DOI: 10.1111/cgf.14696
Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 41 (2022), Number 7

Efficient Texture Parameterization Driven by Perceptual-Loss-on-Screen

Haoran Sun[1], Shiyi Wang[1], Wenhai Wu[2], Yao Jin[2], Hujun Bao[1] and Jin Huang[1†]

[1]State Key Lab of CAD&CG, Zhejiang University, Hangzhou, China
[2]School of Computer Science and Technology, Zhejiang Sci-Tech University, Hangzhou, China

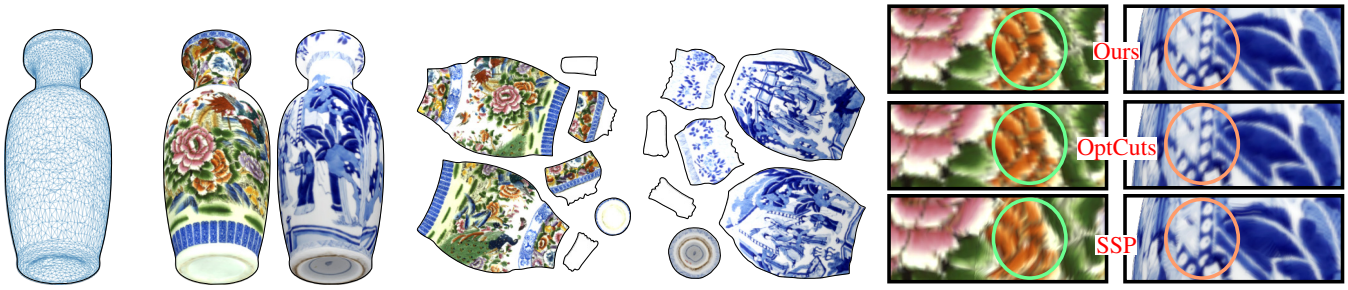

Figure 1: *Given the same surface (a) and different signals (b), our perceptual driven method can generate different parameterizations optimized for the texture (c). As shown in (d), our parameterizations can achieve lower perceptual loss with same number of pixels than the parameterizations from other automatic methods, including geometry only methods (e.g. OptCuts [LKK*18] leads to blur problem) and previous signal-aware methods (e.g. SSP [SGSH02] has aliasing artifacts). For the explanation of the blur and aliasing artifacts mentioned, please refer to Figure 2.*

Abstract

Texture mapping is a ubiquitous technique to enrich the visual effect of a mesh, which represents the desired signal (e.g. diffuse color) on the mesh to a texture image discretized by pixels through a bijective parameterization. To achieve high visual quality, large number of pixels are generally required, which brings big burden in storage, memory and transmission. We propose to use a perceptual model and a rendering procedure to measure the loss coming from the discretization, then optimize a parameterization to improve the efficiency, i.e. using fewer pixels under a comparable perceptual loss. The general perceptual model and rendering procedure can be very complicated, and non-isotropic property rooted in the square shape of pixels make the problem more difficult to solve. We adopt a two-stage strategy and use the Bayesian optimization in the triangle-wise stage. With our carefully designed weighting scheme, the mesh-wise optimization can take the triangle-wise perceptual loss into consideration under a global conforming requirement. Comparing with many parameterizations manually designed, driven by interpolation error, or driven by isotropic energy, ours can use significantly fewer pixels with comparable perception loss or vise vesa.

Keywords: Geometric Modeling, Surface Parameterization, Texture Mapping, Perceptual Loss

CCS Concepts
• **Computing methodologies** → *Shape modeling;*

† Corresponding author. E-mail address: hj@cad.zju.edu.cn

1. Introduction

In computer graphics, parameterization is a technique for mapping a 3D mesh to a 2D domain. Given a signal (e.g. diffuse color) on a mesh, one can map it along with the mesh to the 2D domain, then discretize the 2D signal by sampling it at pixels in a texture image. Increasing the density of pixels, i.e. using a larger texture image, can reduce the discretization error, but increases the cost in storage, memory and transmission etc. To use fewer pixels to faithfully represent the signal on a mesh, some previous works [SGSH02, TSS*04, CW15] propose to guide the parameterization by interpolation error. Roughly speaking, triangles containing non-smooth signals require denser sampling than the ones containing smooth signals.

We argue that the common polynomial interpolation error used in previous works is not good enough. This error does not take the downstream rendering procedure and human's perception [XZMB13] into account. For instance, comparing with low frequency error, the same amount of high frequency error could be less visible especially when the camera is relatively far away (or screen resolution is relatively low). Besides, although the discretization defined by a texture image looks uniform, actually is not isotropic because the texture image is composed of square-shaped pixels in a square lattice structure. As a consequence, rotating the parameterized mesh will change the error. However, all the previous works ignore this point, and just use a metric to characterize the optimal parameterization for each triangle. This brings artifacts (e.g. aliasing) for textures containing strong sharp edges.

Thus, we replace the previous "numerical" error by *perceptual-loss-on-screen* which involves a customizable rendering procedure $\mathcal{H}$ and a full-reference image quality assessment (FR-IQA) $\mathcal{D}$. The basic idea is to take a rendering image as the ground truth. Then, the quality of another image is measured by the visual difference between them, which helps find the *blur* problem (See Figure 2). This error is also aware of *aliasing artifacts*, so it helps to optimize the orientation of triangles in the texture domain.

Figure 2: *Left: high quality rendering; Middle: medium quality rendering (blur); Right: low quality rendering with aliasing artifacts. Aliasing artifacts are easily recognizable even without comparison while blur means the loss of high frequency details and usually must be recognized by comparison with the high quality rendering.*

The perceptual loss and rendering procedure are generally complicated, so we adopt a two-step strategy (see Figure 3). First, in the triangle-wise step, we find the optimal local parameterization (not only a metric) for each triangle via Bayesian optimization. To make the problem friendly for Bayesian optimization, we carefully re-parametrize the deformation Jacobian from a common 2×2 matrix into a 4D box-constrained vector space. Then, in the mesh-wise step, we solve the global parameterization best matching the local optimal deformation Jacobian. To further improve the results, we propose to use perceptual loss guided weighting scheme to replace commonly used Frobenius norm when measuring the deviation to the local optimal deformation Jacobian. Results show that using our parameterization, one can use less number of pixels to achieve comparable visual similarity or vise vesa. In short, our parameterization is more efficient.

In summary, the novelty and contribution of this work include:

- guiding the parameterization by customizable perceptual loss and rendering procedure,
- re-parametrizing the deformation Jacobian in favor of Bayesian optimization in the triangle-wise step,
- and a perceptual loss induced weighting scheme in the mesh-wise parameterization.

2. Related work

Parameterization is a fundamental problem of geometry processing, and many methods have been proposed. A lot of topics about parameterization have been explored, such as kinds of distortion measurements, constraints like bounded distortion, cross parameterization between different meshes, high performance numerical methods, and its downstream applications like remeshing [RLL*06, BZK09], texture synthesis [YHBZ01, LH06] etc. Some representative works are introduced in surveys like [SPR06].

Parameterization for texture mapping Most parameterization methods measure the quality by distortion criteria like isometric or conformal, and suggest that such a measurement is good for texture mapping. However, it does not generally hold. Even some texture packing methods [NS11, LVS18, LFY*19] try to reduce useless pixels in the texture images. They keep the resulting parameterization locally as-isometric-as-possible to the input one, so the number of pixels covered by the mesh, i.e. the useful pixels, in the texture image will not change significantly. A few signal-aware methods [SWB98, SGSH02, CW15] argue that the mapping should be adaptive and anisotropic according to the signal field, i.e. allocates more texture samples in areas with greater signal details. [SWB98] warps the texture domain to evenly distribute a user-defined importance field. [BBT03] automatically computes the field by wavelet-based technique to guide the texture optimization. Unlikely, [SGSH02, TSS*04] compute signal-aware parameterization by minimizing a signal approximation error. Noticing the error of representing the signal are rooted in the nature of discretization by limited samples, and the desired anisotropic density is usually induced from the polynomial interpolation error. [CW15] computes an importance map from averaged gradient magnitudes, which indeed is also about the error of piece-wise constant interpolation. [MA10] presents a perception-based metric to generates space-optimized texture atlases from 3D scenes with per-polygon textures. Such methods can successfully reduce the number of useful pixels without significantly sacrificing the signal reconstruction error. We take a similar view, but go a step further that measures the loss at the end of the rendering procedure with a human perceptual

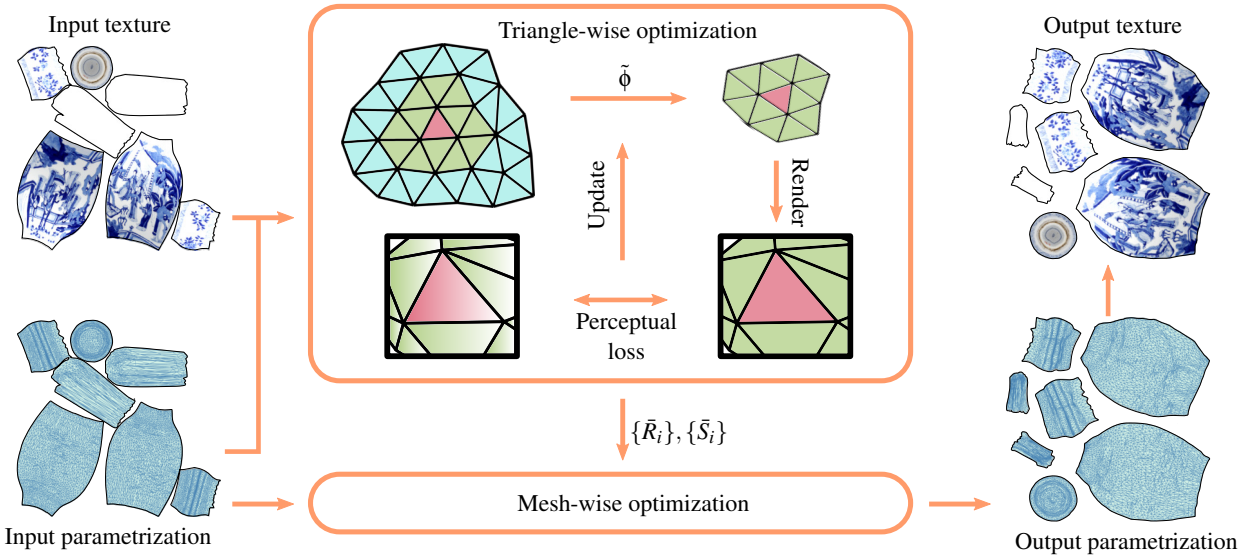

Figure 3: *Overview of our approach. With the input parameterization and texture (left), our method first optimizes the efficiency for each triangle using perceptual-loss-on-screen (middle top). Then we solve a mesh-wise optimization to fit the triangle-wise result (middle bottom) and obtain the optimized parameterization and texture in the end (right).*

model. We also noticed that end-to-end fashion get a lot of attention in recent years. Differential rendering is obviously related to our work. Such methods [GLD*19, BLD20] often explicitly model the whole rendering procedure, so the derivatives of the loss with respect to a few variables can be computed. However, the problem of parameterization involves large number of variables (the texture coordinates of every vertex), and thus they are still very costly even if computing the derivatives is feasible.

Local-global parameterization Local-global strategy is a common alternating optimization method, and widely used in geometric processing [SA07, BDS*12]. Typically, the optimal local Jacobian is computed on each element without the consideration of conforming condition, then a Poisson-like problem guided by these Jacobians is solved to obtain a conforming mesh. For example, in as-rigid-as-possible parameterization [LZX*08], the optimal local Jacobian is optimized to be the rotation best matching the current local Jacobian in the conforming mesh, then the conforming mesh is updated by minimizing the difference between the Jacobian of each triangle and the local rotation. The local problem on each element usually involves small number of variables, so one can use more complicated objective function and constraints here. For example, [RPPSH17] enhances the local energy for flip-preventing. Our method takes a two-stage strategy. First we optimize a local deformation for each triangle (triangle-wise optimization), then solve a global deformation (mesh-wise optimization). There is no iteration at this level, so it is not a typical local-global optimize strategy. But we can exploit the advantage of local-step and formulate the perceptual loss in the triangle-wise step. In the global step, Frobenius-norm [LZX*08, LYNF18] is generally used to measure the deviation of the Jacobian to the optimal local one. Though it works well in many situations, our triangle-wise problem related measurement leads to better results in our mesh-wise problem. Our mesh-wise optimization takes typical local-global iterations.

3. Problem statement

For a 3D triangular mesh $M \subset \mathbb{R}^3$, a single channel signal (e.g. diffuse) $\hat{f} : M \to \mathbb{R}$ on it is usually provided by a texture mapping $(\hat{\psi}, \hat{I}^t)$: a parameterization $\hat{\psi}$ and a texture image $\hat{I}^t$. For convenience, we assume that parameterization $\hat{\psi} : M \to [0,1]^2$ maps M to a subset of a unit square, and the texture image of resolution $\hat{r} \times \hat{r}$ (with possible padding) covers this square. In other words, each pixel covers a square with edge length $1/\hat{r}$ in the texture domain, and number of pixels covered by $\hat{M}^t = \hat{\psi}(M) \subset [0,1]^2$ is about $\hat{r}^2|\hat{M}^t|$. The texture domain signal is a 2D function $\hat{f}^t = \mathcal{S}(\hat{I}^t) : [0,\hat{r}]^2 \to \mathbb{R}$ interpolated from $\hat{I}^t$ according to certain interpolation scheme $\mathcal{S}$ (e.g. bi-linear), so signal $\hat{f}$ at $p \in M$ is $\hat{f}(p) = \hat{f}^t(\hat{r}\hat{\psi}(p))$, i.e. $\hat{f} = \mathcal{S}(\hat{I}^t) \circ \hat{r} \circ \hat{\psi}$. Figure 4 shows the illustration of some important symbols used in this paper.

Using another parameterization ψ along with a new texture image I^t with resolution $r \times r$ will change the texture domain signal $\hat{f}^t$ and surface signal $\hat{f}$. The difference is estimated based on piecewise constant interpolation [SGSH02], piece-wise linear interpolation [TSS*04], or averaged gradient magnitudes [CW15]. Guided by such error estimations, these methods stretch each triangle in the texture domain by a signal-stretch metric so that the error is uniformly distributed on the mesh.

This error does not take the downstream rendering procedure and human's perception into account. For instance, comparing with low frequency error, the same amount of high frequency error could be less visible if the camera is relatively far away (or screen resolu-

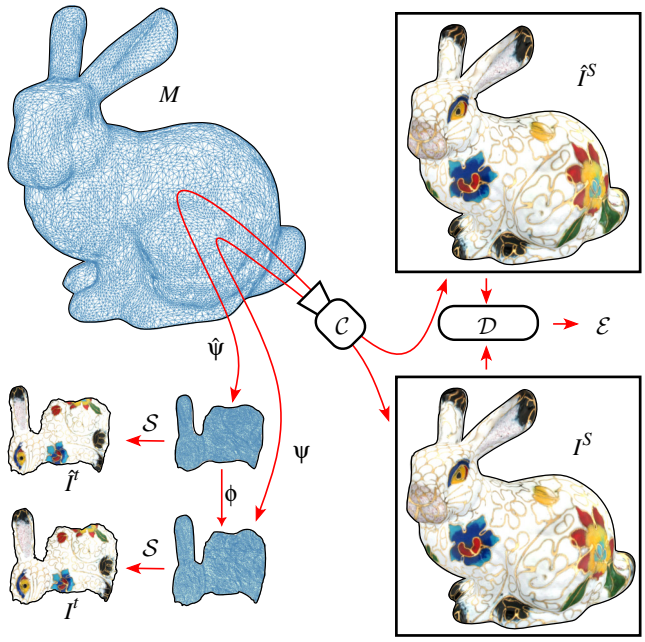

Figure 4: *Illustration of some important symbols.*

tion is relatively low). Thus, we replace the previous "numerical" error by *perceptual-loss-on-screen* which involves a customizable rendering procedure $\mathcal{H}$ and a Full-reference image quality assessment (FR-IQA) $\mathcal{D}$. Roughly speaking, the mesh M along with the texture mapping $(\hat{\psi},\hat{I}^t)$ can be rendered into a screen image $\hat{I}^s$ via a rendering procedure $\mathcal{H}$ involving a camera $\mathcal{C}$ and other rendering settings (i.e. screen resolution):

$$\hat{I}^s = \mathcal{H}(M,\mathcal{C},\hat{\psi},\hat{I}^t). \tag{1}$$

We can also get the rendering result from ψ and I^t with the same procedure:

$$I^s = \mathcal{H}(M,\mathcal{C},\psi,I^t). \tag{2}$$

The difference between the two rendering results, i.e. *perceptual-loss-on-screen*, is

$$\mathcal{E}_{\mathcal{C},\mathcal{H}}(\psi,I^t) = \mathcal{D}(\hat{I}^s,I^s). \tag{3}$$

Indeed, the texture image I^t under a given resolution $r \times r$ can be formulated as a function of parameterization ψ via

$$I^{t*} = \arg\min_{I^t} \mathcal{E}_{\mathcal{C},\mathcal{H}}(\psi,I^t), \tag{4}$$

and then *perceptual-loss-on-screen* of ψ under $r,\mathcal{C}$ and $\mathcal{H}$ is

$$\mathcal{E}_{r,\mathcal{C},\mathcal{H}}(\psi) = \mathcal{E}_{\mathcal{C},\mathcal{H}}(\psi,I^{t*}). \tag{5}$$

3.1. The two-stage approach

Obviously, $\mathcal{E}$ is camera-dependent, and one may immediately notice that there does not exist a method of selecting $\mathcal{C}$ where every triangle $\triangle \subset M$ contributes to the rendering for general meshes due to the perspective and hidden face removal. So it is not suitable for an optimization objective on the whole mesh.

Therefore, we would take a similar two-stage strategy with [SGSH02], where we firstly optimize the parameterization ψ_i for $\mathcal{E}$ on every individual triangle $\triangle_i$ (Section 4.1), and then solve a mesh-wise optimized ψ matching ψ_i without direct involve of $\mathcal{E}$(Section 4.2). Later, we would verify the efficiency of ψ on the view of the whole mesh with $\mathcal{E}$ for several testing cameras in experiments (Section 5).

4. Method

Without loss of generality, we will elaborate our method as a reparametrization problem regarding the input parameterization ψ composed of per-triangle deformations ϕ_i, i.e. we would like to find a locally injective mapping ϕ such that $\psi = \phi \circ \hat{\psi}$ is the optimized parameterization. Such mapping ϕ can be viewed as a deformation on the parameterization mesh $\hat{M}^t$.

4.1. Triangle-wise optimization for ϕ_i

Now we would try to improve the efficiency for a single $\triangle_i \in M$. A strightforward way is to reduce the number of pixels with the same loss $\mathcal{E} = \varepsilon$. When texture resolution r is fixed as $r = \hat{r}$, the number of pixels is approximately proportional to the area of the triangle, so we can write the objective as:

$$\begin{aligned} &\min_{\phi_i} \det(\nabla\phi_i)|\hat{\triangle}_i^t|, \\ &\text{s.t.} \quad \mathcal{E}(\tilde{\phi}\circ\hat{\psi}) = \varepsilon, \quad \det(\nabla\phi_i) > 0, \end{aligned} \tag{6}$$

where $\tilde{\phi}_j = \phi_i, \forall \triangle_j$. Note that the triangles used for rendering here are the whole mesh instead of a single triangle $\triangle_i$, which would help avoid ambiguity pixels at boundary of $\triangle_i$. Practically, due to the setting of the camera described in Section 4.1.1, we can only take one-ring neighbors that share a vertex with $\triangle_i$.

However, for triangle with simple signal (e.g. constant), $\mathcal{E}$ may always be less than ε, so a more reasonable way is to relax it to an inequality constraint:

$$\begin{aligned} &\min_{\phi_i} \det(\nabla\phi_i)|\hat{\triangle}_i^t|, \\ &\text{s.t.} \quad \mathcal{E}(\tilde{\phi}\circ\hat{\psi}) \le \varepsilon, \quad \det(\nabla\phi_i) > 0, \end{aligned} \tag{7}$$

which turns out to be a bounded error compression problem on each triangle, and thus improves the efficiency (see Figure 5 for the intuition). In the following, we first introduce the computation of $\mathcal{E}$ and then present the Bayesian optimizing strategy for solving an optimal ϕ_i.

4.1.1. Computation of perceptual-loss-on-screen

There are several abstract procedures involved in Equation (5). We would describe the one used in this paper of the triangle-wise *perceptual-loss-on-screen* in this section.

Camera. The camera $\mathcal{C}_i$ used is a parallel camera pointing towards the center of the triangle along its normal direction and the visible region is roughly the bounding box of $\triangle_i$(details can be found in Appendix A). This setting would allow every direction to contribute to the rendering equally.

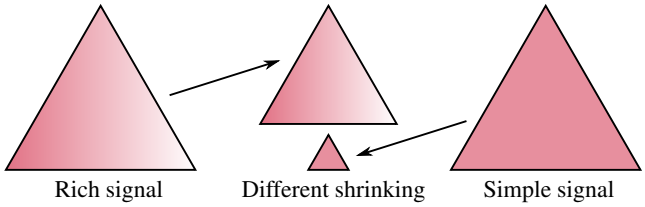

Figure 5: *For two triangles of the same size in the input parameterization, one with rich signal and another with simple signal, they would be shrinked to different sizes in the triangle-wise optimization. Then we can use relatively fewer pixels for the simple signal, and the efficiency is thus improved.*

Rendering and interpolation. The rendering procedure $\mathcal{H}$ used is a simple rasterization that does not consider lighting:

$$\begin{aligned} I_i^s &= \mathcal{H}(M, C_i, \tilde{\phi} \circ \hat{\psi}, I^t) \\ I_i^s(q) &= (\mathcal{S}(I_i^t) \circ \hat{r} \circ \tilde{\phi} \circ \hat{\psi} \circ \mathcal{C}^{-1})(q), \text{ for all pixel } q \in \triangle^s. \end{aligned} \tag{8}$$

The interpolation is the common bi-linear interpolation:

$$\begin{aligned} \mathcal{S}(I_i^t)(p) = (1-u)(1-v) I_i^t(\lfloor p \rfloor) + (1-u) v I_i^t(\lfloor p \rfloor + (1,0)) \\ + u(1-v) I_i^t(\lfloor p \rfloor + (0,1)) + uv I_i^t(\lfloor p \rfloor + (1,1)), \end{aligned} \tag{9}$$

where $(u,v) = p - \lfloor p \rfloor$. It is easy to see that the screen pixels are linear combinations of the texture pixels under such settings.

Texture. When $\psi = \tilde{\phi} \circ \hat{\psi}$ is obtained, we need to compute the corresponding texture I_i^t, which is known as *baking*. However, directly solving Equation (4) even locally

$$I_i^{t*} = \arg\min_{I_i^t} \mathcal{E}(\tilde{\phi} \circ \hat{\psi}, I_i^t) \tag{10}$$

is usually very difficult because I_i^t is in high dimensions and $\mathcal{D}$ may have very complicated forms. Therefore, we take an approximation in our experiments:

$$I_i^t \approx \arg\min_{I_i^t} \|\hat{I}_i^s - I_i^s\|^2, \tag{11}$$

which turns out to be a simple least square problem when $\mathcal{H}$ is a linear mapping from texture pixels to screen pixels. Intuitively, this is the same as optimizing $\mathcal{E}(\tilde{\phi} \circ \hat{\psi}, I_i^t)$ with $\mathcal{D}(\hat{I}_i^s, I_i^s) = \text{MSE}(\hat{I}_i^s, I_i^s)$. In other words, we use MSE to approximate $\mathcal{D}$ when optimizing I_i^t for efficiency .

FR-IQA Having both $\hat{I}^s$ and I^s at hand, we use Gradient Magnitude Similarity Deviation (GMSD for short) [XZMB13] to compute the visual difference as the loss: $\mathcal{E}(\tilde{\phi} \circ \hat{\psi}) = \mathcal{D}(\hat{I}_i^s, I_i^s) = \text{GMSD}(\hat{I}_i^s, I_i^s)$. For an intuition of visual quality about different GMSD, please refer to Figure 6. Of course, other perceptual model can also be used here, even MSE is also feasible though it is inaccurate enough to measure visual difference.

4.1.2. Bayesian optimization

Because the visual difference $\mathcal{D}$ and rendering procedure $\mathcal{H}$ for $\hat{I}_i^s$ and I_i^s are complicated in general, it is difficult to solve such local optimization problem using common deviation dependent methods. Thus, we resort to Bayesian optimization strategy [CCAM18], which just needs to evaluate the value of the objective function at some samples in a domain. To apply [CCAM18], we first parameterize ϕ_i properly, then approximate Equation (7) into an optimization with box-constraints.

As ϕ is piecewise-linear, we have $\phi_i(x) = (\nabla\phi_i)x + t_i$ locally, where $\nabla\phi_i \in \mathbb{R}^{2\times 2}$ and $t_i \in \mathbb{R}^2$ is the offset. The offset part would have less impact on the visual quality and bring difficulties for the comforming requirement. Therefore, we choose to set $t_i = 0$ always and only need to take 4 DoFs of $\nabla\phi_i$.

To turn the anti-flip constraint $\det(\nabla\phi_i) > 0$ into a box constraint, we decompose $\nabla\phi_i = R_i S_i$ into rotational part R_i and stretching part S_i by polar decomposition. The symmetric part S_i can be further decomposed by eigen decomposition as $S_i = V_i \Sigma_i V_i^\top$. We restrict both R_i, V_i in $SO(2)$, i.e. 2D rotations in angles $\alpha, \beta \in (-\pi, \pi]$ respectively. It is easy to see that the anti-flip constraint can be easily replaced by $\lambda_1, \lambda_2 > 0$, i.e. the two eigenvalues in Σ_i are positive. The eigenvalues indicate the stretch of the triangle under deformation ϕ_i, so they should be less than 1 because the input texture is taken as ground truth and any bigger triangle would not provide extra benefits to visual quality.

We then approximate the bounded constraint about $\mathcal{E}$ as a penalty term $L_\varepsilon(\mathcal{E}(\tilde{\phi} \circ \hat{\psi}))$ using the following softer barrier function,

$$L_\varepsilon(x) = \begin{cases} \infty & x > \varepsilon, \\ -\log(\varepsilon - x) & \text{otherwise.} \end{cases} \tag{12}$$

By $\det(\nabla\phi_i) = \lambda_1\lambda_2$, we now have a box-constrained optimization with only four variables:

$$\begin{aligned} &\min_{\alpha,\lambda_1,\lambda_2,\beta} \lambda_1\lambda_2 |\hat{\triangle}_i^t| + L_\varepsilon(\mathcal{E}(\tilde{\phi} \circ \hat{\psi})), \\ &\text{s.t. } \lambda_1, \lambda_2 \in (0,1], \alpha, \beta \in (-\pi, \pi]. \end{aligned} \tag{13}$$

Symmetry of the optimization. Because the square-shaped pixels are packed in a square lattice structure in the texture image, it is easy to see that $\mathcal{H}\left(M, C_i, \tilde{\phi} \circ \hat{\psi}, I_i^t|_{\tilde{\phi}}\right) = \mathcal{H}\left(M, C_i, R(\pi/2) \circ \tilde{\phi} \circ \hat{\psi}, I_i^t|_{R(\pi/2)\circ\tilde{\phi}}\right)$, i.e. the 90-degree rotation of R_i will not change I_i^s. It means that the perceptual loss $\mathcal{D}(\hat{I}_i^s, I_i^s)$ is symmetric with respect to 90-degree rotation of R_i as shown in Figure 7. For β, we have $S(\lambda_1, \lambda_2, \beta + \frac{\pi}{2}) = S(\lambda_2, \lambda_1, \beta)$obviously,

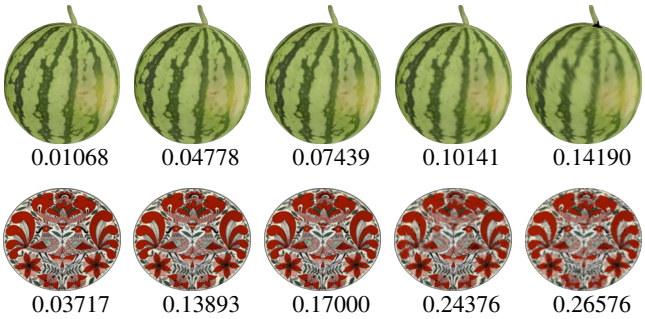

Figure 6: *Illustration of visual quality at different GMSD*

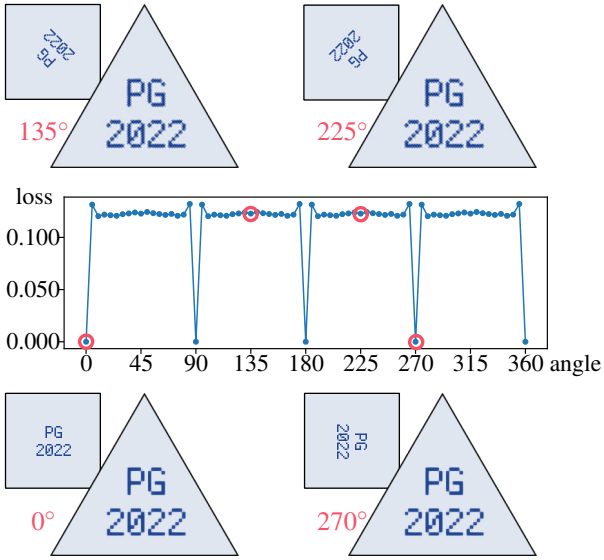

Figure 7: *The perceptual loss is related to the rotation α with a symmetry property. Triangle: rendering result; Square: texture.*

so there is also a symmetry of 90-degree rotation for V_i. Therefore, we restrict $\alpha,\beta \in [0,\pi/2)$. It shrinks the sampling domain of Bayesian optimization for efficiency, and more importantly, eliminates the problem of having multiple symmetric optimal solutions. Finally, the Bayesian optimization problem is:

$$\begin{aligned} &\min_{\alpha,\lambda_1,\lambda_2,\beta} \lambda_1\lambda_2|\hat{\triangle}_i^t| + L_\varepsilon(\mathcal{E}(\tilde{\phi}\circ\hat{\psi})), \\ &\text{s.t. } \lambda_1,\lambda_2 \in (0,1], \alpha,\beta \in [0,\frac{\pi}{2}]. \end{aligned} \tag{14}$$

4.2. Mesh-wise optimization of ϕ

For i-th triangle, now we get an optimal deformation Jacobian $\nabla\bar{\phi}_i = \bar{R}_i\bar{S}_i$. In consideration of the 90-degree rotational symmetry of R_i, the optimal Jacobian are indeed $\nabla\bar{\phi}_{i,k} = R(k\pi/2)\bar{R}_i\bar{S}_i, k = 0,1,2,3$. In the mesh-wise step, we are seeking for a global parameterization ϕ that best matches one of them on each triangle. Taking the common Frobenius norm, the deviation on i-th triangle can be written as

$$e(\phi_i) = \min_{k\in\{0,1,2,3\}} \|\nabla\phi_i - R(k\pi/2)\bar{R}_i\bar{S}_i\|_F^2. \tag{15}$$

However, as shown in our experiments (see Figure 17), such a measurement of deviation is not good enough since it totally ignores the perceptual loss $\mathcal{E}$.

Taking the polar decomposition $\nabla\phi_i = R_iS_i$ and using R_i, S_i as the variables, we approximate the triangle-wise perceptual loss $\mathcal{E}$ by a positive defined quadratic approximation on i-th triangle as:

$$e(R_i,S_i) = \min_{k_i\in\{0,1,2,3\}} w_i^R\|R_i - R(k_i\pi/2)\bar{R}_i\|_F^2 + w_i^S\|S_i - \bar{S}_i\|_F^2, \tag{16}$$

where w_i^R and w_i^S measure the sensitivity of $\mathcal{E}$ with respect to the rotation R_i and stretch S_i respectively. Specifically, we first rewrite the loss as $\mathcal{E}(\alpha,\lambda_1,\lambda_2,\beta)$ and keep other variables fixed. Then the weights are defined as variance of the loss:

$$\begin{aligned} w_i^R &= \mathrm{Var}[\mathcal{E}(\alpha,\bar{\lambda}_1,\bar{\lambda}_2,\bar{\beta})], \\ w_i^S &= \mathrm{Var}[\mathcal{E}(\bar{\alpha},\lambda_1,\lambda_2,\beta)]. \end{aligned} \tag{17}$$

With this perceptual loss related weighting scheme, we solve the following optimization in the mesh-wise step:

$$\begin{aligned} &\min_{\phi,\{k_i\}} \sum_{\triangle_i\in M} w_i^R\|R_i - \bar{R}_iR_{\pi/2}^{k_i}\|_F^2 + w_i^S\|S_i - \bar{S}_i\|_F^2, \\ &\text{s.t.} \quad \nabla\phi_i = R_iS_i, R_i \in SO(2), S_i^\top = S_i, \\ &\det(\nabla\phi_i) > 0, \quad k_i \in \{0,1,2,3\}. \end{aligned} \tag{18}$$

Different from the conventional Poisson-like problem, our formulation uses specialized weight scheme, and also needs to count the symmetry of rotation. To evaluate the variance in the weight scheme, we uniformly sample the angle α and the singular values λ_1, λ_2 in the domain. For the symmetry of rotation, we enumerate $k \in 1,2,3,4$ to find the optimal rotation $R_{\pi/2}^k$ that minimize $\|R_i - \bar{R}_iR_{\pi/2}^k\|_F^2$, after the polar decomposition of $\nabla\phi_i$ on each triangle.

After handling the difficulties coming from the weights and the symmetry of rotation, we introduce the slack variables $\{R_i\}$ and $\{S_i\}$ into above problem:

$$\begin{aligned} &\min_{\phi,\{k_i\},\{R_i\},\{S_i\}} \sum_{\triangle_i\in M} E_{\text{fit}}(\phi_i,k_i,R_i,S_i) + w^{rs}E_{\text{rs}}(\phi_i,R_i,S_i), \\ &\text{s.t.} \quad R_i \in SO(2), S_i^\top = S_i, \det(\nabla\phi_i) > 0, k_i \in \{0,1,2,3\}. \end{aligned} \tag{19}$$

Here E_{fit} fits the rotation and stretch: $E_{\text{fit}} = w_i^R\|R_i - \bar{R}_iR_{\pi/2}^{k_i}\|_F^2 + w_i^S\|S_i - \bar{S}_i\|_F^2$ and E_{rs} recovers the mapping ϕ_i from the rotation and stretch: $E_{\text{rs}} = \|\nabla\phi_i - R_iS_i\|_F^2$. We iteratively solve the problem via the standard local-global update strategy [SA07]. To be specific, we find the optimal k_i, R_i and S_i in the local step and solve ϕ via an unconstrained optimization in the global step. For the detailed algorithm, we provide the updating strategies in Appendix B.

Bijectivity The method described above is designed for local injectivity instead of bijectivity, and the resulting parameterization may have self-intersections. To handle this problem, one can use [JSP17] after triangle-wise optimization as an optional step to guarantee the bijectivity with the following steps: 1. Construst scaffolding triangles by [JSP17]; 2. Perform mesh-wise parameterization in Section 4.2 with anti-flip constraint of the scaffolding triangles; 3. Remove the scaffolding triangles. The results are shown in Figure 19.

5. Experiments

Our method can generate efficient parameterization on models with various signals. Figure 8 shows a gallery of our results, and the performance statistics about the results are listed in Table 1. All the results are generated on a desktop with AMD Ryzen™ 9 5900X and NVIDIA® GeForce® GTX 1650. As formulated in Equation (7),

Figure 8: *The gallery of the rendered models, the reparametrized textures with our method. $c_r \approx 0.3$. The input parameterizations are shown as the untextured planar meshes.*

the perceptual loss is controlled via the parameter ε, and we experimentally take $\varepsilon = 0.01$ by default (see Section 5.3.5). To demonstrate the efficacy of our method, we first introduce the assessment of texture parameterization in our experiment, then make comparisons with state-of-the-arts, and finally present the algorithm behaviors under different experiment settings.

Model	#V / #F	r	c_{input}	t_{triangle} / t_{mesh}
Bunny	15258 / 30338		678742	841.94s / 68.50s
Feline	5131 / 10266		624538	277.56s / 3.17s
Plate	7482 / 14956		500777	402.70s / 65.09s
Statuette	1443 / 2882		556417	76.33s / 1.23s
Tiger	5396 / 10788	1024	283775	253.43s / 8.05s
Vase	6366 / 12723		598658	321.88s / 9.43s
Vase-2	6366 / 12723		598658	314.66s / 10.60s
Watemelon	4870 / 9737		575943	193.71s / 8.90s
Zebra	5040 / 10076		570767	264.24s / 8.57s

Table 1: *Statistics of the results. All the models are shown in Figure 8. c_{input} is roughly the number of pixels used for texture mapping in the input texture image, as defined in Equation (20). $t_{triangle}$ and t_{mesh} list the time taken for the triangle-wise and mesh-wise optimization respectively.*

5.1. Assessment of texture parameterization

As mentioned in Section 3.1, perceptual-loss-on-screen can be defined for a single triangle or for the whole mesh. For typical applications, the surface is viewed as a whole instead of many triangles. Therefore, we would use perceptual loss defined for the whole mesh as the "visual quality" metric in our evaluation.

Generation of texture images. To compare the efficiency of parameterizations, we need to first generate corresponding textures with the same number of pixels used for texture mapping. Similar to what we do for single triangles, we would take cost

$$c(\psi, r) = r^2 \sum |\triangle_i^t| \tag{20}$$

as an approximation. Given a desired $\bar{c}$ where the comprasion is made, we can then find proper r to satisfy

$$r^2 \sum |\triangle_i^t| \approx \bar{c}.$$

The symbol "$\approx$" here is due to the fact that resolution $r \in \mathbb{Z}$.

For convinience, we use relative cost defined as following:

$$c_r(\psi, r) = c(\psi, r)/c(\hat{\psi}, \hat{r}) \tag{21}$$

And we can still easily solve r such that $c_r(\psi, r) \approx \bar{c}_r$. Once we get proper r for both parameterizations, we generate the texture images I^t with Blender [Ble22].

Perceptual loss for the whole mesh. Computing the perceptual-loss-on-screen for the whole mesh is generally the same as the one

for a single triangle except for the camera. To get a comprehensive metric, we use 20 cameras located at the vertices of a regular dodecahedron and pointing to the center of the mesh to calcuate $\mathcal{E}$, and take the average as the visual quality at texture resolution r:

$$q(\psi, r) = \frac{1}{20} \sum_{i=1}^{20} \mathcal{E}_{r,\mathcal{C}_i,\mathcal{H}}(\psi), \tag{22}$$

or at the relative cost $\bar{c}_r$:

$$q_r(\psi, \bar{c}_r) = q(\psi, r), \text{s.t. } c_r(\psi, r) \approx \bar{c}_r. \tag{23}$$

Texture packing. We do not change the cuts of the input mesh. If the mesh is cut into several disconnected patches, we pack the corresponding patches in the parameterization domain by [LPRM02] to reduce the whitespaces between the patches. It should be noticed that to keep the optimized orientation, we only allow limited operations of translation and multiply-of-90-degree rotation during packing to avoid affecting the efficiency of the parameterization. Other methods that only take these limited operations can also be used here, and the cost and loss described above will not be affected.

5.2. Comparisons

In real world applications, a parameterization would usually not be limited to a single resolution. Moreover, it is not fair enough to compare parameterizations on a single resolution. We follow [SGSH02] to compare parameterizations in a series of $\bar{c}_r$. The result will be presented as a $q_r - c_r$ curve.

5.2.1. Comparison with input parameterizations

With the guidance of the perceptual loss, our method can generate efficient parameterization for texture mapping, while the input parameterizations (e.g., designed manually or generated automatically) are usually not optimized for this. Thus, it is not surprising that our method can supply parameterizations with lower perceptual loss in the same relative cost as showed in Figure 9 and the results on "Vase" model are shown in Figure 10. Please notice that the term "Input" in Figures 9 and 10 only refers to "input parameterization" instead of the whole input which includes the input texture image. In this experiment, the texture image associated with the input parameterization are re-generated by the procedure described in Section 5.1.

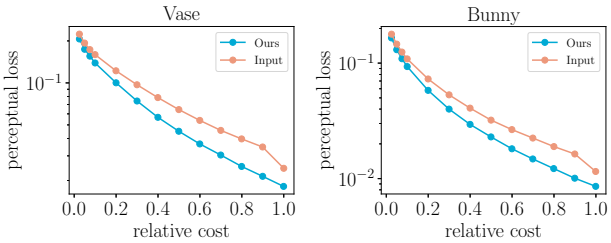

Figure 9: *Comparison with input parameterizations. Y-axis is the logarithm of the perceptual loss, i.e.* $\log(q_r)$.

5.2.2. Comparison with SSP [SGSH02]

SSP [SGSH02] attempts to find the best triangle-wise Jacobian minimizing the interpolation error of the discretized signals. However, their expected stretch $\lambda_1, \lambda_2, \theta$ are not always be able to make improvement on the result because they ignore the rendering procedure. In our experiments, we compare our methods with theirs (SSP). Further, since their global parameterization technique is relatively old, to compare their triangle-wise parameterization deformation with ours fairly, we use their expected stretch in our global parameterization with rotation weight $w^R = 0$ (SSPOP). As shown in Figure 11, our method can generate results with lower perceptual loss (q_r) than both SSP and SSPOP under different relative cost (c_r). We also show the rendered texture models in Figure 10 and Figure 12.

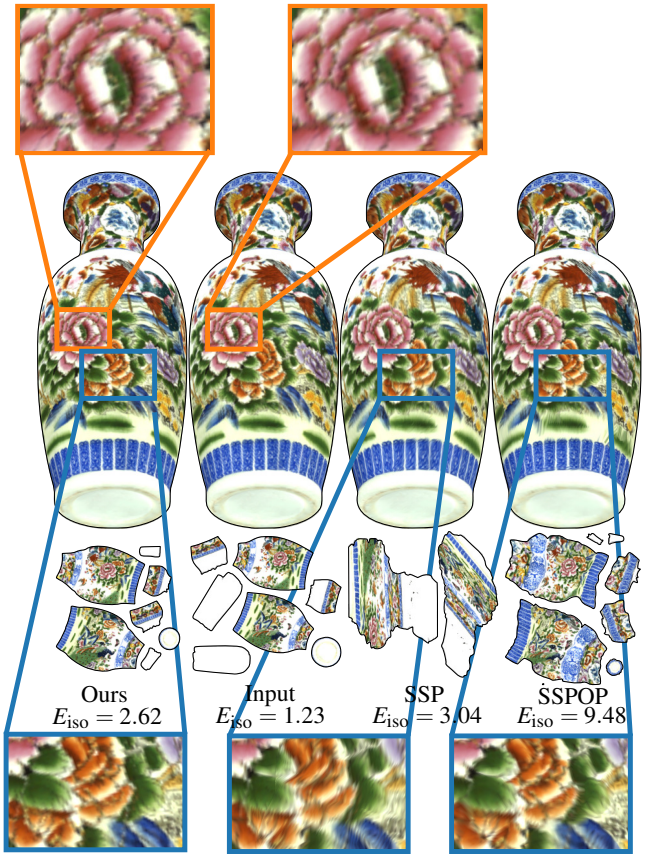

Figure 10: *Detailed comparison of the Vase model with input parameterization, SSP [SGSH02] and SSPOP. The relative cost* $c_r \approx 0.3$. *Similar to OptCuts in Figure 1, the rendering result of input parameterization also suffers from the blur problem, especially at the center region.*

5.2.3. Comparison with OptCuts [LKK*18]

One may expect that a parameterization can usually lead to high quality texture mapping when each triangle reserves isometry to its original shape. However, such isometric parameterization only focuses on the geometry but ignores the texture signals, so it usually

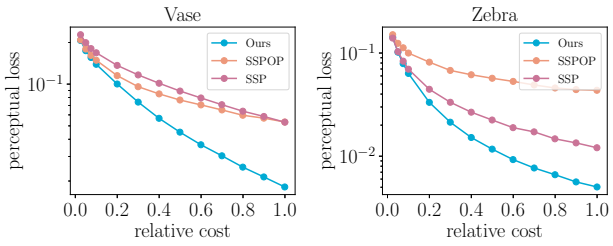

Figure 11: *Comparison with SSP [SGSH02]. Y-axis is the logarithm of the perceptual loss, i.e.* $\log(q_r)$.

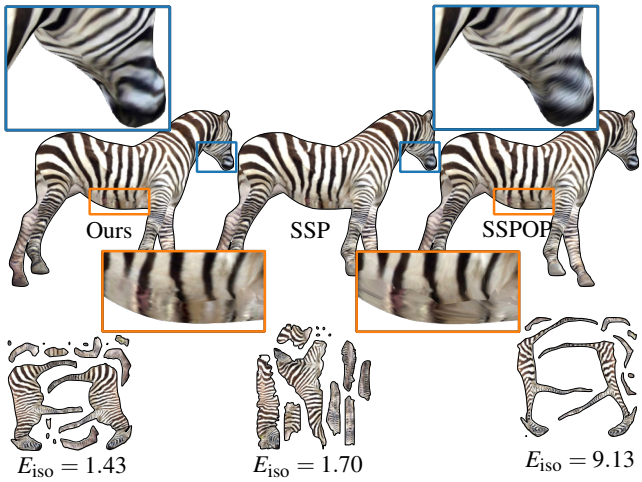

Figure 12: *Detailed comparison of the Zebra model with SSP [SGSH02] and SSPOP. The relative cost* $c_r \approx 0.3$.

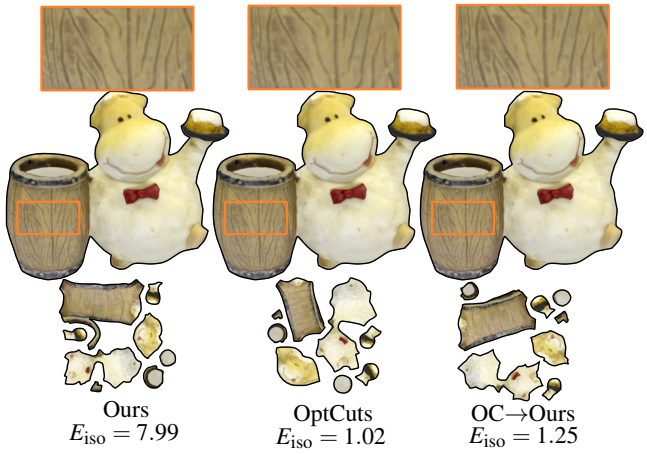

Figure 13: *Detailed comparison of the Statuette model with OptCuts [LKK*18]. The relative cost* $c_r \approx 0.3$.

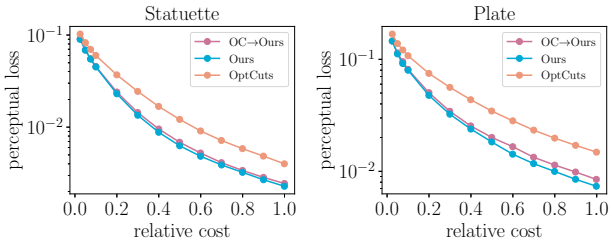

Figure 14: *Comparison with OptCuts [LKK*18]. Y-axis is the logarithm of the perceptual loss, i.e.* $\log(q_r)$.

leads to inefficient results. Here, we compare with one of the state-of-the-art isometric parameterization methods [LKK*18].

5.2.4. Comparisons about distortion

One may also be interested in the comparisons about common parameterization distortion. Here, we use the symmetric Dirichlet energy defined in [SS15, LYNF18] to measure the isometric distortion:

$$E_{\text{iso}} = \frac{1}{4}\sum a_i(\|J_i\|_F^2 + \|J_i^{-1}\|_F^2), \tag{24}$$

where a_i is the area of i-th triangle in the input mesh and J_i is its Jacobian of parameterization with respect to the input mesh. Under this measurement, our method can generate parameterizations with lower distortion than SSP and SSOP (see Figures 10 and 12). Though our results have higher distortion than the input parameterization, they are more efficient because the distortion and efficiency are not strongly related. Figures 13 and 14 show the same behavior that our method generates parameterizations with higher E_{iso} compared to OptCuts [LKK*18], but we can achieve more efficient results for texture mapping.

5.3. Algorithm behaviors

We show the behaviors of our method in the following.

5.3.1. Sensitivity to the input parameterization

Our method can be viewed as a reparametrization, and the input parameterization has some influences on the results. To show the sensitivity to the input parameterizations of our method, we first apply OptCuts to the input, then use its resulting parameterization $\hat{\psi}_{\text{OC}}$ to generate a new texture image $\hat{I}^t_{\text{OC}}$. Taking them as the input of our method, the efficiency (see "OC→Ours" in Figures 13 and 14) is slightly worse than using $\hat{\psi}, \hat{I}^t$. Considering the quality loss when generating $\hat{I}^t_{\text{OC}}$, our method shows good stability in this experiment. However, the distortion is much lower than directly using the input parameterization. So first applying OptCuts or other isometric parameterization methods could be a practical way to reduce the distortion if distortion is important, or generate the input for our method if the quality of original parameterization is bad.

5.3.2. Convergency of the triangle-wise stage

Though lacking assistant of derivative information, the Bayesian optimization used in our triangle-wise stage would converge in fast pace. As shown in Figure 15, taking 100 iterations is usually enough.

5.3.3. Perceptual loss after mesh-wise parameterization

Although we can solve the optimal rotation and stretch in triangle-wise optimization with a bounded perceptual loss, it is usually im-

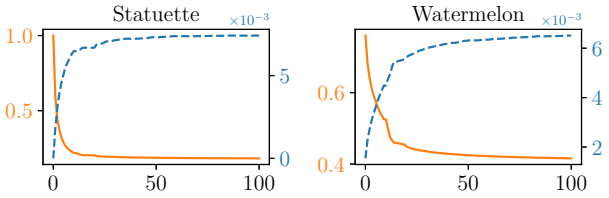

Figure 15: *Convergence of the triangle-wise optimization. The X-axis is the iteration counts. For the Y-axises in each sub-figure, the left one (marked in orange) is the average* $\det(\phi_i)$*, and the right one (marked in blue) is the average perceptual loss* $\mathcal{E}$*.*

possible to find a parameterization on the whole mesh that matches all the optimal solutions on each triangle due to the conforming requirement. Experimentally, our method can help to find a mesh-wise parameterization that most of the perceptual loss are under the given bar and Figure 16 shows some distribution. For the "Statuette" and "Watermelon" model, there are 93% and 66% triangles under the perceptual bound ($\mathcal{E} \leq 0.01$) after mesh-wise parameterization respectively.

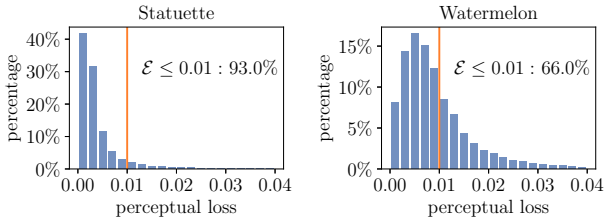

Figure 16: *The distribution of triangle-wise perceptual loss after mesh-wise parameterization.*

5.3.4. Benefit of the weight scheme

In the stage of mesh-wise parameterization, our method fully takes the perceptual loss $\mathcal{E}$ into consideration, and seperates the rotation and stretch into two items. Compared to the common parameterization strategy which mixes rotation and stretch together under the Frobenius-norm as formulated in Equation (15), our method can generate more efficient parameterization results. Moreover, our carefully designed weighting scheme is well aware of the perceptual loss, so it can provide parameterizations with lower loss under the same relative cost. The comparisons under different weighting settings are shown in Figure 17.

5.3.5. Customized-control of perceptual loss

Our method can support the control of preceptual loss with a customized parameter ε in Equation (7). Generally, a larger ε usually lead to higher perceptual loss as shown in Figure 18, and we pratically find $\varepsilon = 0.01$ works well in all experiments.

6. Conclusions

This paper has proposed a novel two-stage method for parametrization driven by the perceptual loss. The parametrization results are efficient for texture mapping because the triangle-wise optimal rotations and stretches fully take the *perceptual-loss-on-screen* into

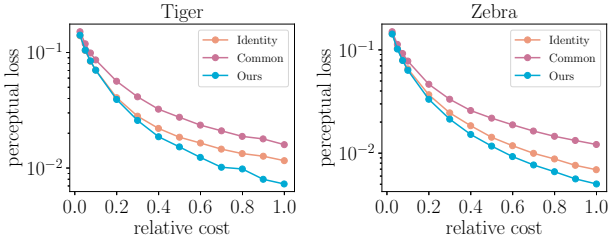

Figure 17: *Comparisons under different weight settings. "Identity" is our global parameterization with* $w_i^R = w_i^S = 1, \forall \triangle_i$*. "Common" is directly solving Equation* (15)*.*

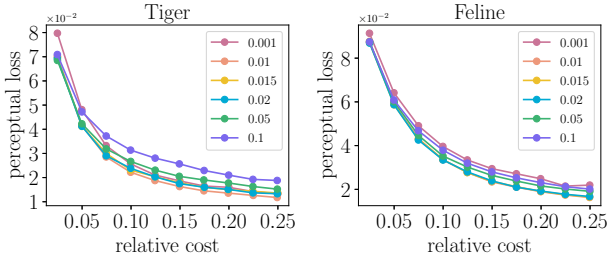

Figure 18: *The balance between the relative cost and perceptual loss controlled by* ε*.*

consideration via a Bayesian optimization. To alleviate the numerical difficuties in the mesh-wise optimization, we first count the symmetry of the rotation and then use a percepture loss awared weight scheme to solve the problem. Experimental results show that the proposed method outperforms many previous methods on a lower relative cost under a comparible perceptual loss. Even the input parametrization is fairly efficient, our method can also help to further improve its efficiency as our experiments indicate.

Limitations and future work One limitation is that there is no guarantee of such bounded-loss in the mesh-wise stage, though the Bayesian optimization can achieve the optimal rotations and stretches on each triangle under the perceptual loss bound. Besides, our method only focuses on the effect of rotation and stretch on the perceptual loss but ignores the visual artifacts on seams (see Figure 20). One can apply [LFJG17] as a post-processing to alleviate the artifacts around the seams. Although the current mesh-

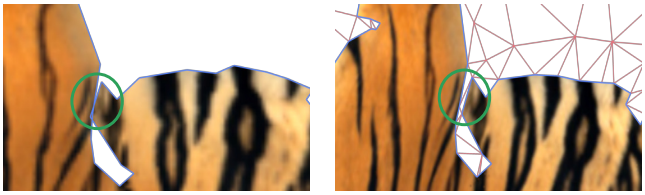

Figure 19: *Use [JSP17] to guarantee the bijectivity of the parameterization. Left: the result with self-intersections. Right: the result without self-intersection by applying [JSP17].*

wise optimization method works well on many models, the results may be further improved by applying better initialization strategies (see Appendix B) and solvers. Besides, Bayesian optimization in triangle-wise step leads to good results but is slow. Replacing it by recently developed differentiable rendering technique may greatly improve the performances. Finally, the current perceptual loss is only calculated from the naive rendering and baking procedure, thus it is a valuable future work to integrated our framework with more complicated and advanced rendering and baking techniques.

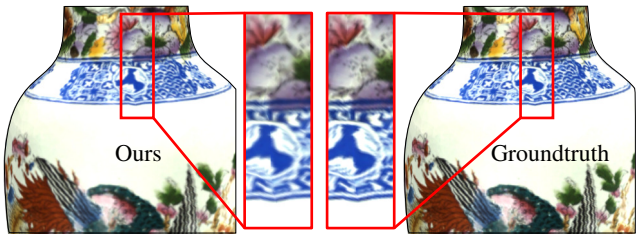

Figure 20: *Visual artifacts on the seam. The relative cost $c_r \approx 1.0$. If zoomed in, the seam on ours is indeed more visible, as we do not take any special attension to the seams. .*

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments and suggestions. Authors in Zhejiang University were partially supported by National Key RD Program of China (No. 2020AAA0108901) and Zhejiang Provincial Science and Technology Program in China (No. 2021C01108). Authors in Zhejiang Sci-Tech University were partially supported by National Natural Science Foundation of China (No. 61702458, 61602416).

References

[BBT03] BALMELLI L., BERNARDINI F., TAUBIN G.: Space-optimized texture maps. *Computer Graphics Forum 21*, 3 (2003), 411–420.

[BDS*12] BOUAZIZ S., DEUSS M., SCHWARTZBURG Y., WEISE T., PAULY M.: Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum 31*, 5 (2012), 1657–1667.

[BLD20] BANGARU S. P., LI T., DURAND F.: Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph. 39*, 6 (Nov 2020), 245:1–245:18.

[Ble22] BLENDER ONLINE COMMUNITY: *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2022. URL: http://www.blender.org.

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

[CCAM18] CULLY A., CHATZILYGEROUDIS K., ALLOCATI F., MOURET J.-B.: Limbo: A Flexible High-performance Library for Gaussian Processes modeling and Data-Efficient Optimization. *The Journal of Open Source Software 3*, 26 (2018), 545.

[CW15] CHIU Y., WANG Y.: Content aware texture compression. *J. Inf. Sci. Eng. 31*, 6 (2015), 2075–2088.

[GLD*19] GAO D., LI X., DONG Y., PEERS P., XU K., TONG X.: Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. *ACM Trans. Graph. 38*, 4 (Jul 2019), 134:1–134:15.

[JSP17] JIANG Z., SCHAEFER S., PANOZZO D.: Simplicial complex augmentation framework for bijective maps. *ACM Trans. Graph. 36*, 6 (Nov 2017), 186:1–186:9.

[LFJG17] LIU S., FERGUSON Z., JACOBSON A., GINGOLD Y.: Seamless: Seam erasure and seam-aware decoupling of shape from mesh resolution. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 216:1–216:15.

[LFY*19] LIU H.-Y., FU X.-M., YE C., CHAI S., LIU L.: Atlas refinement with bounded packing efficiency. *ACM Trans. Graph. 38*, 4 (July 2019), 33:1–33:13.

[LH06] LEFEBVRE S., HOPPE H.: Appearance-space texture synthesis. *ACM Trans. Graph. 25*, 3 (Jul 2006), 541–548.

[LKK*18] LI M., KAUFMAN D. M., KIM V. G., SOLOMON J., SHEFFER A.: Optcuts: Joint optimization of surface cuts and parameterization. *ACM Trans. Graph. 37*, 6 (Dec 2018), 247:1–247:13.

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graphics 21*, 3 (Jul 2002), 362–371.

[LVS18] LIMPER M., VINING N., SHEFFER A.: Box cutter: Atlas refinement for efficient packing via void elimination. *ACM Trans. Graph. 37*, 4 (Jul 2018), 153:1–153:13.

[LYNF18] LIU L., YE C., NI R., FU X.-M.: Progressive parameterizations. *ACM Trans. Graph. 37*, 4 (July 2018), 41:1–41:12.

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. *Computer Graphics Forum 27*, 5 (2008), 1495–1504.

[MA10] MARTINEZ J., ANDUJAR C.: Space-optimized texture atlases for 3d scenes with per-polygon textures. In *2010 18th Pacific Conference on Computer Graphics and Applications* (2010), IEEE, pp. 14–23.

[NS11] NÖLL T., STRIEKER D.: Efficient packing of arbitrary shaped charts for automatic texture atlas generation. *Computer Graphics Forum 30*, 4 (2011), 1309–1317.

[RLL*06] RAY N., LI W. C., LÉVY B., SHEFFER A., ALLIEZ P.: Periodic global parameterization. *ACM Trans. Graph. 25*, 4 (Oct 2006), 1460–1485.

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Trans. Graph. 36*, 2 (Apr 2017), 16:1–16:16.

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Symposium on Geometry processing* (Goslar, DEU, 2007), vol. 4 of *SGP '07*, Eurographics Association, p. 109–116.

[SGSH02] SANDER P. V., GORTLER S. J., SNYDER J., HOPPE H.: Signal-specialized parametrization. In *Proceedings of the 13th Eurographics Workshop on Rendering* (Goslar, DEU, 2002), EGRW '02, Eurograph. Association, p. 87–98.

[SKPS13] SCHÜLLER C., KAVAN L., PANOZZO D., SORKINE-HORNUNG O.: Locally injective mappings. *Computer Graphics Forum 32*, 5 (2013), 125–135.

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

[SS15] SMITH J., SCHAEFER S.: Bijective parameterization with free boundaries. *ACM Trans. Graph. 34*, 4 (Jul 2015), 70:1–70:9.

[SWB98] SLOAN P.-P. J., WEINSTEIN D. M., BREDERSON J.: Importance driven texture coordinate optimization. *Computer Graphics Forum 17*, 3 (1998), 97–104.

[TSS*04] TEWARI G., SNYDER J., SANDER P. V., GORTLER S. J., HOPPE H.: Signal-specialized parameterization for piecewise linear reconstruction. In *Eurographics Symposium on Geometry Processing (2004)* (July 2004), ACM, pp. 55–64.

[XZMB13] XUE W., ZHANG L., MOU X., BOVIK A. C.: Gradient Magnitude Similarity Deviation: A Highly Efficient Perceptual Image Quality Index. *IEEE Trans. Image Process. 23*, 2 (Dec 2013), 684–695.

[YHBZ01] YING L., HERTZMANN A., BIERMANN H., ZORIN D.: Texture and shape synthesis on surfaces. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques, London, UK, June 25-27, 2001* (2001), Eurographics, Springer, pp. 301–312.

Appendix A: Camera $\mathcal{C}_i$ in our experiment

The camera $\mathcal{C}_i$ used in our experiment is an orthographic camera, which would view on a square region centered at the barycenter of $\triangle_i$. For $\triangle_i$ with vertices a_i, b_i, c_i, it can be calculated from the following pseudocode Algorithm 1.

Algorithm 1 Compute $\mathcal{C}_i$ for $\triangle_i$

procedure CAMERA(a,b,c)
  $C = (a+b+c)/3$
  $N = ((b-a)\times(c-a))/\|(b-a)\times(c-a)\|$
  $R = (a-C)/\|a-C\|$
  $U = (N\times R)/\|N\times R\|$
  $(P_l, P_r, P_b, P_t) = \text{BBOX}(a,b,c,C,U,R)$
  View $= \text{LOOKAT}(\text{eye} = C+500N, \text{front} = C, \text{up} = U)$
  Project $= \text{ORTHO}(P_l, P_r, P_b, P_t, \text{near} = 0.1, \text{far} = 1000.0)$
  Viewport $= \text{VIEWPORT}(\text{width} = 64\text{px}, \text{height} = 64\text{px})$
  return Viewport $\circ$ Project $\circ$ View
end procedure
procedure BBOX(a,b,c,C,U,R)
  $a_p = \text{PROJECT}(C,a,R,U)$
  $b_p = \text{PROJECT}(C,b,R,U)$
  $c_p = \text{PROJECT}(C,c,R,U)$
  $P_l = \min\{a_p.x, b_p.x, c_p.x\}$
  $P_r = \max\{a_p.x, b_p.x, c_p.x\}$
  $P_b = \min\{a_p.y, b_p.y, c_p.y\}$
  $P_t = \max\{a_p.y, b_p.y, c_p.y\}$
  $w = P_r - P_l$, $h = P_t - P_b$
  if $w > h$ **then**
    $P_b = P_b - (w-h)/2$
    $P_t = P_t + (w-h)/2$
  else
    $P_l = P_l - (h-w)/2$
    $P_r = P_r + (h-w)/2$
  end if
  return (P_l, P_r, P_b, P_t)
end procedure
procedure PROJECT(C,x,R,U)
  $d = x - C$
  return $(\langle d,R\rangle, \langle d,U\rangle)$
end procedure

Appendix B: Local-global strategy

The problem in Equation (19) is highly nonlinear. We thus apply a local-global strategy with an alternating iterative scheme to decouple the whole optimization problem into several relatively simple ones. In this section, we will describe the numerical strategy to solve Equation (19). For the local step, we update k_i, R_i and S_i with the fixed ϕ_i on each triangle, and in the global step, we optimize ϕ under the given $\{k_i\}$, $\{R_i\}$ and $\{S_i\}$. In the following, we will introduce the two steps individually.

Locally updating k_i, R_i and S_i. With a given ϕ_i, we use its Jacobian $\nabla\phi_i$ to first update the rotation part (k_i and R_i), and then optimize the stretch part S_i. For the rotation part, we parametrize the rotation matrix R_i via a rotational angle θ_i and the optimization of rotation can be approximated as:

$$\min_{\theta_i, k_i} w_i^R \|R(\theta_i) - \bar{R}_i R_{\pi/2}^{k_i}\|_F^2 + w^{rs}\|R(\theta_i) - \nabla\phi_i S_i^{-1}\|_F^2. \tag{25}$$

For a better recovery of $\nabla\phi_i$, we first search $k_i \in \{0,1,2,3\}$ via: $\min_{k_i}\|\nabla\phi_i - \bar{R}_i R_{\pi/2}^{k_i}\|_F^2$, and θ_i can be solved analytically under a given k_i.

Similarly, we can also solve the stretch S_i with fixed k_i and R_i on each triangle as the following approximation:

$$\min_{S_i} w_i^S \|S_i - \bar{S}_i\|_F^2 + w^{rs}\|S_i - R_i^\top \nabla\phi_i\|_F^2, \tag{26}$$

which can be solved via a simple linear system. Notice that, both $\bar{S}_i$ and $R_i^\top\nabla\phi_i$ are symmetric matrices, so the optimal solution of the above equation satisfies the constraint $S_i^\top = S_i$. In our implementation, w_i^R and w_i^S are carefully designed as described in Equation (17), and w^{rs} is a factor that balances the first fitting and the second recovery terms. We empirically set w^{rs} to 10^{-5} in our experiments.

Globally updating ϕ. In the global step, we solve ϕ in Equation (19) with the fixed $\{k_i\}, \{R_i\}, \{S_i\}$ and the only constraint comes from the flip-free requirement $\det(\nabla\phi_i) > 0$, and we also turn it into penalty as:

$$\min_{\phi} \sum_{\triangle_i \in M} w^{rs}\|\nabla\phi_i - R_i S_i\|_F^2 + w^b E_{\text{barrier}}(\nabla\phi_i). \tag{27}$$

For the barrier term, we borrow the similar spline-like function with nice properties in [SKPS13], but other approaches, i.e. in logarithmic term, can also be used.

The minimization of Equation (27) can be solved through the similar damped Newton solver [SKPS13]. In practice, an amount of triangles of the optimized parameterization approach degeneration, which may squeeze the corresponding triangles dramatically during the iteration. In such a moment, E_{barrier} may dominate the whole energy which results in poor fitting effect. Therefore, we use an adaptive weighting scheme to decrease w^b, where the denominator 20 is an empirical value:

$$w_{j+1}^b = \begin{cases} w_j^b & E_{\text{rs}} \geq E_{\text{barrier}}, \\ w_j^b/20.0 & E_{\text{rs}} < E_{\text{barrier}}. \end{cases} \tag{28}$$

The whole optimization of Equation (19) is initialized by the input parameterization. Specifically, the local-global strategy begins from the local step, and the non-linear optimization is initialized with given $\nabla\phi_i = Id$. Equation (26) is a simple quadratic problem, and does not require initialization. For the first global step in Equation (27), ϕ is initialized by the input parameterization coordinates.