International Journal of Shape Modeling
Vol. 11, No.1 (2005) 25-42

A MOVING MESH APPROACH TO STRETCH-MINIMIZING MESH PARAMETERIZATION

SHIN YOSHIZAWA
AG4, MPI Informatik, Stuhlsatzenhausweg 85, 66123, Saarbrücken, Germany
shin.yoshizawa@mpi-sb.mpg.de

ALEXANDER BELYAEV
AG4, MPI Informatik, Stuhlsatzenhausweg 85, 66123, Saarbrücken, Germany
belyaev@mpi-sb.mpg.de

HANS-PETER SEIDEL
AG4, MPI Informatik, Stuhlsatzenhausweg 85, 66123, Saarbrücken, Germany
hpseidel@mpi-sb.mpg.de

We propose to use a moving mesh approach, a popular grid adaption technique in computational mechanics, for fast generating low-stretch mesh parameterizations. Given a triangle mesh approximating a surface, we construct an initial parameterization of the mesh 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.

2000 Mathematics Subject Classification: 68U05, 65D18, 65D17

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

[†]It is an extension of our previous work [25].

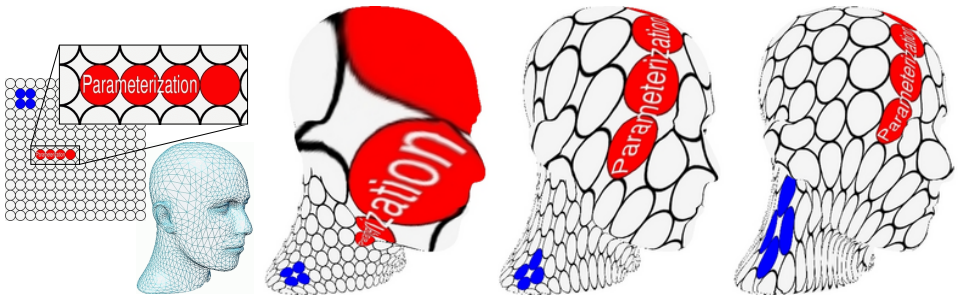


Fig. 1. Texture mapping of the Mannequin Head model with three mesh parameterizations used in our method. (a) Texture and model. (b) Floater's shape preserving parameterization is used as an initial mesh parameterization. (c) After a single optimization pass. (d) Our optimal low-stretch parameterization.

triangulation of a planar polygon. An excellent survey of recent advances in mesh parameterization is given in [10], 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 [21] have not yet been proposed.

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

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 $E = \mathbf{p}_s^2$, $F = \mathbf{p}_s \cdot \mathbf{p}_t$, and $G = \mathbf{p}_t^2$. 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 $\mathbf{p}$. For example, if $\Gamma(s,t) = \gamma(s,t)$, the parameterization is conformal and mapping $\mathbf{p} = \mathbf{p}(s,t)$ preserves angles.

Since the conformal mappings are well understood mathematically, discrete approximations of conformal mappings are widely used for mesh parameterization purposes [12,14,6,11]. 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 $\mathbf{p} = \mathbf{p}(s,t)$ by $\sqrt{(\Gamma^2+\gamma^2)/2} = \sqrt{(E+G)/2}$ [21]. Stretch minimizing mesh parameterizations were considered in [21,20,18]. See also [23] where a similar stretch measure is proposed and [17,27] where the Green-Lagrange tensor is used to measure the stretch.

While the stretch minimization approach proposed in [21] and further developed in [20] and [27] leads to generating high-quality mesh parameterizations, the computational procedure used in [21,20,27] for stretch minimization is time consuming. Besides the mesh parameterization procedure of [21,20] 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*. Figure 2 demonstrates an appearance of such parameter cracks on the textured Mannequin Head model parametrized by the stretch minimization method from [21].

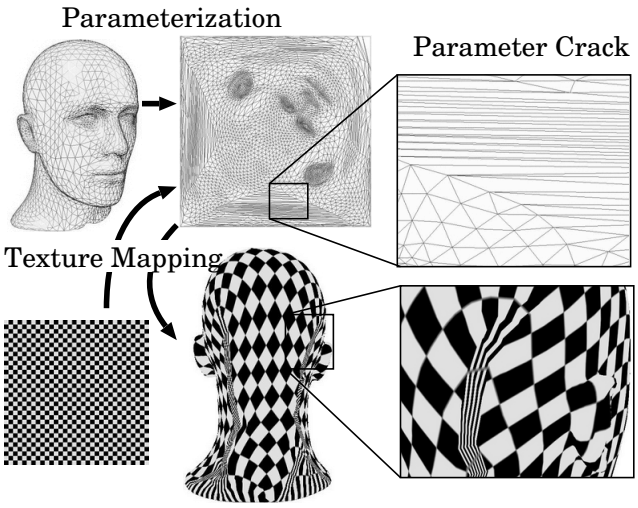


Fig. 2. Parameter cracks on textured Mannequin Head model parametrized by the stretch minimization method of Sander et al.

In [18] 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 an initial mesh parameterization and then 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 [8].

Our method can be considered as an error redistribution (diffusion) procedure applied to local stretches. The error redistribution (also known as the moving mesh method or r-method) is a powerful mesh adaption technique in computational mechanics (see, for example, [15,4] and references therein). It has become popular after seminal works of De Boor [5] and Bab**u**uska and Rheiboldt [2]. The general idea behind the approach is extremely simple: let us move mesh vertices to positions where they are mostly needed. Obviously this leads to error equalization w.r.t. a user-specified error measure (energy) often called a monitor function in computational mechanics studies. Error equalization resembles a diffusion process and can be governed by a system of partial differential equations [4,24]. In the geometric modeling field, it generalizes Laplacian smoothing and similar ideas were used for mesh parameterization purposes [22,25,26] and optimizing texture maps [3].

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.

Figure 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. In Section 4, we discuss how the procedure depends on the initial mesh parameterization and consider meshes with multiple boundaries. We conclude in Section 5.

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 [21,27]. As we mentioned above, quantity

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

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 [8]: 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

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

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 [8].

Notice that modifying weights $\{w_{ij}\}$ in quadratic energy (2) and, consequently, in (3) modifies the mesh parameterization. Thus one can improve the mesh parameterization initially determined by (3) with weights $\left\{w_{ij}^{\text{old}}\right\}$ via selecting better weights $\left\{w_{ij}^{\text{new}}\right\}$. In our mesh optimization procedure, we exploit this simple observation and choose weights $\left\{w_{ij}^{\text{new}}\right\}$ such that vertices $\{\mathbf{u}_j\}$ are moved toward locations where they are mostly needed.

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

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

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) resembles mesh moving techniques discussed in the previous section.

We start from an initial parameterization $\mathcal{U}^0 = \left\{\mathbf{u}_i^0\right\}$ 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

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

We select w_{ij}^0 as the shape preserving weights proposed by Floater [8] . 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. [21]

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

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 $\mathcal{U}^{\text{opt}} = \{\mathbf{u}_i^h\}$ as an optimal low stretch mesh parameterization.

Besides $\mathcal{U}^{\text{opt}}$ we also consider $\mathcal{U}^1 = \{\mathbf{u}_i^1\}$, 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):

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

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 Figure 9 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 $\mathcal{U}^{\text{opt}}$ 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 [19] 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 Figure 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

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

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 [7]
(b)	Floater's shape preserving parameterization [8]
(c)	Desbrun et al. intrinsic parameterization [6]
(d)	Sander et al. stretch minimizing parameterization [21]
(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 6-8, and 11 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 [6], 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. [6]. 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. [21] which is very slow. Moreover, by contrast with [21], $\mathcal{U}^{\mathrm{opt}}$ does not generate parameter cracks (see Figure 11) 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 [21].

Figure 12 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 [6]. Notice that the stretch distortions near the boundary are substantially reduced in the latter case.

In Figure 13 mesh parameterizations $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\text{opt}}$ are evaluated and compared using the checkerboard texture. Sometimes $\mathcal{U}^{\text{opt}}$ does not produce the best visual result because of high anisotropy and $\mathcal{U}^1$ is preferable. Finally, in Figure 14 we analyze how the stretch distribution over a complex geometry model is changing during the optimization process $\mathcal{U}^0 \rightarrow \mathcal{U}^1 \rightarrow \mathcal{U}^{\text{opt}}$. 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 $\mathcal{U}^{\text{opt}}$. The four remaining images of the model show the stretch distribution over the model for $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\text{opt}}$ 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 columns of Figures 6-8 and in Figure 10 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 $\mathcal{U}^{\text{opt}}$ has nice area-preserving properties and the initial Floater's parameter-

ization $\mathcal{U}^0$ is close to a conformal one, we use $\mathcal{U}^{\rm opt}$ 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.

The right images of rows (a)-(c) of Figures 6-8 demonstrate results of the single-parameterization remeshing scheme if the discrete harmonic map parameterization [7], Floater's shape preserving parameterization [8], and intrinsic discrete conformal parameterization are used, respectively. The right images of rows (d)-(f) of Figures 6-8 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. [21] and $\mathcal{U}^1$ and $\mathcal{U}^{\rm opt}$. These low-stretch parameterizations were used as substitutes of an area-preserving parameterization. Figure 10 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 $\{\mathcal{U}^0,\mathcal{U}^{\rm opt}\}$ 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 $\{\mathcal{U}^0,\mathcal{U}^{\rm opt}\}$ yields the best results.

4. Discussion

The final result of our mesh optimization method depends on the choice of initial weights $\{\mathbf{u}_i^0\}$. In particular we found out that selecting Floater's shape preserving weights [8] leads to a very effective stretch minimization procedure. Even better results are often obtained if the so-called cotangent weights [6] are used for generating the initial parameterization $\mathcal{U}^0$. However since cotangent weights are not necessary positive, using them may generate triangle flips.

One interesting situation when the choice of shape preserving weights is not very appropriate consists of parameterizing meshes with multiple boundaries, see the left image of Figure3 for such a mesh topologically equivalent to a sphere with holes. One solution to create a good initial parameterization of such a mesh consists of the following. Let us choose one hole (the biggest one) as the outer hole and the remaining holes as inner holes. Let us triangulate the inner holes and then use the shape preserving weights. Alternatively, for each edge $[\mathbf{x}_i,\mathbf{x}_j]$ of an inner hole, according to the right image of Figure3, we can compute angles needed to generate either the mean value weights [9]

$$\frac{\tan(\theta_{ij}/2)+\tan(\phi_{ij}/2)}{|\mathbf{x}_i-\mathbf{x}_j|}$$

or cotangent weights

$$\cot(\alpha_{ij})+\cot(\beta_{ij})$$

and use either of these sets of weights for generating the initial parameterization $\mathcal{U}^0$.

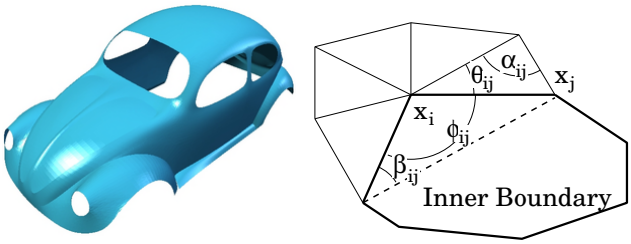


Fig. 3. Left: a mesh with multiple boundaries. Right: the angles needed to define the cotangent and mean value weights for boundary vertices.

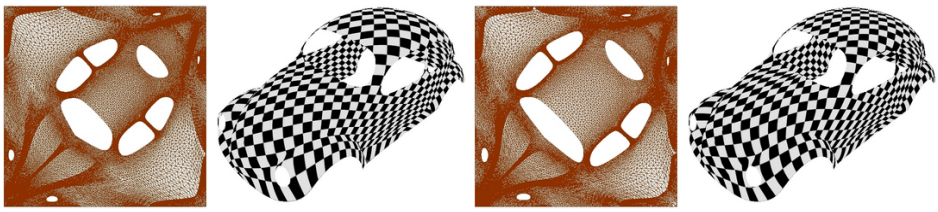


Fig. 4. Left: the cotangent (harmonic) weights are used to generate $\mathcal{U}^0$; stretch L^2 error = 1.495, stretch L^∞ error = 360.4. Right: $\mathcal{U}^{\rm opt} = \mathcal{U}^1$; stretch L^2 error = 1.178, stretch L^∞ error = 20.13.

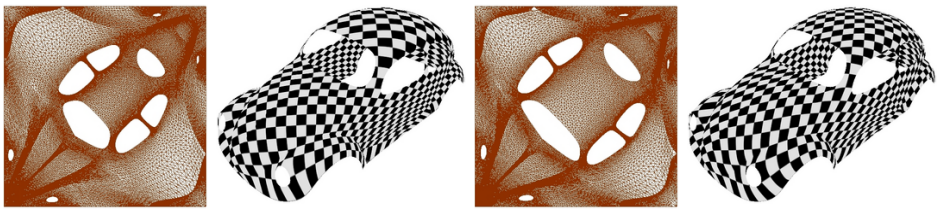


Fig. 5. Left: the mean value weights are used to generate $\mathcal{U}^0$; stretch L^2 error = 1.395, stretch L^∞ error = 172.7. Right: $\mathcal{U}^{\rm opt} = \mathcal{U}^1$; stretch L^2 error = 1.181, stretch L^∞ error = 21.37.

This technique as well as the virtual boundary method of Lee et al [13] is developed for dealing with mesh parameterizations defined over non-convex parameter domains. In contrast to [13] our approach is especially designed for processing meshes with holes. The use of the virtual boundary method [13] for meshes with holes would require a nontrivial hole filling procedure (see, for example, [16]) as a preprocessing step.

Figures 4 and 5 demonstrate the power of this our technique and show parameterizations $\mathcal{U}^0$ and $\mathcal{U}^{\rm opt}$ obtained for the Car model.

5. 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. [21] (note that their more recent coarse-to-fine stretch optimization procedure [20] is significantly faster than that of [21] but still slower than ours) and often produces better quality results. In particular, it does not generate parameter cracks.

Our approach is heuristic. Although it has much in common with mesh moving techniques widely used in computational mechanics and often justified mathematically, at present we are not able to support our approach by rigorous mathematical results. In future we would be glad to justify the effectiveness of our approach rigorously.

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. Bab**u**uska and Rheiboldt. A posteriori error estimates for the finite element method. *Int. J. Numer. Meth. Eng.*, 12:1597–1615, 1978.
3. L. Balmelli, G. Taubin, and F. Bernardini. Space-optimized texture maps. In *Proceedings of EUROGRAPHICS 2002*, pages 411–420, 2002.
4. W. Cao, W. Huang, and R. D. Russell. Approaches for generating moving adaptive meshes: location versus velocity. *Appl. Numer. Math.*, 47:121–138, 2003.
5. C. De Boor. Good approximation by splines with variable knots ii. In *Conference on the Numerical Solution of Differential Equations, Lecture Notes in Mathematics. No. 363*, pages 12–20, 1973.
6. M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. In *Proceedings of EUROGRAPHICS 2002*, pages 209–218, 2002.
7. 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.
8. M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.
9. M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.
10. M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In *Multiresolution in Geometric Modelling*, pages 157–186, 2004.
11. X. Gu and S.-T. Yau. Global conformal surface parameterization. In *Proceedings of Eurographics Symposium on Geometry Processing 2003*, pages 135–146, 2003.
12. 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.
13. Y. Lee, H. S. Kim, and S. Lee. Mesh parameterization with a virtual boundary. *Computers and Graphics*, 26(5):677–686, 2002.

14. 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.
15. R. Li, T. Tang, and P. Zhang. Moving mesh methods in multiple dimensions based on harmonic maps. *Journal of Computational Physics*, 170:562–588, 2001.
16. P. Liepa. Filling holes in meshes. In *Proceedings of Eurographics Symposium on Geometry Processing 2003*, pages 200–205, 2003.
17. J. Maillot, H. Yahia, and A. Verroust. Interactive texture mapping. In *Proceedings of ACM SIGGRAPH 1993*, pages 27–34, 1993.
18. E. Praun and H. Hoppe. Spherical parametrization and remeshing. In *Proceedings of ACM SIGGRAPH 2003*, pages 340–349, 2003.
19. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipies in C*. Cambridge University Press, 1988.
20. 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.
21. 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.
22. A. Sheffer and E. de Sturler. Smoothing an overlay grid to minimize linear distortion in texture mapping. *ACM Transactions on Graphics*, 21(4):874–890, 2002.
23. O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of IEEE Visualization*, pages 355–362, 2002.
24. A. M. Winslow. Numerical solution of the quasilinear poisson equation in a nonuniform. *J. Comput. Phys.*, 2:149–172, 1967.
25. S. Yoshizawa, A. Belyaev, and H.-P. Seidel. A fast and simple stretch-minimizing mesh parameterization. In *Proceedings of Shape Modeling International 2004*, pages 200–208, 2004.
26. R. Zayer, C. Rössl, and H.-P. Seidel. Discrete tensorial quasi-harmonic maps. In *Shape Modeling International 2005*, 2005.
27. E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24(1):1–27, 2005.

	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

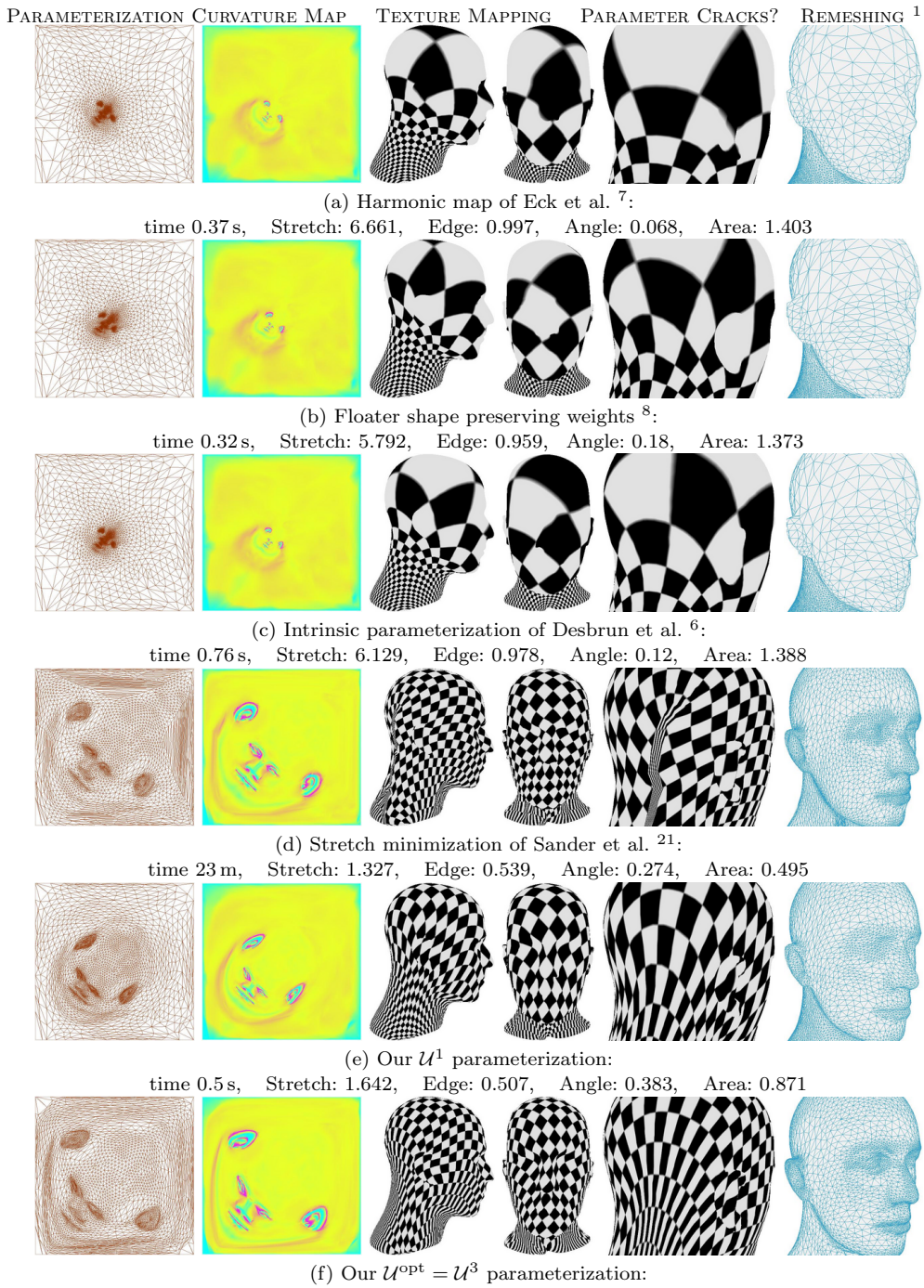
Table 10. Stanford Bunny model: $V = 31272$, $F = 62247$

	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: $V = 199169$, $F = 398043$



(a) Harmonic map of Eck et al. [7]:
time 0.37 s, Stretch: 6.661, Edge: 0.997, Angle: 0.068, Area: 1.403

(b) Floater shape preserving weights [8]:
time 0.32 s, Stretch: 5.792, Edge: 0.959, Angle: 0.18, Area: 1.373

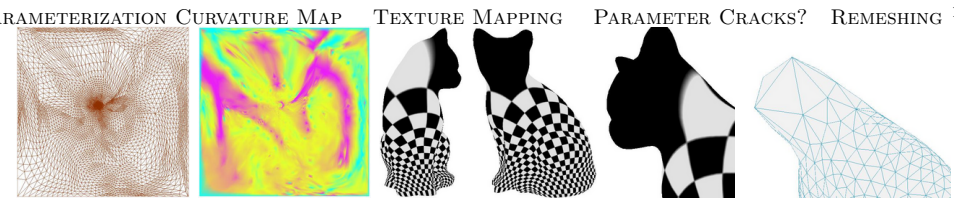
(c) Intrinsic parameterization of Desbrun et al. [6]:
time 0.76 s, Stretch: 6.129, Edge: 0.978, Angle: 0.12, Area: 1.388

(d) Stretch minimization of Sander et al. [21]:
time 23 m, Stretch: 1.327, Edge: 0.539, Angle: 0.274, Area: 0.495

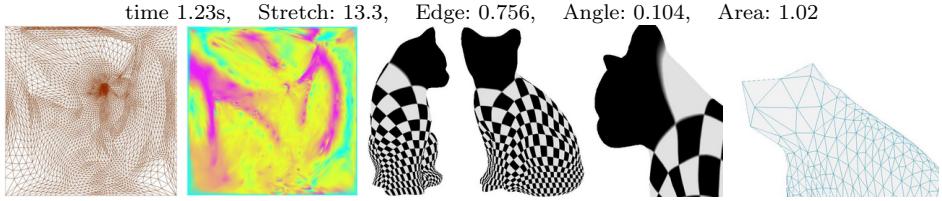
(e) Our $\mathcal{U}^1$ parameterization:
time 0.5 s, Stretch: 1.642, Edge: 0.507, Angle: 0.383, Area: 0.871

(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

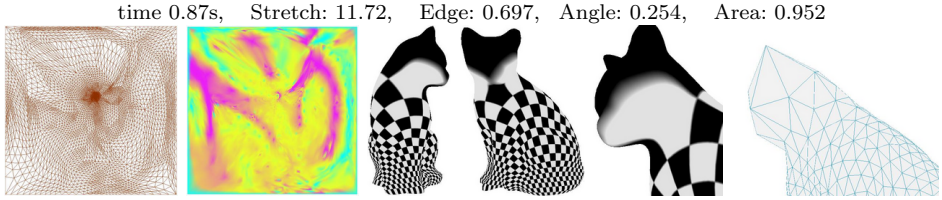
Fig. 6. Comparison of various mesh parameterization schemes on the Mannequin Head model ($V = 2732$, $F = 5420$).



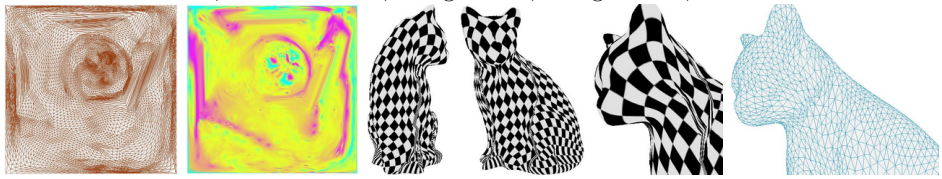
(a) Harmonic map of Eck et al. [7]:
time 1.23s, Stretch: 13.3, Edge: 0.756, Angle: 0.104, Area: 1.02



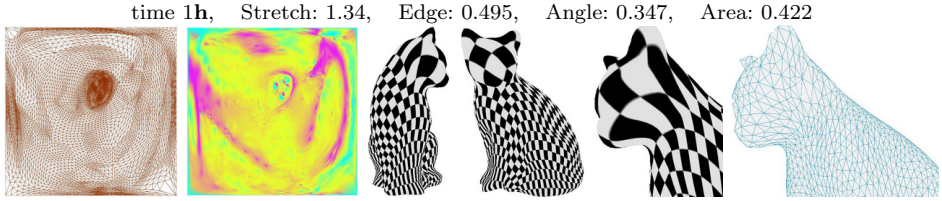
(b) Floater shape preserving weights [8]:
time 0.87s, Stretch: 11.72, Edge: 0.697, Angle: 0.254, Area: 0.952



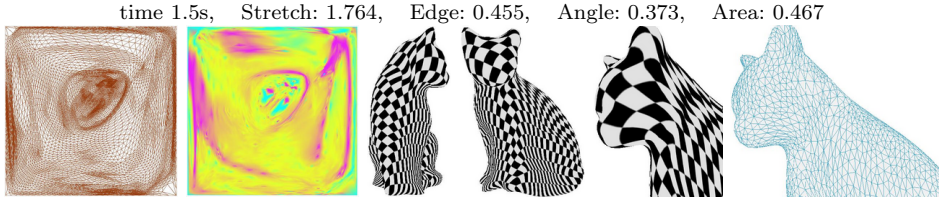
(c) Intrinsic parameterization of Desbrun et al. [6]:
time 1.81s, Stretch: 12.26, Edge: 0.723, Angle: 0.176, Area: 0.979



(d) Stretch minimization of Sander et al. [21]:
time 1**h**, Stretch: 1.34, Edge: 0.495, Angle: 0.347, Area: 0.422



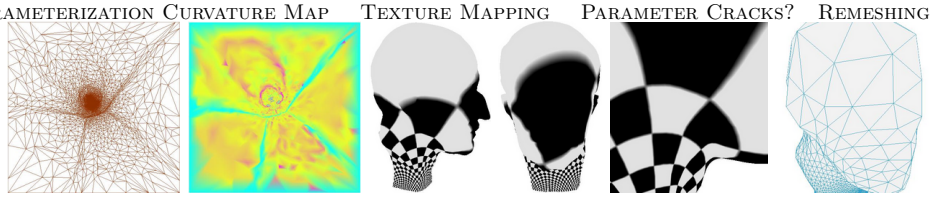
(e) Our $\mathcal{U}^1$ parameterization:
time 1.5s, Stretch: 1.764, Edge: 0.455, Angle: 0.373, Area: 0.467



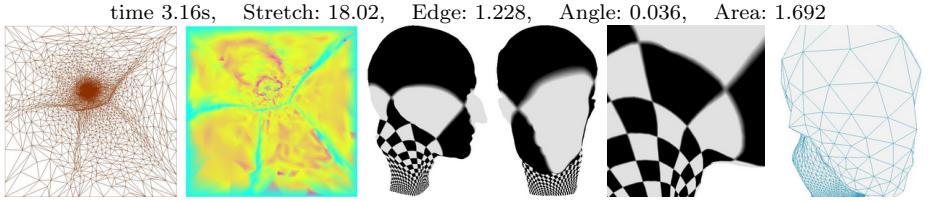
(f) Our $\mathcal{U}^{\text{opt}} = \mathcal{U}^3$ parameterization:
time 3.44s, Stretch: 1.479, Edge: 0.466, Angle: 0.522, Area: 0.361

Fig. 7. Comparison of various mesh parameterization schemes on the Cat model ($V = 5649$, $F = 11168$).

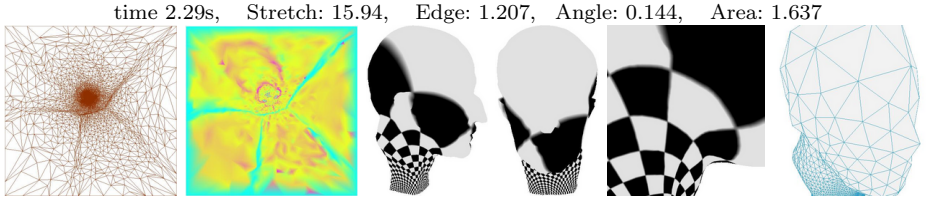
Parameterization Curvature Map Texture Mapping Parameter Cracks? Remeshing [1]



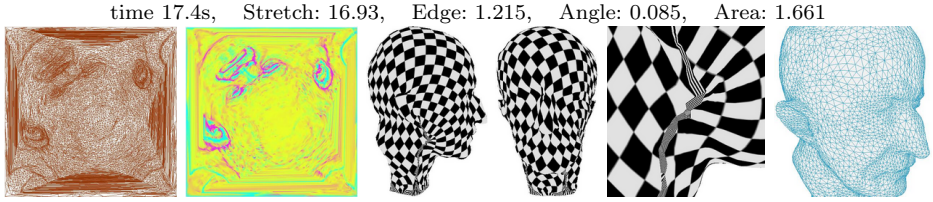
(a) Harmonic map of Eck et al. [7]:
time 3.16s, Stretch: 18.02, Edge: 1.228, Angle: 0.036, Area: 1.692



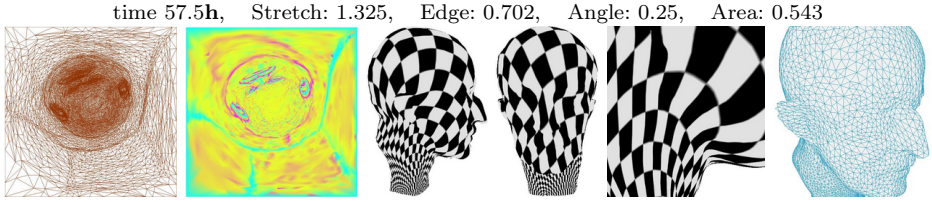
(b) Floater shape preserving weights [8]:
time 2.29s, Stretch: 15.94, Edge: 1.207, Angle: 0.144, Area: 1.637



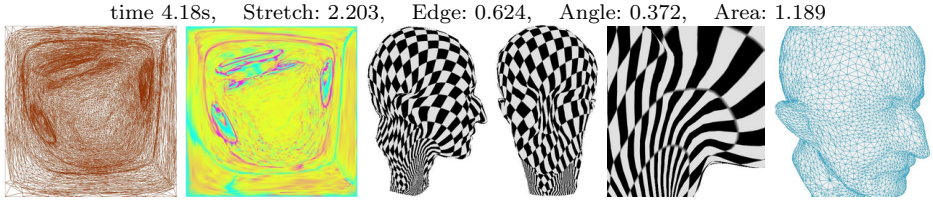
(c) Intrinsic parameterization of Desbrun et al. [6]:
time 17.4s, Stretch: 16.93, Edge: 1.215, Angle: 0.085, Area: 1.661



(d) Stretch minimization of Sander et al. [21]:
time 57.5**h**, Stretch: 1.325, Edge: 0.702, Angle: 0.25, Area: 0.543



(e) Our $\mathcal{U}^1$ parameterization:
time 4.18s, Stretch: 2.203, Edge: 0.624, Angle: 0.372, Area: 1.189



(f) Our $\mathcal{U}^{\text{opt}} = \mathcal{U}^3$ parameterization:
time 9.22s, Stretch: 1.539, Edge: 0.562, Angle: 0.49, Area: 0.621

Fig. 8. Comparison of various mesh parameterization schemes on the decimated Max-Planck bust model ($V = 9462$, $F = 18866$).

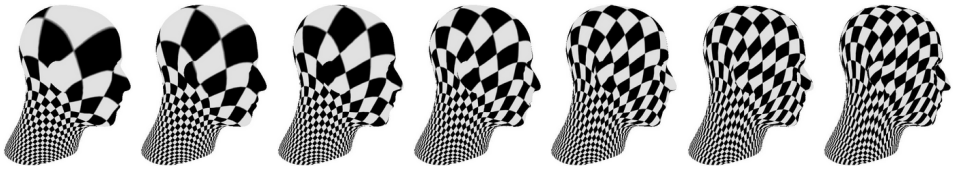


Fig. 9. Choosing smaller values for η leads to a less aggressive stretch minimization. From left to right: $\mathcal{U}^1$ parameterization of Mannequin Head with $\eta = \{0, 0.1, 0.2, 0.4, 0.6, 0.8, 1\}$.

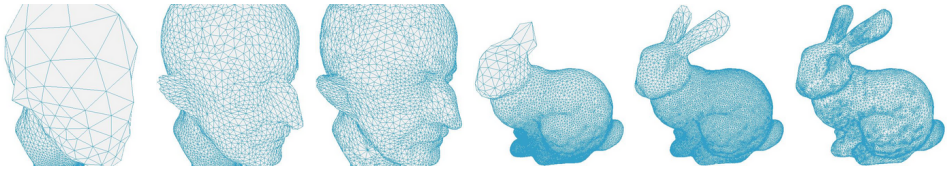


Fig. 10. Remeshing of Max-Planck bust model (three left images) and Stanford bunny (three right images) models. For each model remeshings according to $\mathcal{U}^0$, $\{\mathcal{U}^0, \mathcal{U}^1\}$, and $\{\mathcal{U}^0, \mathcal{U}^{\text{opt}}\}$ are shown. See the text for details.

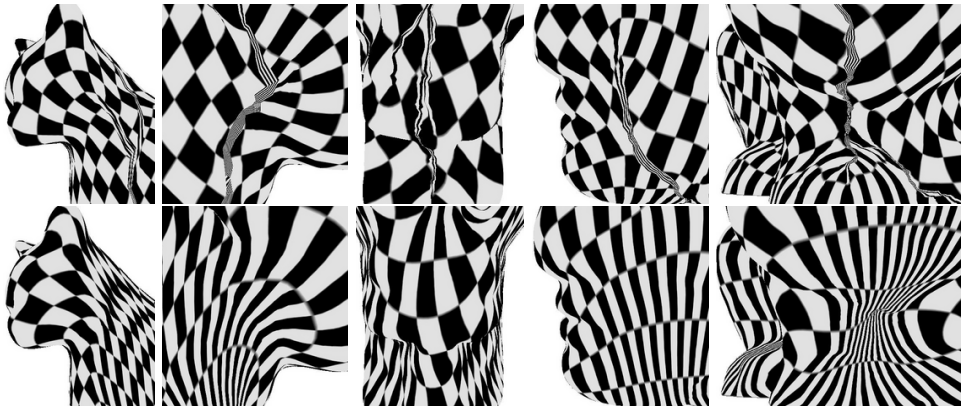
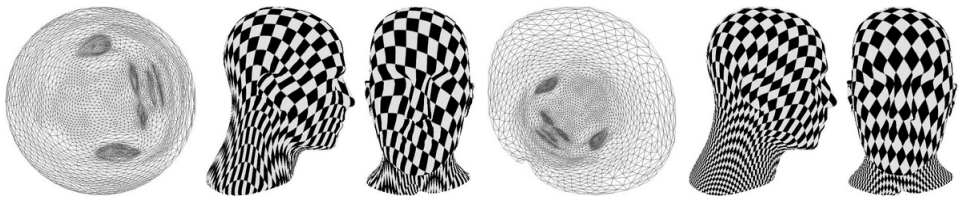


Fig. 11. Parameter cracks on various models textured with checkerboard texture. The images of the upper row demonstrate parameter cracks generated by the stretch-minimization method of Sander et al. The images of the bottom row show the same parts of the models parameterized by our $\mathcal{U}^{\text{opt}}$.



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

$\mathcal{U}^{\text{opt}} = \mathcal{U}^1$ with natural boundary [6]. Time:1.67 s, Stretch: 1.68, Edge: 0.5, Angle: 0.37, Area: 0.9.

Fig. 12. Using various parameter domains for $\mathcal{U}^{\text{opt}}$.

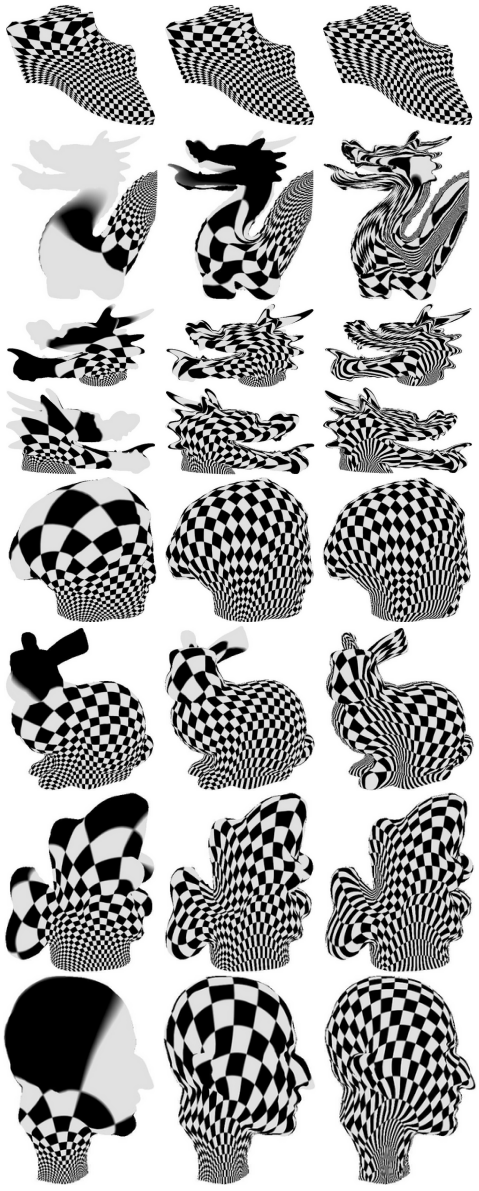


Fig. 13. Checkerboard texture mapping with U^0 (left), U^1 (middle), and U^{opt} (right).

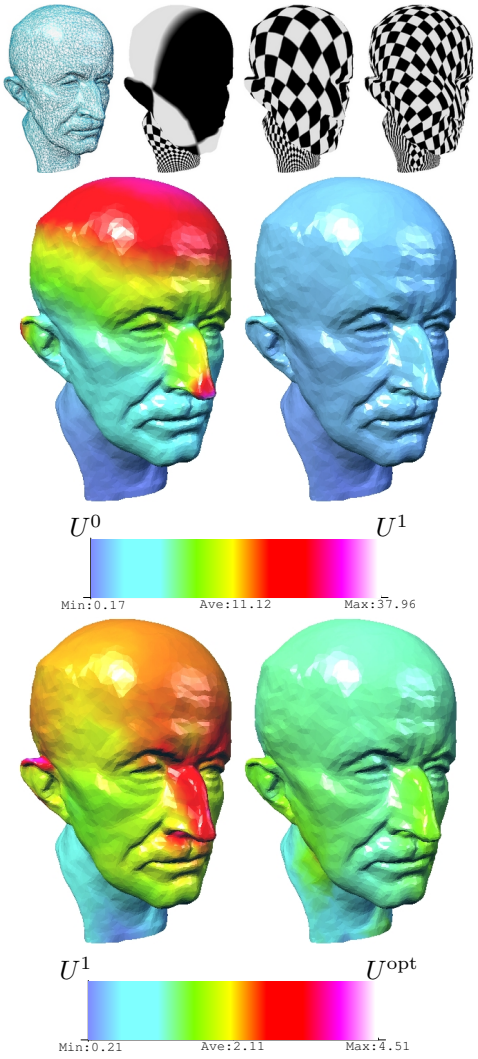


Fig. 14. Top row: a decimated Max-Planck bust model and results of checkerboard texture mapping with $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\mathrm{opt}}$ parameterizations. The four remaining images of the model show the distribution of the vertex stretches over the model for $\mathcal{U}^0$, $\mathcal{U}^1$, and $\mathcal{U}^{\mathrm{opt}}$. 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 $\mathcal{U}^{\mathrm{opt}}$. Here the bounds of the interval are equal to the maximal and minimal stretch values.