Bijective parameterizations are enabled in both our method and the OptCuts method for fair comparisons. We use the method from [JSP17] to achieve our bijective parameterizations. The input isometric distortion bound in the OptCuts method is set as our resulting distortion, so the distortion values for the OptCuts method and our method are almost the same. We run the OptCuts method and our method on our data set.

The running timings for our method and the OptCuts method are denoted as t^{ours} and t^{opt}. We denote the resulting cut length for our method and OptCuts as L^{ours} and L^{opt}, respectively. We report the



(a) $\ln(\rho^{\text{time}})$ **(b)** $\ln(\rho^{\text{length}})$

Figure 21: *Distributions of* $\ln(\rho^{time})$ *and* $\ln(\rho^{length})$*. The average and standard deviation of* ρ^{time} *and* ρ^{length} *over all models are denoted as* ρ^{time}_{avg}, ρ^{time}_{std}, $\rho^{length}_{\text{avg}}$*, and* $\rho^{length}_{\text{std}}$*, respectively.*



(a) AutoCuts $E^{\text{r}}_{\text{iso}} = 4.106$ $L_{\mathcal{C}} = 4.16L_{\text{bb}}$ 234 seconds **(b)** Ours $E^{\text{r}}_{\text{iso}} = 4.106$ $L_{\mathcal{C}} = 2.58L_{\text{bb}}$ 5.43 seconds **(c)** AutoCuts $E^{\text{r}}_{\text{iso}} = 4.148$ $L_{\mathcal{C}} = 4.32L_{\text{bb}}$ 315 seconds **(d)** Ours $E^{\text{r}}_{\text{iso}} = 4.148$ $L_{\mathcal{C}} = 3.48L_{\text{bb}}$ 10.52 seconds

Figure 22: *Comparisons to the OptCuts method for cut construction on the Teddy and Spider models.*

ratios: $\rho^{\text{time}} = t^{\text{opt}}/t^{\text{ours}}$ and $\rho^{\text{length}} = L^{\text{opt}}/L^{\text{ours}}$. The distributions of ρ^{time} and ρ^{length} are illustrated via histograms, as shown in Figure 21. As shown in the statistics, when producing similar isometric distortion levels, our method is one order of magnitude faster and generates shorter cuts than the OptCuts method on average. Comparisons using a Teddy bear model and a Spider model are shown in Figure 22, again verifying that our method is much faster and generates shorter cuts than the OptCuts method.

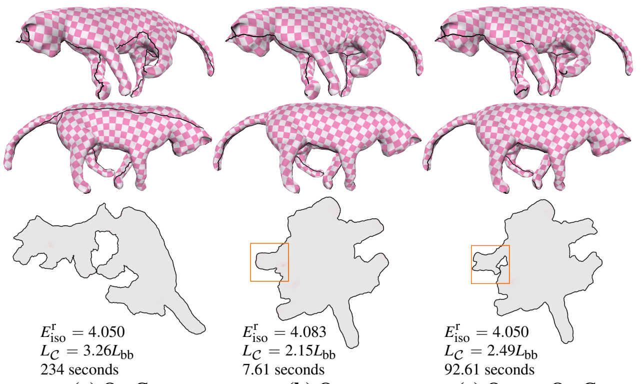
Fairness in comparison with OptCuts. There are two commonly used libraries, Eigen and Pardiso, to solve linear systems. OptCuts adopts Eigen as the default linear system solver, while we use Pardiso by default. However, to thoroughly compare the performance, we test the performances of OptCuts and our method via four models with both Pardiso and Eigen respectively in the same computer and operating system. Table 2 lists the running time of each result. For Optcuts, Pardiso has a small advantage over Eigen only when the model has more than 50,000 vertices. Besides, it takes about 1.39 times longer to run with Pardiso than Eigen when the model has around 13,000 vertices. As for our method, running with Pardiso is slightly faster than running with Eigen. From the perspective of algorithm efficiency, we use the faster setting of each method to have a fair comparison since the average number of model vertices in our data set is about 13,000. Overall, our method is faster than OptCuts with both solvers.

5. Conclusion

Our method provides a novel technique for constructing short cuts for parameterizations while maintaining low isometric distortion. Due to two greedy strategies for filtering redundant feature points

Model	#vert	OptCuts (Eigen)	OptCuts (Pardiso)	Ours (Eigen)	Ours (Pardiso)
Swallow	10,000	162	228	10.64	8.95
Toy	13,000	225	337	10.21	8.86
Plane	15,000	113	143	12.67	11.20
Elephant	50,000	3,578	3,501	138.62	117.32

Table 2: *Running time (in seconds) of two different linear system solvers on four models.*


$E^{\mathrm{r}}_{\mathrm{iso}} = 4.050$
$L_{\mathcal{C}} = 3.26L_{\mathrm{bb}}$
234 seconds
$E^{\mathrm{r}}_{\mathrm{iso}} = 4.083$
$L_{\mathcal{C}} = 2.15L_{\mathrm{bb}}$
7.61 seconds
$E^{\mathrm{r}}_{\mathrm{iso}} = 4.050$
$L_{\mathcal{C}} = 2.49L_{\mathrm{bb}}$
92.61 seconds

(a) OptCuts **(b)** Ours **(c)** Ours + OptCuts

Figure 23: *Bounding isometric distortion. (a) The isometric distortion bound in the OptCuts method is set as 4.050. (b) E^r_{iso} in our method is larger than the bound. (c) We initialize the OptCuts method as our result. The frames in the first row and the second row show the model from two perspectives.*

Model	#tri	$\#\mathcal{F}^{\mathrm{b}}/L^{\mathrm{b}}/E^{\mathrm{b}}_{\mathrm{iso}}$	$\#\mathcal{F}^{\mathrm{a}}/L^{\mathrm{a}}/E^{\mathrm{a}}_{\mathrm{iso}}$	$\#\mathcal{A}/L_{\mathcal{C}}/E^{\mathrm{r}}_{\mathrm{iso}}$	$t_g/t_f/t_c/t_{\mathrm{total}}$ (s)
Fish (Figure 1)	13000	54/2.48/4.060	7/1.87/4.043	4/1.79/4.050	1.78/4.09/0.04/5.91
Frog (Figure 1)	13000	41/4.29/4.112	16/3.88/4.190	15/3.19/4.179	2.39/10.08/0.54/13.01
Torso (Figure 2 (c))	11392	87/3.73/4.048	12/2.13/4.131	6/2.01/4.141	2.17/9.78/0.13/12.08
Hand (Figure 3 (b))	13000	96/4.07/4.029	7/3.50/4.083	5/2.55/4.095	1.92/8.41/0.07/10.39
GardenPig (Figure 4)	19022	92/4.74/4.036	9/2.74/4.067	5/2.33/4.072	2.81/11.80/0.12/14.73
Cow (Figure 5)	16452	71/4.84/4.043	13/3.39/4.087	9/3.10/4.096	2.74/8.10/0.22/11.05
Robot (Figure 7&8)	13000	122/5.03/4.043	15/3.32/4.091	11/2.80/4.098	2.11/15.72/0.29/18.12
Kitten (Figure 9)	15462	72/4.60/4.044	11/2.81/4.057	6/2.67/4.063	4.75/12.08/0.11/16.95
Bunny (Figure 12 (a))	13000	79/6.07/4.048	11/2.94/4.092	9/2.67/4.093	2.23/9.14/0.12/11.49
Bunny (Figure 12 (b))	2000	33/4.93/4.073	12/2.94/4.103	6/2.68/4.101	0.52/1.22/0.02/1.76
Bunny (Figure 12 (c))	120000	188/7.49/4.031	19/3.40/4.082	11/2.96/4.093	22.75/175.91/3.63/202.30
Bunny (Figure 12 (d))	13000	265/10.72/4.039	12/2.90/4.103	5/2.76/4.114	1.95/22.83/0.09/24.87
Bunny (Figure 12 (e))	14797	59/5.42/4.058	8/3.36/4.146	4/2.94/4.098	2.32/7.38/0.07/9.77
Bunny (Figure 12 (f))	10043	49/5.27/4.060	11/3.05/4.136	6/2.68/4.095	1.49/6.06/0.08/7.63
Head (Figure 13)	109945	219/7.57/4.008	41/4.57/4.020	17/4.32/4.019	17.47/146.06/22.83/186.93
Oni (Figure 13)	107810	99/5.59/4.054	20/4.01/4.097	11/3.71/4.086	16.90/78.73/4.09/100.25
Buddha (Figure 13)	114335	235/8.60/4.015	26/4.04/4.035	9/3.88/4.034	18.00/190.40/5.08/214.07
Armchair (Figure 14 (a))	18342	98/5.03/4.018	9/2.51/4.061	2/2.43/4.075	2.65/15.46/0.05/18.16
Dragon (Figure 15)	16326	1192/15.66/4.019	14/2.82/4.101	11/2.52/4.103	3.67/119.12/0.56/123.36
Kangaroo (Figure 15)	14958	156/6.58/4.033	12/3.57/4.097	12/3.19/4.123	2.49/17.39/0.35/20.23
Octopus (Figure 15)	16272	42/4.61/4.078	9/5.42/4.072	10/3.91/4.113	4.04/5.78/0.28/10.10
Uu-memento (Figure 15)	15945	67/5.53/4.089	16/4.87/4.112	12/4.09/4.116	2.96/11.41/0.51/14.88
Asclepius (Figure 15)	24994	410/10.19/4.054	15/3.08/4.113	14/2.88/4.120	14.96/83.05/1.10/99.11
Rockerarm (Figure 15)	10044	79/5.10/4.026	12/2.96/4.139	5/2.83/4.157	3.36/7.73/0.10/11.20
Fertility (Figure 15)	13971	61/4.85/4.061	14/4.03/4.161	9/3.88/4.084	4.87/8.51/0.31/13.69
Pegaso (Figure 15)	15319	154/7.99/4.100	21/4.94/4.190	21/4.71/4.131	5.93/32.67/1.92/40.52
Red (Figure 17)	12767	644/20.35/4.030	11/2.78/4.085	3/2.66/4.090	3.67/62.60/0.05/66.33
Vase (Figure 17)	13315	93/4.03/4.022	9/2.38/4.041	3/2.24/4.048	2.14/8.67/0.03/10.84
Alien (Figure 19)	17732	102/5.85/4.096	17/4.98/4.119	19/3.73/4.129	3.16/14.44/1.11/18.72
Rabbit (Figure 19)	13697	66/5.03/4.058	10/3.30/4.125	7/2.79/4.133	1.95/6.65/0.10/8.70
Buddha (Figure 20)	5002	102/7.76/4.032	11/2.92/4.071	4/2.87/4.062	0.96/8.13/0.03/9.14
Tiger (Figure 20)	5396	68/4.28/4.105	18/3.697/4.104	17/2.92/4.131	1.52/10.67/0.40/12.60
Teddy (Figure 22)	11596	36/3.51/4.141	10/2.85/4.073	6/2.58/4.106	1.61/3.74/0.08/5.43
Spider (Figure 22)	16220	51/4.36/4.145	12/4.10/4.107	12/3.48/4.148	2.35/7.82/0.35/10.52
Cat (Figure 23)	13000	69/3.73/4.044	8/2.52/4.090	5/2.15/4.083	2.15/5.39/0.072/7.61

Table 3: *Statistics and timings of our results. We report the number of triangles of the input mesh ("#tri"), the number of feature points, the cut length, and parameterization distortion before ("$\#\mathcal{F}^b$, L^b, and E^b_{iso}") and after filtering ("$\#\mathcal{F}^a$, L^a, and E^a_{iso}"), the number of auxiliary points ("$\#\mathcal{A}$"), the length of the resulting cut ("$L_\mathcal{C}$"), the isometric distortion metric of the resulting parameterization ("E^r_{iso}"), and the computational time in seconds for redundant feature point generation ("t_g"), redundant feature point filtering ("t_f") and feature point connection ("t_c"). L^b, L^a, and $L_\mathcal{C}$ are in L_{bb}, which is the diagonal length of the bounding box of $\mathcal{M}$. Note that the cuts before and after filtering are constructed by the MST-based method using the candidate feature points $\mathcal{F}^b$ and final feature points $\mathcal{F}^a$, respectively. For high-genus models, we report the statistics of $\mathcal{M}_{one}$.*

and connecting feature points, our method achieves short cuts and low isometric distortion. It performs very fast, and the overall cut quality is superior to previous methods. We have demonstrated the efficacy of our method on a data set containing 3,757 models.

Bounded isometric distortion. Although our method tries to detect necessary feature points in order to reduce isometric distortion, we cannot explicitly bound it. For example, the maximum isometric distortion in our data set is 4.888. Fortunately, our method is very fast, so our method can be used as a pre-processing step in the OptCuts method to achieve bounded distortion. As shown in Figure 23, to achieve the same user-specified isometric distortion bound as the OptCuts method, combining our method with the OptCuts method uses much less time and generates a shorter cut.

Cut length. For 21.2% of the examples, our method generates longer cuts than the OptCuts method. This indicates that our output leaves room for improvement in reducing cut length. Therefore, efficiently and effectively reducing cut length while bounding isometric distortion is still an intriguing direction for future research.

Acknowledgments

We would like to thank Minchen Li for sharing their source code for [LKK*18], and the anonymous reviewers for their constructive suggestions and comments. This work is supported by the National Natural Science Foundation of China (61802359), the Fundamental Research Funds for the Central Universities (WK0010460006, WK0010450004), the Anhui Provincial Natural Science Foundation (1808085QF208), and the Key R&D Program of Zhejiang Province (2018C01112).

References

[BCGB08] BEN-CHEN M., GOTSMAN C., BUNIN G.: Conformal flattening by curvature prescription and metric scaling. *Comput. Graph. Forum 27*, 2 (2008), 449–458. 3

[Bea89] BEASLEY J. E.: An SST-based algorithm for the Steiner problem in graphs. *Networks 19*, 1 (1989), 1–16. 3

[BGRS13] BYRKA J., GRANDONI F., ROTHVOSS T., SANITÀ L.: Steiner tree approximation via iterative randomized rounding. *Journal of the ACM (JACM) 60*, 1 (2013), 6. 3

[BR94] BERMAN P., RAMAIYER V.: Improved approximations for the Steiner tree problem. *Journal of Algorithms 17*, 3 (1994), 381–408. 3

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (2009), 77:1–77:10. 3

[CFH*18] CHAI S., FU X.-M., HU X., YANG Y., LIU L.: Sphere-based Cut Construction for Planar Parameterizations. *Computer & Graphics 74* (2018), 66–75. 2, 3, 6, 9, 10

[CFL19] CHAI S., FU X.-M., LIU L.: Voting for Distortion Points in Geometric Processing. *IEEE. T. Vis. Comput. Gr.* (2019). 3

[DFW13] DEY T. K., FAN F., WANG Y.: An Efficient Computation of Handle and Tunnel Loops via Reeb Graphs. *ACM Trans. Graph. 32*, 4 (2013), 32:1–32:10. 6

[FGK08] FOMIN F. V., GRANDONI F., KRATSCH D.: Faster Steiner tree computation in polynomial-space. In *European Symposium on Algorithms* (2008), pp. 430–441. 3

[FH05] Floater M. S., Hormann K.: Surface parameterization: a tutorial and survey. In *In Advances in Multiresolution for Geometric Modelling* (2005), Springer, pp. 157–186. 2

[FKM*07] Fuchs B., Kern W., Molle D., Richter S., Rossmanith P., Wang X.: Dynamic programming for minimum Steiner trees. *Theory of Computing Systems 41*, 3 (2007), 493–500. 9

[FLG15] Fu X.-M., Liu Y., Guo B.: Computing locally injective mappings by advanced MIPS. *ACM Trans. Graph. 34*, 4 (2015), 71:1–71:12. 2

[Flo03] Floater M. S.: One-to-one piecewise linear mappings over triangulations. *Math. Comput. 72* (2003), 685–696. 2

[FLSG14] Fu X.-M., Liu Y., Snyder J., Guo B.: Anisotropic simplicial meshing using local convex functions. *ACM Trans. Graph. 33*, 6 (2014), 182:1–182:11. 7

[GGH02] Gu X., Gortler S. J., Hoppe H.: Geometry Images. *ACM Trans. Graph. 21*, 3 (2002), 355–361. 2, 3, 9

[GSC18] Golla B., Seidel H.-P., Chen R.: Piecewise linear mapping optimization based on the complex view. *Comput. Graph. Forum 37*, 7 (2018), 233–243. 2, 4, 7

[Hak71] Hakimi S. L.: Steiner's problem in graphs and its implications. *Networks 1*, 2 (1971), 113–133. 3

[HFL18] Hu X., Fu X.-M., Liu L.: Advanced Hierarchical Spherical Parameterizations. *IEEE. T. Vis. Comput. Gr. 24*, 6 (2018), 1930–1941. 3

[HG00] Hormann K., Greiner G.: MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*. Vanderbilt University Press, 2000, pp. 153–162. 2, 3

[HLS07] Hormann K., Lévy B., Sheffer A.: Mesh Parameterization: Theory and Practice. In *ACM SIGGRAPH 2007 Courses* (2007), SIGGRAPH '07. 2, 3

[HRW92] Hwang F. K., Richards D. S., Winter P.: The steiner tree problem. *Annals of Discrete Mathematics 53* (1992). 3

[JKS05] Julius D., Kraevoy V., Sheffer A.: D-Charts: Quasi-Developable Mesh Segmentation. In *Comput. Graph. Forum* (2005), vol. 24, pp. 581–590. 2

[JSP17] Jiang Z., Schaefer S., Panozzo D.: Simplicial Complex Augmentation Framework for Bijective Maps. *ACM Trans. Graph. 36*, 6 (2017), 186:1–186:9. 2, 10

[KCPS13] Knöppel F., Crane K., Pinkall U., Schröder P.: Globally Optimal Direction Fields. *ACM Trans. Graph. 32*, 4 (2013), 59:1–59:10. 3

[KMB81] Kou L., Markowsky G., Berman L.: A fast algorithm for Steiner trees. *Acta informatica 15*, 2 (1981), 141–145. 2, 3, 5, 6, 9

[LDB17] Lucquin V., Deguy S., Boubekeur T.: SeamCut: Interactive Mesh Segmentation for Parameterization. In *ACM SIGGRAPH 2017 Technical Briefs* (2017). 3

[LFY*19] Liu H.-Y., Fu X.-M., Ye C., Chai S., Liu L.: Atlas Refinement with Bounded Packing Efficiency. *ACM Trans. Graph. 38*, 4 (2019), 33:1–33:13. 3

[LKK*18] Li M., Kaufman D. M., Kim V. G., Solomon J., Sheffer A.: OptCuts: Joint Optimization of Surface Cuts and Parameterization. *ACM Trans. Graph. 37*, 6 (2018), 247:1–247:13. 1, 2, 10, 11

[LPRM02] Lévy B., Petitjean S., Ray N., Maillot J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3 (2002), 362–371. 2

[LVS18] Limper M., Vining N., Sheffer A.: Box Cutter: Atlas Refinement for Efficient Packing via Void Elimination. *ACM Trans. Graph. 37*, 4 (2018), 153:1–153:13. 3

[LYNF18] Liu L., Ye C., Ni R., Fu X.-M.: Progressive Parameterizations. *ACM Trans. Graph. 37*, 4 (2018), 41:1–41:12. 2, 5, 6, 7

[LZF*19] Liu H., Zhang X.-T., Fu X.-M., Dong Z.-C., Liu L.: Computational Peeling Art Design. *ACM Trans. Graph. 38*, 4 (2019), 64:1–64:12. 3

[MZ12] Myles A., Zorin D.: Global Parametrization by Incremental Flattening. *ACM Trans. Graph. 31*, 4 (2012), 109:1–109:11. 3

[PTH*17] Poranne R., Tarini M., Huber S., Panozzo D., Sorkine-Hornung O.: Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Trans. Graph. 36*, 6 (2017), 215:1–215:11. 1, 2, 3, 10

[PUW18] Pajor T., Uchoa E., Werneck R. F.: A robust and scalable algorithm for the Steiner problem in graphs. *Mathematical Programming Computation 10*, 1 (2018), 69–118. 3

[RPPSH17] Rabinovich M., Poranne R., Panozzo D., Sorkine-Hornung O.: Scalable Locally Injective Mappings. *ACM Trans. Graph. 36*, 2 (2017), 16:1–16:16. 2

[RZ05] Robins G., Zelikovsky A.: Tighter bounds for graph Steiner tree approximation. *SIAM Journal on Discrete Mathematics 19*, 1 (2005), 122–134. 3

[SC18] Sharp N., Crane K.: Variational Surface Cutting. *ACM Trans. Graph. 37*, 4 (2018), 156:1–156:13. 3

[SCOGL02] Sorkine O., Cohen-Or D., Goldenthal R., Lischinski D.: Bounded-distortion piecewise mesh parameterization. In *Proceedings of the Conference on Visualization '02* (2002), pp. 355–362. 2

[SGSH02] Sander P. V., Gortler S. J., Snyder J., Hoppe H.: Signal-specialized parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering* (2002), pp. 87–98. 2

[SH02] Sheffer A., Hart J. C.: Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of the conference on Visualization'02* (2002), pp. 291–298. 2, 3, 9, 10

[She02] Sheffer A.: Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Shape Modeling International* (2002), pp. 61–66. 2, 3

[SJZP19] Shen H., Jiang Z., Zorin D., Panozzo D.: Progressive Embedding. *ACM Trans. Graph. 38*, 4 (2019), 32:1–32:13. 2

[SOG09] Sun J., Ovsjanikov M., Guibas L.: A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Comput. Graph. Forum 28*, 5 (2009), 1383–1392. 3

[SPR06] Sheffer A., Praun E., Rose K.: Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis. 2*, 2 (2006), 105–171. 2

[SPSH*17] Shtengel A., Poranne R., Sorkine-Hornung O., Kovalsky S., Lipman Y.: Geometric Optimization via Composite Majorization. *ACM Trans. Graph. 36*, 4 (2017), 38:1–38:11. 2

[SS15] Smith J., Schaefer S.: Bijective Parameterization with Free Boundaries. *ACM Trans. Graph. 34*, 4 (2015), 70:1–70:9. 2, 3

[SSC18] Soliman Y., Slepčev D., Crane K.: Optimal Cone Singularities for Conformal Flattening. *ACM Trans. Graph. 37*, 4 (2018), 105:1–105:17. 3

[SSP08] Springborn B., Schröder P., Pinkall U.: Conformal equivalence of triangle meshes. *ACM Trans. Graph. 27*, 3 (2008), 77:1–77:11. 3

[TA80] Takahashi H., Akira M.: An approximate solution for the Steiner problem in graphs. *Mathematica Japonica 24*, 6 (1980), 573–577. 3, 8, 9

[Tut63] Tutte W. T.: How to draw a graph. In *Proceedings of the London Mathematical Society* (1963), vol. 13, pp. 747–767. 2

[VCD*16] Vaxman A., Campen M., Diamanti O., Panozzo D., Bommes D., Hildebrandt K., Ben-Chen M.: Directional field synthesis, design, and processing. *Comput. Graph. Forum 35*, 2 (2016), 545–572. 3

[ZSGS04] Zhou K., Synder J., Guo B., Shum H.-Y.: Iso-charts: Stretch-driven Mesh Parameterization Using Spectral Analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2004), pp. 45–54. 2