A fast and simple stretch-minimizing mesh parameterization

Shin Yoshizawa   Alexander Belyaev   Hans-Peter Seidel

Computer Graphics Group, MPI Informatik, Saarbrücken, Germany
Phone: [+49](681)9325-414   Fax: [+49](681)9325-499
E-mails: {shin.yoshizawa|belyaev|hpseidel}@mpi-sb.mpg.de

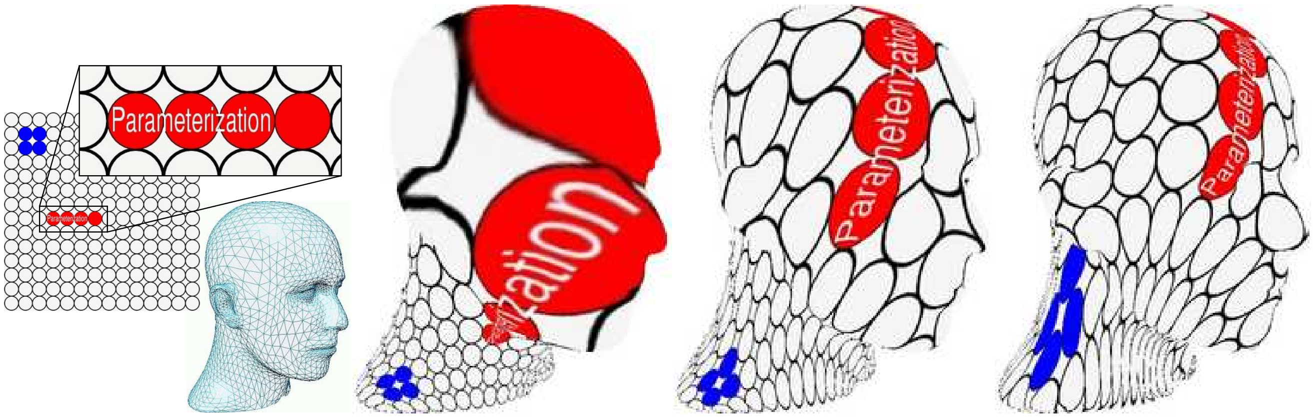


Figure 1: Texture mapping of the Mannequin Head model with three mesh parameterizations used in our method. (a) Texture and model. (b) Floater's shape preserving parameterization [6] is used as an initial mesh parameterization. (c) After a single optimization pass. (d) Our optimal low-stretch parameterization.

Abstract

We propose a fast and simple method for generating a low-stretch mesh parameterization. Given a triangle mesh, we start from the Floater shape preserving parameterization and then improve the parameterization gradually. At each improvement step, we optimize the parameterization generated at the previous step by minimizing a weighted quadratic energy where the weights are chosen in order to minimize the parameterization stretch. This optimization procedure does not generate triangle flips if the boundary of the parameter domain is a convex polygon. Moreover already the first optimization step produces a high-quality mesh parameterization. We compare our parameterization procedure with several state-of-art mesh parameterization methods and demonstrate its speed and high efficiency in parameterizing large and geometrically complex models.

Keywords: mesh parameterization, stretch minimization, remeshing

1 Introduction

Surface parameterization consists of a surface decomposition into a set of patches, also referred to as an atlas of charts, and establishing one-to-one mappings between the patches and reference domains. Numerous applications of surface parameterization in computer graphics and geometric modeling include texture mapping, shape morphing, surface reconstruction and repairing, and grid generation.

In this paper, we deal with a planar parameterization for a triangle mesh approximating a smooth surface, a bijective mapping between the mesh and a triangulation of a planar polygon. An excellent survey of recent advances in mesh parameterization is given in [7], see also references therein. While various algorithms are developed for mesh parameterization approaches based on solid mathematical theories (e.g., conformal mappings), effective computational schemes for generating practically important low-stretch mesh parameterizations [15] have not yet been proposed.

Consider a surface $S \in \mathbb{R}^3$ topologically equivalent to a disk and given parametrically by $\mathbf{r}(s,t) = [x(s,t), y(s,t), z(s,t)]$. The Jacobian matrix corresponding to the mapping $\mathbf{r}$ is given by $J = [\partial \mathbf{r}/\partial s, \partial \mathbf{r}/\partial t,]$. The Jacobian J determines all the first-order geometric properties of the parameterization $\mathbf{r}(s,t)$, including the area, angle, and length distortions caused by the mapping $\mathbf{r}$.

Denote by $\Gamma(s,t)$ and $\gamma(s,t)$ the maximal and minimal singular values of J. Consider the first fundamental form of S:

$$dl^2 = E(s,t)ds^2 + 2F(s,t)dsdt + G(s,t)dt^2,$$

where $E = \mathbf{r}_s^2$, $F = \mathbf{r}_s \cdot \mathbf{r}_t$, and $G = \mathbf{r}_t^2$. Then Γ^2 and γ^2 are the eigenvalues of the metric tensor

$$J^T J = \begin{bmatrix} E & F \\ F & G \end{bmatrix}.$$

It is convenient to use Γ and γ for measuring various properties of $\mathbf{r}$. For example, if $\Gamma(s,t) = \gamma(s,t)$, the parameterization is conformal and mapping $\mathbf{r} = \mathbf{r}(s,t)$ preserves angles.

Since the conformal mappings are well understood mathematically, discrete approximations of conformal mappings are widely used for mesh parameterization purposes [9, 10, 4, 8]. However conformal mappings often produce high stretch regions where texture mappings have severe undersampling artifacts.

It is natural to measure the local stretch of mapping $\mathbf{r} = \mathbf{r}(s,t)$ by $\sqrt{(\Gamma^2 + \gamma^2)/2}$ [15]. Stretch minimizing mesh parameterizations were considered in [15, 14, 12]. See also [16] where a similar stretch measure is proposed and [11, 19] where the Green-Lagrange tensor is used to measure the stretch.

While the stretch minimization approach proposed in [15] and further developed in [14] and [19] leads to generating high-quality mesh parameterizations, the computational procedure used in [15, 14, 19] for stretch minimization is time consuming. Besides the mesh parameterization procedure of [15, 14] often generates regions of high anisotropic stretch, consisting of slim triangles. Such the regions on a parameterized and textured mesh look like cracks and we call them *parameter cracks*. Fig. 2 demonstrates an appearance of such parameter cracks on the textured Mannequin Head model parametrized by the stretch minimization method from [15].

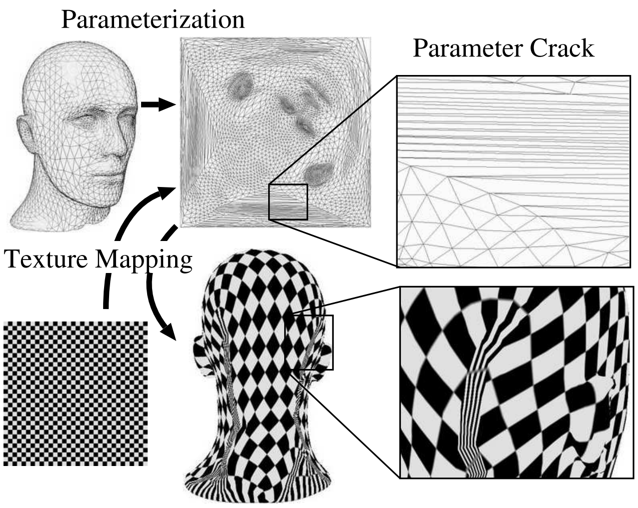


Figure 2: Parameter cracks on textured Mannequin Head model parametrized by the stretch minimization method of Sander et al. [15].

In [12] the authors propose to add a regularization term to the stretch energy in order to avoid parameter cracks. The term depends on two parameters. Besides minimizing the resulting energy does not produce a minimal stretch parameterization.

In this paper, we develop a simple and fast method for generating low-stretch mesh parameterizations. Given a triangle mesh, we first construct the Floater shape preserving mesh parameterization [6]. We improve the parameterization gradually: at each improvement step we optimize the parameterization generated at the previous step. The optimization is achieved by minimizing a weighted quadratic energy with positive weights chosen to minimize the parameterization stretch. Thus the single optimization step is fast since it is based on solving a sparse system of linear equations. Besides if the boundary of the parameterization domain forms a convex polygon, triangle flips never happen [6].

Roughly speaking, our method achieves a low-stretch mesh parameterization via a redistribution (diffusion) of local stretches. It also resembles quasi-Newton type optimization procedures.

We compare our low-stretch mesh parameterization procedure with several state-of-art mesh parameterization methods and demonstrate its speed and high efficiency in parameterizing large and geometrically complex models. Besides we show how our mesh parameterization approach can be combined with the interactive geometry remeshing scheme of Alliez et al. [1] in order to achieve fast and high quality remeshing.

Fig. 1 shows the three stages of our mesh parameterization method: generating an initial parameterization, our single-pass low-stretch parameterization, and the optimal low-stretch parameterization.

The rest or the paper is organized as follows. In Section 2 we explain our low-stretch mesh parameterization procedure and give a motivation behind it. We evaluate our method and compare it with state-of-art mesh parameterization techniques in Section 3. We conclude in Section 4.

2 Low stretch mesh parameterization

Given a parametrized triangle mesh $\mathcal{M} \in \mathbb{R}^3$, consider a mesh triangle $T = \langle \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \rangle \in \mathcal{M}$ and its corresponding triangle $U = \langle \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \rangle$ in the parametric plane $\mathbb{R}^2_{s,t}$. Triangles $\{U\}$ define a planar mesh $\mathcal{U} \in \mathbb{R}^2_{s,t}$ and the parameterization of $\mathcal{M}$ is given by one-to-one mapping between meshes $\mathcal{U}$ and $\mathcal{M}$. The correspondence between the vertices of T and U uniquely defines an affine mapping $P : U \to T$. Let us denote by $\Gamma(T)$ and $\gamma(T)$ the maximal and minimal eigenvalues of the metric tensor induced by the mapping [15, 19]. As we mentioned above, quantity

$$\sigma(U) = \sqrt{(\Gamma^2 + \gamma^2)/2}$$

characterizes the stretch of mapping P.

For each vertex $\mathbf{u}_i$ in the parameter domain let us define its stretch $\sigma_i = \sigma(\mathbf{u}_i)$ by

$$\sigma_i = \sqrt{\sum A(T_j)\sigma(U_j)^2 \Big/ \sum A(T_j)} \tag{1}$$

where $A(T)$ denotes the area of triangle T and the sums are taken over all triangles T_j surrounding mesh vertex $\mathbf{p}_i$ corresponding to $\mathbf{u}_i$.

Our method to build a low stretch mesh parameterization consists of several steps. First we construct an initial mesh parameterization using the Floater approach [6]: the boundary vertices of mesh $\mathcal{M}$ are mapped into the boundary vertices of $\mathcal{U}$ which form a polygon in the parameter plane $\mathbb{R}^2_{s,t}$ and for each inner vertex $\mathbf{p}_i$ of $\mathcal{M}$ its corresponding vertex $\mathbf{u}_i$ inside the polygon is selected such that the following local quadratic energy

$$E(\mathbf{u}_i) = \sum\nolimits_j w_{ij}(\mathbf{u}_j - \mathbf{u}_i)^2, \tag{2}$$

achieves its minimal value. Here $\{\mathbf{u}_j\}$ are vertices corresponding to the mesh one-link neighbors of $\mathbf{p}_i \in M$ and $\{w_{ij}\}$ are positive weights. Now the optimal positions for $\mathbf{u}_i$ are found by solving a sparse system of linear equations

$$\sum\nolimits_j w_{ij}\left(\mathbf{u}_j - \mathbf{u}_i\right) = 0. \tag{3}$$

This computationally simple procedure produces a valid parameterization of mesh $\mathcal{M}$ and avoids triangle flips if the boundary of $\mathcal{U}$ is a convex polygon [6].

Now let us estimate local stretch $\sigma_i = \sigma(\mathbf{u}_i)$ for each inner vertex $\mathbf{u}_i$ in the parametric plane. We redistribute the local stretches by assigning

$$w_{ij}^{\mathrm{new}} = w_{ij}^{\mathrm{old}} \big/ \sigma_j \tag{4}$$

in (2). The new positions of $\{\mathbf{u}_i\}$ are now found by solving (3).

We can think about vertices $\{\mathbf{u}_i\}$ and corresponding energies (2) in terms of a mass-spring system. For an area preserving parameterization, if a high (low) stretch is observed at $\mathbf{u}_i$, that is $\sigma_i > 1$ ($\sigma_i < 1$), we relax (strengthen) the springs connected with $\mathbf{u}_i$ by solving (3) with new weights (4). It works similarly for a general parameterization.

Our idea to diffuse the local stretches iteratively by (1), (3), (4) can resemble the relaxation approach of Balmelli et al. [2]. Notice however that in [2] the authors use a gradient-descent minimization approach (an explicit mesh evolution scheme) while our method is similar to quasi-Newton type minimization algorithms and an implicit mesh evolution scheme is used.

One can also find a similarity between (2), (4) and Winslow's variable diffusion method [18, 3] for adaptive grid generation (see also [17] for a comprehensive analysis of numerical methods used to minimize Winslow's variable diffusion functional).

We start from the shape preserving parameterization of Floater [6, § 6] $\mathcal{U}^0 = \left\{\mathbf{u}_i^0\right\}$ and then improve it gradually: $\mathcal{U}^{h+1} = \left\{\mathbf{u}_i^{h+1}\right\}$ is obtained from $\mathcal{U}^h = \left\{\mathbf{u}_i^h\right\}$ by solving (3) with weights w_{ij}^{h+1} defined by

$$w_{ij}^{h+1} = w_{ij}^h \big/ \sigma\left(\mathbf{u}_j^h\right).$$

Here w_{ij}^0 are the shape preserving weights proposed by Floater. The boundary vertices of the evolving mesh $\mathcal{U}^h$, $h = 0, 1, 2, \ldots,$ remain fixed. When solving (3) with $w_{ij} = w_{ij}^{h+1}$ numerically we use $\mathcal{U}^h$ as the initial guess for the numerical solver we employ.

We use the L^2 stretch metric of Sander et al. [15]

$$E_s^h = E_s(\mathcal{U}^h) = \sqrt{\sum A(T)\sigma(U^h)^2 \big/ \sum A(T)}, \tag{5}$$

where the sums are taken over all the triangles T of mesh $\mathcal{M}$, to define a stopping criterion. Namely, if $E_s^{h+1} \geq E_s^h$ we consider $\mathcal{U}^{\mathrm{opt}} = \left\{\mathbf{u}_i^h\right\}$ as an optimal low stretch mesh parameterization.

Besides $\mathcal{U}^{\mathrm{opt}}$ we also consider $\mathcal{U}^1 = \left\{\mathbf{u}_i^1\right\}$, the mesh parameterization obtained after one step of our optimization procedure since, according to our experiments, already the first step dramatically improves the parameterization quality.

We also can vary the strength of stretch redistribution (diffusion) step (4) by using the weights $\{\sigma_i^\eta\}$, $0 < \eta \leq 1$, instead of $\{\sigma_i\}$ in (4):

$$w_{ij}^{\mathrm{new}} = w_{ij}^{\mathrm{old}} \big/ \sigma_j^\eta. \tag{6}$$

Using (6) with $\eta < 1$ slows down the stretch minimization process but, on the other hand, often improves the mesh parameterization quality. The influence of exponent η in (6) is demonstrated in Fig. 4 for our single-step parameterization $\mathcal{U}^1$. Choosing smaller values for η leads to a less aggressive stretch minimization.

In the next section, we compare $\mathcal{U}^1$ and $\mathcal{U}^{\mathrm{opt}}$ with results produced by conventional mesh parameterization schemes.

3 Results and comparisons

Computing. All the examples presented in this section are computed using gcc 2.95 C++ compiler on a 1.7GHz Pentium 4 computer with 512MB RAM. To solve a system of linear equation $\mathbf{Ax} = \mathbf{b}$ we use PCBCG [13] with the maximum number of iterations equal to 10^4 and the approximation error $|\mathbf{Ax} - \mathbf{b}| / |\mathbf{b}|$ set to 10^{-6}.

Error metrics. To evaluate the visual quality of a parameterization we use the checkerboard texture shown in the bottom-left image of Fig. 2. For a quantitative evaluation of various mesh parameterization methods we employ L^2 stretch metric (5) and consider edge, angle, and area distortion error functions defined below. To measure the edge distortion error we use

$$\sum \left| \frac{|\mathbf{p}_i - \mathbf{p}_j|}{\sum |\mathbf{p}_i - \mathbf{p}_j|} - \frac{|\mathbf{u}_i - \mathbf{u}_j|}{\sum |\mathbf{u}_i - \mathbf{u}_j|} \right|,$$

where the sums are taken over all the edges of meshes $\mathcal{M}$ and $\mathcal{U}$. The angle distortion error is defined by

$$\frac{1}{3F}\sum_{j}\sum_{i=1}^{3}\left|\theta_{j,i}-\phi_{j,i}\right|,$$

where the sums are taken over all the angles $\theta_{j,i}$ and $\phi_{j,i}$ of the triangles of meshes $\mathcal{M}$ and $\mathcal{U}$, respectively, and F is the total number of triangles (faces) of $\mathcal{M}$. The area distortion is measured by

$$\sum\left|A\left(T_j\right)/\sum A\left(T_j\right)-A\left(U_j\right)/\sum A\left(U_j\right)\right|,$$

where the sums are taken over all the triangles of meshes $\mathcal{M}$ and $\mathcal{U}$.

Comparison and evaluation. We have implemented a number of conventional mesh parameterization methods and compared them with our low stretch technique:

(a)	Eck et al. harmonic map [5]
(b)	Floater's shape preserving parameterization [6, § 6]
(c)	Desbrun et al. intrinsic parameterization [4]
(d)	Sander et al. stretch minimizing parameterization [15]
(e)	Our single-step parameterization $\mathcal{U}^1$
(f_h)	Our optimal parameterization $\mathcal{U}^{\mathrm{opt}}$

The subindex h in (f_h) in the bottom row of the above table shows the total number of optimization steps (3), (4) needed to generate $\mathcal{U}^{\mathrm{opt}}$.

Tables 1-12 and Figures 3 and 6 present qualitative and visual comparisons of the above mesh parameterization schemes tested on various models topologically equivalent to a disk. The unit square is used as the parameter domain and for each models its the boundary vertices are fixed on the boundary of the square. The errors and computational times measured in seconds (s) and sometimes in minutes (**m**) and hours (**h**) are given.

For the intrinsic parameterization method [4], we use the equal blending of the Dirichlet and Authalic energies for all the models, except for the Fish model (Table 11) where we use only the Dirichlet energy in order to avoid triangle flips.

Our single-step mesh parameterization procedure (generating $\mathcal{U}^1$) is only slightly slower than the fast Floater and Eck et al. parameterization methods and faster than the intrinsic parameterization of Desbrun et al. [4]. Besides $\mathcal{U}^1$ demonstrates competitive results in minimizing the stretch, edge, area, and angle distortions.

Our optimal mesh parameterization procedure is also fast enough and sometimes achieves better results in stretch minimizing than the probabilistic minimization of Sander et al. [15] which is very slow. Moreover, by contrast with [15], $\mathcal{U}^{\mathrm{opt}}$ does not generate parameter cracks (see Fig. 6) because (3) acts like a diffusion process. Besides, if a very low stretch parameterization is needed, $\mathcal{U}^{\mathrm{opt}}$ can be used as an initial parameterization for [15].

Fig. 7 shows $\mathcal{U}^{\mathrm{opt}}$ parameterization of the Mannequin Head model when the parameter domain has boundaries of various shapes. The left images show the parameterization and corresponding texture mapping results when the boundary is the unit circle. The right images demonstrate similar results when the boundary of the parameter domain was obtained as the so-called natural boundary for the conformal parameterization of [4]. Notice that the stretch distortions near the boundary are substantially reduced in the latter case.

In Fig. 8 mesh parameterizations $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\mathrm{opt}}$ are evaluated and compared using the checkerboard texture. Sometimes $\mathcal{U}^{\mathrm{opt}}$ does not produce the best visual result because of high anisotropy and $\mathcal{U}^1$ is preferable. Finally, in Fig. 9 we analyze how the stretch distribution over a complex geometry model is changing during the optimization process $\mathcal{U}^0 \to \mathcal{U}^1 \to \mathcal{U}^{\mathrm{opt}}$. The top row of images presents the model (a decimated Max-Planck bust model) and results of checkerboard texture mapping with $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\mathrm{opt}}$. The four remaining images of the model show the stretch distribution over the model for $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\mathrm{opt}}$ parameterizations. The images demonstrate how well our stretch minimization procedures minimize and equalize the stretch. It is interesting to notice that near the mesh boundary the optimized meshes have large area and angle distortions (the same effect is observed in all the other tested models) but relatively low stretch distortions. One can hope that an appropriate relaxation of boundary conditions will reduce those area and angle distortions while maintaining low stretch.

Application to remeshing. In the right column of Fig. 3 and in Fig. 5 we demonstrate how our mesh parameterization technique can be used for fast and high quality remeshing of complex surfaces. We have chosen the interactive geometry remeshing scheme of Alliez et al. [1] and implemented its main steps:

1. Create a mesh parameterization.
2. Compute area, curvature, and control maps using hardware accelerated OpenGL commands.
3. Sample points by applying an error diffusion to the control map.
4. Connect the points using the Delaunay triangulation.
5. Use the parameterization to map the points into 3D.

A conformal mesh parameterization is the best choice for the described remeshing scheme.

It is clear that the remeshing quality depends on the size of an image used for the hardware assisted acceleration: the bigger size, the better result. On the other side, the image size is restricted by the graphics card memory. It turns out that a high quality remeshing can be obtained even for a relatively small image size. Let us assume that we have two parameterizations of a 3D mesh: a conformal parameterization and an area-preserving one. Then let us the area-preserving parameterization for computing the control map and resampling the points via an error diffusion process. Finally, the points are mapped from the area-preserving parameterization to the conformal one and are connected using the Delaunay triangulation.

The above remeshing modification has one drawback: it requires two parameterizations, conformal and area-preserving. However since our low-stretch parameterization $\mathcal{U}^{\mathrm{opt}}$ has nice area-preserving properties and the initial Floater's parameterization $\mathcal{U}^0$ is close to a conformal one, we use $\mathcal{U}^{\mathrm{opt}}$ and $\mathcal{U}^0$ instead of the conformal and area-preserving parameterizations in the above modification of the interactive geometry remeshing scheme of Alliez et al.

Right images (a)-(c) of Fig. 3 demonstrate results of the single-parameterization remeshing scheme if the discrete harmonic map parameterization [5], Floater's shape preserving parameterization [6], and intrinsic discrete conformal parameterization are used, respectively. Right images (d)-(f) of Fig. 3 present our experiments with the double-parameterization remeshing scheme. We set Floater's parameterization $\mathcal{U}^0$ as a substitute of a conformal parameterization and used $\mathcal{U}^0$ as an initial parameterization to generate the stretch-minimizing parameterization of Sander et al. [15] and $\mathcal{U}^1$ and $\mathcal{U}^{\mathrm{opt}}$. These low-stretch parameterizations were used as substitutes of an area-preserving parameterization. Fig. 5 presents remeshed Max-Planck bust and Stanford bunny models obtained by the remeshing schemes based on (from left to right) $\{\mathcal{U}^0\}$, $\{\mathcal{U}^0,\mathcal{U}^1\}$, and $\{\mathcal{U}^0,\mathcal{U}^{\mathrm{opt}}\}$ parameterizations (here using $\{\mathcal{U}',\mathcal{U}''\}$ parameterizations means that we use $\mathcal{U}'$ as a substitute of a conformal parameterization and $\mathcal{U}''$ as a substitute of an area preserving one). Notice that the double-parameterization remeshing scheme with $\{\mathcal{U}^0,\mathcal{U}^{\mathrm{opt}}\}$ produces the best results.

4 Conclusion

We have presented a fast and powerful method for generating low-stretch mesh parameterizations and demonstrate its applicability to high quality texture mapping and remeshing. Our method is much faster than the stochastic stretch minimization procedure of Sander et al. [15] (note that their more recent coarse-to-fine stretch optimization procedure [14] is significantly faster than that of [15] but still slower than ours) and often produces better quality results. In particular, it does not generate parameter cracks.

Our method is heuristic. At present we are not able to support it by rigorous mathematical results.

In future we plan to extend our approach to spherical parameterizations and use quadratic energies with matrix weights for incorporating an anisotropy in our optimization process.

Acknowledgments

The authors would like to thank Hugues Hoppe for a helpful discussion and the anonymous reviewers of this paper for their valuable and constructive comments. The models are courtesy of the Stanford University (bunny and dragon), the University of Washington (mannequin head and fish), Cyberware Inc. (Igea), and MPI für Informatik (Max-Planck bust).

References

[1] P. Alliez, M. Meyer, and M. Desbrun. Interactive geometry remeshing. In *Proceedings of ACM SIGGRAPH 2002*, pages 347–354, 2002.

[2] L. Balmelli, G. Taubin, and F. Bernardini. Space-optimized texture maps. In *Proceedings of EUROGRAPHICS 2002*, pages 411–420, 2002.

[3] W. Cao, W. Huang, and R. D. Russell. Approaches for generating moving adaptive meshes: location versus velocity. *Appl. Numer. Math.*, 47:121–138, 2003.

[4] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. In *Proceedings of EUROGRAPHICS 2002*, pages 209–218, 2002.

[5] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzl. Multiresolution analysis of arbitrary meshes. In *Proceedings of ACM SIGGRAPH 1995*, pages 173–182, 1995.

[6] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

[7] M. S. Floater and K. Hormann. Recent advances in surface parameterization. In *Multiresolution in Geometric Modelling*, pages 259–284, 2003.

[8] X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proceedings of Eurographics Symposium on Geometry Processing 2003*, pages 135–146, 2003.

[9] S. Haker, S. Angenent, A. Tannenbaum, R. Kikinis, G. Sapiro, and M. Halle. Conformal surface parameterization for texture mapping. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):181–189, 2000.

[10] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generations. In *Proceedings of ACM SIGGRAPH 2002*, pages 362–371, 2002.

[11] J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Proceedings of ACM SIGGRAPH 1993*, pages 27–34, 1993.

[12] E. Praun and H. Hoppe. Spherical parametrization and remeshing. In *Proceedings of ACM SIGGRAPH 2003*, pages 340–349, 2003.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies in C.* Cambridge University Press, 1988.

[14] P. V. Sander, S. J. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. In *Proceedings of Eurographics Workshop on Rendering 2002*, pages 87–98, 2002.

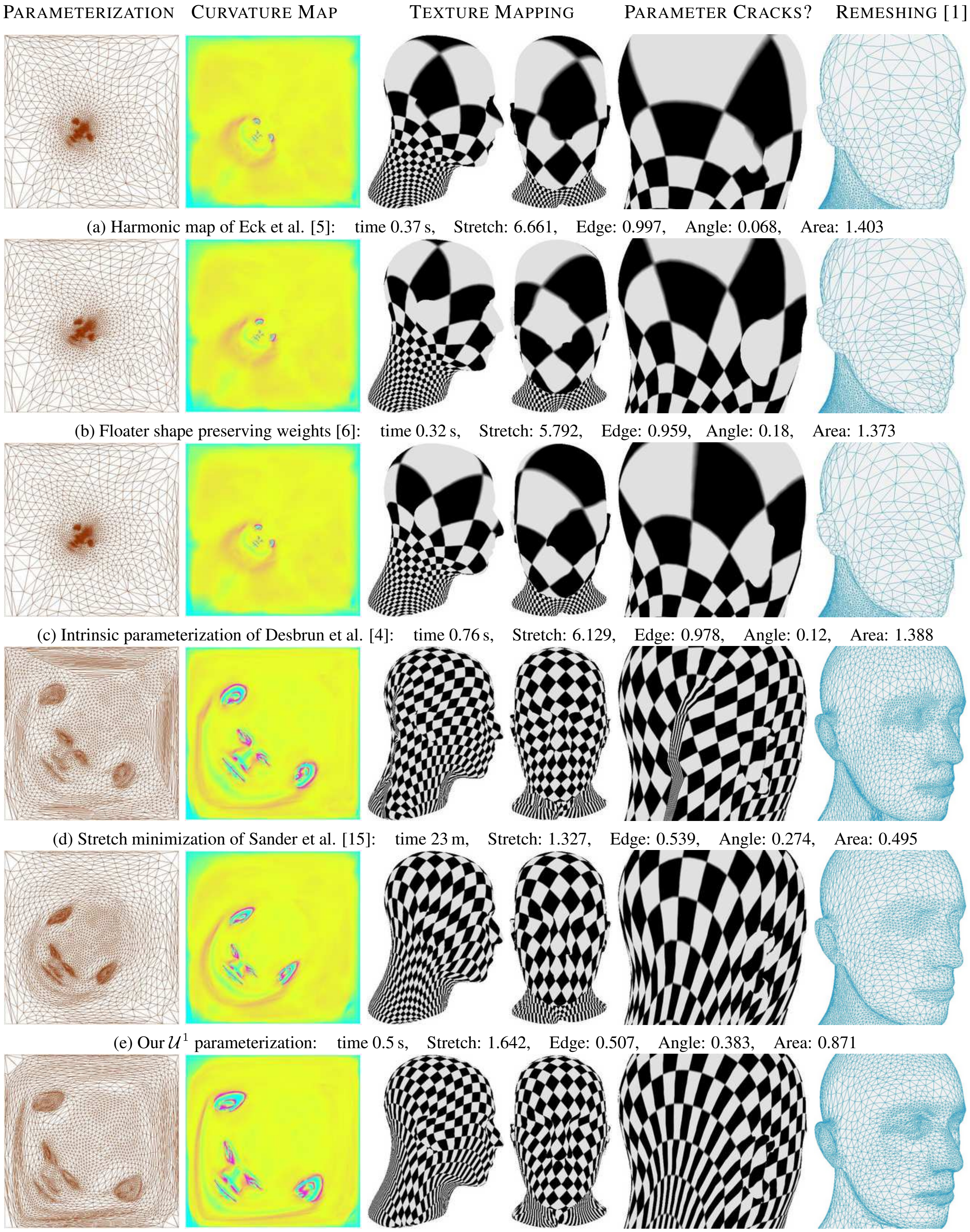
[15] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe. Texture mapping progressive meshes. In *Proceedings of ACM SIGGRAPH 2001*, pages 409–416, 2001.

[16] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization*, pages 355–362, 2002.

[17] A. M. Winslow. Numerical solution of the quasilinear poisson equation in a nonuniform. *J. Comput. Phys.*, 2:149–172, 1967.

[18] A. M. Winslow. Adaptive mesh zoning by equipotential method. Technical report, Lawrence Livermore Laboratory, 1981. Report UCID-19062.

[19] E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. Technical report, Georgia Institute of Technology, 2003. GVU Tech Report 03-29.



(a) Harmonic map of Eck et al. [5]: time 0.37 s, Stretch: 6.661, Edge: 0.997, Angle: 0.068, Area: 1.403

(b) Floater shape preserving weights [6]: time 0.32 s, Stretch: 5.792, Edge: 0.959, Angle: 0.18, Area: 1.373

(c) Intrinsic parameterization of Desbrun et al. [4]: time 0.76 s, Stretch: 6.129, Edge: 0.978, Angle: 0.12, Area: 1.388

(d) Stretch minimization of Sander et al. [15]: time 23 m, Stretch: 1.327, Edge: 0.539, Angle: 0.274, Area: 0.495

(e) Our $\mathcal{U}^1$ parameterization: time 0.5 s, Stretch: 1.642, Edge: 0.507, Angle: 0.383, Area: 0.871

(f) Our $\mathcal{U}^{\text{opt}} = \mathcal{U}^3$ parameterization: time 1.09 s, Stretch: 1.382, Edge: 0.4748, Angle: 0.4132, Area: 0.3832

Figure 3: Comparison of various mesh parameterization schemes on the Mannequin Head model ($V = 2732$, $F = 5420$).

	time	Stretch	Edge	Angle	Area
(a)	0.06 s	6.6507	0.9918	0.125	1.4032
(b)	0.06 s	5.9171	0.9635	0.1995	1.3801
(c)	0.12 s	6.2751	0.9778	0.1619	1.3931
(d)	80.91 s	1.375	0.5162	0.2952	0.5232
(e)	0.08 s	1.6691	0.5084	0.3717	0.8836
(f_3)	0.16 s	1.4084	0.4814	0.4479	0.4165

Table 1: Mannequin Head model: $V = 689$, $F = 1355$

	time	Stretch	Edge	Angle	Area
(a)	0.21 s	1.9708	0.4935	0.0969	0.8455
(b)	0.17 s	1.8084	0.4648	0.1568	0.8409
(c)	0.33 s	1.8511	0.4753	0.1189	0.84
(d)	213 s	1.172	0.2996	0.2239	0.3043
(e=f_1)	0.3 s	1.2057	0.2862	0.2881	0.3179

Table 2: Cat Head model: $V = 1856$, $F = 3660$

	time	Stretch	Edge	Angle	Area
(a)	0.37 s	6.6617	0.9971	0.0685	1.4036
(b)	0.32 s	5.7921	0.9599	0.1807	1.3733
(c)	0.76 s	6.1295	0.9784	0.1209	1.3886
(d)	23 **m**	1.3279	0.5393	0.2744	0.4956
(e)	0.5 s	1.6425	0.5073	0.3838	0.8717
(f_3)	1.09 s	1.382	0.4748	0.4132	0.3832

Table 3: Refined Mannequin Head model: $V = 2732$, $F = 5420$

	time	Stretch	Edge	Angle	Area
(a)	1.23 s	13.306	0.7563	0.1041	1.0207
(b)	0.87 s	11.729	0.6976	0.2545	0.9526
(c)	1.81 s	12.266	0.7232	0.176	0.9795
(d)	1 **h**	1.3408	0.4955	0.3477	0.4227
(e)	1.5 s	1.7643	0.4551	0.3735	0.4676
(f_3)	3.44 s	1.4791	0.4661	0.5226	0.3613

Table 4: Cat model: $V = 5649$, $F = 11168$

	time	Stretch	Edge	Angle	Area
(a)	3.16 s	18.027	1.2288	0.0361	1.692
(b)	2.29 s	15.941	1.2074	0.1441	1.6373
(c)	17.4 s	16.933	1.2157	0.0857	1.6618
(d)	57.5**h**	1.3257	0.7021	0.2501	0.5436
(e)	4.18 s	2.2037	0.6249	0.372	1.1899
(f_3)	9.22 s	1.5392	0.5623	0.4905	0.6217

Table 5: Decimated Max-Planck bust model: $V = 9462$, $F = 18866$

	time	Stretch	Edge	Angle	Area
(a)	12.9 s	1.5348	0.3025	0.1313	0.5063
(b)	6.21 s	1.485	0.3412	0.1748	0.5651
(c)	25.8 s	43.947	0.7602	0.3622	1.0085
(d)	4.5 **h**	1.2226	0.2833	0.1934	0.4338
(e)	17.9 s	1.2105	0.2477	0.2112	0.3876
(f_3)	42.6 s	1.1718	0.24	0.2636	0.2375

Table 6: Fandisk model: $V = 9919$, $F = 19617$

	time	Stretch	Edge	Angle	Area
(a)	5.55 s	9179549	1.6037	0.0915	1.7599
(b)	4.24 s	1120318	1.5049	0.3491	1.7175
(c)	21.1 s	231989	1.5494	0.2707	1.7387
(d)	39.7 **h**	7635.3	1.1442	0.3544	0.8435
(e)	6.99 s	313.64	0.9883	0.6341	1.4739
(f_8)	33.2 s	3.5688	0.8522	0.8253	0.7897

Table 7: Half-of-Dragon model: $V = 13927$, $F = 27782$

	time	Stretch	Edge	Angle	Area
(a)	12.4 s	9462.1	0.9729	0.0704	1.5132
(b)	8.95 s	181.05	0.9983	0.3852	1.5725
(c)	90.7 s	320.53	0.9845	0.2281	1.5425
(d)	43.4 **h**	1.6816	0.7193	0.2917	0.6665
(e)	14.7 s	3.3929	0.5041	0.6184	0.8078
(f_3)	32.3 s	2.884	0.6399	0.7747	0.5344

Table 8: Dragon Head model: $V = 23929$, $F = 47783$

	time	Stretch	Edge	Angle	Area
(a)	11.2 s	3.4799	0.7924	0.0542	1.3399
(b)	8.46 s	4.676	0.8678	0.1627	1.3664
(c)	93.8 s	34.621	0.8104	0.1831	1.3525
(d)	18.6 **h**	1.3092	0.4603	0.2265	0.5492
(e)	15.2 s	1.4373	0.4166	0.3446	0.6868
(f_2)	27.2 s	1.304	0.385	0.3923	0.4123

Table 9: Igea model: $V = 24720$, $F = 49301$

	time	Stretch	Edge	Angle	Area
(a)	17.9 s	712.33	0.7097	0.0797	1.098
(b)	13.2 s	85.181	0.7241	0.1522	1.0861
(c)	231 s	672.45	0.7062	0.2866	1.0957
(d)	55.6 **h**	1.5159	0.4982	0.3109	0.4868
(e)	22.5 s	4.7926	0.4582	0.387	0.5632
(f_6)	79.8 s	1.8755	0.6143	0.6065	0.5241

Table 10: Stanford Bunny model: $V = 31272$, $F = 62247$

	time	Stretch	Edge	Angle	Area
(a)	92.4 s	6.3061	0.8241	0.0445	1.3021
(b)	66.3 s	6.092	0.7752	0.1782	1.2613
(c')	486 s	6.306	0.8241	0.0445	1.3021
(d)	120 **h**	2.5689	0.6481	0.2444	0.926
(e)	125 s	1.5683	0.4252	0.3476	0.6387
(f_2)	206 s	1.5041	0.4414	0.4678	0.3946

Table 11: Fish model: $V = 64982$, $F = 129664$

	time	Stretch	Edge	Angle	Area
(a)	250 s	18.207	1.2578	0.03	1.6936
(b)	204 s	18.1025	1.25	0.0512	1.6912
(c)	52.1 **m**	2.8434	1.2341	0.3068	1.6924
(e)	384 s	2.2094	0.6598	0.3698	1.2017
(f_3)	848 s	1.4926	0.5939	0.4865	0.4812

Table 12: Max-Planck bust model: $V = 199169$, $F = 398043$

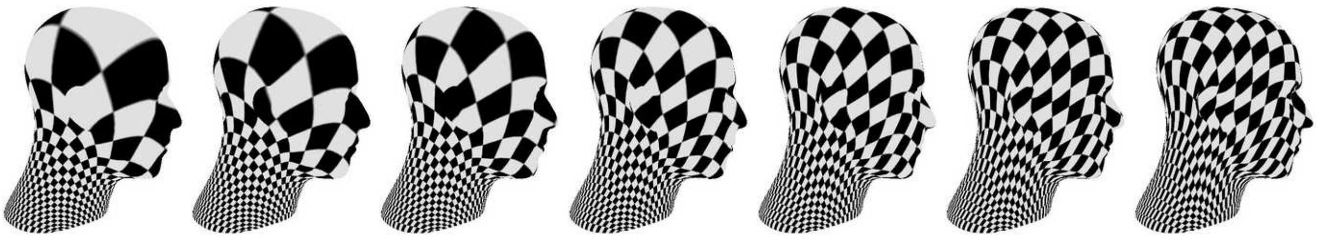


Figure 4: Choosing smaller values for η leads to a less aggressive stretch minimization. From left to right: $\mathcal{U}^1$ parameterization of Mannequin Head with $\eta = \{0, 0.1, 0.2, 0.4, 0.6, 0.8, 1\}$.

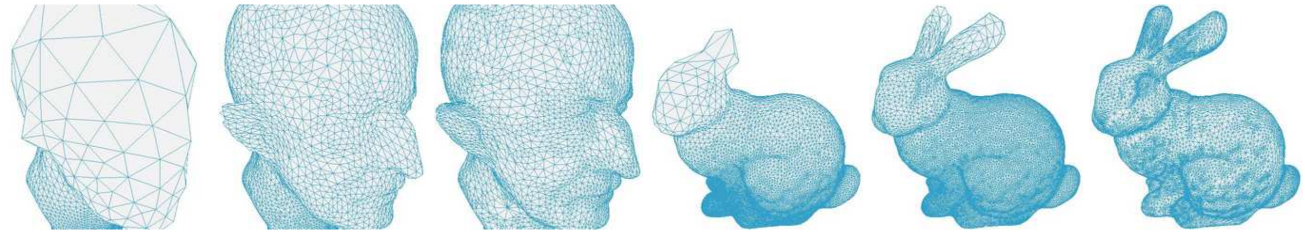


Figure 5: Remeshing of Max-Planck bust model (three left images) and Stanford bunny (three right images) models. For each model remeshings according to $\mathcal{U}^0$, $\{\mathcal{U}^0, \mathcal{U}^1\}$, and $\{\mathcal{U}^0, \mathcal{U}^{\mathrm{opt}}\}$ are shown. See the text for details.

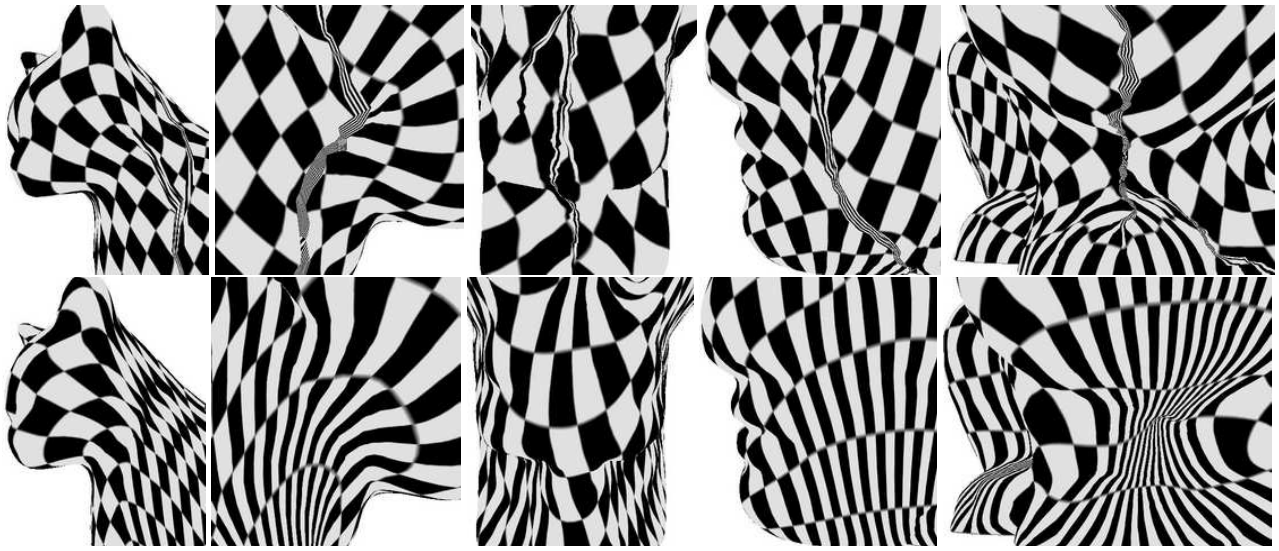
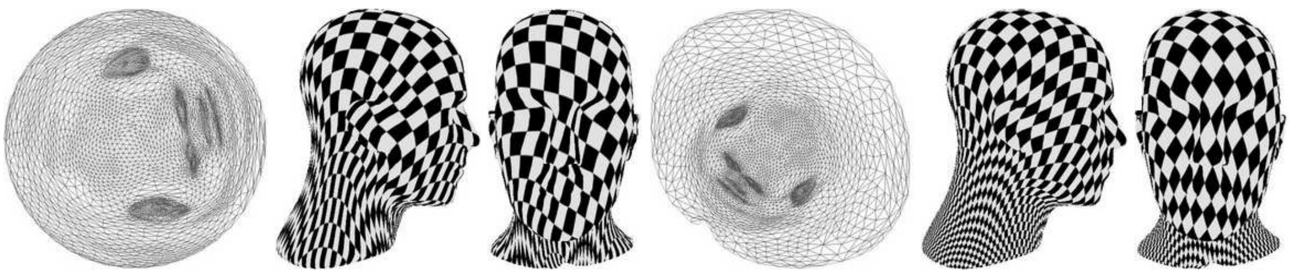


Figure 6: Parameter cracks on various models textured with checkerboard texture. The images of the upper row demonstrate parameter cracks generated by the stretch-minimization method of Sander et al. [15]. The images of the bottom row show the same parts of the models parameterized by our $\mathcal{U}^{\mathrm{opt}}$.



$\mathcal{U}^{\mathrm{opt}} = \mathcal{U}^5$ with circular parameter domain. Time: 1.51 s, Stretch: 1.34, Edge: 0.43, Angle: 0.47, Area: 0.4.

$\mathcal{U}^{\mathrm{opt}} = \mathcal{U}^1$ with natural boundary [4]. Time:1.67 s, Stretch: 1.68, Edge: 0.5, Angle: 0.37, Area: 0.9.

Figure 7: Using various parameter domains for $\mathcal{U}^{\mathrm{opt}}$.

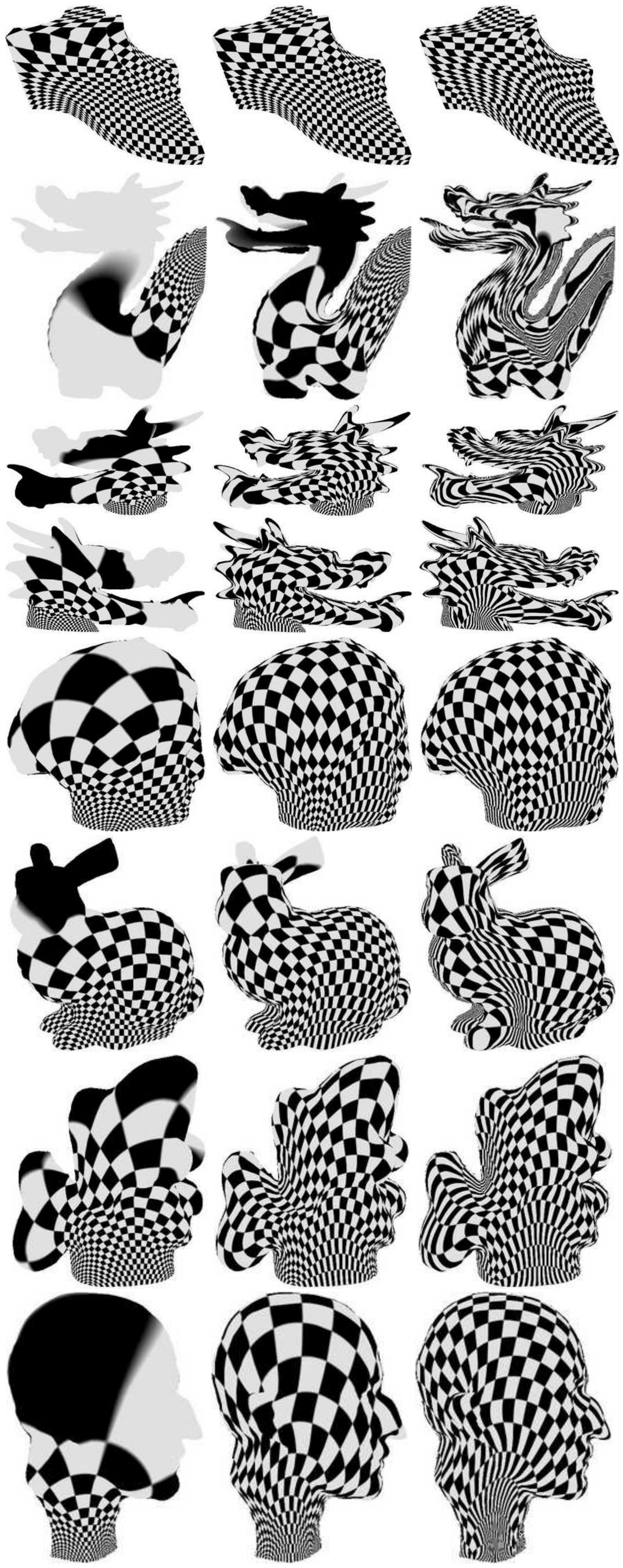


Figure 8: Checkerboard texture mapping with U^0 (left), U^1 (middle), and U^{opt} (right).

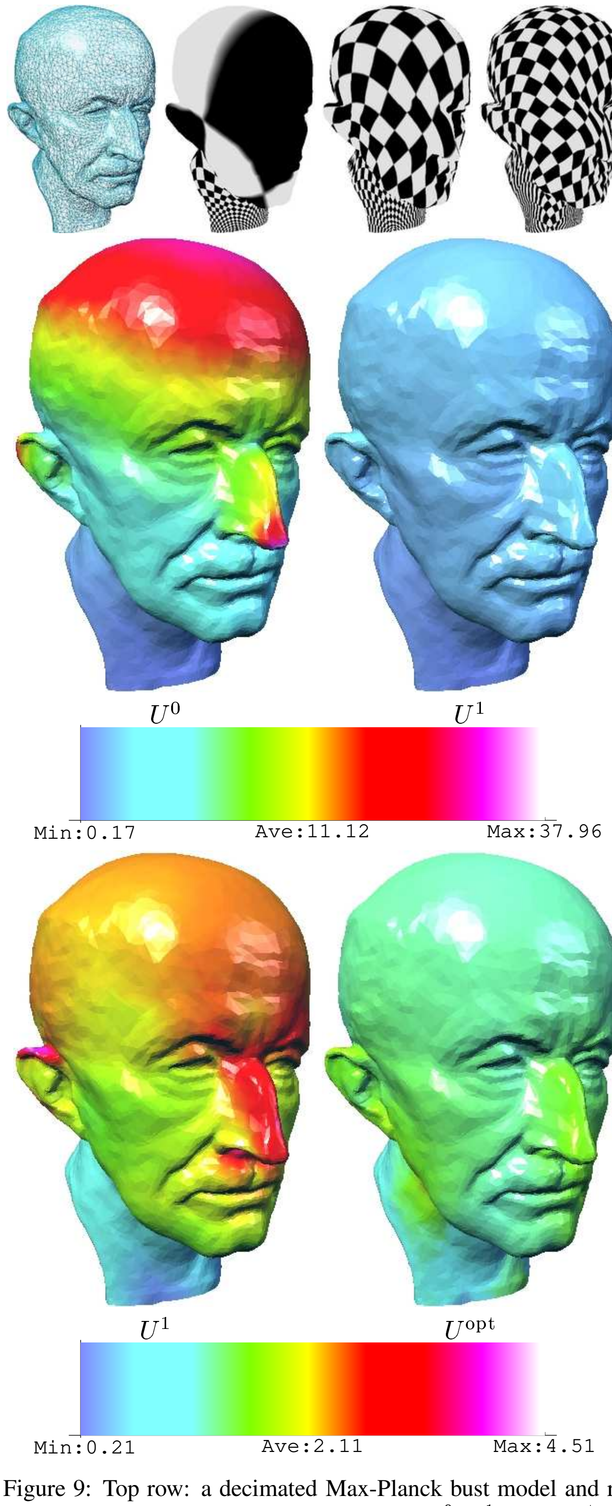


Figure 9: Top row: a decimated Max-Planck bust model and results of checkerboard texture mapping with $\mathcal{U}^0, \mathcal{U}^1$, and $\mathcal{U}^{\text{opt}}$ parameterizations. The four remaining images of the model show the distribution of the vertex stretches over the model for $\mathcal{U}^0, \mathcal{U}^1$, and $\mathcal{U}^{\text{opt}}$. Firstly coloring by stretch $\sigma \in [0.17, 37.96]$ is used to compare $\mathcal{U}^0$ and $\mathcal{U}^1$. Then the same coloring scheme on the stretch interval $[0.21, 4.51]$ is employed to compare the stretch distributions for $\mathcal{U}^1$, and $\mathcal{U}^{\text{opt}}$. Here the bounds of the interval are equal to the maximal and minimal stretch values.