# Atlas Refinement with Bounded Packing Efficiency

HAO-YU LIU, University of Science and Technology of China, China
XIAO-MING FU*, University of Science and Technology of China, China
CHUNYANG YE, University of Science and Technology of China, China
SHUANGMING CHAI, University of Science and Technology of China, China
LIGANG LIU, University of Science and Technology of China, China

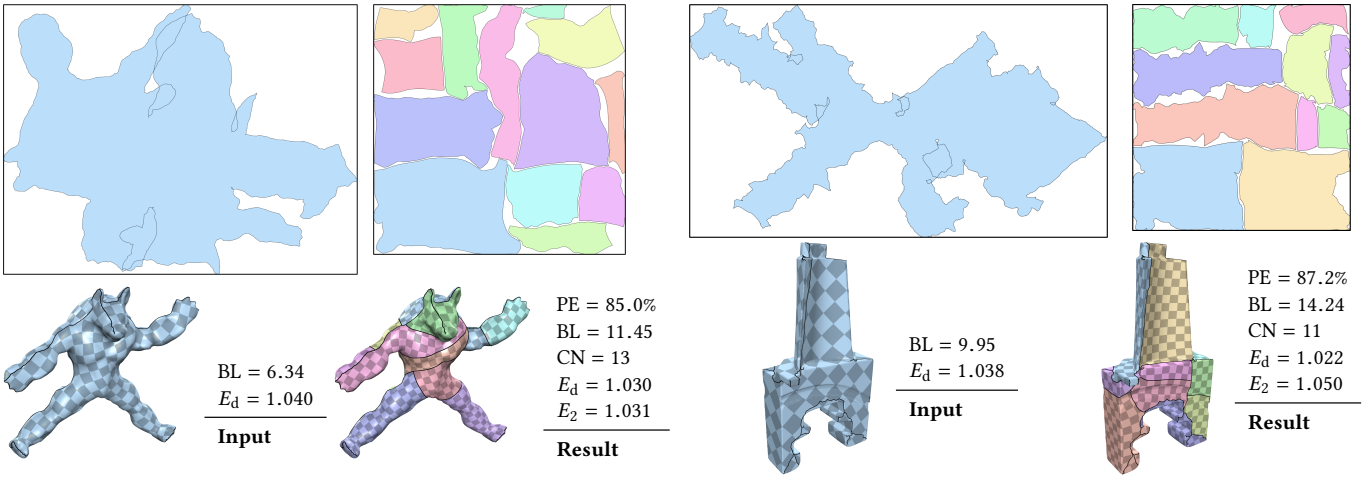



Fig. 1. Refining input parameterized charts to obtain packing efficiency that is greater than or equal to the given bounds. Compared to the inputs that have low packing efficiency and are not bijective, our method significantly improves the packing efficiency and ensures the bijection. We set the packing efficiency bounds as 80% for these two models. PE, BL, CN, $E_d$, and $E_2$ represent the packing efficiency, the boundary length, the number of charts, the symmetric Dirichlet distortion metric with respect to the input 3D surface, and the symmetric Dirichlet distortion metric with respect to the input atlas, respectively.

We present a novel algorithm to refine an input atlas with bounded packing efficiency. Central to this method is the use of the axis-aligned structure that converts the general polygon packing problem to a rectangle packing problem, which is easier to achieve high packing efficiency. Given a parameterized mesh with no flipped triangles, we propose a new angle-driven deformation strategy to transform it into a set of axis-aligned charts, which can be decomposed into rectangles by the motorcycle graph algorithm. Since motorcycle graphs are not unique, we select the one balancing the trade-off between the packing efficiency and chart boundary length, while maintaining bounded packing efficiency. The axis-aligned chart often contains greater distortion than the input, so we try to reduce the distortion while bounding the packing efficiency and retaining bijection. We demonstrate the efficacy of our method on a data set containing over five thousand complex models. For all models, our method is able to produce packed atlases with bounded packing efficiency; for example, when the packing efficiency bound is set to 80%, we elongate the boundary length by an average of 78.7% and increase the distortion by an average of 0.0533%. Compared to state-of-the-art methods, our method is much faster and achieves greater packing efficiency.

CCS Concepts: • **Computing methodologies** → **Shape modeling**.

Additional Key Words and Phrases: atlas refinement, bounded packing efficiency, axis-aligned chart

**ACM Reference Format:**
Hao-Yu Liu, Xiao-Ming Fu, Chunyang Ye, Shuangming Chai, and Ligang Liu. 2019. Atlas Refinement with Bounded Packing Efficiency. *ACM Trans. Graph.* 38, 4, Article 33 (July 2019), 13 pages. https://doi.org/10.1145/3306346.3323001

## 1 INTRODUCTION

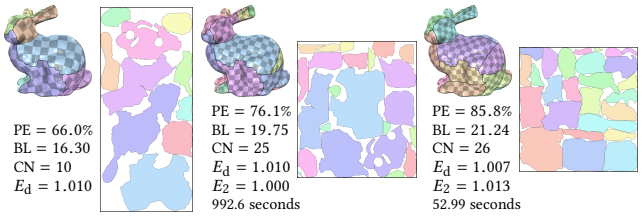
In computer graphics, atlases are constructed by packing 2D parameterized charts into rectangular texture image domains. They are commonly used to store surface signals, such as colors, normals, and textures. Due to the regularity of texture image grids, they are beneficial for some geometric processing tasks, such as signal smoothing and sharpening, texture stitching, geodesic distance computation, and line integral convolution [Prada et al. 2018].

A good atlas usually satisfies the following properties: (1) the parameterization of each chart is bijective and contains as little as possible isometric distortion; (2) the packed parameterized charts

*The corresponding author

Authors' addresses: Hao-Yu Liu, University of Science and Technology of China, China, optexeon@mail.ustc.edu.cn; Xiao-Ming Fu, University of Science and Technology of China, China, fuxm@ustc.edu.cn; Chunyang Ye, University of Science and Technology of China, China, yechyang@mail.ustc.edu.cn; Shuangming Chai, University of Science and Technology of China, China, kfckfckf@mail.ustc.edu.cn; Ligang Liu, University of Science and Technology of China, China, lgliu@ustc.edu.cn.


0730-0301/2019/7-ART33 $15.00
https://doi.org/10.1145/3306346.3323001




**(a)** [Julius et al. 2005] **(b)** [Limper et al. 2018] **(c)** Ours

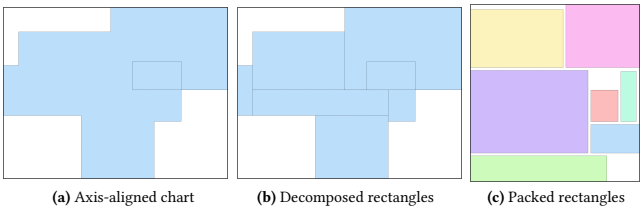
Fig. 2. Refining an input atlas (a) to improve packing efficiency. (b) When the boundary length is allowed to increase by 100%, the method of [Limper et al. 2018] improves packing efficiency by only 15%. (c) The packing efficiency improvement of our method is 30%, and the resulting boundary length and parameterization distortion are similar to [Limper et al. 2018].

have no global overlap; (3) the boundary length of all the charts is as short as possible, otherwise the rendering performance may be decreased [Hakura and Gupta 1997], and texturing artifacts may arise [Poranne et al. 2017]; (4) the packing efficiency, which is defined as the ratio between the areas of an atlas and its bounding box, is as high as possible, otherwise it causes big waste of space.

It is non-trivial to generate atlases that adequately satisfy the aforementioned four requirements. One common way to generate atlases contains three steps [Julius et al. 2005; Lévy et al. 2002; Zhou et al. 2004]: (i) compute seams that are as short as possible to segment an input mesh into charts; (ii) parameterize the charts with as little isometric distortion as possible; and (iii) pack the parameterized charts into a rectangular domain. Some algorithms consider the first and second steps using a whole optimization problem [Li et al. 2018; Poranne et al. 2017]. These methods are able to compute atlases that achieve trade-offs between the first three properties. However, their output leaves room to improve the packing efficiency (Fig. 2 (a)).

To that end, given a previously generated atlas that may have overlaps, Limper *et al.* [2018] propose an iterative cut-and-repack process to achieve the desired trade-offs between packing efficiency improvement and boundary elongation without changing distortion. However, there are two main limitations. First, the practical algorithm for improving packing efficiency may be trapped by the local minimum, resulting in limited improvement (Fig. 2 (b)). Second, although the packing method is optimized to achieve a balance between packing efficiency and computation time, it is still costly to use it many times throughout the whole atlas refinement process.

In this paper, we present a novel method to refine a given atlas with a bounded packing efficiency. Our algorithm achieves a desired trade-off between boundary elongation and improved packing efficiency while guaranteeing global bijection and increasing as little as possible distortion. Our key insight is that instead of handling the packing problem of general polygons, we achieve high packing efficiency by efficiently and effectively solving a rectangle packing problem (Fig. 3). As we know, an axis-aligned 2D polygon, which has a boundary that is axis-aligned, is easily broken down into a set of rectangles. To this end, we propose a three-step procedure (Fig. 4): (1) deform the input charts into axis-aligned charts; (2) decompose the axis-aligned charts into a set of rectangles that are packed into a rectangular domain with high packing efficiency; (3) reduce the parameterization distortion while maintaining bijections and bounding the packing efficiency. For the first step, we design a novel angle-driven strategy to robustly deform the input chart to be axis-aligned with as little isometric distortion as possible and without $2\pi$ ambiguity. Then, motorcycle graphs are used for decomposing the axis-aligned chart to achieve a desired trade-off between packing efficiency and chart boundary length. Since the axis-aligned chart construction process increases the isometric distortion of the input atlas, we use a similar method to [Jiang et al. 2017] to decrease isometric distortion without violating bijection properties and packing efficiency bounds.



**(a)** Axis-aligned chart **(b)** Decomposed rectangles **(c)** Packed rectangles

Fig. 3. An axis-aligned chart (a) can be decomposed into rectangles (b) that can be packed highly efficiently (c). The packing efficiency in (c) is 87.6%.

Our method is able to efficiently produce atlases with bounded packing efficiency for various input atlases. We demonstrate the practical robustness and computational efficiency of our method on a data set containing 5,588 models. Compared to state-of-the-art methods, our method achieves higher efficiency and better quality.

## 2 RELATED WORK

Atlas generation consists of three main components: chart generation, chart parameterizations, and parameterized chart packing.

*Chart generation.* Generating charts also indicates cut detection. Many methods first detect the distortion vertices that are most likely to introduce high isometric distortion. Then, they connect them with minimal spanning trees to compute the resulting cut paths. To detect distortion vertices, the Gaussian Curvature [Sheffer 2002; Sheffer and Hart 2002] and the distortion measurement [Chai et al. 2018; Gu et al. 2002] are used. In addition, conformal cone singularities [Ben-Chen et al. 2008; Kharevych et al. 2006; Myles and Zorin 2012; Soliman et al. 2018; Springborn et al. 2008] can also be treated as distortion vertices. The above methods try to balance parameterization distortion and chart boundary lengths. In [Li et al. 2018; Poranne et al. 2017; Sorkine et al. 2002], the parameterization distortion and boundary lengths are simultaneously and directly optimized. However, none of the above methods consider packing efficiency optimization. Packing efficiency is hard to optimize, so the compactness measurement (e.g., convexity or roundness) of charts is used to implicitly improve packing efficiency and reduce the possibility of overlaps [Julius et al. 2005; Lévy et al. 2002; Sander et al. 2002, 2001; Zhou et al. 2004]. As observed by [Limper et al. 2018], there is no clear, direct relationship between compactness measurements and packing efficiency, and their results still leave room for improving packing efficiency. Each chart consists of two to six triangles [Carr and Hart 2002], or is a tiny quadrilateral patch [Carr et al. 2006; Purnomo et al. 2004], which can be packed highly efficiently; however, the number of charts is too long and the boundary length is too long. Similar to [Limper et al. 2018], the outputs of these method can serve as the inputs of our method.

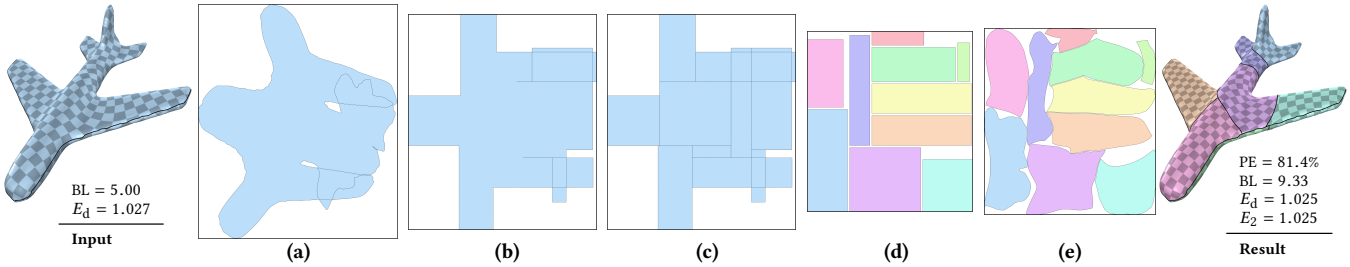



Fig. 4. The pipeline of our method. Given a parameterized chart (a), our method first constructs an axis-aligned chart (b), then partitions the axis-aligned chart into a set of rectangles (c) that are packed into a rectangular domain with high packing efficiency 87.1% (d), and finally it reduces distortion while bounding packing efficiency and ensuring bijection (e). Furthermore, the charts are not allowed to be too close to each other in the distortion reduction step.

*Foldover-free parameterizations.* Given a disk topology chart, numerous parameterization techniques have been proposed in the past thirty years (cf. the surveys in [Floater and Hormann 2005; Hormann et al. 2007; Sheffer et al. 2006]). Since the input parameterization of each chart should be foldover-free, we only review prior works that guarantee no foldovers. Although Tutte's embedding method [Floater 2003; Tutte 1963] is theoretically guaranteed to generate bijective parameterizations, high isometric distortion often exists for complex inputs. Many methods first use Tutte's embedding method to compute foldover-free initializations, and then try to reduce isometric distortion while ensuring no flipped triangles [Claici et al. 2017; Fu et al. 2015; Golla et al. 2018; Hormann and Greiner 2000; Kovalsky et al. 2016; Liu et al. 2018; Rabinovich et al. 2017; Schüller et al. 2013; Shtengel et al. 2017; Zhu et al. 2018]. Furthermore, bijective parameterizations are guaranteed [Jiang et al. 2017; Smith and Schaefer 2015]. Our method uses the scaffold-based method [Jiang et al. 2017; Zhang et al. 2005] to decrease isometric distortion while ensuring bijection and bounding packing efficiency.

*Chart packing.* Given a set of 2D parameterized charts, maximizing the packing efficiency is an NP-hard problem [Garey and Johnson 1979; Milenkovic 1999]. Heuristic strategies, which achieve a trade-off between packing efficiency and computational cost by rotating and translating charts, have been developed [Lévy et al. 2002; Nöll and Strieker 2011; Sander et al. 2003]. In addition to rotations and translations, Limper *et al.* [2018] add new cuts for obtaining the desired trade-off between packing efficiency and boundary length. In contrast, we convert the original problem to a rectangle packing problem that makes it easier to achieve high packing efficiency. In Section 4, we provide a detailed comparison to [Limper et al. 2018].

*Axis-aligned chart.* PolyCube and PolySquare are two commonly used axis-aligned structures in computer graphics. PolyCube is essential to many computer graphics applications, such as texture mapping [Chang and Lin 2010; Tarini et al. 2004; Yao and Lee 2008], all-hexahedral meshing [Fang et al. 2016; Fu et al. 2016; Gregson et al. 2011; Huang et al. 2014; Livesu et al. 2013; Yu et al. 2014] and GPU-based subdivision [Xia et al. 2011]. PolySquare is also very useful, such as quad mesh generation for 2D domains [Liu et al. 2017], and isogeometric analysis parameterizations [Xiao et al. 2018]. Since axis-aligned charts are composed of rectangles, which can be rapidly packed into a rectangular domain with high packing efficiency, we deform the input parameterized charts to axis-aligned charts for initial high packing efficiency. Different from PolyCubes and PolySquares, the corners of our axis-aligned charts are not required to locate at integers. Similar to the former methods [Liu et al. 2017; Xiao et al. 2018], our axis-aligned chart construction method is inspired by [Fu et al. 2016]. However, they use a direction vector to represent the orientations of boundary edges in the construction process, and this representation may result in a failure to generate foldover-free maps from input charts to axis-aligned charts. To this end, we propose a novel angle-driven deformation approach.

## 3 METHOD

We first introduce our novel angle-driven axis-aligned chart construction method in Section 3.1, then present techniques for decomposing an axis-aligned chart and packing rectangles in 3.2, and finally reduce distortion while keeping bijection and bounding packing efficiency in 3.3. Fig. 4 shows the workflow of our method.

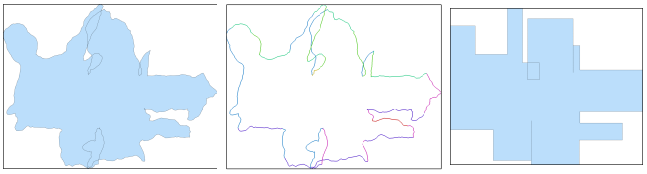
*Inputs.* We receive as input a 3D triangular mesh $\mathbf{M}$ containing $N_f$ triangles $\mathbf{F} = \{\mathbf{f}_i\}_{i=1}^{N_f}$ and $N_c$ parameterized charts $\mathbf{C} = \{\mathbf{c}_i\}_{i=1}^{N_c}$ that are disk topologies. The initial parameterized charts do not contain flipped triangles and may contain global overlaps. And if the normal of a chart is in the negative $z$-direction, we flip it at first. Each triangle $\mathbf{f}_i$ has a corresponding parameterized triangle in $\mathbf{C}$, which is denoted as $\mathbf{f}_i^c$. The mapping from $\mathbf{f}_i$ to $\mathbf{f}_i^c$ is an affine transformation whose Jacobian matrix is denoted as $J_i$.

### 3.1 Angle-driven axis-aligned chart construction

Since the axis-aligned chart construction of each chart is independent, we only discuss one chart $\mathbf{c}$ in this Section for concision. We map $\mathbf{c}$ into an axis-aligned chart while ensuring no flipped triangles.

*Chart* $\mathbf{c}$. The chart $\mathbf{c}$ contains $N_b$ boundary edges, denoted as $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^{N_b}$. The indices of $\mathbf{b}_i$ are defined in counterclockwise order, and expanded periodically. That is to say, we have $\mathbf{b}_{i+kN_b} = \mathbf{b}_i$ for $i = 1, \ldots, N_b$ and $k \in \mathbf{Z}$. We denote the boundary vertex between $\mathbf{b}_i$ and $\mathbf{b}_{i+1}$ as $\mathbf{v}_i^b$, and also expand the indices of $\mathbf{v}_i^b$ periodically. The interior angle of $\mathbf{v}_i^b$ is denoted as $\alpha_i$.

*Orientation representation.* Our goal is to deform $\mathbf{c}$ into an axis-aligned chart that requires the orientation of each $\mathbf{b}_i$ to be axis-aligned. To represent the orientation of $\mathbf{b}_i$, there are two commonly used candidates: (1) the direction vector [Liu et al. 2017; Xiao et al. 2018] and (2) the polar angle. The former representation is easy to compute, but if we rotate $\mathbf{b}_i$ by $2\pi$, the direction vector is exactly the same as the one before rotation, which may lead to ambiguities



(a) (b) (c)

Fig. 5. The pipeline of axis-aligned chart construction. The input atlas is the same as the left example of Fig. 1. (a) After applying global rotation. (b) The computed target polar angles $\frac{\pi}{2}\Theta_i$. We colorize the edge according to its $\Theta_i$. (c) The resulting axis-aligned chart.

(see Fig. 6). Thus, we choose the polar angle, which is denoted as $\theta_i$ for $\mathbf{b}_i$. Since the interior angles are always single-valued, we compute the polar angles in a recurrence relation to deal with the multivalued problem:

$$\theta_{i+1} = \theta_i + \pi - \alpha_i, \quad i = s, s+1, ..., s+N_b-2, \tag{1}$$

The starting term $\theta_s$ is evaluated in $(-\pi, \pi]$. How to choose the starting edge $\mathbf{b}_s$ is discussed in Section 3.1.1.

*Axis-aligned chart.* An axis-aligned chart is a 2D shape with boundary edges that are parallel to the axes. If a chart $\mathbf{c}$ is already axis-aligned, all $\alpha_i$ are integer multiples of $\pi/2$, and so are all $\theta_i$. Thus, we define integer-valued $K_i$ and $\Theta_i$ as $K_i = 2 - \alpha_i/(\pi/2)$, $\Theta_i = \theta_i/(\pi/2)$. If $K_i \neq 0$, we call $\mathbf{v}_i^b$ a corner of an axis-aligned chart. From (1), we have a recurrence relation for $\Theta_i$:

$$\Theta_{i+1} = \Theta_i + K_i, \quad i = s, s+1, ..., s+N_b-2. \tag{2}$$

*Construction overview.* There are two steps in our construction.

(1) *Angle computation*: For each $\mathbf{b}_i$, we compute its target polar angle $\frac{\pi}{2}\Theta_i$ (Section 3.1.1).
(2) *Deformation*: We drive $\theta_i$ to $\frac{\pi}{2}\Theta_i$ via a mesh deformation while retaining foldover-free properties (Section 3.1.2).

Fig. 5 shows an example. In the step of determining the target polar angles, we first compute the target interior angles. According to the Gauss-Bonnet formula, it requires $\sum_{i=1}^{N_b}(\pi - \alpha_i) = 2\pi$; thus, any modification of $\alpha_i$ should also follow this equation. In addition, it requires $\sum_{i=1}^{N_b} K_i = 4$.

### 3.1.1 Computing target polar angles.

*Boundary Smoothing.* The number of rectangles, which are decomposed from an axis-aligned chart with a lot of corners, is also large, thereby causing long boundary lengths. To obtain an axis-aligned chart with a small number of corners, we smooth the boundary edges to compute the new interior angles $\widehat{\alpha}_i$.

(1) Set the iteration number $k = 1$. Set $\mathbf{s}_i^1$ to the unit direction vector of $\mathbf{b}_i$, and $\widehat{\alpha}_i^1 = \alpha_i$.
(2) For each $\mathbf{b}_i$, we apply the following Gaussian smoothing:

$$G_\sigma(\mathbf{s}_i^k) = \sum_{\mathbf{b}_j} l_j \exp\left(-\frac{\text{dist}(\mathbf{b}_i, \mathbf{b}_j)^2}{2\sigma^2}\right)\mathbf{s}_j^k$$
$$\widehat{\mathbf{s}}_i^k = G_\sigma(\mathbf{s}_i^k)/\|G_\sigma(\mathbf{s}_i^k)\|, \tag{3}$$

where $l_j$ is the length of $\mathbf{b}_j$, $\text{dist}(\mathbf{b}_i, \mathbf{b}_j)$ measures the geodesic distance between the centers of $\mathbf{b}_i$ and $\mathbf{b}_j$ along the boundary, and the parameter $\sigma$ controls the strength of smoothing. In our experiments, we set $\sigma = 0.004 l_{\mathbf{b}}$, where $l_{\mathbf{b}} = \sum_{j=1}^{N_b} l_j$.

(3) We update $\widehat{\alpha}_i$ as follows:

$$\widehat{\alpha}_i^{k+1} = \widehat{\alpha}_i^k + \angle(\mathbf{s}_i^k, \mathbf{s}_i^{k+1}) - \angle(\mathbf{s}_{i+1}^k, \mathbf{s}_{i+1}^{k+1}). \tag{4}$$

To determine the value of $\mathbf{s}_i^{k+1}$, the angle between $\mathbf{s}_i^k$ and $\mathbf{s}_i^{k+1}$ should not be too large so as to avoid a multivalued problem with the angle. Thus, we accept $\widehat{\mathbf{s}}_i^k$ as $\mathbf{s}_i^{k+1}$ if $\widehat{\mathbf{s}}_i^k \cdot \mathbf{s}_i^k \geq 0$; otherwise, we select $\mathbf{s}_i^k$ as $\mathbf{s}_i^{k+1}$. It is easy to prove that this modification ensures $\sum_{i=1}^{N_b}(\pi - \widehat{\alpha}_i^{k+1}) = 2\pi$.

(4) $k := k + 1$. If the maximum change of the interior angles is less than $10^{-4}$ or the maximum iteration number $k_{\max} = 5$ is reached, we stop the smoothing. Otherwise, we go to Step 2.

After smoothing, the smoothed direction of $\mathbf{b}_i$ is denoted as $\mathbf{s}_i$, and the updated interior angle at $\mathbf{v}_i^b$ is denoted as $\widehat{\alpha}_i$.

*Global rotation.* As we know, an axis-aligned chart is related to global rotations. Since the function $\Phi(x, y) = x^2y^2$ is equal to zero when the vector $(x, y)$ is axis-aligned [Fu et al. 2016], we compute a 2D rotation matrix $R$, which makes $\mathbf{c}$ more similar to be axis-aligned by solving the following problem using the L-BFGS method:

$$\min_R \; E_{\text{rotation}}(R) = \sum_{i=1}^{N_b} l_i \Phi(R\mathbf{s}_i). \tag{5}$$

After solving (5), the obtained rotation is applied to $\mathbf{s}_i$ and the chart $\mathbf{c}$. Without a loss of generality, we still denote the rotated chart as $\mathbf{c}$, and the rotated smoothed direction of $\mathbf{b}_i$ as $\mathbf{s}_i$.

*Target polar angles.* In order to compute $\Theta_i$ based on (2), we first compute the target interior angles to determine $K_i$, and then find the starting edge $\mathbf{b}_s$.

We project $\mathbf{s}_i$ to its closest axis direction $\Gamma(\mathbf{s}_i)$, and update $\widehat{\alpha}_i$ as follows:

$$\widehat{\alpha}_i \leftarrow \widehat{\alpha}_i + \angle(\mathbf{s}_i, \Gamma(\mathbf{s}_i)) - \angle(\mathbf{s}_{i+1}, \Gamma(\mathbf{s}_{i+1})). \tag{6}$$

Since $\angle(\mathbf{s}_i, \Gamma(\mathbf{s}_i)) \in [-\pi/4, \pi/4]$, $\sum_{i=1}^{N_b}(\pi - \widehat{\alpha}_i)$ is still equal to $2\pi$. Therefore, the updated $\widehat{\alpha}_i$ (i.e., the target interior angle) is already an integer multiple of $\pi/2$. Then, we compute $K_i = 2 - \widehat{\alpha}_i/(\pi/2)$. We select a boundary edge $\mathbf{b}_i$ as the starting edge $\mathbf{b}_s$ if the angle between $\Gamma(\mathbf{s}_i)$ and the direction of $\mathbf{b}_i$, i.e. $\mathbf{s}_i^1$, is the minimum among all of the boundary edges. Then, $\theta_s$ is evaluated in $(-\pi, \pi]$ to represent the direction of $\mathbf{b}_s$. As we know, $\frac{\pi}{2}\Theta_i$ should represent the same orientation as $\Gamma(\mathbf{s}_i)$. Thus $\Theta_i = 4k_i + \Lambda(\Gamma(\mathbf{s}_i)), k_i \in Z$, where the one-to-one mapping $\Lambda$ is defined as follows:

$$\Lambda((1,0)^T) = 0,\ \Lambda((0,1)^T) = 1,\ \Lambda((-1,0)^T) = 2,\ \Lambda((0,-1)^T) = 3. \tag{7}$$

To compute $\Theta_s = 4k_s + \Lambda(\Gamma(\mathbf{s}_s))$, we should compute $k_s$, which is determined by minimizing $|\theta_s - \frac{\pi}{2}\Theta_s|$. After $\Theta_s$ is computed, we update $\Theta_i$ based on (2).

*Foldover-free requirements.* The foldover-free constraints imply $\widehat{\alpha}_i > 0$, i.e., $K_i \leq 1$ for all $\mathbf{v}_i^b$. According to $\sum_{i=1}^{N_b} K_i = 4$, we change $K_i$ locally while maintaining this equation to ensure $K_i \leq 1$. For a corner $\mathbf{v}_i^b$ with $K_i > 1$, we introduce a *splitting* operation to change $K_i$. This operation first selects $K_i - 1$ non-corner boundary vertices

**ALGORITHM 1:** Axis-aligned chart deformation

**Input** : 3D triangular mesh $\mathbf{M}$ and a chart $\mathbf{c}$
**Output**: A axis-aligned chart of $\mathbf{c}$
$n \leftarrow 1$ ;
$\lambda \leftarrow \frac{E_{\mathrm{d}}(\mathbf{c})}{E_{\mathrm{align}}(\mathbf{c})}$ ;
**while** $(\max_i |\theta_i - \frac{\pi}{2}\Theta_i| \geq \epsilon_a)$ *and* $(n < n_{\max})$ **do**
  $\mathbf{c} \leftarrow$ `DeformChart` via solving (10) by L-BFGS method;
  $n \leftarrow n + 1$ ;
  //$\beta$ is greater than 1;
  $\lambda \leftarrow \beta\lambda$ ;
**end**
$\mathbf{c} \leftarrow$ `FlattenChart`;

adjacent to $\mathbf{v}_i^b$, then modifies the values of their $K$ to 1, and finally sets $K_i = 1$. After modifying $K_i$, we then update $\Theta_i$ according to (2).

We further reduce the number of corners or make the axis-aligned shape more similar to the input chart by changing the values of $K_i$. The details are provided in the supplementary material.

*3.1.2 Deformation.* Our deformation is driven by two energy terms: (i) boundary alignment energy and (ii) isometric distortion energy.

*Boundary alignment energy.* This is defined as follows.

$$\begin{aligned} E_{\mathrm{edge}}(\mathbf{b}_i) &= \frac{1}{2}(1-\gamma)(\theta_i - \frac{\pi}{2}\Theta_i)^2 + \frac{1}{2}\gamma(\frac{l_i}{l_i^0} - 1)^2, \\ E_{\mathrm{align}}(\mathbf{c}) &= \sum_{i=1}^{N_b} \frac{l_i^0}{l^0} E_{\mathrm{edge}}(\mathbf{b}_i), \end{aligned} \tag{8}$$

where $l_i^0$ is the length of $\mathbf{b}_i$ from the input chart, and $l^0 = \sum_{i=1}^{N_b} l_i^0$. $E_{\mathrm{edge}}(\mathbf{b}_i)$ measures the difference between the polar angle and the target angle of $\mathbf{b}_i$, and it requires the edge length $l_i$ to remain close to the input $l_i^0$. The weight $\gamma$ balances the angle term and the length term, and we set $\gamma = 0.3$ in our experiments.

*Isometric distortion energy.* Isometric distortion energy measures the quality of the parameterizations, and we use the symmetric Dirichlet energy [Smith and Schaefer 2015]:

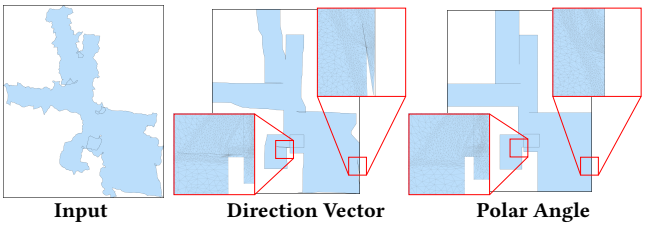
$$E_{\mathrm{d}}(\mathbf{c}) = \frac{1}{4}\sum_{\mathbf{f}_i \in \mathbf{F}^{\mathbf{c}}} \frac{\mathrm{Area}(\mathbf{f}_i)}{\mathrm{Area}(\mathbf{M}^{\mathbf{c}})}(\|J_i\|_F^2 + \|J_i^{-1}\|_F^2), \tag{9}$$

where $\mathbf{F}^{\mathbf{c}}$ is the set of corresponding faces for $\mathbf{c}$ on $\mathbf{M}$, $\mathrm{Area}(\mathbf{M}^{\mathbf{c}})$ is the total area of these faces, and $\|\cdot\|_F$ indicates the Frobenius norm.

*Deformation.* The deformation is solved as a nonlinear constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{c}} \quad & E_{\mathrm{d}}(\mathbf{c}) + \lambda E_{\mathrm{align}}(\mathbf{c}) \\ \text{s.t.} \quad & \det J_i > 0, \forall i, \end{aligned} \tag{10}$$

where $\lambda$ is the weight that increases during the iteration to make $\theta_i$ approach $\frac{\pi}{2}\Theta_i$. The optimization process is shown in Alg. 1. After the optimization in the While loop, the boundary is almost axis-aligned, and we perform a flattening process [Fu et al. 2016] to ensure an exactly axis-aligned output. In our experiments, we set $\epsilon_a = 0.1$, $n_{\max} = 10$, and $\beta = 6$.



**Input** **Direction Vector** **Polar Angle**

Fig. 6. Polar angle vs. direction vector. The deformation using the direction vector representation is trapped, so it fails to generate a valid axis-aligned chart. But, the polar angle representation succeeds.

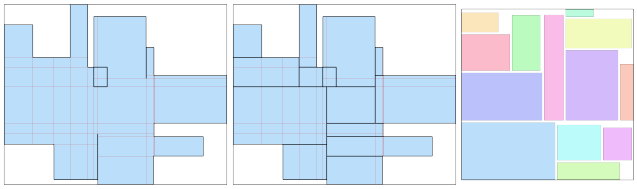
*Polar angle vs. direction vector.* The direction vector representation may create ambiguity. During the deformation process, the boundary edge rotates clockwise or counterclockwise to its target direction vector. In fact, rotating in one of these two directions cannot reach the target direction vector under the foldover-free constraint. The deformation prefers shorter rotations, which may cause the boundary edges to fail to reach their targets, thereby failing to generate a valid axis-aligned chart. On the contrary, if the polar angle representation is used, the boundary edge rotates in only one direction, so there is no ambiguity. Fig. 6 presents a comparison.

## 3.2 Rectangle decomposition and packing

*Naive decomposition.* It is obvious that an axis-aligned shape is easily partitioned into several rectangles. From each corner, we trace the axis directions inside the axis-aligned chart until it arrives at the boundary. There are $1 - K_i$ axis directions that can be traced from a corner $\mathbf{v}_i^b$. These trails and the axis-aligned chart boundary form a 2D quadrilateral mesh, denoted as $\mathcal{Q}$. Each quadrilateral face of $\mathcal{Q}$ is actually a rectangle (Fig. (7) (a)). However, when rectangles are produced in this way, there are usually too many generated.

*Motorcycle graph.* In order to obtain a coarser partition, we use the motorcycle graph algorithm [Eppstein et al. 2008] to partition the 2D quad mesh $\mathcal{Q}$. The goal of this algorithm is to trace some motorcycles starting from the corners and continuing straight along the interior edges until arriving at the boundary or the trail of the other motorcycles in the quad mesh. After all of the motorcycles stop, their trails partition $\mathcal{Q}$ into several rectangular charts (Fig. (7) (b)).

It is not necessary to assign a motorcycle for each of the interior edges of a corner, as long as for each two consecutive interior edges of a corner at least one edge is assigned [Eppstein et al. 2008]. In this way, fewer rectangles are generated while we can still guarantee



**(a)** Quad Mesh **(b)** Decomposition **(c)** Packing

Fig. 7. The pipeline of rectangle decomposition and packing. The input axis-aligned shape is from Fig. 5. (a) Naive decomposition, where the red lines form the quad mesh $\mathcal{Q}$. (b) Decomposed rectangles with boundaries colored in black. (c) Packed rectangles with high packing efficiency, 87.0%, and a high evaluation score, 0.688.

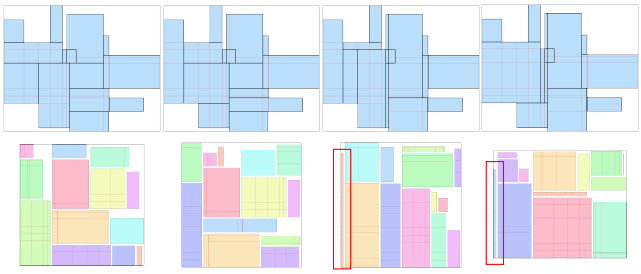


Fig. 8. The input axis-aligned shape is also from Fig. 5. Its rectangle decomposition is not unique. Here, we show four other cases not including the one from Fig. 7. From left to right: PE = 83.6%, 84.4%, 83.9%, 87.7%, $E_{\text{motor}}$ = 0.659, 0.658, 0.637, 0.696. Note that we do not select the right two results since they have thin rectangles (shown in red boxes).

that all of the charts are rectangles. Therefore, for a corner with $n$ interior edges in $\boldsymbol{Q}$, we need at least $\lfloor n/2 \rfloor$ motorcycles, and at most $n$ motorcycles. So, we can generate various motorcycle graphs for $\boldsymbol{Q}$ (Fig. (8)).

*Rectangle packing.* After decomposing all of the axis-aligned shapes, we first collect all the rectangles and then quickly and robustly pack them with high efficiency using an existing rectangle packing algorithm [TeamHypersomnia 2018]. All of the rectangles are slightly padded to leave space for further distortion reduction.

Since the motorcycle graph is not unique, and both the motorcycle graph algorithm and the rectangle packing algorithm are very fast, we generate $N_{\text{motor}}$ random decomposition candidates and select one based on the following evaluation score, while guaranteeing that the packing efficiency is greater than $\text{PE}_{\text{bound}}$:

$$E_{\text{motor}} = \text{PE} - \omega \frac{\text{BL}_1}{\text{BL}_0}. \tag{11}$$

For one candidate, PE, $\text{BL}_0$, and $\text{BL}_1$ denote the packing efficiency and the boundary length before and after decomposition, respectively. Since the long and narrow rectangles are likely to lead to poor results, we discard the decomposition candidates containing any rectangle with an aspect ratio larger than 20 (the right two in Fig. (8)). Then, we choose the candidate with the highest score. We set $\omega = 0.1$ and $N_{\text{motor}} = 500$ in our experiments. Some of the decomposition candidates may be duplicates. The initial width of the padding space is set as 0.25% of the diagonal length of the bounding box of the input atlas.

If the packing efficiencies of all candidates are lower than $\text{PE}_{\text{bound}}$, we reduce the padding space when packing rectangles, or sample some non-corner vertices and regard them as corners for generating $\boldsymbol{Q}$ and motorcycle graphs to get decompositions with more rectangles. Since the packing efficiency can be very close to 100% in an extreme case where $\boldsymbol{Q}$ is decomposed to very tiny rectangles and the padding space is very small, the packing efficiency bound is guaranteed to be achieved.

*Cutting.* According to the selected decomposition candidate, we cut each triangular axis-aligned chart along the trails of its corresponding motorcycle into a set of parameterized rectangle charts. At the same time, $\mathbf{M}$ is also cut to correspond to the parameterized charts. Then, we put together the parameterized charts based on the packed rectangles to generate an atlas, denoted as $\widehat{\mathbf{C}}$ (Fig. (7) (c)).

## 3.3 Distortion reduction

*Goal and formulation.* Since the parameterization distortion increases in the axis-aligned chart construction process, it should be optimized while maintaining bijection and bounding packing efficiency. This task can be formulated as a nonlinear constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{C}} \quad & E_{\text{d}}(\mathbf{C}), \\ \text{s.t.} \quad & \Phi \text{ is bijective}, \\ & \text{PE}(\mathbf{C}) \geq \text{PE}_{\text{bound}}. \end{aligned} \tag{12}$$

where $\mathbf{C}$ indicates the parameterized charts and $\Phi$ denotes the parameterization.

*Reformulation.* The parameterized charts after the rectangle packing step satisfy the constraints of (12). Thus, we start from this stage and ensure that $\mathbf{C}$ remains in the feasible solution space. A barrier term is used to ensure that $\text{PE}(\mathbf{C})$ is always greater than $\text{PE}_{\text{bound}}$:

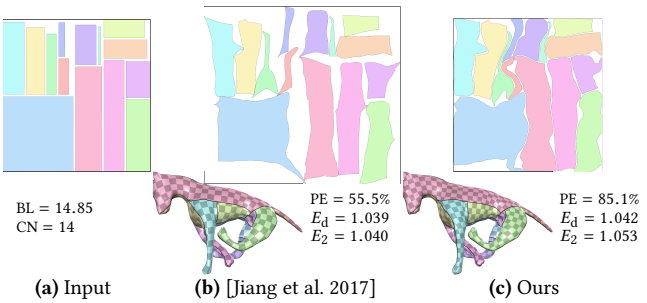
$$E_{\text{PE}}(\mathbf{C}) = \begin{cases} \max\left\{0, \dfrac{\epsilon_{\text{PE}}}{\text{PE}(\mathbf{C}) - \text{PE}_{\text{bound}}} - 1\right\}, & \text{if } \text{PE}(\mathbf{C}) > \text{PE}_{\text{bound}}, \\ +\infty, & \text{otherwise.} \end{cases} \tag{13}$$

When $\text{PE}(\mathbf{C}) \geq \text{PE}_{\text{bound}} + \epsilon_{\text{PE}}$, $E_{\text{PE}}(\mathbf{C}) = 0$. Then, we solve following problem to achieve our goal:

$$\begin{aligned} \min_{\mathbf{C}} \quad & E_{\text{reduction}} = E_{\text{d}}(\mathbf{C}) + E_{\text{PE}}(\mathbf{C}), \\ \text{s.t.} \quad & \Phi \text{ is bijective}, \end{aligned} \tag{14}$$

To maintain the bijective property and reduce $E_{\text{d}}$, we use the scaffold-based method [Jiang et al. 2017]. The initializations of (14) include a scaffold bounding box that is required by [Jiang et al. 2017] and an initial atlas.

*Scaffold bounding box.* The scaffold bounding box is computed by conducting uniform scaling three times for the bounding box of the image from the current parameterization in [Jiang et al. 2017]. In our problem, this treatment is unsuitable and may cause the packing efficiency to quickly fall below the bound (see Fig. 9). Thus, we fix a scaffold bounding box that is slightly bigger than the bounding box of $\mathbf{C}$ during the optimization of (14).



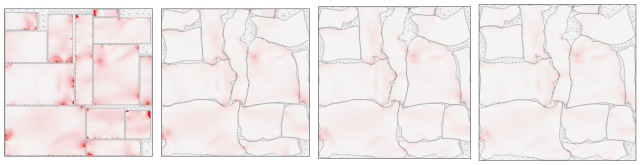

**(a)** Input **(b)** [Jiang et al. 2017] **(c)** Ours

Fig. 9. Comparison to [Jiang et al. 2017]. From a common input (a), [Jiang et al. 2017] uses a large initial bounding box, and their result has low packing efficiency (b). Our method uses incremental bounding boxes, and produces high packing efficiency (c). Both methods stop when the symmetric Dirichlet distortion is less than the input distortion.

**ALGORITHM 2:** Distortion reduction

**Input** : 3D triangular mesh **M** and a chart $\widehat{\mathbf{C}}$
**Output**: A bijective **C** with bounded packing efficiency

$s_i \leftarrow \min\{\frac{\text{Area}(\mathbf{c}_i^{\mathbf{M}})}{\text{Area}(\mathbf{c}_i)}, i = 1, \ldots, N_c\}$ ;
$\mathbf{C} \leftarrow$ `ScaleAtlas`$(\widehat{\mathbf{C}}, s_i)$;
$(w, h) \leftarrow$ `GetBoundingBoxSize`$(\mathbf{C})$;
$(w, h) \leftarrow (\delta w, \delta h)$; // $\delta$ is slightly greater than 1;
`SetScaffoldBoundingBoxSize`$(w, h)$;
$k \leftarrow 1$;
**repeat**
  $\mathbf{C} \leftarrow$ `DistortionReductionWithFixedScaffold`;
  $(w, h) \leftarrow (w, h) + (\Delta\delta w, \Delta\delta h)$; // $\Delta\delta$ is slightly greater than 0;
  `SetScaffoldBoundingBoxSize`$(w, h)$;
  $k \leftarrow k + 1$;
**until** $\frac{\left|E^k_{\text{reduction}} - E^{k-1}_{\text{reduction}}\right|}{E^{k-1}_{\text{reduction}}} < 10^{-4}$ *or* $E_d \le E_d^{input}$;



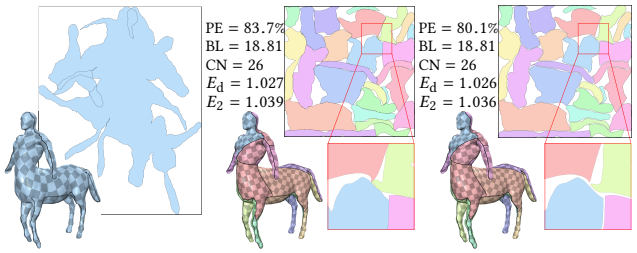
**(a)** **(b)** **(c)** **(d)**

Fig. 10. Progressive distortion reduction results on the Aramdilo model (Fig. 1 - left) whose initial atlas is shown in Fig. 7 (c). (a) Initial atlas (PE = 85.0%, $E_\text{d}$ = 1.139). (b) After the first iteration (PE = 87.4%, $E_\text{d}$ = 1.058). (c) After the seventh iteration (PE = 86.0%, $E_\text{d}$ = 1.044). (d) Final result (PE = 84.0%, $E_\text{d}$ = 1.036) after 14 iterations.

*Initial atlas.* A straightforward approach is to use $\widehat{\mathbf{C}}$ as the initial atlas. However, most parameterized charts may shrink to reduce isometric distortion, thereby lowering the packing efficiency. Since the packing efficiency is scale-independent, one possible solution is to zoom out $\widehat{\mathbf{C}}$ so that most charts are likely to expand to increase the packing efficiency while optimizing an isometric energy.

*Practical solutions.* Based on the above two considerations, we propose a practical solution for solving (14), whose pseudocode is shown in Alg. 2. Fig. 10 shows the progressive atlas results. The initial scale $s_i$ ensures that most charts contain a smaller area than the source area. Therefore, they may expand during the distortion optimization in order to have the potential to increase the packing efficiency. Similarly, as the size of the scaffold bounding box slowly increases, the charts compress each other, which is likely to improve packing efficiency. Here we provide another termination condition: when $E_\text{d}(\mathbf{C}) \le E_\text{d}^\text{input}$, where $E_\text{d}^\text{input}$ is the distortion of the input charts, we also stop the algorithm. When $\text{PE}(\mathbf{C}) \ge \text{PE}_\text{bound} + \epsilon_\text{PE}$, $E_\text{PE}(\mathbf{C}) = 0$, and we solve (14) by [Liu et al. 2018] with explicit checks in combination with line searches for bounding packing efficiency; otherwise, the L-BFGS solver is used. In our experiments, we set $\delta = 1.005$, $\Delta\delta = 0.005$, and $\epsilon_\text{PE} = 0.0001$. In practice, these empirical techniques work well.

*Gaps between charts.* Alg. 2 often produces atlases, where some charts are too close. When charts lack adequate space, they may not be acceptable for subsequent applications. To that end, we add a thin scaffold triangle layer with a fixed width to the outside of each chart to form a new chart before running Alg. 2 (Fig. 11). During the optimization, the new charts are used for Alg. 2, and the added layers prevent the charts from coming too close. Note that the reserved space in the rectangle packing should include the newly added layers. Although this treatment may decrease the packing efficiency, we still include it in our algorithm by default. Fig. 11 shows a comparison between the optimizations with and without this treatment.




**(a)** Initial **(b)** Without Gaps **(c)** With Gaps

Fig. 11. Gaps between charts. The resulting atlas without gaps between charts usually produces higher packing efficiency. However, charts that are too close to each other may have a significant negative impact on subsequent applications.

## 3.4 Discussions

*Padding space.* In the rectangle packing phase, we set the padding space to be a fixed value. However, it is possible to specify a chart-dependent padding space, e.g., the regions with higher distortion get more padding space. Since the distortion reduction makes the charts compress each other, the chart-dependent and fixed padding spaces may only have slight difference on the final results.

*Packing algorithm.* The packing algorithm we choose is very fast and the result is already good enough, so we can run it for 500 candidates and choose the one with highest score. If other rectangle packing algorithm could provide a better tradeoff than the currently used one, we can use it here.

*Distortion type in distortion reduction.* In practice, users may want to minimize the distortion between the input and the resulting atlases, our reduction step can be easily modified to optimize this energy instead of the distortion with respect to the input 3D surface. In fact, the energy used in our reduction step can be customized by users (see Fig. 18). If the specified energy cannot prevent the foldovers, an extra barrier term should be added to ensure bijection.

# 4 EXPERIMENTS

Our method generates bounded packing efficiency for atlases with strong practical robustness, and we apply it to various atlases. We select the Box Cutter [Limper et al. 2018] as the competitor. We report the timings and the atlas quality statistics, which are defined below. Our experiments were performed on a desktop PC with a 4.0 GHz Intel Core i7-4790K and 16 GB of RAM. We use the Intel® Math Kernel Library for the linear solve in our method.

*Quality metrics.* The commonly used metrics, including the packing efficiency, the boundary length, the number of charts (denoted

as CN), and the distortion metric, are used to measure the quality of an atlas. For the packing efficiency (denoted as PE), we measure the ratio between the areas of the atlas and its bounding box. For the boundary length (denoted as BL), we measure the ratio between the length of the cut on $\mathbf{M}$ and the diagonal length of the bounding box of $\mathbf{M}$. And we use the symmetric Dirichlet energy $E_d$ with respect to the input surface as a distortion metric. As we modify the parameterized atlas, the symmetric Dirichlet energy (denoted as $E_2$) from the input atlas to the resulting atlas is also reported.

The charts with very small areas may increase the difficulty of texture editing. Thus, we also include two metrics regarding areas: (i) the smallest area (denoted as SA), which is the ratio between the areas of the smallest chart and $\mathbf{M}$, and (ii) the number of charts with an area less than 3% the area of $\mathbf{M}$, denoted as $N_{\leq 3\%}$.

In the signal storage context, the signal on surface is reconstructed from a texture using the parameterized atlas. Therefore, the error between this reconstructed signal and the ground truth signal, which is directly defined on the surface, should be as small as possible. To measure this error, we compute the *signal approximation error* (denoted as SAE) as [Sander et al. 2002] on a $4096 \times 4096$ texture. For general cases, we define a high-frequency signal on the surface of $\mathbf{M}$ using a 3D procedural checkerboard pattern: the bounding box of $\mathbf{M}$ is uniformly divided into a $100 \times 100 \times 100$ 3D black and write checkerboard, and the color signal of each point on the surface of $\mathbf{M}$ is defined as the color of the checkerboard at this point.

## 4.1 Evaluations and comparisons

*Different $\sigma$s in Eq. (3).* The $\sigma$ in Eq. (3) controls the balance between the number of corners in an axis-aligned chart and the parameterization distortion. A larger $\sigma$ usually results in fewer corners, fewer rectangles, and shorter boundary lengths. At the same time, a larger $\sigma$ causes higher distortion and lower packing efficiency. As shown in Fig 12, three different $\sigma$s are used for a Camel model. To achieve a resulting atlas with balanced quality metrics, we set $\sigma = 0.004 l_{\mathrm{b}}$ by default. From all of the testing examples, it is apparent that the selected $\sigma$ works well.

*Packing efficiency bounds.* An example with various packing efficiency bounds is shown in Fig. 13. The packed rectangles have a packing efficiency of 90.7%. Then, for each bound, we run our final distortion reduction until the packing efficiency reaches the

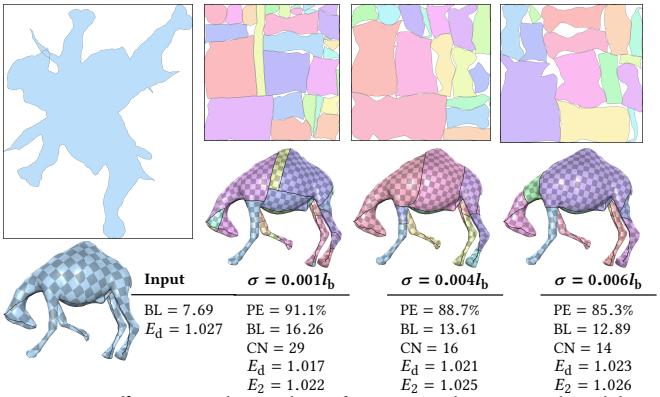


Fig. 12. Different $\sigma$s. Three values of $\sigma$ are tested on a Camel model.

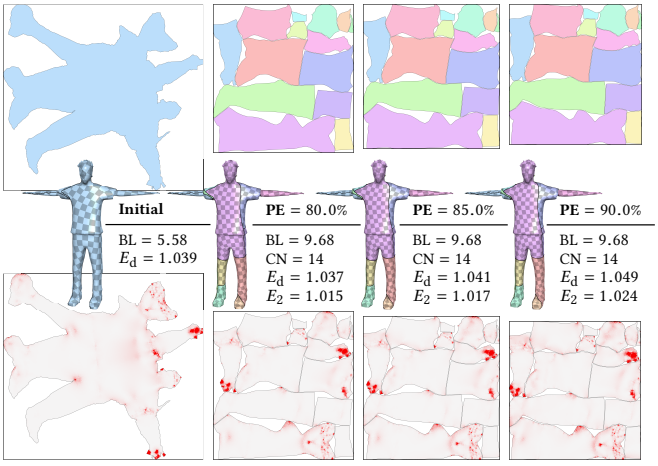


Fig. 13. Various bounds for the packing efficiency. We tested three bounds, i.e., 80%, 85%, and 90%.

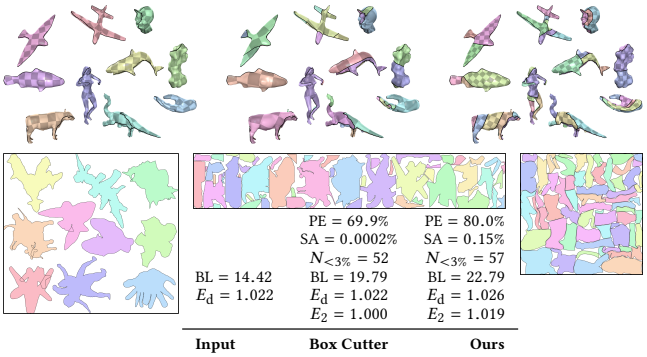


Fig. 14. A collection of models. The Box Cutter results were generated by an executable file that was provided by the authors.

set bound. That is to say, the distortion termination condition is disabled in this example. The higher the packing efficiency bound is set, the greater the distortion in the resulting atlas is. However in this example, the slight distortion differences cause almost no visible textural changes.

If users set a very large packing efficiency bound (e.g., 95%), we should divide the rectangles into smaller and smaller ones to satisfy it; however, this causes a very long cut and a lot of small charts. Thus, to achieve a trade-off between the boundary length and the packing efficiency, we set $\mathrm{PE}_{\mathrm{bound}} = 80\%$ by default.

*A collection of models.* We generate one atlas for a set of models. In fact, these models are considered as one model with separate parameterized charts. Thus, our method first deforms all input charts to axis-aligned shapes, then decomposes each axis-aligned shape into rectangles, and finally collects all rectangles for packing and distortion optimization. In Fig 14, ten models are considered, and a comparison to Box Cutter is also performed. Our resulting atlas contains bounded packing efficiency. The resulting packing efficiency of Box Cutter is 69.9%, which is much smaller than ours.

*Comparison to Box Cutter [Limper et al. 2018].* Here, we use the models provided by Box Cutter for comparison. Since gaps are not addressed in these models, we also do not include gaps between

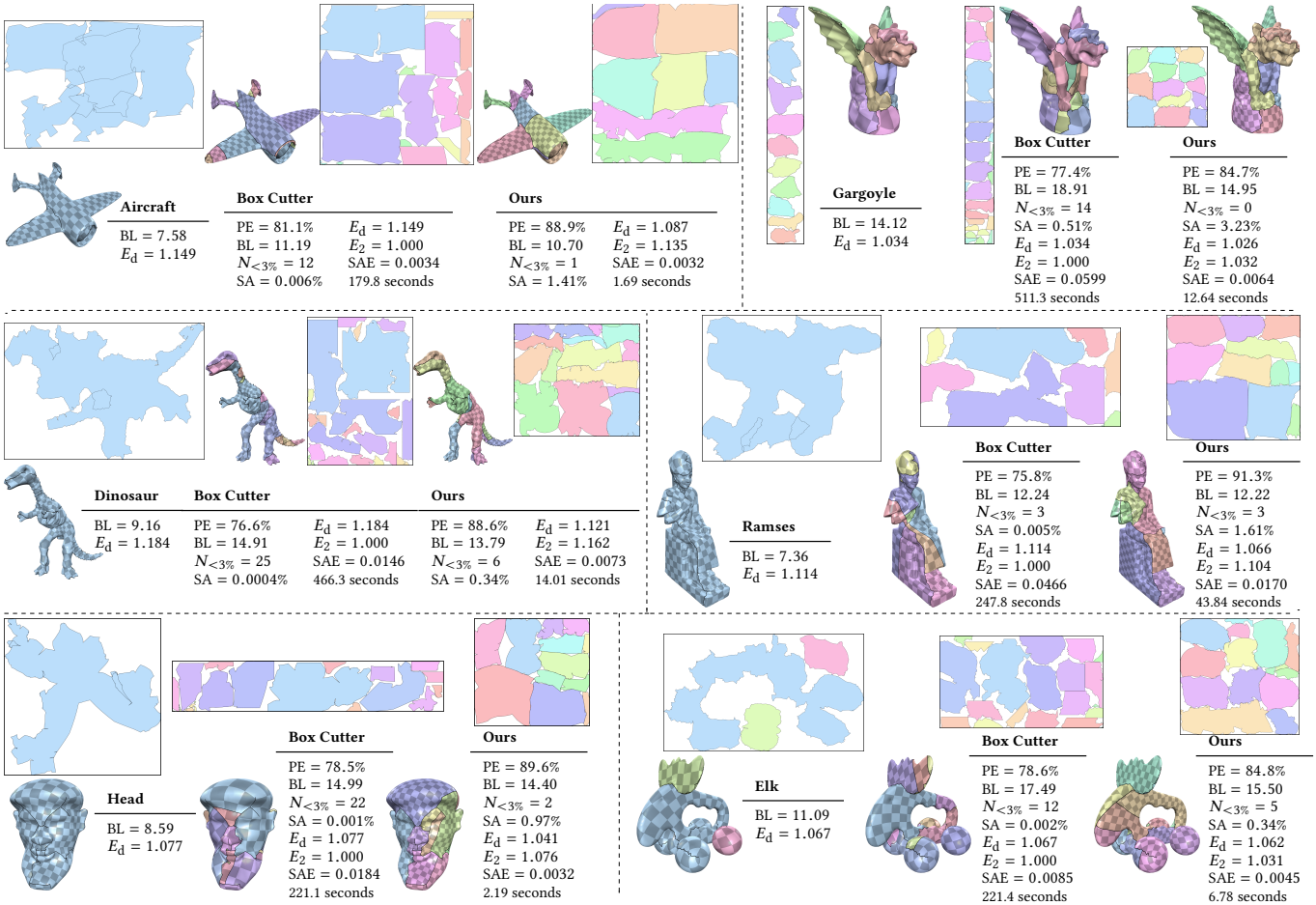



Fig. 15. Comparison to [Limper et al. 2018]. Six examples are used for comparisons. The results of Box Cutter are provided by the authors. Judging from the reported times, our method is much faster than Box Cutter.

charts for fair comparisons. In Fig. 15, six examples are used for comparison and the comparisons on 55 models are included in the supplementary material. Compared to Box Cutter, our method achieves higher packing efficiency with similar boundary length elongation. In practice, Box Cutter may be trapped by a local minimum while improving the packing efficiency; so, the same packing efficiency may be obtained for different boundary elongation constraints (Fig. 2 and Fig. 14). However, our method ensures that the packing efficiency is bounded. Due to the high packing efficiency, both methods produce small signal approximation errors.

To improve the packing efficiency with boundary length elongation constraints, Box Cutter tends to generate short cuts on the input charts. Although the boundary length is constrained, it usually results in a lot of small charts, as indicated by the metrics SA and $N_{\leq 3\%}$. Box Cutter can prevent small charts in the cut-and-pack step, but there is no such prevention in the overlap removal step. Since the overlap removal process aims to find an overlap-free result while minimizing the cut length, it may produce very small charts (see the right column in Fig. 16 of [Limper et al. 2018]). But, the charts generated by our method are of a much more uniform size.

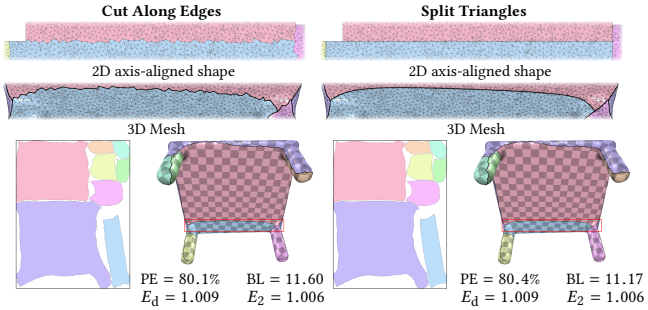
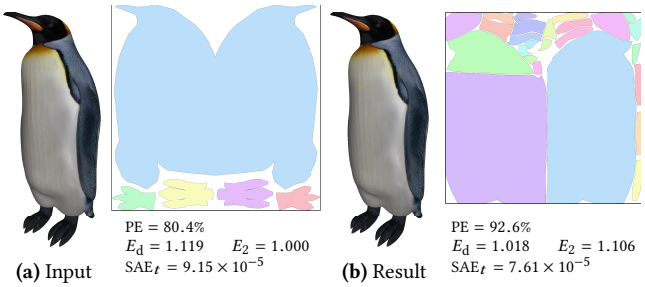



Fig. 16. Cutting along the existing edges. The cuts across the triangles result is slightly better atlases than the cuts along the edges.

*Cutting along the existing edges.* By default, we cut the parameterized and input meshes by subdividing triangles traversed by the trails of the selected motorcycle (Fig. 16-right). However, one may want to keep the connectivity of the input mesh. Then, we need to cut the mesh along the edges (Fig. 16-left). To that end, we use the shortest edge paths to replace the straight lines. After these shortest edge paths are detected, we use Tutte's embedding method to map each chart to its corresponding rectangle. Then, we can still use

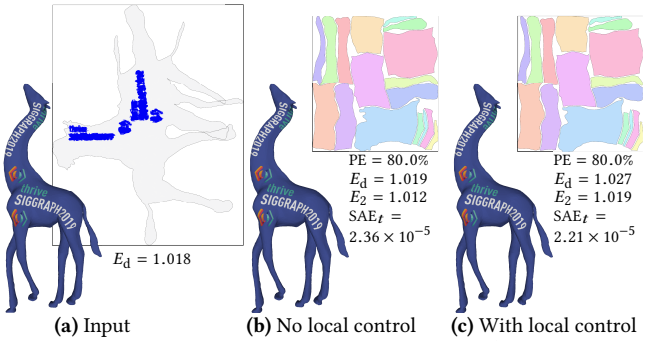



**(a)** Input **(b)** Result

Fig. 17. Signal approximation error. (a) Input textured Penguin model. (b) Our result. The size of the input texture is $4096 \times 4096$.

the rectangle packing algorithm to generate high initial packing efficiency and our distortion reduction technique for further optimization. In Fig. 16, we show a comparison in which the resulting atlases are similar.

*Signal approximation error.* Our optimization of the geometric distortion between the atlas and the input surface in the distortion reduction step will also change the signal approximation error. In Fig. 17, given a textured model, we refine its atlas to improve packing efficiency and measure the signal approximation error. The signal approximation error of the color signal (denoted as $SAE_t$) on our refined atlas is small. In practice, the low symmetric Dirichlet energy between the resulting atlas and the input surface indicates that the atlas reserves isometry to its original shape as much as possible. Besides, the higher the packing efficiency is, the more pixels are used to reconstruct the signal. Therefore, we conjecture that low symmetric Dirichlet energy and high packing efficiency may imply small signal approximation error.




**(a)** Input **(b)** No local control **(c)** With local control

Fig. 18. Local control. (a) Given an input textured Giraffe model (left), we compute the importance map (right), which is encoded by color with deep blue being the most important. (b) Without local control. (c) With local control. The size of the input texture is $4096 \times 4096$.

*Local control.* In practice, not all charts are equally important. Thus, sometimes the distortion reduction step should provide local control of the charts. Here, we first define importance levels on the input charts according to the input texture and then add importance weights to the triangles when reducing the distortion. The more important, the larger the weight. Fig. 18 shows an example. The symmetric Dirichlet energy with local control is slightly higher than the resulting energy without local control, and the signal approximation error of this specific texture $SAE_t$ with local control is smaller. Although the two resulting signal approximation errors are different, both of them are very small, indicating that there is almost no visual difference between the textured models.

Table 1. Statistics and timings for atlas refinement. The timings for the axis-aligned chart construction, the axis-aligned chart decomposition and rectangle packing, and the distortion reduction are denoted as $t_1$, $t_2$, and $t_3$, respectively.

| Model | Input | | | | Quality metrics | | | | Timings (s) |
|---|---|---|---|---|---|---|---|---|---|
| Name | $N_f$ | BL | $E_d$ | SAE | PE | BL/CN/SA/$N_{\leq 3\%}$ | $E_d/E_2$ | SAE | $t_1/t_2/t_3/t_{\text{total}}$ |
| Fig. 1 Armadillo | 18,894 | 6.34 | 1.040 | 0.0100 | 85.0% | 11.45/13/0.90%/3 | 1.030/1.031 | 0.0052 | 2.59/0.16/2.70/5.45 |
| Fig. 1 Blade | 58,546 | 9.95 | 1.038 | 0.0417 | 87.2% | 14.24/11/2.39%/3 | 1.022/1.050 | 0.0115 | 15.99/0.13/14.01/30.13 |
| Fig. 2 Bunny | 69,451 | 16.30 | 1.010 | 0.0321 | 85.8% | 21.24/26/0.19%/12 | 1.007/1.013 | 0.0118 | 12.32/0.25/40.42/52.99 |
| Fig. 4 Plane | 15,298 | 5.00 | 1.027 | 0.0070 | 81.4% | 9.33/10/1.87%/1 | 1.025/1.025 | 0.0046 | 2.14/0.14/3.04/5.32 |
| Fig. 9 Cheetah | 17,598 | 8.29 | 1.061 | 0.0141 | 85.1% | 14.85/14/0.79%/5 | 1.042/1.053 | 0.0071 | 2.67/0.15/4.38/7.20 |
| Fig. 11 Centaur-nogap | 18,868 | 10.45 | 1.037 | 0.0135 | 83.7% | 18.81/26/0.25%/11 | 1.027/1.039 | 0.0055 | 3.37/0.22/5.83/9.42 |
| Fig. 11 Centaur-gap | 18,868 | 10.45 | 1.037 | 0.0135 | 80.1% | 18.81/26/0.25%/11 | 1.026/1.036 | 0.0057 | 3.37/0.22/8.72/12.31 |
| Fig. 12 Camel1-0.001 | 18,816 | 7.69 | 1.027 | 0.0131 | 91.1% | 16.26/29/0.07%/19 | 1.017/1.022 | 0.0047 | 3.82/0.20/8.58/12.60 |
| Fig. 12 Camel1-0.004 | 18,816 | 7.69 | 1.027 | 0.0131 | 88.7% | 13.61/16/0.81%/5 | 1.021/1.025 | 0.0049 | 6.26/0.14/14.32/20.72 |
| Fig. 12 Camel1-0.006 | 18,816 | 7.69 | 1.027 | 0.0131 | 85.3% | 12.89/14/0.72%/4 | 1.023/1.026 | 0.0051 | 3.98/0.12/11.60/15.70 |
| Fig. 13 Boy-80 | 34,036 | 5.58 | 1.039 | 0.0177 | 80.0% | 9.68/14/0.26%/5 | 1.037/1.015 | 0.0099 | 6.56/0.13/30.67/37.36 |
| Fig. 13 Boy-85 | 34,036 | 5.58 | 1.039 | 0.0177 | 85.0% | 9.68/14/0.26%/5 | 1.041/1.017 | 0.0094 | 6.56/0.13/22.02/28.71 |
| Fig. 13 Boy-90 | 34,036 | 5.58 | 1.039 | 0.0177 | 90.0% | 9.68/14/0.26%/5 | 1.049/1.024 | 0.0090 | 6.56/0.13/7.45/14.14 |
| Fig. 14 Collection | 106,058 | 14.42 | 1.022 | 0.0238 | 80.0% | 22.79/64/0.15%/57 | 1.026/1.019 | 0.0119 | 19.80/0.60/73.11/93.51 |
| Fig. 15 Aircraft | 4,656 | 7.58 | 1.149 | 0.0050 | 88.9% | 10.70/8/1.41%/1 | 1.050/1.135 | 0.0032 | 0.81/0.06/0.82/1.69 |
| Fig. 15 Gargoyle | 20,000 | 14.12 | 1.034 | 0.0573 | 84.7% | 14.95/13/3.23%/0 | 1.025/1.032 | 0.0064 | 2.88/0.10/9.66/12.64 |
| Fig. 15 Dinosaur | 28,136 | 9.16 | 1.184 | 0.0257 | 88.6% | 13.79/16/0.34%/6 | 1.049/1.162 | 0.0073 | 7.91/0.14/5.96/14.01 |
| Fig. 15 Ramses | 100,000 | 7.36 | 1.114 | 0.0369 | 91.3% | 12.22/11/1.61%/3 | 1.036/1.104 | 0.0170 | 31.49/0.19/12.16/43.84 |
| Fig. 15 Head | 7,232 | 8.59 | 1.077 | 0.0056 | 89.6% | 14.40/12/0.97%/2 | 1.032/1.076 | 0.0032 | 0.99/0.08/1.12/2.19 |
| Fig. 15 Elk | 10,387 | 11.09 | 1.067 | 0.0120 | 84.8% | 15.50/15/0.34%/5 | 1.062/1.031 | 0.0045 | 1.90/0.10/4.78/6.78 |
| Fig. 16 Desk-edges | 37,394 | 6.65 | 1.010 | 0.0232 | 80.1% | 11.60/8/1.76%/3 | 1.009/1.006 | 0.0118 | 6.22/0.09/28.37/34.68 |
| Fig. 16 Desk-splitting | 37,394 | 6.65 | 1.010 | 0.0232 | 80.4% | 11.17/8/1.76%/3 | 1.009/1.006 | 0.0120 | 6.22/0.10/29.53/35.85 |
| Fig. 17 Penguin | 14,976 | 7.53 | 1.119 | 0.0067 | 92.6% | 12.84/25/0.21%/22 | 1.018/1.106 | 0.0050 | 2.48/0.16/9.13/11.77 |
| Fig. 18 Giraffe-noLC | 19,994 | 8.01 | 1.018 | 0.0266 | 80.0% | 12.54/17/0.63%/7 | 1.019/1.012 | 0.0088 | 2.84/0.16/9.38/12.38 |
| Fig. 18 Giraffe-LC | 19,994 | 8.01 | 1.018 | 0.0266 | 80.0% | 12.54/17/0.63%/7 | 1.027/1.019 | 0.0089 | 2.84/0.16/9.21/12.21 |
| Fig. 19 Orangutan | 18,560 | 7.14 | 1.034 | 0.0106 | 86.7% | 12.10/10/2.97%/1 | 1.024/1.024 | 0.0068 | 2.78/0.10/4.91/7.79 |
| Fig. 19 Frogbuddha | 27,236 | 5.92 | 1.021 | 0.0099 | 83.6% | 12.37/12/0.56%/6 | 1.019/1.013 | 0.0072 | 4.16/0.11/15.53/19.80 |
| Fig. 19 Lamp | 28,174 | 6.10 | 1.026 | 0.0219 | 86.2% | 9.82/8/4.37%/0 | 1.020/1.016 | 0.0091 | 3.82/0.10/5.82/9.74 |
| Fig. 19 Turboslot | 22,520 | 15.15 | 1.013 | 0.0235 | 91.2% | 27.59/16/0.32%/6 | 1.008/1.011 | 0.0122 | 2.82/0.13/4.53/7.48 |
| Fig. 19 Dog | 18,230 | 7.22 | 1.044 | 0.0089 | 87.1% | 12.88/19/0.41%/8 | 1.027/1.036 | 0.0048 | 2.59/0.15/4.31/7.05 |
| Fig. 19 Octopus | 87,992 | 12.60 | 1.023 | 0.0297 | 85.9% | 18.51/25/0.04%/15 | 1.019/1.018 | 0.0140 | 28.73/0.27/31.32/60.32 |
| Fig. 19 Mermaid | 25,994 | 6.46 | 1.028 | 0.0200 | 85.4% | 10.67/12/1.55%/3 | 1.021/1.019 | 0.0076 | 3.76/0.11/7.50/11.37 |
| Fig. 19 Mechapart | 2,998 | 8.53 | 1.014 | 0.0027 | 90.5% | 17.87/10/1.46%/2 | 1.011/1.009 | 0.0021 | 0.38/0.08/1.01/1.47 |
| Fig. 22 Ramses | 100,000 | 7.36 | 1.114 | 0.0369 | 89.1% | 12.22/11/1.61%/3 | 1.041/1.105 | 0.0175 | 31.49/0.19/10.22/41.90 |
| Fig. 23 Fender | 122,510 | 5.75 | 1.001 | 0.0277 | 80.0% | 10.14/19/0.31%/11 | 1.003/1.004 | 0.0115 | 152.31/0.27/145.58/298.16 |

## 4.2 Practical robustness

*Data set.* To verify the practical robustness of our method, we run our algorithm on a data set containing 5,588 models. This data set is provided by [Liu et al. 2018], and the input parameterizations have low isometric distortions, i.e., $E_d < 1.100$ for any model (Fig. 20 (a)). For all models, we set $\sigma = 0.004 l_{\mathbf{b}}$ and the packing efficiency bound as 80%. The gaps between charts are enabled by default.

*Results.* Our method succeeds in generating bounded packing efficiency atlases for all models. The maximum, average, and standard deviation of the packing efficiency for all the meshes are 97.3%, 82.3% and 0.0295, respectively. These results indicate the practical robustness of our method. We show eight models in Fig. 19. Table 1 summarizes the results.

*Distortion statistics of results.* Our method may increase distortion in some examples. We measure the distortion increase as $\Delta E = (E_d^{\text{result}} - E_d^{\text{input}})/E_d^{\text{input}}$, where $E_d^{\text{input}}$ is the symmetric Dirichlet energy of the input parameterization, and $E_d^{\text{result}}$ indicates the symmetric Dirichlet energy of the resulting atlas. The histograms in Fig. 20 show the distributions of the isometric distortion. The average increase is only 0.0533%, indicating that the resulting atlases and input parameterizations have similar isometric distortions. In addition, the left two histograms in Fig. 20 also demonstrate the high similarity between $E_d^{\text{result}}$ and $E_d^{\text{input}}$. Moreover, the rightmost

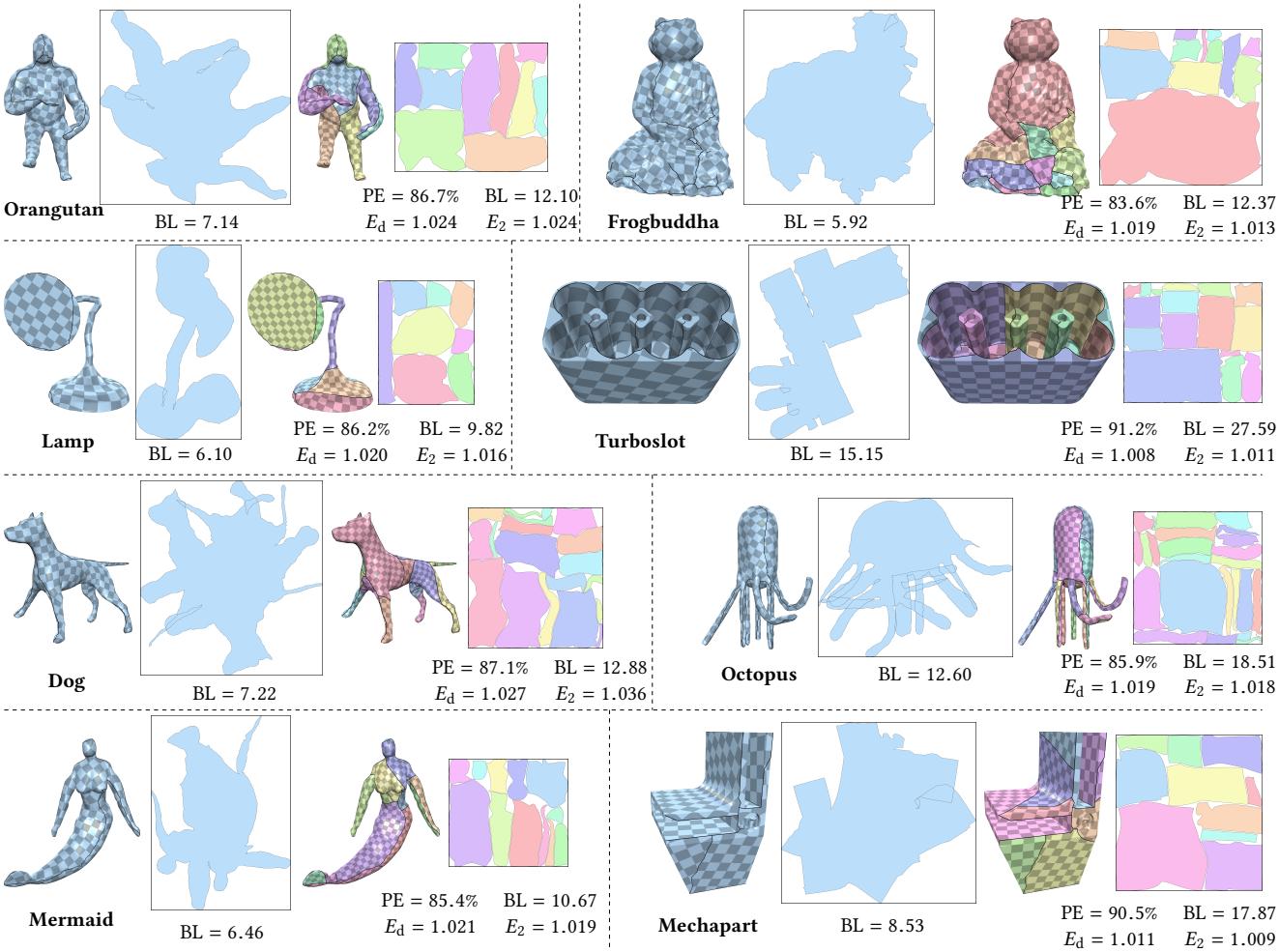



Fig. 19. Gallery. Our method succeeds in generating atlases with bounded packing efficiency.

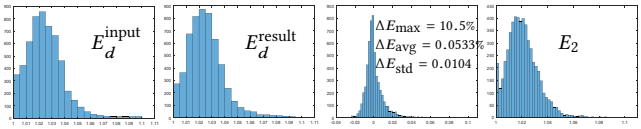



Fig. 20. Distributions of isometric distortion. $\Delta E_{\max}$, $\Delta E_{\text{avg}}$, and $\Delta E_{\text{std}}$ are the maximum, average, and standard deviation of $\Delta E$ for all the meshes, respectively.

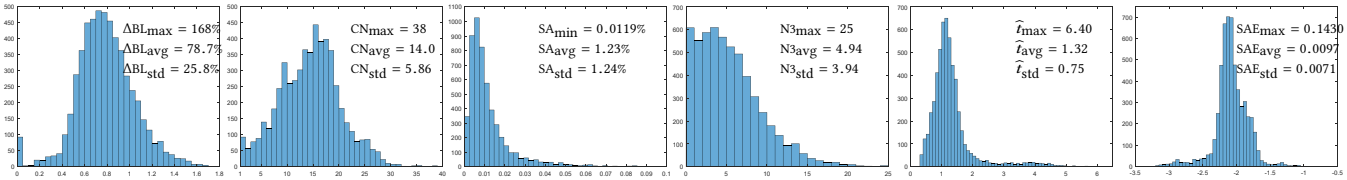
histogram in Fig. 20 shows the distributions of $E_2$, and the maximum, average, and standard deviation of $E_2$ are 1.106, 1.021, 0.012. It indicates that the deformation of the input atlas is slight.

*Statistics data.* We measure the boundary length elongation as $\Delta\text{BL} = (\text{BL}^{\text{result}} - \text{BL}^{\text{input}})/\text{BL}^{\text{input}}$. Since each input model in the data set only has one chart, we record the resulting chart number. Furthermore, we also measure $N_{\leq 3\%}$, SA and SAE for each mesh. We report the above quality metrics, including the worse case, average, and standard deviation for all meshes. This is denoted as $\Delta\text{BL}_{\max}$, $\Delta\text{BL}_{\text{avg}}$, $\Delta\text{BL}_{\text{std}}$, $\text{CN}_{\max}$, $\text{CN}_{\text{avg}}$, $\text{CN}_{\text{std}}$, $\text{N3}_{\max}$, $\text{N3}_{\text{avg}}$, $\text{N3}_{\text{std}}$, $\text{SA}_{\max}$, $\text{SA}_{\text{avg}}$, $\text{SA}_{\text{std}}$, and $\text{SAE}_{\max}$, $\text{SAE}_{\text{avg}}$, $\text{SAE}_{\text{std}}$. The histograms of these metrics are shown in Fig. 21. In our data set, there are 207 examples with SA greater than 10%. And we believe there is no small chart in these result atlases. Thus, we omit them to create the histogram in which SA is in Fig. 21. Although we modify the input atlas, the signal approximation error is still very small.

*Timings.* For the Armadillo model in Fig. 1 with 9,904 vertices, it took 2.59 seconds, 0.16 seconds, and 2.70 seconds to construct the axis-aligned chart, decompose and pack rectangles, and reduce distortion, respectively. The axis-aligned shape construction, axis-aligned shape decomposition and rectangle packing, and distortion reduction took 28.73 seconds, 0.27 seconds, and 31.32 seconds respectively for the Octopus model with 45,579 vertices (Fig. 19). Since the running time $t$ is mainly affected by the size of the mesh, it is weighted with the inverse of vertex number, which is denoted as $\widehat{t} := t/N$. The histograms of $\widehat{t}$ are shown in Fig. 21 (e).

## 5 CONCLUSION

Our method provides a novel technique to refine previously generated atlases with bounded packing efficiency. Due to the conversion from a polygon packing problem to a rectangle packing problem, our method achieves high packing efficiency. It performs well, and




**(a)** ΔBL **(b)** CN **(d)** SA **(c)** $N_{<3\%}$ **(e)** $\widehat{t}$ **(f)** log(SAE)

Fig. 21. Distributions of boundary length elongation, chart number, $N_{<3\%}$, smallest area, timings, and signal approximation error.

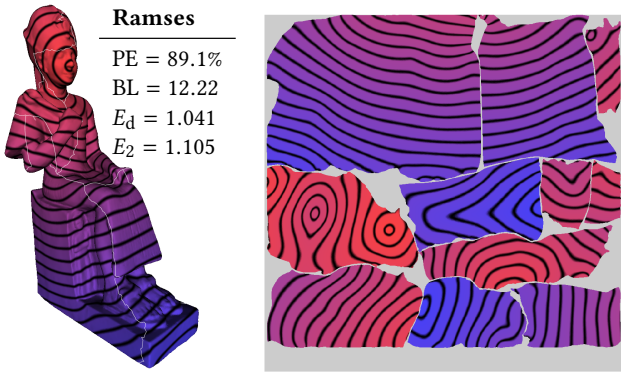
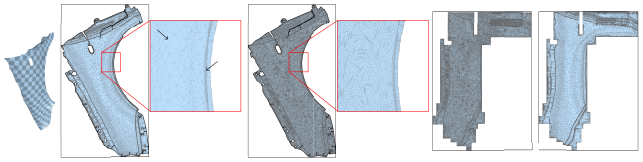



Fig. 22. Single-source geodesics calculation using the method from [Prada et al. 2018] and an atlas generated by our method. The geodesics distance from the nose is continuous across all charts. The input model is the same as the Ramses model in Fig. 15, but the atlas is refined with gaps in distortion reduction.



**(a)** **(b)** **(c)** **(d)** **(e)**

Fig. 23. A failure case. (a) The input mesh. (b) Some edges of the parameterized chart are very different in length (arrows in the red box) cause the axis-aligned chart construction to fail. (c) If the mesh is subdivided, the edge lengths are more similar. (d) Then, it is turned into an axis-aligned chart successfully. (e) The original chart is mapped to the axis-aligned shape.

the overall atlas quality is far superior to previous methods. We have demonstrated the practical robustness of our method on a large data set containing 5,588 models.

*Mapping continuity.* The resulting atlas of our method does not maintain the mapping continuity at the newly generated cut paths. We argue that this is acceptable based on the following two reasons. First, the existing method [Prada et al. 2018] for processing geometric signals on atlases can handle the discontinuity (Fig. 22). Second, the mapping continuity is usually not satisfied at the boundaries of the input charts. In addition, we also provide atlases with mapping continuity at cut paths, i.e., the one after rectangle packing and before distortion reduction. In the future, an interesting topic for research would be to develop a method that can bound packing efficiency without changing mapping distortion and maintaining mapping continuity at newly introduced cuts.

*Modification of the input charts.* Our introduced modification of the input atlas changes the signal that has been defined on the input atlas. It may not meet the original intention of the creator of the original signal. Although our method can minimize the distortion between the input and output atlases in the distortion reduction step for reducing this modification, it is still a limitation of our method.

*Boundary length elongation.* Although our method considers the boundary length elongation in the axis-aligned chart construction and decomposition, we cannot explicitly bound it. For example, the maximum boundary length elongation in our data set is 168%. Thus, simultaneously bounding the packing efficiency and the boundary length elongation would be an intriguing direction for future research.

*Theoretical guarantee.* Although we could achieve high quality refined atlases on more than five thousand models, there is no theoretical guarantee of success for any model, especially for the axis-aligned chart construction process. Our method may fail in the axis-aligned chart construction step because of extremely poor triangulation of the input (e.g., a model with extremely varying edge lengths), as shown in Fig. 23. One possible method to resolve this issue is as follows: (1) split the long edges until all edges are shorter than a specified small length; (2) deform the new atlas to be axis-aligned; (3) map the original atlas to the axis-aligned shape by the SA method [Fu and Liu 2016].

*Quad layout.* Our axis-aligned chart based decomposition method generates a non-conforming quad layout of the input 2D parameterized chart. In principle, other automatic quad layout generation methods, which are robust, high-quality, and suitable for various complex 2D domains, can be used. However, as pointed out by [Campen 2017], all proposed automatic methods are rarely able to provide strict assurance of the quality and suitability of the results for any particular application scenario with hard requirements. Therefore, an interesting future work is to investigate methods that enable the creation of quad layouts particularly suited for the PE improvement.

## ACKNOWLEDGMENTS

We would like to thank Max Limper for sharing their data and executable file of [Limper et al. 2018], and the anonymous reviewers for their constructive suggestions and comments. This work is supported by the National Natural Science Foundation of China (61802359, 61672482, 11626253), the Anhui Provincial Natural Science Foundation (1808085QF208), the Fundamental Research Funds for the Central Universities (WK0010460006, WK0010450004), and the One Hundred Talent Project of the Chinese Academy of Sciences.

## REFERENCES


Mirela Ben-Chen, Craig Gotsman, and Guy Bunin. 2008. Conformal flattening by curvature prescription and metric scaling. In *Comput. Graph. Forum*, Vol. 27. 449–458.

Marcel Campen. 2017. Partitioning surfaces into quadrilateral patches: a survey. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 567–588.

Nathan A. Carr and John C. Hart. 2002. Meshed Atlases for Real-time Procedural Solid Texturing. *ACM Trans. Graph.* 21, 2 (2002), 106–131.

Nathan A. Carr, Jared Hoberock, Keenan Crane, and John C. Hart. 2006. Rectangular multi-chart geometry images. In *Proceedings of the fourth Eurographics symposium on Geometry processing*.

Shuangming Chai, Xiao-Ming Fu, Xin Hu, Yang Yang, and Ligang Liu. 2018. Sphere-based Cut Construction for Planar Parameterizations. *Computer & Graphics (SMI 2018)* 74 (2018), 66–75.

Chin-Chen Chang and Chen-Yu Lin. 2010. Texture tiling on 3D models using automatic PolyCube-maps and wang tiles. *Journal of Information Science and Engineering* 26, 1 (2010), 291–305.

S Claici, M Bessmeltsev, S Schaefer, and J Solomon. 2017. Isometry-Aware Preconditioning for Mesh Parameterization. *Comput. Graph. Forum (SGP)* 36, 5 (2017), 37–47.

David Eppstein, Michael T. Goodrich, Ethan Kim, and Rasmus Tamstorf. 2008. Motorcycle Graphs: Canonical Quad Mesh Partitioning. *Comput. Graph. Forum (SGP)* 27, 5 (2008), 1477–1486.

Xianzhong Fang, Weiwei Xu, Hujun Bao, and Jin Huang. 2016. All-hex meshing using closed-form induced Polycube. *ACM Trans. Graph. (SIGGRAPH)* 35, 4 (2016), 124:1–124:9.

Michael S. Floater. 2003. One-to-one piecewise linear mappings over triangulations. *Math. Comput.* 72 (2003), 685–696.

Michael S. Floater and Kai Hormann. 2005. Surface parameterization: a tutorial and survey. In *In Advances in Multiresolution for Geometric Modelling*. Springer, 157–186.

Xiao-Ming Fu, Chong-Yang Bai, and Yang Liu. 2016. Efficient Volumetric PolyCube-Map Construction. *Computer Graphics Forum (Pacific Graphics)* 35, 7 (2016).

Xiao-Ming Fu and Yang Liu. 2016. Computing inversion-free mappings by simplex assembly. *ACM Trans. Graph. (SIGGRAPH ASIA)* 35, 6 (2016), 216:1–216:12.

Xiao-Ming Fu, Yang Liu, and Baining Guo. 2015. Computing locally injective mappings by advanced MIPS. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 71:1–71:12.

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co.

Björn Golla, Hans-Peter Seidel, and Renjie Chen. 2018. Piecewise linear mapping optimization based on the complex view. 37, 7 (2018), 233–243.

James Gregson, Alla Sheffer, and Eugene Zhang. 2011. All-Hex mesh generation via volumetric PolyCube deformation. *Comput. Graph. Forum (SGP)* 30 (2011), 1407–1416.

Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. 2002. Geometry Images. *ACM Trans. Graph. (SIGGRAPH)* 21, 3 (2002), 355–361.

Ziyad S. Hakura and Anoop Gupta. 1997. The Design and Analysis of a Cache Architecture for Texture Mapping. *SIGARCH Comput. Archit. News* 25, 2 (1997), 108–120.

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

Kai Hormann, Bruno Lévy, and Alla Sheffer. 2007. Mesh Parameterization: Theory and Practice. In *ACM SIGGRAPH 2007 Courses (SIGGRAPH '07)*.

Jin Huang, Tengfei Jiang, Zeyun Shi, Yiying Tong, Hujun Bao, and Mathieu Desbrun. 2014. $l_1$-based construction of PolyCube maps from complex shapes. *ACM Trans. Graph.* 33, 3 (2014), 25:1–25:11.

Zhongshi Jiang, Scott Schaefer, and Daniele Panozzo. 2017. Simplicial Complex Augmentation Framework for Bijective Maps. *ACM Trans. Graph. (SIGGRAPH ASIA)* 36, 6 (2017), 186:1–186:9.

Dan Julius, Vladislav Kraevoy, and Alla Sheffer. 2005. D-Charts: Quasi-Developable Mesh Segmentation. In *Comput. Graph. Forum*, Vol. 24. 581–590.

Liliya Kharevych, Boris Springborn, and Peter Schröder. 2006. Discrete conformal mappings via circle patterns. *ACM Trans. Graph.* 25, 2 (2006), 412–438.

Shahar Z. Kovalsky, Meirav Galun, and Yaron Lipman. 2016. Accelerated Quadratic Proxy for Geometric Optimization. *ACM Trans. Graph. (SIGGRAPH)* 35, 4 (2016), 134:1–134:11.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. (SIGGRAPH)* 21, 3 (2002), 362–371.

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. (SIGGRAPH ASIA)* 37, 6 (2018).

Max Limper, Nicholas Vining, and Alla Sheffer. 2018. Box Cutter: Atlas Refinement for Efficient Packing via Void Elimination. *ACM Trans. Graph. (SIGGRAPH)* 37, 4 (2018), 153:1–153:13.

Celong Liu, Wuyi Yu, Zhonggui Chen, and Xin Li. 2017. Distributed poly-square mapping for large-scale semi-structured quad mesh generation. *Computer Aided Design* 90 (2017), 5 – 17.

Ligang Liu, Chunyang Ye, Ruiqi Ni, and Xiao-Ming Fu. 2018. Progressive Parameterizations. *ACM Trans. Graph. (SIGGRAPH)* 37, 4 (2018), 41:1–41:12.

Marco Livesu, Nicholas Vining, Alla Sheffer, James Gregson, and Riccardo Scateni. 2013. Polycut: monotone graph-cuts for PolyCube base-complex construction. *ACM Trans. Graph. (SIGGRAPH ASIA)* 32, 6 (2013), 171:1–171:12.

Victor J. Milenkovic. 1999. Rotational Polygon Containment and Minimum Enclosure Using Only Robust 2D Constructions. *Comput. Geom. Theory Appl.* 13, 1 (1999), 3–19.

Ashish Myles and Denis Zorin. 2012. Global Parametrization by Incremental Flattening. *ACM Trans. Graph. (SIGGRAPH)* 31, 4 (2012), 109:1–109:11.

Tobias Nöll and D Strieker. 2011. Efficient packing of arbitrary shaped charts for automatic texture atlas generation. *Comput. Graph. Forum* 30, 4 (2011), 1309–1317.

Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Trans. Graph. (SIGGRAPH ASIA)* 36, 6 (2017), 215:1–215:11.

Fabián Prada, Misha Kazhdan, Ming Chuang, and Hugues Hoppe. 2018. Gradient-domain Processing Within a Texture Atlas. *ACM Trans. Graph. (SIGGRAPH)* 37, 4 (2018), 154:1–154:14.

Budirijanto Purnomo, Jonathan D. Cohen, and Subodh Kumar. 2004. Seamless Texture Atlases. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 65–74.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Maps. *ACM Trans. Graph.* 36, 2 (2017).

Pedro V. Sander, Steven J. Gortler, John Snyder, and Hugues Hoppe. 2002. Signal-specialized Parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering*. 87–98.

Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. 2001. Texture mapping progressive meshes. In *SIGGRAPH*. 409–416.

P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. 2003. Multi-chart Geometry Images. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 146–155.

Christian Schüller, Ladislav Kavan, Daniele Panozzo, and Olga Sorkine-Hornung. 2013. Locally Injective Mappings. *Comput. Graph. Forum (SGP)* 32, 5 (2013), 125–135.

Alla Sheffer. 2002. Spanning tree seams for reducing parameterization distortion of triangulated surfaces. In *Shape Modeling International*. 61–66.

Alla Sheffer and John C Hart. 2002. Seamster: inconspicuous low-distortion texture seam layout. In *Proceedings of the conference on Visualization'02*. 291–298.

Alla Sheffer, Emil Praun, and Kenneth Rose. 2006. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.* 2, 2 (2006), 105–171.

Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z. Kovalsky, and Yaron Lipman. 2017. Geometric Optimization via Composite Majorization. *ACM Trans. Graph. (SIGGRAPH)* 36, 4 (2017), 38:1–38:11.

Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 70:1–70:9.

Yousuf Soliman, Dejan Slepčev, and Keenan Crane. 2018. Optimal Cone Singularities for Conformal Flattening. *ACM Trans. Graph. (SIGGRAPH)* 37, 4 (2018), 105:1–105:17.

Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. 2002. Bounded-distortion piecewise mesh parameterization. In *Proceedings of the Conference on Visualization '02*. 355–362.

Boris Springborn, Peter Schröder, and Ulrich Pinkall. 2008. Conformal equivalence of triangle meshes. *ACM Trans. Graph. (SIGGRAPH)* 27, 3 (2008), 77:1–77:11.

Marco Tarini, Kai Hormann, Paolo Cignoni, and Claudio Montani. 2004. PolyCube-Maps. *ACM Trans. Graph. (SIGGRAPH)* 23, 3 (2004), 853–860.

TeamHypersomnia. 2016–2018. rectpack2D. https://github.com/TeamHypersomnia/rectpack2D.

W. T. Tutte. 1963. How to draw a graph. In *Proceedings of the London Mathematical Society*, Vol. 13. 747–767.

Jiazhil Xia, Ismael Garcia, Ying He, Shi-Qing Xin, and Gustavo Patow. 2011. Editable polycube map for GPU-based subdivision surfaces. In *Symp. on Interactive 3D Graphics and Games*. 151–158.

Shiwei Xiao, Hongmei Kang, Xiao-Ming Fu, and Falai Chen. 2018. Computing IGA-suitable Planar Parameterizations by PolySquare-enhanced Domain Partition. *Comput. Aided Geom. Des.* 62 (2018), 29–43.

Chih-Yuan Yao and Tong-Yee Lee. 2008. Adaptive geometry image. *IEEE. T. Vis. Comput. Gr.* 14, 4 (2008), 948–960.

Wuyi Yu, Kang Zhang, Shenghua Wan, and Xin Li. 2014. Optimizing Polycube domain construction for hexahedral remeshing. *Computer Aided Design* 46 (2014), 58–68.

Eugene Zhang, Konstantin Mischaikow, and Greg Turk. 2005. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics (TOG)* 24, 1 (2005), 1–27.

Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-charts: Stretch-driven Mesh Parameterization Using Spectral Analysis. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*. 45–54.

Yufeng Zhu, Robert Bridson, and Danny M. Kaufman. 2018. Blended Cured Quasi-newton for Distortion Optimization. *ACM Trans. Graph. (SIGGRAPH)* 37, 4 (2018), 40:1–40:14.