Direct Seamless Parametrization

ZOHAR LEVI, Victoria University of Wellington, New Zealand

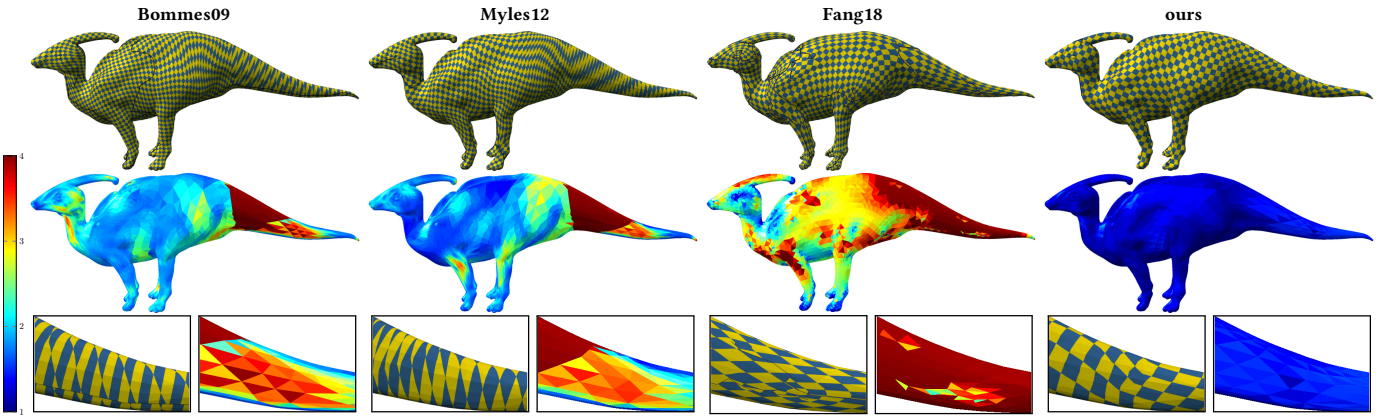


Fig. 1. Dilo. *Top* to *bottom*: texture, a parametrization isometric distortion heat map, and the middle of the tail blowup. On the *left* is the legend of the colors of the heat maps of the isometric distortion measure τ in (3). The tail in previous state-of-the-art results is distorted. In Fang18, while the squares' shape is not very distorted, they are badly scaled w.r.t. the most distorted facet.

We present a method for seamless surface parametrization. Recent popular methods first generate a cross-field, where curvature is concentrated at singular vertices. Next, in a separate step, the surface is laid out in the domain subject to derived seamlessness constraints. This decoupling of the process into two independent problems, each with its own objective, leads to suboptimal results. In contrast, our method solves both problems together using domain variables.

The key ingredient to the robustness of our method is a rounding strategy based on local estimation. The insight is that testing a small patch to decide between two likely possibilities is a good estimator.

Most distortion measures can be used with our method, which get minimized consistently throughout the pipeline. Our method also enables feature alignment, as well as alignment to principle curvatures, and isotropic and anisotropic scaling.

Additional Key Words and Phrases: parametrization, locally injective mappings, bounded distortion

ACM Reference Format:
Zohar Levi. 2020. Direct Seamless Parametrization. *ACM Trans. Graph.* 1, 1, Article 1 (January 2020), 15 pages. https://doi.org/10.1145/3439828

1 INTRODUCTION

Surface parametrization is a classic problem, and methods in the field have evolved considerably to produce high quality mappings. Nowadays, the mappings are often required to be locally injective (foldover-free) as well as exhibit low distortion.

Whereas classic methods usually minimize a quadratic objective, which requires solving only an inexpensive linear system, recent methods for mapping optimization [Aigerman et al. 2014; Chien et al. 2016; Levi and Zorin 2014; Lipman 2012; Rabinovich et al. 2017; Shtengel et al. 2017; Smith and Schaefer 2015] employ more expensive solutions to generate high quality mappings. Where *seamless parametrization* is required, previous approaches do not take into account the new optimization costs, and employing it within their frameworks is impractical. As a consequence, advanced mapping optimization is not fully integrated into the seamless parametrization pipeline, and the new mapping methods are used only to optimize the final layout. The objective of this work is to remedy this limitation by incorporating the mapping optimization through the whole pipeline. The result is seamless parametrization with increased quality.

We take a closer look at the issue. Given a triangle mesh, the most common approach to the seamless parametrization problem is to divide the problem into two stages. In the first stage, a cross field is constructed, and *matchings* (or period jumps) are defined. This can be viewed as defining a conic metric where most vertices are flat (incident angles summing to 2π), and all the curvature is concentrated in a few cone singularities, as determined by the matchings. In the second stage, the cross field is used as local frames to lay out the mesh in the parametric domain (assigning UV coordinates to vertices) subject to seamlessness constraints that are derived from the field matchings.

These two steps are considered independent to such an extent that there is specialized related work that addresses each of the two problems on its own. The main reason that the stages are separated is that solving the first stage is hard. It is mostly tackled using integer-programming or non-convex minimization. After solving

Author's address: Zohar Levi, Victoria University of Wellington, New Zealand.

0730-0301/2020/1-ART1 $15.00
https://doi.org/10.1145/3439828

for the field, the layout problem can then be solved more efficiently, using a more general mapping optimization method to solve a standard planar mapping problem with the addition of cheap linear equality seamlessness constraints. The problem with separating the two stages is that the objectives of the two steps are generally inconsistent. More specifically, the distortion measure of the mapping that we would like to minimize is used as an objective only in the second stage of optimizing the layout. This means that the field that was constructed in the first stage, from which local frames are derived, is not necessarily optimal for this objective. Usually, the objective in the first stage is a smooth field, which only indirectly promotes isometry.

This drawback motivates the key idea for our algorithm. Instead of a two-stage approach, our approach is to define a single problem with a metric-based objective, using domain variables. This allows us to employ recent popular distortion metrics and minimize the same energy consistently in every step of the algorithm. Our algorithm also uses a new rounding strategy that is based on local estimation, which is key to our method robustness. We also offer a variant of the algorithm that follows the same paradigm to handle feature and curvature alignment. Finally, we demonstrate how anisotropy is handled within the framework in a consistent way.

1.1 Paper Outline

The structure of the paper is as follows:

- Section 2 reviews recent work.
- Section 3 defines the formal setup.
- Section 4 introduces the main algorithm for direct seamless parametrization.
- Section 5 introduces a rounding strategy based on local estimation.
- Section 6 discusses a variant of the algorithm that handles feature alignment, as well as a generalization that handles principal curvature alignment.
- Section 7 treats isotropic and anisotropic scaling.
- Section 8 provides an evaluation of the method, where a practical, small set of challenging benchmark models for seamless parametrization is offered.

2 RELATED WORK

We review recent work related to seamless parametrization. A popular approach to the problem is based on cross field generation [Vaxman et al. 2016], and the main application of the problem is quad mesh generation [Bommes et al. 2013].

Bommes et al. [2009] smoothly interpolates a cross field that is aligned with sparse salient principal curvature directions over the mesh. The cross field is used as frames to lay out the mesh. Diamanti et al. [2015] generate a field that integrates to a seamless parametrization and is based on complex polynomials [Diamanti et al. 2014; Knöppel et al. 2013].

In our context, the ultimate objective of these methods is to construct a seamless parametrization with low isometric distortion. This is done in two steps: i) constructing a field, ii) optimizing the layout according to the chosen distortion measure. The objective of the first step is to construct the smoothest frame field. The motivation is that a smoother field usually promotes isometry in the end result. As a simple illustration, consider a developable surface, discretized as triangle mesh. A perfectly smooth field (with vanishing connection) can be constructed over the whole surface. This leads to the same local coordinate frame for all triangles in the second step, which results in a perfect isometric layout. However, when the surface has Gaussian curvature, a smoother field only promotes isometry indirectly. In other words, solving for a smooth field is not equivalent to solving for an isometric mapping, let alone minimizing a specific isometric distortion measure.

Another inconsistency is the type of norm used in the minimized energy in the two steps. In the first step, when the field is constructed, the triangle areas are not taken into account, and the matchings are rounded in a greedy way. These promote L_∞-norm minimization (minimizing the maximal error). On the other hand, the objective of the second step is an L_2-norm, a sum weighted by triangle area. Optimizing for two different objectives, where one of them only indirectly encourages low metric distortion, results in a suboptimal mapping.

Myles and Zorin [2012] describe a method for finding cone locations and angles by evolving the metric of the surface during a flattening stage. They approximate the *As-Rigid-As-Possible* energy by the scale factor squared. Feature alignment capability is added in [Myles and Zorin 2013]. While the metric is considered during the field generation, the objective that they minimize is only an approximation to make the problem more tractable. Moreover, their objective is quadratic, and extension to more expensive objectives is not straight forward.

Based on tracing of the motorcycle graph, Myles et al. [2014] generate a locally injective mapping that aligns with a given cross field.

Fang et al. [2018] introduce a hybrid method for quad mesh generation, which combines field-based parametrization with the Morse-Smale complex. Their field generation is metric-based [Jiang et al. 2015], but the chosen distortion measure is not flexible.

In the context of surface parametrization, [Li et al. 2018; Poranne et al. 2017] also work directly with domain variables. They optimize UV coordinates and topology to improve mapping distortion and seam quality.

3 FORMAL SETUP

Let $\mathcal{M} = (V; E; T)$ be a surface triangle mesh (consisting of vertices, edges, and triangles). Let $f\colon \mathcal{M} \subset \mathbb{R}^3 \to \mathbb{R}^2$ be a piecewise linear surface parametrization that maps a point $p \in M$ to the UV plane, $f(p) = (u, v)$. To create a planar global parametrization, the surface is cut along a seam into a disk-topology mesh M_c. The seam is defined as a set of *seam edges* $S \subset E$ and a set of corresponding boundary half-edges S_c of the cut mesh M_c. Each seam edge in S is mapped into two *twin half-edges* $e, e' \in S_c$ in the plane corresponding to the two triangles incident to the edge.

Definition 3.1 (seamless parametrization). We say a parametrization is *seamless* if the (u, v) domain values for corresponding vertices on both sides of a seam edge are related by

$$(u', v') = R^r_{90^\circ}(u, v) + (s, t), \quad r \in \{0, 1, 2, 3\}, \quad (s, t) \in \mathbb{R}^2 \tag{1}$$

Algorithm 1: The main algorithm

```
Lay out the mesh isometrically.
Initialize the rounding error threshold ϵ.
while there are edges with non-integer matchings do
    Round matchings with rounding error below ϵ.
    Derive seamlessness constraints (1) from integer
      matchings and add them to the constraint list.
    Layout optimization subject to the constraints in the list.
    Increase ϵ.
```

where $R^r_{90^\circ}$ is a $90r^\circ$ rotation. r is called a *matching*. r, s, and t are associated with an edge.

Definition 3.2 (integer seamless parametrization). We say a parametrization is *integer seamless* if domain values across seam edges are related by

$$(u', v') = R^r_{90^\circ}(u, v) + (s, t), \quad r \in \{0, 1, 2, 3\}, \quad (s, t) \in \mathbb{Z}^2$$

and $(u, v) \in \mathbb{Z}^2$ for all cone vertex copies.

In this work, our objective is seamless parametrization, and we consider making it integer a separate problem, e.g. tackled by Campen et al. [2015].

4 THE ALGORITHM

This section provides details for the main algorithm of the paper. The main objective is seamless parametrization, solving for UV coordinates. The problem can be formulated using a set of domain variables x:

$$\min_{\substack{x \in \mathbb{R}^{2\times n} \\ r \in \{0,1,2,3\}^{|S|}}} \quad \mu(x) \tag{2a}$$

$$\text{s.t.} \quad \vec{e'} = R^{r_e}_{90^\circ}\vec{e}\,, \quad \forall e \in S\,, \tag{2b}$$

where x is the UV coordinates of n vertex copies in the cut mesh. $r = (r_1, ..., r_{|S|})$ is an integer vector of rotation types for each seam edge. $\mu \in \mathbb{R}$ represents the minimized objective (which may encapsulate additional constraints), and it varies according to the chosen distortion measure. (2b) are the constraints on the seam edge vectors derived from (1).

The problem in (2) is Mixed Integer (MI) due to the matchings, which makes it hard to solve. Bommes et al. [2009] offered a greedy Mixed Integer Quadratic Programming (MIQP) solver that minimizes a quadratic objective. Their approach is to start with a smooth solution and perform one-by-one greedy rounding, which is not practical for more expensive types of objectives, for example, those related to bounded distortion or inversion-free mappings. Similarly, we also start with a smooth solution but proceed to round the matchings in batches.

Definition 4.1 (rounding error). Given a scalar $x \in \mathbb{R}$, the rounding error is $\left|x - \lfloor x \rceil\right| \in [0, 0.5]$.

Alg. 1 outlines the main algorithm, and we now give details of the steps.

Initialization. We cut the surface along a (dual) spanning tree of the facets and lay out the cut mesh in the plane without distortion: we first map a seed triangle isometrically to the domain and then lay out its triangle neighbors isometrically. We proceed laying out neighbors of the growing patch until the whole mesh is laid out. The seam edges (of the primal cut graph) will have matchings that are not integer in general.

Rounding Step. Given a rounding error threshold ϵ, we round the matchings of all edges with rounding error smaller than ϵ. The rounding is done based on local estimation (Section 5). The choice of the increment step for ϵ can vary between two extremes: i) rounding all matchings at once; ii) rounding matchings one by one. Both extremes are suboptimal. The first fails on challenging models, and the second is too slow and often still results in suboptimal mappings. Through experiments, we found a sweet spot between the two extremes: increase the threshold in increments of 0.1 (up to 0.5).

Layout optimization. We optimize the layout using a method that fits the selected objective μ. The layout from the previous iteration is used to initialize the method (whether as a starting point or a source to extract initial frames from). For edges with integer matchings, linear constraints are derived from the seamlessness condition in (1) [Bommes et al. 2009] and added to the layout problem.

5 LOCAL ESTIMATION

Rounding the matchings is usually done naively: greedily rounding to the nearest integer [Bommes et al. 2009; Myles and Zorin 2012]. This may lead to suboptimal results or bad layouts with foldovers. In the supplement, we offer two approaches to recover from bad layouts. Resorting to these recovery approaches, though, is not ideal. Runtime aside, even if the resulting layout is admissible (locally injective), its distortion may be too high; a better approach would be to somehow round the matchings more cleverly in the first place. We suggest a new metric-based rounding scheme, which proved successful on all our benchmark models and is the main ingredient for our algorithm's robustness. Our approach is motivated by two insights:

- The optimal rounding is likely to be one of the two nearest integers, either the floor or the ceiling of a given non-integer edge matching.
- A small neighborhood is usually enough to judge which option is better.

This leads to the following local estimation routine for rounding. Given an edge matching to round, we consider the local triangle patch neighborhood of the edge (see Section 8.1 for details). For both possible values of the rounded matching (floor and ceiling), we lay out the patch independently, using the same layout optimization method that we selected in the main algorithm, and check the distortion of its resulting mapping. We choose the rounded value for the matching that results in the lower distortion. Small patches suffice for a good estimation, and they are fast to lay out (see Section 8.1 for details). One can also consider a tailored solver for small problems based on an analytical solution in the spirit of [Levi and Zorin 2014], but an off-the-shelf SOCP solver was fast enough in

Algorithm 2: Feature edge alignment together with matching rounding

Lay out the mesh isometrically.
Initialize the rounding error threshold ϵ.
while *there are edges with non-integer matchings or unlabeled feature edges* **do**
  Round matchings with rounding error smaller (greater) than ϵ.
  Derive seamlessness constraints (1) from integer matchings and add them to the constraint list.
  Label the feature edges with alignment error smaller (greater) than ϵ.
  Derive alignment constraints from feature edge labeling and add to constraint list.
  Layout optimization subject to constraints.
  Update ϵ.

Algorithm 3: Feature edge alignment after matching rounding

Lay out the mesh isometrically.
Initialize the matching rounding error threshold ϵ_m and the feature edge alignment error threshold ϵ_a.
while *there are edges with non-integer matchings* **do**
  Round matchings with rounding error smaller (greater) than ϵ_m.
  Derive seamlessness constraints (1) from integer matchings and add them to the constraint list.
  Lay out the mesh subject to constraints.
  Update ϵ_m.
while *there are unlabeled feature edges* **do**
  Label the feature edges with alignment error smaller (greater) than ϵ_a.
  Derive alignment constraints from feature edge labeling and add to constraint list.
  Layout optimization subject to constraints.
  Update ϵ_a.

our experiments. We round edges with smaller rounding error first. The algorithm can be easily parallelized if the employed solver is not already using parallelism (e.g. Mosek).

6 FEATURE ALIGNMENT

For certain models, usually CAD-related, aligning the parametrization to sharp features is required. For example, when quadrangulating a cube, we would like the sharp cube edges to stay sharp. To preserve a surface edge in the source triangle (to force it to coincide with an edge of the generated quad mesh), its corresponding domain (twin half-)edges need to be mapped to integer valued isolines. We augment our previous definitions to define seamless parametrization as having feature domain edges parallel to the axes and integer seamless parametrization as having feature domain edges mapped to integer valued isolines. Our objective remains seamless parametrization.

Algorithm 4: Optimal initial global rotation angle of feature edges

$\{v_i\} \coloneqq$ normalized feature domain edge vectors
  ▷ `Unit vectors to points on a unit circle`
$\{p_i\} \coloneqq \{origin + v_i\}$
Run k-means on $\{p_i\}$ to partition it into 4 groups.
$\bar{p} \coloneqq$ the mean of the largest group
Return the angle between $(\bar{p} - origin)$ and the x-axis.

In the algorithm, a feature edge is labeled u-aligned or v-aligned, according to the axis it needs to align with. As we did for the seamlessness constraints, labeled feature edges are aligned by adding appropriate linear equality constraints [Bommes et al. 2009] to the layout problem in (2). We use the same mechanism that we used to round the matchings for feature alignment. Since a matching r represents a rotation angle of $90r^\circ$, we have the following relation between a rotation angle α and a matching r:

$$r = \frac{\alpha}{90} \ .$$

Definition 6.1 (alignment error). Let α be the angle between a feature domain edge and the u-axis. We define the alignment error of the feature edge as the rounding error of $\frac{\alpha}{90}$.

The algorithm is outlined in Alg. 2, and we now give the details.

Initialization. As in Alg. 1, we start with an isometric layout of a dual spanning tree with no constraints and zero distortion. Next, we find an optimal global planar rotation (see Alg. 4) that minimizes the alignment error of all feature edges, and we rotate the mapping accordingly.

Threshold steps. We use the same threshold and incremental steps for both matching rounding error and alignment error. As a natural, intuitive choice, we start with a low threshold and increase it gradually by positive steps [Bommes et al. 2009; Myles and Zorin 2012]. The intuition behind this standard strategy is clear: we first round edges with the smallest rounding errors since their rounding is more likely to be correct.

Another choice is to do the rounding in a top-down fashion using negative steps. Then, edges with larger rounding error, which are considered more "challenging" to round, are processed first. We consider them more challenging since their change of position (due to the rounding) is more pronounced, creating more distortion in the mapping. Rounding them first provides more flexibility, in terms of how many edges are constrained before they are rounded, and how much the mapping can be deformed (considering degrees of freedom) and remain valid. This allows the "easier" edges—those with less rounding error—the opportunity to adapt and compensate for the distortion introduced by the "challenging" edges. This can

also be viewed as sharing the distortion caused by the more "challenging" edges with the "easier" ones. This second strategy usually results in more cones, which is necessary in some cases.

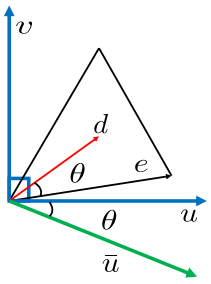
Another variation of the algorithm is to first run Alg. 1 to completion to round the matchings and then label feature edge alignment separately; see Alg. 3. In this approach, we use soft constraints for the edge alignment. The advantage of this algorithm is that it is usually easy enough to parameterize the surface without alignment constraints, and then adding them as soft constraints guarantees that the parametrization stays within distortion bounds if such constraints are present. The price, though, is potential alignment error, which in our experiments did not exceed 10^{-5}.

Labeling feature edges. Similar to the local-patch-based rounding strategy for the matchings, we do the following for the alignment error to determine feature edge labels. We lay out the local neighborhood of a feature edge twice. In the first time, we constrain the edge to be parallel to the u-axis, and in the second time, we constrain the edge to be parallel to the v-axis. Depending on which of the two patch layouts has lower distortion, we label the feature edge as u-aligned or v-aligned.

Furthermore, we impose a maximum distortion threshold for feature edges: when the increase in distortion due to the selected alignment constraint (estimated by the local estimation procedure) is larger than the threshold, we disable the feature edge constraint. This is useful when we have low confidence in the procedure that detected features. When used in Alg. 3, this distortion threshold roughly sets the max distortion for the whole mapping. Here, we consider feature alignment constraints to be less critical than matching constraints.

6.1 Alignment with Principal Curvature Directions

Sometimes, aligning quad facets with principal curvature directions is desired [Bommes et al. 2009]. Alignment constraints based on the mapping Jacobian can be added to the layout optimization. We opted for a simpler solution that exploits feature alignment. Consider a facet with given principal curvature directions. Let e be an arbitrary reference triangle edge vector, d the main principal curvature direction, and θ the angle between them; see inset. To align d, for example, with the canonical u-axis, we need to align e with the u-axis rotated by $-\theta$, which is notated as $\bar{u}$ in the figure. We use the same procedure of feature alignment to align e with an arbitrary $\bar{u}$ direction (in a cross). Note that in case of anisotropy (Section 7), the aligned direction should be calculated with respect to the scaled triangle edge. The solution works well when expecting low isometric distortion. See Fig. 4 for an example.



7 ISOTROPIC AND ANISOTROPIC SCALING

Anisotropic elements may be desired as a designer choice or to better approximate the surface local shape [Kovacs et al. 2011]. Bommes et al. [2009] implement anisotropy by penalizing stretch differently along desired isoline directions. Panozzo et al. [2014] use frame fields, which generalize cross fields and allow to incorporate anisotropy and scaling. Given sparse user defined anisotropic constraints, they decouple the symmetric scaling part of the frame and interpolate it over the surface. They observed that treating anisotropy in the layout stage alone is not enough, and it should be taken into consideration when generating a field. For that purpose, they scale the metric according to the interpolated scaling field and deform the mesh accordingly. They proceed with finding a smooth cross field on the deformed embedding.

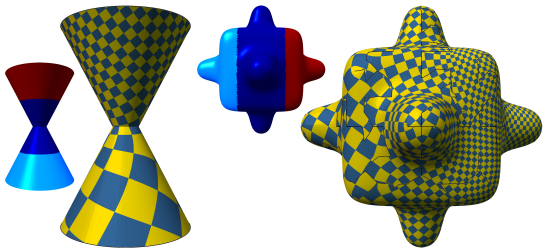


Fig. 2. Parameterizing the hourglass and bumpy-cube with isotropic scaling. The heat maps show the user defined scale: 2 in red, 0.5 in light blue, and free in deep blue. The parts with the free scale were optimized using the As-Similar-As-Possible energy.

Following the same consideration, instead of deforming the mesh and incurring an amount of distortion, we naturally incorporate anisotropy into our metric-based framework. Moreover, we take the target distortion measure into account while the method in [Panozzo et al. 2014] still operates in two decoupled steps of field generation and layout. Given the symmetric scaling part of a frame field over the mesh, we incorporate it into the objective of the layout stage (see supplement). More specifically, instead of using a source mesh triangle in the selected distortion measure objective, we use a transformed version of it. The rest of the algorithm remains unchanged.

In Fig. 2, we demonstrate isotropic scaling. A third of the mesh facets is given a desired scale of 2, a third is given 0.5, and a third remains free. Panozzo et al. [2014] interpolate the free part using a harmonic field. We opted for interpolating the free part by solving for the optimal scaling. For that, we use conformal distortion in the layout stage. The supplement provides the details for generating a quasi-conformal mapping, as well as incorporating an arbitrary transformation (derived, for example, from an anisotropic frame) into an optimized mapping objective.

In Figures 3 and 4, we illustrate anisotropy based on principal curvature. While [Bommes et al. 2009] chair result is comparable to ours (though the quads are a bit too elongated at some places compared to the supplied desired scaling), the surfboard result demonstrates why decoupling the field generation from the layout can be problematic.

The surfboard was modeled as a deformed sphere, which was stretched and flattened. With isometric (i.e. rotation—cross) frames and no alignment, both methods produce similar results (distortion-wise). When alignment is introduced (still isometric frames), [Bommes et al. 2009] produces a field with only eight 3-cones (0.25 field index) versus six cones without alignment. The field was generated using sparse fixed frames, set to align with principle curvature directions;

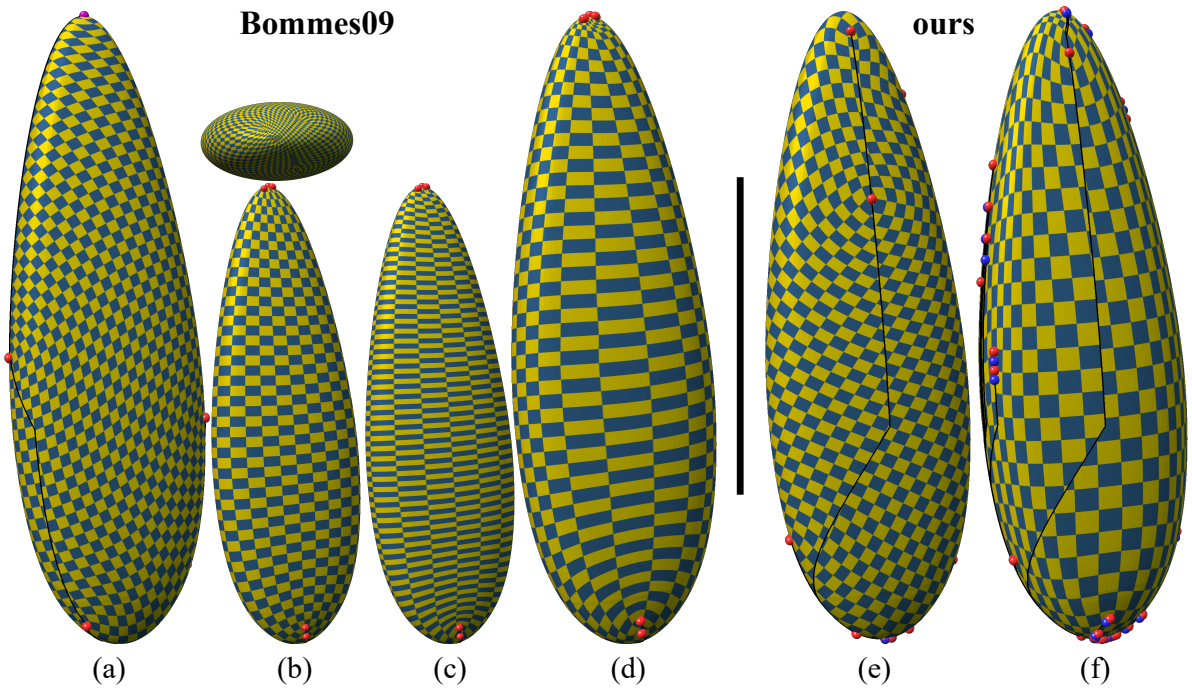


Fig. 3. Anisotropy. (d,f) the frames of the surfboard were anisotropically scaled based on principal curvatures' magnitude, and the mapping was aligned with principle curvature directions. (a,e) regular cross frames were used with no alignment. (b,c) generated using isotropic frames with principle curvature direction alignment. (b) generated by minimizing in the layout stage the L_2-ARAP energy (see supplement)—with fixed frames—which is more suited for frame alignment. The top view shows areas with high distortion and flipped triangles near the cones. (c) generated by minimizing the L_∞-ARAP energy.

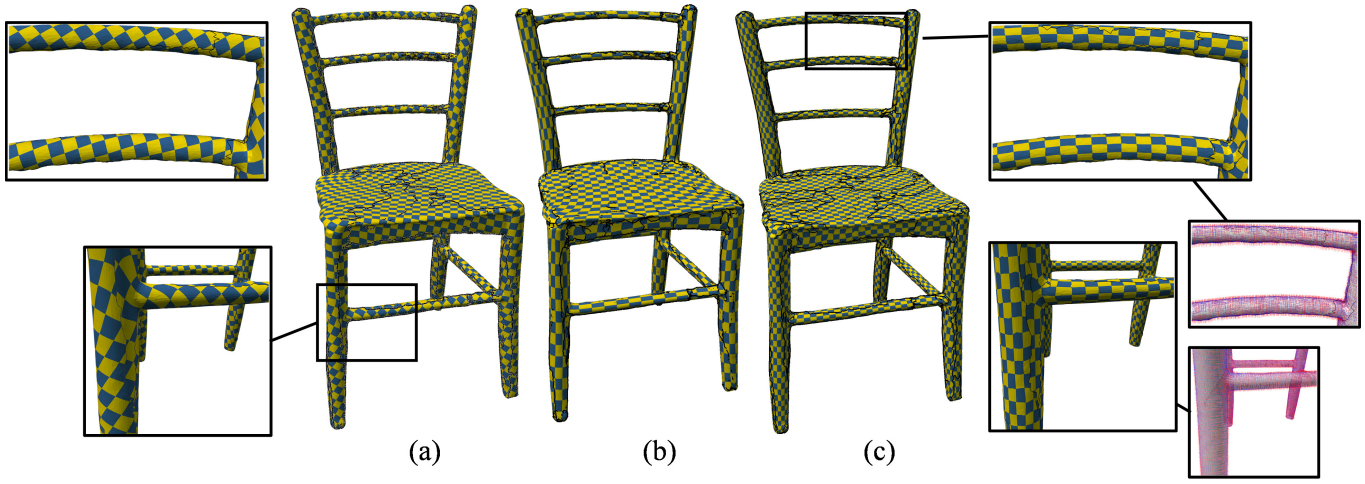


Fig. 4. (c) The frames of the chair were anisotropically scaled based on principal curvatures' magnitude. Moreover, the frames were aligned with principal curvature directions in places where they were prominent. (a) regular cross frames with no alignment. Principal curvature directions are illustrated in the blowup on the *bottom right corner*. (b) [Bommes et al. 2009] result with anisotropy and alignment.

the rest of the cross field was smoothed. The result is that along the surface, between the two poles, the field is relatively smooth with no cone interruption. As the field gets near the poles, the board is getting narrower, and four cones are introduced near each pole.

This field poses an issue in the layout step. There are not enough cones to account for the narrowing of the surface near the poles (e.g. compared to our results with 12 cones in the isometric, unaligned case and 88 cones in anisotropic, aligned case). When an L_2-norm-based optimization is used, the distortion is concentrated near the

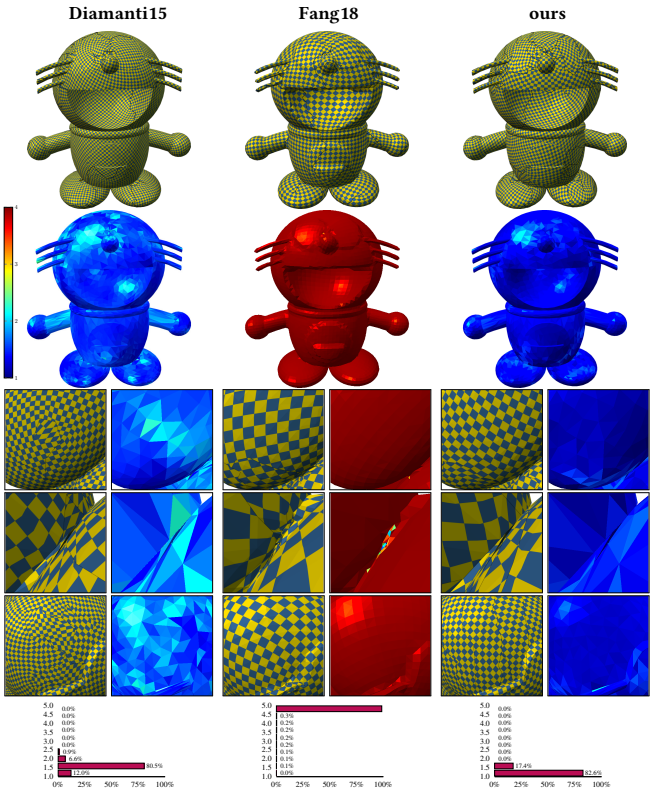


Fig. 5. Robocat. *Top* to *bottom*: texture, a heat map of the isometric distortion τ in (3), three other viewpoints, and a distortion histogram.

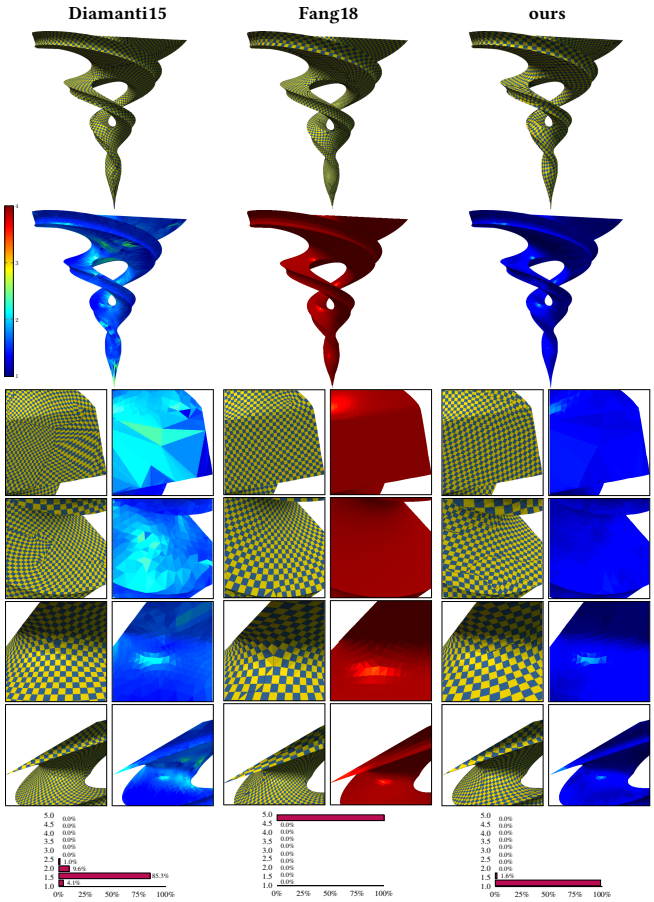


Fig. 6. Twirl. *Top* to *bottom*: texture, an isometric distortion τ heat map, four other viewpoints, and a distortion histogram.

cones, which forms highly distorted areas and foldovers. When an L_∞-norm-based optimization is used, the distortion is spread across the surface, which prevents flips and highly distorted areas. However, to account for the narrowing near the poles, the mapping is stretched (and hence distorted)—it is supposed to be squares due to the isometric frames—throughout the surface. After adding anisotropy and scaling the frames in the layout step, the mapping remains stretched in general.

Note that since this problem is inherent to decoupling the two steps in [Bommes et al. 2009] approach, and it happens even when isometric frames are used, the approach in [Panozzo et al. 2014] faces the same issue. More specifically, when isometric frames are used, the deformation of the surface according to the scaled frames, suggested in [Panozzo et al. 2014], does not change the surface, and the result would be identical to [Bommes et al. 2009].

8 EVALUATION

In the evaluation, we chose as our main objective to minimize the scale-invariant, isometric distortion measure

$$\tau = \begin{cases} \sqrt{\frac{\sigma_1^{max}}{\sigma_2^{min}}} & \sigma_2^{min} > 0 \\ \infty & else \end{cases}, \tag{3}$$

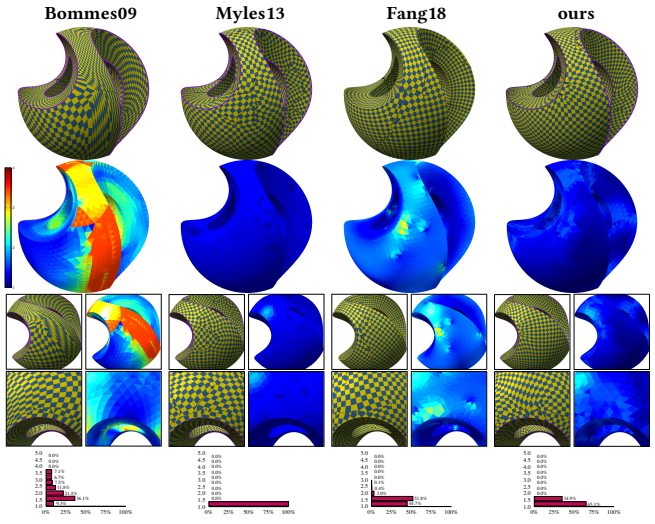


Fig. 7. Sculpt. *Top* to *bottom*: texture, an isometric distortion τ heat map, two other viewpoints, and a distortion histogram.

model	#tri	ours						Bommes09			Myles12			Myles14				Diamanti15			Fang18's τ
		#it	1 it(sec)	τ	#patch	#improved	time(sec)	#it	τ	time(sec)	#it	τ	time(sec)	#tri	#it	τ	time(sec)	#it	τ	time(sec)	
dancer	18K	13	4	1.5	762	79	76	3	1.9	12	1	∞		42K	7	2.6	28	3	1.5	12	5.0
armadillo	100K	12	32	1.7	4727	716	528	1	∞		1	∞		155K	7	3.8	225	2	1.7	64	7.9
dilo	54K	12	10	1.5	2163	310	188	2	4.6	20	2	4.3	20	80K	4	4.7	41	3	1.6	31	7.2
filigree	100K	11	23	1.7	8593	1062	515	3	2.2	70	4	2.7	94	237K	1	∞		3	1.8	70	∞
knot	100K	13	28	1.4	5580	516	530	2	2.5	56	2	2.5	56	128K	4	1.3	112	3	1.6	84	5.4
dragon	105K	12	31	2.0	10847	1444	695	1	∞		1	∞		185K	6	4.9	185	1	∞		9.1
robocat	8K	12	2	1.7	1066	170	51	1	∞		1	∞		23K	34	4.2	54	5	2.4	8	27.1
shark	20K	13	5	1.4	920	85	91	2	3.7	10	4	3.2	19	38K	4	3.8	19	6	1.7	29	9.6
twirl	10K	11	2	1.4	650	87	41	2	1.4	4	1	∞		17K	1	∞		4	2.5	8	74.6
grayloc	69K	10	17	1.6	2926	530	261	1	∞		1	∞		108K	7	6.0	122	3	1.7	52	∞
greek_sculpture	50K	11	9	1.9	3536	472	204	1	∞		1	∞		86K	6	4.2	54	3	1.9	27	61.7
blade_earring	23K	18	4	3.1	1079	173	107	1	∞		1	∞									
octopus	33K	14	7	3.5	1575	238	148	1	∞		1	∞									
octocat	38K	15	12	1.4	1212	188	211	3	2.8	35	1	∞									
linkCupTop	20K	24	5	3.3	2585	489	208	2	4.0	11	1	∞									

Table 1. Comparing 15 models. Missing data means that the result was unavailable. Columns: #tri, number of triangles; #it, number of local-global layout iterations; 1 it, the time in seconds to run one iteration of the local-global layout problem; τ, the maximal distortion measure in (3), where ∞ indicates a failure to generate a locally bijective mapping (Fang18's results: filigree is non-manifold, grayloc is multi-component); #patches, the number of edges to which the local estimation procedure was applied; #improved, the number of edges where the rounding choice was influenced by the local estimation; time, the total time the method spent. For our method, it includes the local estimation and the layout stage. For the other methods, it includes only the layout stage (e.g. the field generation time is not accounted for).

model	#tri	ours									Bommes09			Myles13			Fang18's τ
		#it	algorithm	step inc	1 it(sec)	τ	#patch	#disabled	#improved	time(sec)	#it	τ	time(sec)	#it	τ	time(sec)	
beetle_refined	39K	14	2	+	11	1.9	1541	0	118	201	4	2.7	44	1	∞	11	3.4
casting_refined	37K	27	3	+	12	1.9	3470	1	210	431	4	2.3	48	1	∞	12	3.4
fandisk	14K	13	2	+	5	1.9	856	0	40	88	2	3.5	10	2	1.4	10	2.6
sculpt	7K	18	2	-	3	1.6	556	0	166	64	8	3.4	21	3	1.3	8	3.0
smooth-feature	12K	19	2	-	6	1.8	279	0	64	119	1	∞	6	3	1.4	17	4.0
metatron	50K	27	3	+	22	2.4	6990	1	508	801	2	2.6	44				

Table 2. Comparing 6 models with sharp features. Missing data means that the result was unavailable. Columns: algorithm, the algorithm that was used to generate the result; #disabled, number of feature edges for which the alignment constraint was disabled due to the local estimation; step inc, the direction of the step increment, positive or negative, i.e. performing the rounding in a bottom-up or top-down order.

where σ_1^{max} and σ_2^{min} are the maximal and minimal singular values over all mesh triangles. To optimize the layout w.r.t. to τ, we used the local-global approach with constraint convexification; see supplement for details. This objective was used in all the results unless stated otherwise. Even when other objectives were used, in the histograms and heat maps in the figures we used τ, which is strongly correlated with the other isometric distortion measures.

Runtime. The experiments were performed on a quad core 2.8GHz CPU. In the local step, the patches were about the same size (using default BFS level 5), and it took 30 milliseconds on average to lay out a patch. This means that for the model with the most patches in the tables, the local estimation part took in total about 200 seconds, and on average it took about 20 seconds. Besides that, the runtime was dominated by the layout stage, where we note in the table the average time for one layout iteration (the global step in the local-global scheme). The layout time for all the methods was roughly the same. Our method performed 5 batch-rounding steps (increasing the rounding error threshold), each requiring running the layout procedure, and in total, our method runtime was roughly 5 times slower than the other methods. For the other methods, we did not account for the time to generate the fields (e.g. Diamanti et al. [2015] report 0.8 to 40 minutes per model), and in the tables we note only the layout time.

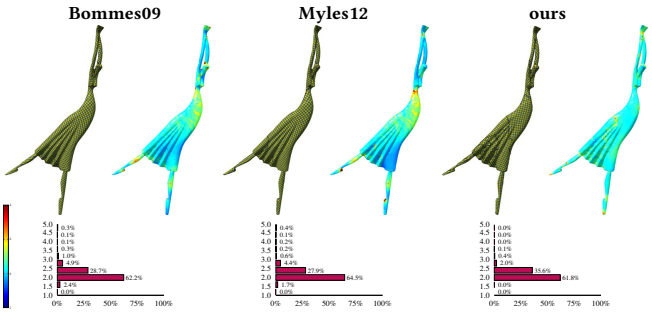


Fig. 8. Dancer. Minimizing the L_2-based ARAP energy. Keeping consistency with the other figures, histograms and heat maps show the distortion τ in (3), which is strongly correlated with the other isometric distortion measures. Bommes09 result contains 5 flips, and the average distortion is 0.052. Myles12 result contains 14 flips, and the average distortion is 0.046. Our result contains no flips, and the average distortion is 0.019.

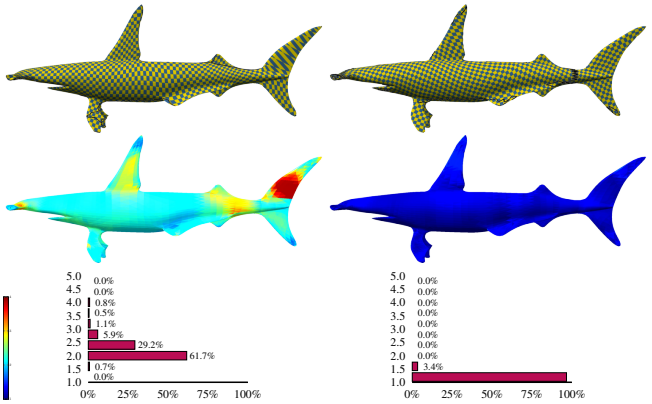


Fig. 9. Shark. Minimizing the L_∞-based ARAP energy. *Left:* Bommes09 with L_∞-ARAP = 0.76. *Right:* ours with L_∞-ARAP = 0.12. Histograms and heat maps show the distortion τ.

model	#tri	Bommes09	Myles12	Myles14	Diamanti15	Fang18	ours
dancer	18K	86	99	57	146	168	105
armadillo	100K	248	274	179	426	1208	357
dilo	54K	76	96	62	140	435	204
filigree	100K	827	889	549	877		1025
knot	100K	38	311	42	110	2693	212
dragon	105K	1136	1019	325	958	2180	1575
robocat	8K	230	247	84	300	379	338
shark	20K	65	83	72	187	370	105
twirl	10K	43	49	20	94	209	49
grayloc	69K	197	284	95	243		281
greek_sculpture	50K	299	314	112	416	1275	426
blade_earring	23K	74	148				155
octopus	33K	131	144				162
octocat	38K	170	183				248
linkCupTop	20K	330	501				695

Table 3. The number of cones of the 15 models.

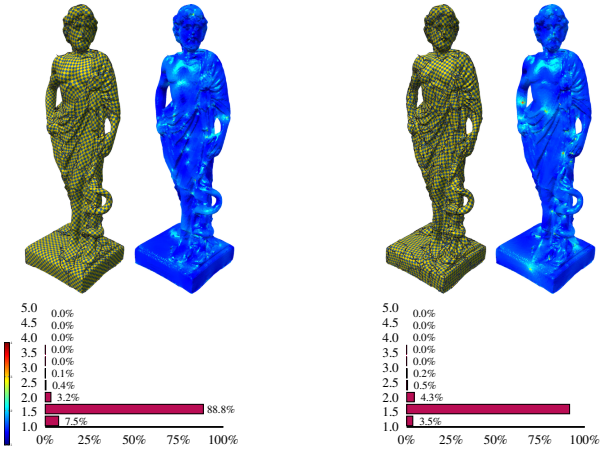


Fig. 10. Greek sculpture. Minimizing the symmetric Dirichlet energy, which approximately averages (weighted by triangle area) $2\tau^2$. *Left:* Diamanti15 with symmetric Dirichlet energy = 2.045 and τ = 3.48. *Right:* ours with symmetric Dirichlet energy = 2.040 and τ = 3.46. Histograms and heat maps show the distortion τ.

model	#tri	Bommes09	Myles13	Fang18	ours
beetle_refined	39K	47	498	46	31
casting_refined	37K	106	436	224	126
fandisk	14K	30	176	66	68
sculpt	7K	8	144	144	217
smooth-feature	12K	10	26	93	132
metatron	50K	80			188

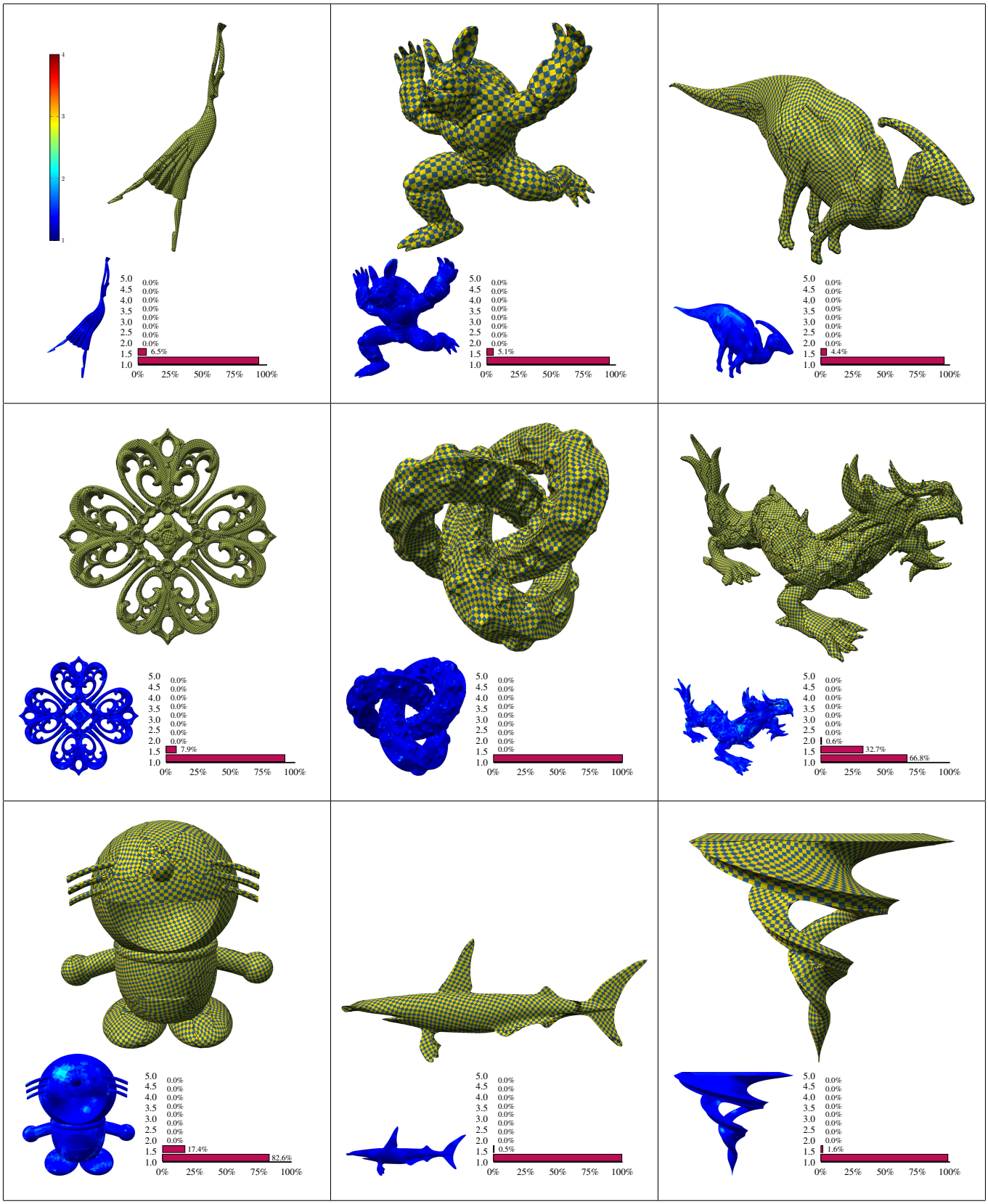
Table 4. The number of cones of the 6 models with sharp features.

Quad mesh generation. We demonstrated the application of quad meshing. We made a seamless parametrization integer by using the method described in the supplement. We chose a grid resolution that is fine enough not to compromise the mapping distortion significantly. This resolution depends on the minimal distance between cones and features. We did not optimize the field in that respect (e.g. performing cone collapse or relocation), which is a separate problem. See Fig. 13 for results.

Number of cones. In Tables 3 and 4, we provide statistics for the number of cones. At Table 3, we compare the methods with [Bommes et al. 2009], which smooths the field without metric consideration. The number of cones that are added on average by each method is: [Myles and Zorin 2012]:69%, [Diamanti et al. 2015]:73%, ours:83%, and [Fang et al. 2018]:989%.

8.1 Parameters

Rounding threshold. In Alg. 1-3, we initialized the rounding (matching and feature alignment) threshold to 0.1. Then, we increased it in steps of 0.1 until it reached 0.5 (the max rounding error). For the top-down variant, the threshold steps were 0.4, 0.3, 0.2, 0.1, 0. The local estimation routine was used in the last three steps (when the rounding threshold was ≥ 0.3 and for the top-down variant ≤ 0.2).


4
3
2
1
5.0 0.0%
4.5 0.0%
4.0 0.0%
3.5 0.0%
3.0 0.0%
2.5 0.0%
2.0 0.0%
1.5 6.5%
1.0
0% 25% 50% 75% 100%
5.0 0.0%
4.5 0.0%
4.0 0.0%
3.5 0.0%
3.0 0.0%
2.5 0.0%
2.0 0.0%
1.5 5.1%
1.0
0% 25% 50% 75% 100%
5.0 0.0%
4.5 0.0%
4.0 0.0%
3.5 0.0%
3.0 0.0%
2.5 0.0%
2.0 0.0%
1.5 4.4%
1.0
0% 25% 50% 75% 100%
5.0 0.0%
4.5 0.0%
4.0 0.0%
3.5 0.0%
3.0 0.0%
2.5 0.0%
2.0 0.0%
1.5 7.9%
1.0
0% 25% 50% 75% 100%
5.0 0.0%
4.5 0.0%
4.0 0.0%
3.5 0.0%
3.0 0.0%
2.5 0.0%
2.0 0.0%
1.5 0.0%
1.0
0% 25% 50% 75% 100%
5.0 0.0%
4.5 0.0%
4.0 0.0%
3.5 0.0%
3.0 0.0%
2.5 0.0%
2.0 0.6%
1.5 32.7%
1.0 66.8%
0% 25% 50% 75% 100%
5.0 0.0%
4.5 0.0%
4.0 0.0%
3.5 0.0%
3.0 0.0%
2.5 0.0%
2.0 0.0%
1.5 17.4%
1.0 82.6%
0% 25% 50% 75% 100%
5.0 0.0%
4.5 0.0%
4.0 0.0%
3.5 0.0%
3.0 0.0%
2.5 0.0%
2.0 0.0%
1.5 0.5%
1.0
0% 25% 50% 75% 100%
5.0 0.0%
4.5 0.0%
4.0 0.0%
3.5 0.0%
3.0 0.0%
2.5 0.0%
2.0 0.0%
1.5 1.6%
1.0
0% 25% 50% 75% 100%

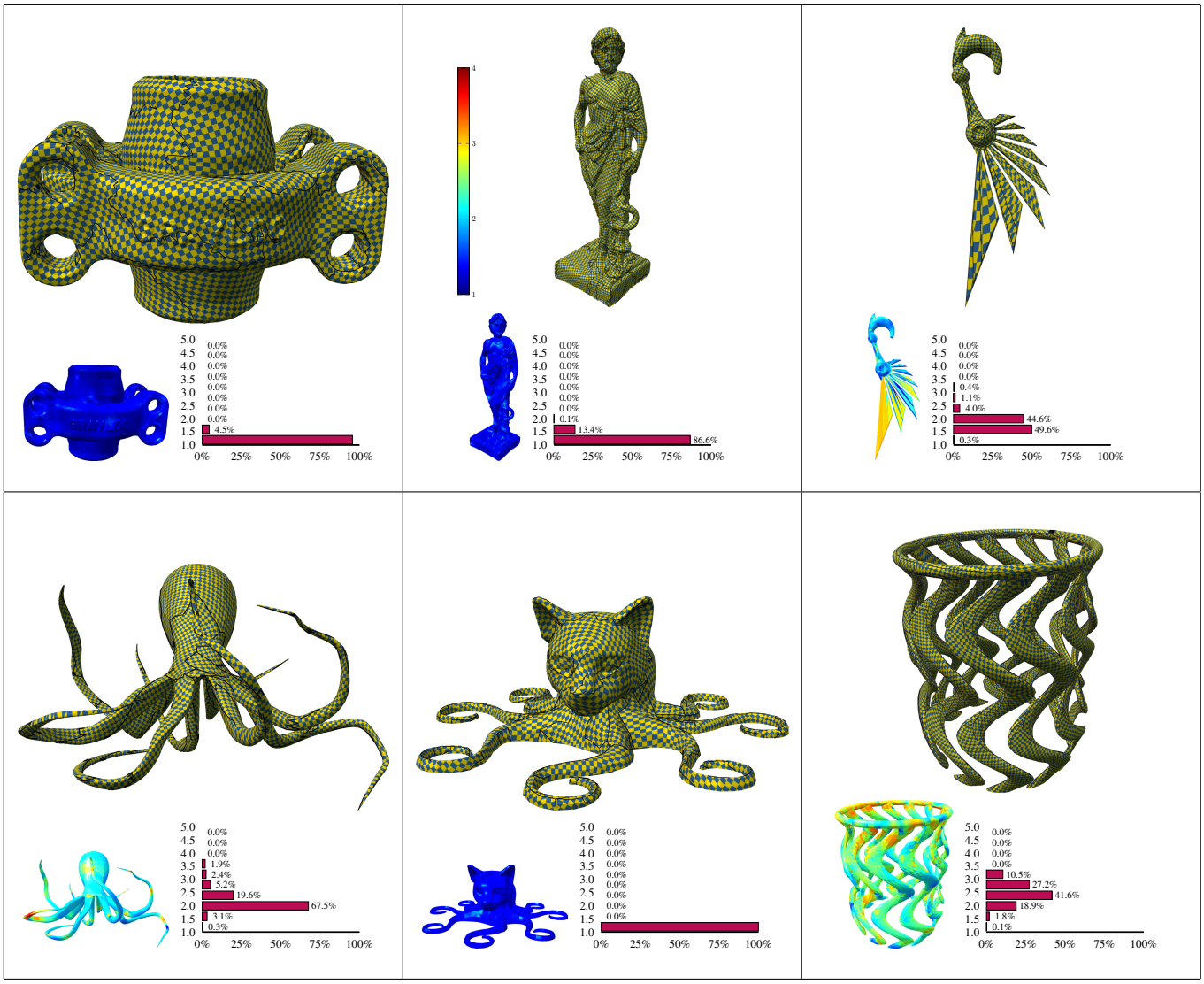


Fig. 11. A gallery of our method's results on the 15 models of our benchmark. Histograms and heat maps show the distortion τ.

After laying out a local patch of a feature edge, we used a threshold of 2.5 for the distortion measure τ to determine if the alignment constraint should be disabled.

Bommes et al. [2010] try to establish variable dependency for simultaneous rounding through a heuristic. Myles and Zorin [2013] round batches of variables where their sum of rounding error is less than 0.5. Our approach uses rounding error threshold that increases (or decreases) in predetermined steps. For our step choice (0.1), we are guaranteed to have no more than 5 batches (or iterations), which proved to be effective and is practical in terms of efficiency, which is important since we use a costly layout step to guarantee mapping quality. When using a simpler distortion measure, such as L_2-ARAP that requires a cheaper solution of a linear system, similar to [Bommes et al. 2009; Myles and Zorin 2012], we can round the matchings one-by-one, and the methods would perform similarly. As discussed before, though, the extreme of rounding one-by-one is not guaranteed to produce an optimal mapping. For example, results of rounding one-by-one the matchings (without local estimation) of the three smallest models in Table 1, minimizing τ: dancer, $\tau = 1.4$; robocat, failed; twirl, $\tau = 2.8$. Compare with Table 1.

Patch size. For the local estimation (Section 5), we used all triangles within a default *Breadth First Search (BFS) level* (or *neighbor ring level*) to create a patch surrounding a given edge. In BFS level 1 of an edge, we add all the faces incident to the two edge vertices. In each following BFS level, we add the incident faces of each vertex in the growing patch. As default, we used BFS level 5. The local estimation routine was run for every edge with rounding error greater than 0.3. Laying out a patch took 30ms on average.

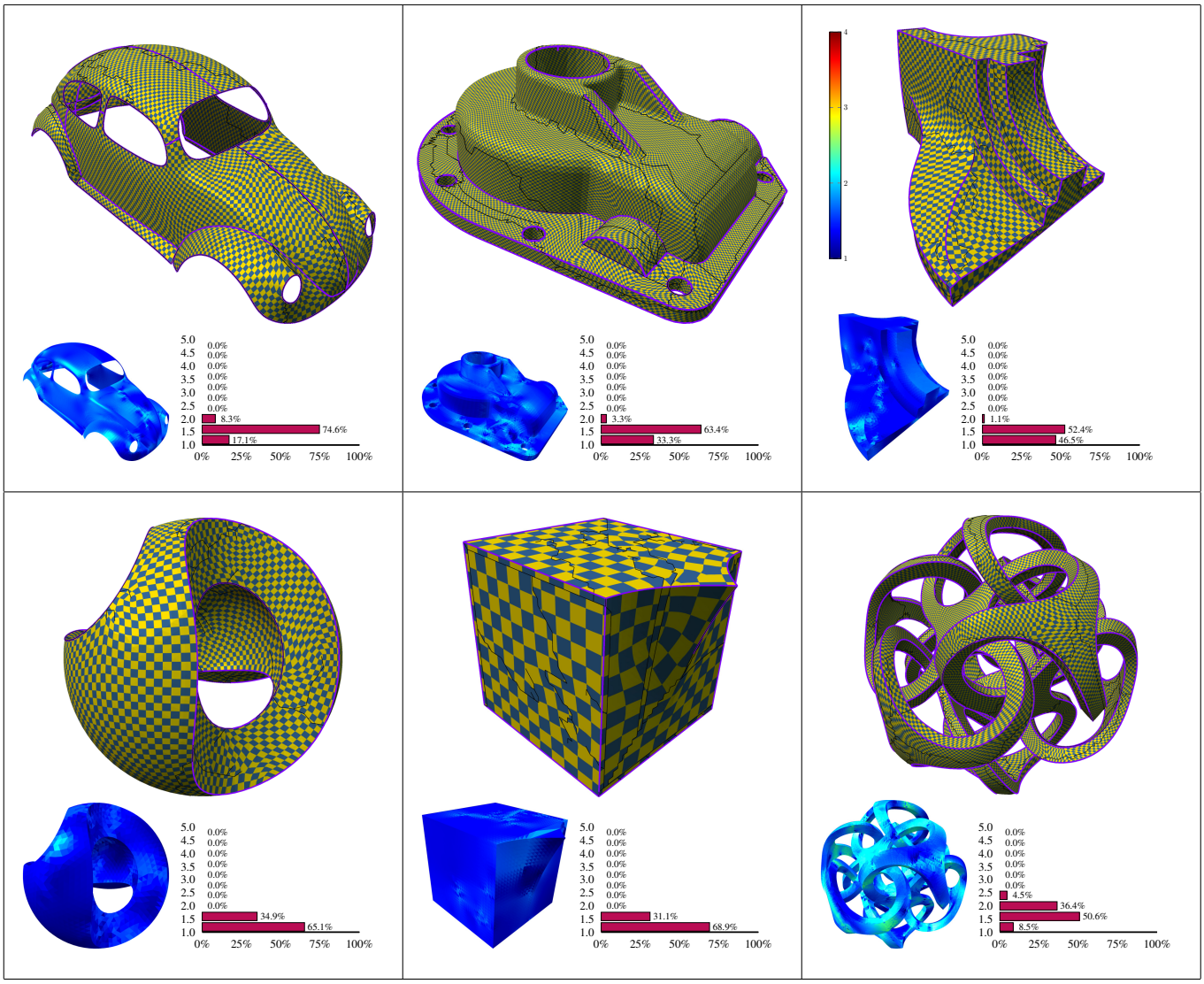


Fig. 12. Our method's results on 6 models with sharp features. Histograms and heat maps show the distortion τ.

For all the models in the paper, we used the default BFS level 5 except for the dilo model, where we increased the patch size to BFS level 8 to lower the distortion further in order to compete with the state-of-the-art (the default BFS level 5 resulted in $\tau = 1.9$). The influence of the BFS level can be seen in Fig. 15. While a small patch may be less reliable, as we consider larger patches, the estimation gets better, and the distortion is improved as a consequence until it is stabilized.

This is one advantage of our method: on most models it runs robustly with the default parameters, but if needed, the user can choose from several variations and parameter settings to improve the result: the local patch size, the rounding step size, and the rounding order can all be customized.

For illustration of how a different set of parameters affects the result, see supplemental Gallery 1.

8.2 Compared Methods

In Tables 1 and 2, we compare with the state-of-the-art that targets isometry.

Cross fields. For the relevant methods, the smoothest field was generated using a seed constraint triangle. Given the fields (or otherwise, a field was extracted from the parametrization result; see supplement), we used the same layout optimization method that minimizes the same target objective (instead of using the compared method layout, which targets a different objective). That is, it would be unfair to compare their parametrization result as is to ours since their objective is not the same as ours (which varies according to the selected distortion measure). For all of the methods, we used the generated field without further processing such as cone collapse

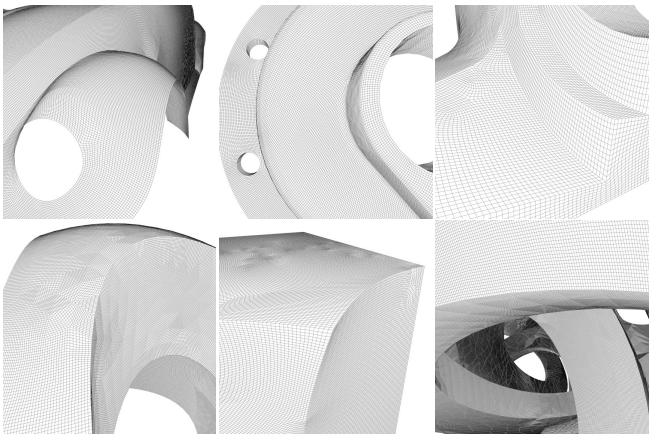


Fig. 13. Quad mesh generation for the models in Fig. 12.

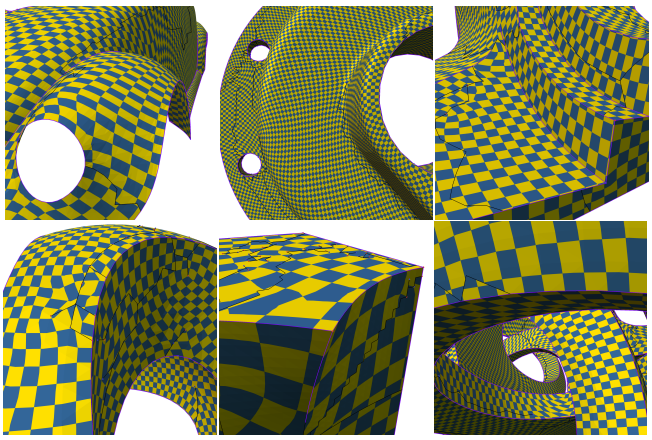


Fig. 14. The same camera view on the parametrization as in Fig. 13.

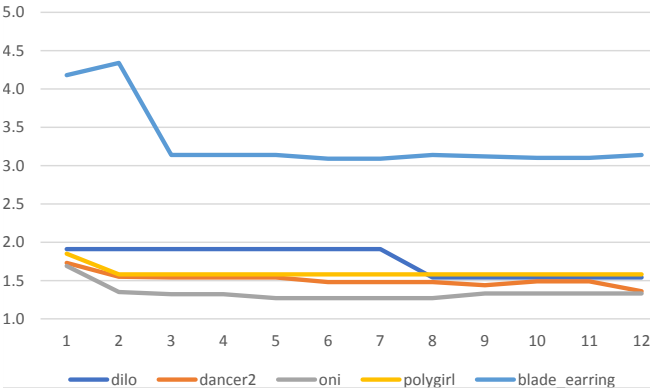


Fig. 15. Patch size influence. The graph shows the distortion amount τ as a function of the BFS level for a few models.

(which may result in a better or worse field).

[Fang et al. 2018]. We used the supplied quad mesh results. To extract the parametrization and measure the distortion, we followed the procedure in the supplement. Since this method has a more restrictive objective (an integer seamless parametrization vs a seamless parametrization for the other methods), the comparison is unfair. Yet, we illustrate here that using this approach, it would be hard, for example, to bound the maximal distortion as in our method. Given the extracted parametrization, we also experimented with extracting the field from it and creating a new layout, in the same way that we did with the other methods. Nevertheless, in most cases, the result still had significantly higher maximal distortion in comparison, and we opted for showing the original mapping. The mapping mostly preserves the shape of the squares, but not the scaling (i.e., it tends to be more conformal). In some models, the color map is almost fully red; see Fig. 5, 6. This is due to the τ measure, which is sensitive to the maximal isometric distortion. That is, bad distortion in even one small region of the map is enough to influence and scale the optimal τ.

[Myles et al. 2014]. The method does not generate a field, but creates an initial feasible layout for a given one (adding cones if needed). The input fields that were used in the paper were a version of [Bommes et al. 2009] (after some post-processing to collapse nearby cones).

[Myles et al. 2014] allows refinement, which we do not (see supplement for discussion), which can simplify impossible situations. The amount of refinement is significant, e.g. 237,000 triangles in the filigree result vs 100,000 in the original model.

While their mapping is guaranteed to be locally injective, in practice there could be nearly-collapsed triangles. For example, the distortion of their supplied mapping result for the filigree and the twirl models is $\tau = 82{,}000$ and $\tau = 56{,}000$ respectively. Extracting frames from these two mappings and feeding them to the layout optimization that minimizes τ resulted in a failure (due to numerical issues).

Since [Myles et al. 2014]'s solution is general, one can apply it to any field, including ours. That is, if our method generates parametrization with foldovers (using a distortion measure that allows it, e.g. L_∞-ARAP), then a field can be extracted from the mapping. The field would need to be corrected (due to the foldovers), e.g. using the matching consistency procedure in the supplement, which guarantees a correct output field. The method in [Myles et al. 2014] can be applied to the corrected field.

8.3 Benchmark Models

Myles et al. [2014] compiled a dataset of 113 models for benchmarking. We find that most of the models are too easy to distinguish state-of-the-art methods and do not represent potential challenges arising in the parametrization problem. From their dataset, we removed all the easy models, which recent state-of-the-art methods parametrized successfully with low distortion. We augmented these with a few more challenging models from other sources [Zhou and Jacobson 2016], where we consider a model challenging if one or

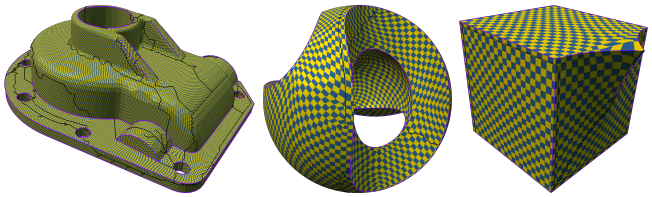


Fig. 16. Using the default Alg. 2 with positive step direction on the four models in Fig. 12 (where a different algorithm or step direction were used). Distortion τ: casting_refined—2.7, sculpt—1.7, smooth-feature—4. This variation failed on the metatron model.

more of the state-of-the-art methods had difficulties generating a locally injective, low distortion mapping. We ended up with a manageable dataset of 15 challenging benchmark models. From Table 1, we can see that our method generates a bijective parametrization successfully on all the models, and the mapping, in general, has lower distortion compared to the other methods.

In the supplemental Gallery 2, we provide results of the rest of [Myles et al. 2014] dataset. Our method achieves maximal distortion of $\tau \leq 1.6$. A comparison to [Diamanti et al. 2015] is included, where it achieves similar low distortion on most of the models, and visually the differences between the two methods are insignificant.

In the supplemental Gallery 1, we experimented with a different set of parameters to decrease the runtime. The cost is one failure and more distortion in some of the models.

8.4 Feature Alignment

There are several ways to detect sharp features, and when comparing methods, it is important that they all use the same features. We mostly detected sharp features by looking for dihedral angles greater than 40° (and discarding sharp edges which created impossible configurations).

We offer a set of 6 models to benchmark feature alignment; see Table 2. When choosing a challenging model for benchmarking feature alignment, one can analyze their distribution (e.g. proximity to cones or each other), their pattern, or the number of corners the model has, as analyzed, for example, in [Huang et al. 2008] for the fandisk model. Instead, as before, we chose models for which some of the candidate methods failed to generate a locally injective parametrization with low distortion.

Our method succeeded in generating a bijective mapping for all the models even if it did not always achieve lowest distortion. As long as the distortion was reasonable compared to the other methods, we opted for using the default Alg. 2. Only when the need arose, we experimented with Alg. 3, as well as the step increment direction of the rounding threshold (Section 6). In Fig. 16, we show the result of only the default variation of the algorithm. Without the challenge of feature alignment, Alg. 1 produced compelling results without the need for variations.

8.5 Other Distortion Measures

We experimented with other distortion measures. See supplement for a distortion measure review.

In Fig. 8, we experimented with lowering the L_2-ARAP energy (see supplement). Bommes et al. [2009] and Myles and Zorin [2012] use this objective in the layout step, and it is inconsistent with the objective these methods use for field design, as explained in the introduction, and our method performs better as a consequence.

Since the distortion is not bounded, the two other methods contain flips (which are not penalized). Our method is prone to flips like the other two (since we solve for the same L_2-norm-based objective without additional constraints), but our rounding strategy by local estimation is robust enough that the mapping result has distortion below the foldover-free threshold (see supplement).

In Figures 9 and 10, we show minimization of the L_∞-ARAP and symmetric Dirichlet energy respectively. We used the compared method results as initialization to the layout optimization method that targets each energy ([Levi and Zorin 2014] and [Shtengel et al. 2017] respectively). In our method, we used each layout optimization method in the whole pipeline.

Fig. 2 illustrates conformal distortion measure.

9 CONCLUSION

We introduced a method for seamless parametrization that formulates a single problem with a single objective. This is made possible by using domain variables.

In terms of limitations, the method is not guaranteed to produce a locally injective mapping. Furthermore, the method parameters determine its success, and due to the mixed-integer nature of the problem, it is hard to determine the optimal ones. Nevertheless, a default set of parameters is offered that works well in most cases, and otherwise the method offers the flexibility to tweak parameters in order to reach the desired quality.

Our method relies on a rounding strategy based on local estimation. The idea of using local estimation as a rounding strategy can be further explored:

- *Rounding of seamless parametrization.* This requires rounding the cone UV coordinates as well as the translations between twin edges. Since the rounding is usually to the nearest integer, local estimation can be used to efficiently test the reduced options.
- *Improving the cone distribution.* Through efficient local estimation, it is possible to test the effects of cone collapse or repulsion (relocating cones and increasing the distance between them) before applying them. These operations are usually necessary if one needs to round a seamless parametrization to a coarse grid.

While we demonstrated our method on a cross field, it can be applied to generate any n-field. Furthermore, it would be interesting to apply these concepts to tetrahedral meshes. Also, addressing other quad meshing attributes can be considered using the proposed paradigm.

ACKNOWLEDGMENTS

Thanks to Julian Panetta, Roi Poranne, and Pondy for their comments.

REFERENCES

Aigerman Noam, Poranne Roi, and Lipman Yaron. 2014. Lifted bijections for low distortion surface mappings. *TOG* 33, 4 (2014), 69.

Bommes David, Lévy Bruno, Pietroni Nico, Puppo Enrico, Silva Claudio, Tarini Marco, and Zorin Denis. 2013. Quad-Mesh Generation and Processing: A Survey. *CGF* 32, 6 (2013), 51–76.

Bommes David, Zimmer Henrik, and Kobbelt Leif. 2009. Mixed-integer Quadrangulation. *ACM Trans. Graph.* 28, 3 (2009), 77:1–77:10.

Bommes David, Zimmer Henrik, and Kobbelt Leif. 2010. Practical Mixed-Integer Optimization for Geometry Processing. In *Curves and Surfaces*, Vol. 6920. 193–206.

Campen Marcel, Bommes David, and Kobbelt Leif. 2015. Quantized Global Parametrization. *ACM Trans. Graph.* 34, 6, Article 192 (2015), 192:1–192:12 pages.

Chien Edward, Levi Zohar, and Weber Ofir. 2016. Bounded Distortion Parametrization in the Space of Metrics. *ACM Trans. Graph.* 35, 6 (2016), 215:1–215:16.

Diamanti Olga, Vaxman Amir, Panozzo Daniele, and Sorkine-Hornung Olga. 2014. Designing N-PolyVector Fields with Complex Polynomials. *Comput. Graph. Forum* 33, 5 (2014), 1–11.

Diamanti Olga, Vaxman Amir, Panozzo Daniele, and Sorkine-Hornung Olga. 2015. Integrable PolyVector Fields. *ACM Trans. Graph.* 34, 4 (2015), 38:1–38:12.

Fang Xianzhong, Bao Hujun, Tong Yiying, Desbrun Mathieu, and Huang Jin. 2018. Quadrangulation Through Morse-parameterization Hybridization. *ACM Trans. Graph.* 37, 4 (2018), 92:1–92:15.

Huang Jin, Zhang Muyang, Ma Jin, Liu Xinguo, Kobbelt Leif, and Bao Hujun. 2008. Spectral Quadrangulation with Orientation and Alignment Control. *ACM Trans. Graph.* 27, 5, Article 147 (Dec. 2008), 9 pages.

Jiang Tengfei, Fang Xianzhong, Huang Jin, Bao Hujun, Tong Yiying, and Desbrun Mathieu. 2015. Frame Field Generation Through Metric Customization. *ACM Trans. Graph.* 34, 4 (2015), 40:1–40:11.

Knöppel Felix, Crane Keenan, Pinkall Ulrich, and Schröder Peter. 2013. Globally optimal direction fields. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 59.

Kovacs Denis, Myles Ashish, and Zorin Denis. 2011. Anisotropic quadrangulation. *Computer Aided Geometric Design* 28, 8 (2011), 449–462.

Levi Zohar and Zorin Denis. 2014. Strict minimizers for geometric optimization. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 185.

Li Minchen, Kaufman Danny M., Kim Vladimir G., Solomon Justin, and Sheffer Alla. 2018. OptCuts: Joint Optimization of Surface Cuts and Parameterization. *ACM Trans. Graph.* (2018).

Lipman Yaron. 2012. Bounded distortion mapping spaces for triangular meshes. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 108.

Myles Ashish, Pietroni Nico, and Zorin Denis. 2014. Robust Field-aligned Global Parametrization. *ACM Trans. Graph.* 33, 4, Article 135 (2014), 14 pages.

Myles Ashish and Zorin Denis. 2012. Global parametrization by incremental flattening. *TOG* 31, 4 (2012), 109.

Myles Ashish and Zorin Denis. 2013. Controlled-distortion constrained global parametrization. *TOG* 32, 4 (2013), 105.

Panozzo Daniele, Puppo Enrico, Tarini Marco, and Sorkine-Hornung Olga. 2014. Frame fields: anisotropic and non-orthogonal cross fields. *ACM Trans. Graph.* 33, 4 (2014), 134:1–134:11.

Poranne Roi, Tarini Marco, Huber Sandro, Panozzo Daniele, and Sorkine-Hornung Olga. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Trans. Graph.* (2017).

Rabinovich Michael, Poranne Roi, Panozzo Daniele, and Sorkine-Hornung Olga. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 4 (2017).

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

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

Vaxman Amir, Campen Marcel, Diamanti Olga, Panozzo Daniele, Bommes David, Hildebrandt Klaus, and Ben-Chen Mirela. 2016. Directional Field Synthesis, Design, and Processing. *CGF* (2016).

Zhou Qingnan and Jacobson Alec. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).