EUROGRAPHICS 2022 / R. Chaine and M. H. Kim
(Guest Editors)

Volume 41 (2022), Number 2

Interactive Editing of Discrete Chebyshev Nets

Rui-Zeng Li Jia-Peng Guo Qi Wang Shuangming Chai Ligang Liu Xiao-Ming Fu†

University of Science and Technology of China

Abstract

We propose an interactive method to edit a discrete Chebyshev net, which is a quad mesh with edges of the same length. To ensure that the edited mesh is always a discrete Chebyshev net, the maximum difference of all edge lengths should be zero during the editing process. Hence, we formulate an objective function using ℓ_p-norm ($p > 2$) to force the maximum length deviation to approach zero in practice. To optimize the nonlinear and non-convex objective function interactively and efficiently, we develop a novel second-order solver. The core of the solver is to construct a new convex majorizer for our objective function to achieve fast convergence. We present two acceleration strategies to further reduce the optimization time, including adaptive p change and adaptive variables reduction. A large number of experiments demonstrate the capability and feasibility of our method for interactively editing complex discrete Chebyshev nets.

CCS Concepts

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

1. Introduction

Discrete Chebyshev nets are quad meshes with all edges of equal length, which are inextensible along the edge directions. The equal edge length of Chebyshev nets makes it possible to simulate the characteristics of quadrilateral meshes built from grids of flexible, nearly inextensible rods. Hence, they have been widely used in many fields, such as architecture, textiles, and art (Figure 1). Many former methods focus on the generation of discrete Chebyshev nets for various 3D models [Aon94, Mas17, SFCBCV19, LLZ*20, GSFD*14]. Different from them, we focus on the editing of the existing discrete Chebyshev nets (Figure 2), i.e., the edge lengths of discrete Chebyshev nets are preserved during deformation.

The motivations for deforming Chebyshev meshes are twofold. First, since such nets are very useful in practice (Figure 1), it is desired to edit and design them in a short time, e.g., less than 0.5 seconds. However, the state-of-the-art methods [Aon94, Mas17, SFCBCV19, LLZ*20] for automatically generating a Chebyshev net from a triangular mesh take lots of time (Figure 3). Second, the automatic generation methods may violate the users' design intentions, e.g., the singularities may not be placed at the desired positions (Figure 3). Therefore, it is challenging to deform triangle meshes and then generate new Chebyshev nets to achieve interactive and effective editing of the Chebyshev nets.

Our goal is to edit discrete Chebyshev nets via a handle-based interface interactively. There are three basic requirements for achieving this goal. First, the mesh after editing is still a discrete Chebyshev net. As an approximation, we require the maximum difference in edge length to be small enough, rather than strictly constraining all edges to the same length. Second, the editing process is fast enough

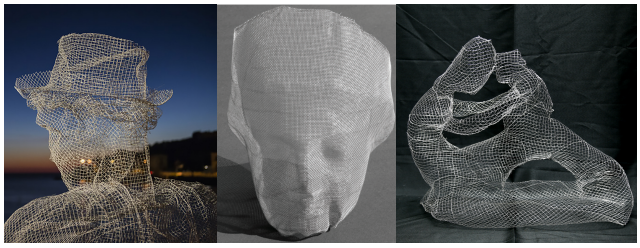

Figure 1: *Left: a physical discrete Chebyshev net created by artist Edoardo Tresoldi. Middle: a wire mesh designed by [GSFD*14]. Right: a discrete Chebyshev net generated by [LLZ*20].*

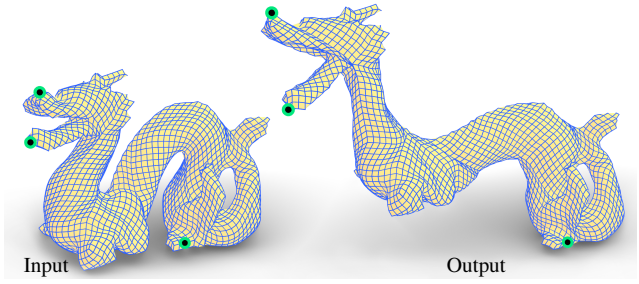

Figure 2: *A discrete Chebyshev net (left) containing 3462 vertices is edited by our system. Given the target positions of the handles, our method takes 0.164 seconds to generate the result (right).*

† The corresponding author

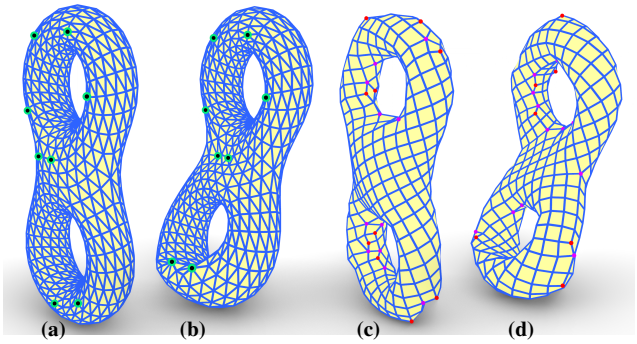

Figure 3: *Generating a Chebyshev net from a deformed triangle mesh. We deform the input triangular mesh (a) by [SA07] to generate a deformed mesh (b). The method in [LLZ*20] is used to generate discrete Chebyshev nets. The net with 37 singularities (colored vertices) in (c) is from the input triangular mesh, and the net with 28 singularities in (d) is from the deformed mesh. It takes about ten minutes to generate each net. The implementation of [LLZ*20] is kindly provided by the authors.*

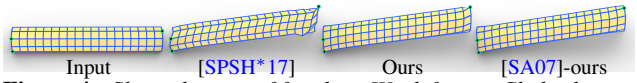

Figure 4: *Shear degrees of freedom. We deform a Chebyshev net using three methods: (1) the method in [SPSH*17] solving our problem (4), (2) our method, and (3) our method initialized by [SA07]. Since we only preserve edge lengths, there is freedom on shearing, and different resulting nets are produced.*

to achieve an interactive rate. Third, handles reach the positions specified by the users after deformation.

Several methods have been proposed to optimize the edge lengths to the targets [DBD*15, LLZ*20, LBOK13]. In general, their objective function is the nonlinear least squares of the difference in edge length. To optimize it, an elegant and efficient local-global solver [BDS*12] is developed. Consequently, their methods work at an interactive rate; however, they cannot constrain the maximum edge length deviation to be very small (Figure 5). Besides, deforming Chebyshev nets have shear degrees of freedom (Figure 4), so this problem cannot be explored using the standard "as-rigid-as-possible" and "nearly isometric" deformation approaches (Figure 17).

To satisfy the first requirement, we can use a hard constraint to control the maximum edge length deviation to be smaller than a tiny bound or a soft constraint to penalize the maximum edge length deviation to approach zero. No matter which form is used to construct the optimization problem, it is a nonlinear and non-convex problem that is difficult to solve. It is even more challenging to solve this optimization problem at the interactive rate. Taking the soft constraint strategy as an example, common numerical optimization solvers can be used, such as first-order approaches, quasi-Newton solvers, and Newton's method [NW06]. Due to the slow convergence rate, it is difficult to achieve our goal using a first-order or quasi-Newton solver. As for Newton's method, although it has a second-order convergence rate, the Hessian matrix in each iteration should be modified to be positive semi-definite to obtain a valid descent direction. In addition, different modifications lead to different convergence speeds. Therefore, it is non-trivial to perform modifications to achieve interactive editing.

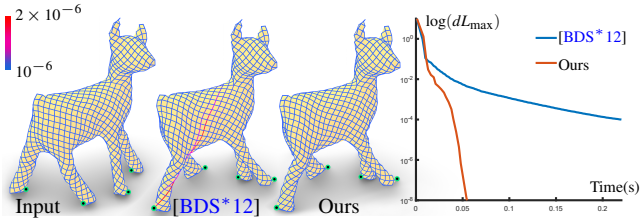

Figure 5: *Comparison with [BDS*12]. The color of each edge* $\mathbf{e}$ *encodes the edge length deviation* $dL_{\mathbf{e}}$ *(defined in (1)), and we use the same colorbar for later figures except for special notes. If* $dL_{\mathbf{e}} < 10^{-6}$ *(or* $dL_{\mathbf{e}} > 2 \times 10^{-6}$*), the edge* $\mathbf{e}$ *is in blue (or red). The rightmost graph shows* $\log(dL_{\max})$ *vs. running time (in seconds), where* $dL_{\max}$ *is the maximum* $dL_{\mathbf{e}}$*.*

In this paper, we propose a novel handle-based tool to edit discrete Chebyshev nets interactively. Our method uses soft constraints to control the maximum edge length deviation to be lower than a tiny threshold. Since the ℓ_p-norm with a large p is a good proxy for the infinite norm, we use the ℓ_p-norm of the edge length deviations as the objective function. The key to achieving an interactive rate is a novel second-order solver. Central to the solver is a new approach constructing a convex majorizer for the objective function. Then, the positive semi-definite Hessian matrix of the convex majorizer is used to compute descent directions. In practice, the solver can make the maximum edge length deviation approach zero quickly. Besides, to make the optimization faster, we develop two acceleration strategies: (1) adaptive p scheme and (2) adaptive variable reduction.

Our interface is straightforward and intuitive to enable users to edit discrete Chebyshev nets creatively. We support real-time editing (about 25 FPS) of nets with around 4000 vertices. If 100 milliseconds is allowed between interactions, nets with around 8000 vertices can be edited. We demonstrate the feasibility and practicality of our system through a large number of experiments (see the supplementary video).

2. Related Work

Discrete Chebyshev nets A Chebyshev net is a coordinate chart defined on $\mathbb{R}^2$ satisfying that the partial derivatives of two coordinate components are equal to one [SD95]. Since the global requirement is difficult to reach, continuous Chebyshev nets consist of several surface patches, each of which satisfies the requirement locally. Discrete Chebyshev nets are the discretization of continuous Chebyshev nets. They are defined as quad meshes with identical edge lengths [BP*96], and each quad is an approximation to a local continuous Chebyshev net. Discrete Chebyshev nets have been widely used in many fields, including woven cloth formation [Aon94, ADBW96], gridshell structures computation [BBC10, HSRG12, BBC14, BSFJR18], wire meshes construction [GSFD*14], etc.

Generating discrete Chebyshev nets In recent years, the generation and fabrication of discrete Chebyshev nets have attracted significant attention. Some methods aim at approximating a local part of the given surface by a discrete Chebyshev net while not

adding singularities [BBC14, GSFD*14]. On the other hand, global discrete Chebyshev nets are needed for some applications. Starting from a single patch [Aon94, ADBW96, ABW01], the generation process gradually introduces seams of tangential discontinuity. A similar method is to gradually add new singularities from an initialization [Mas17]. Some approaches also employ quad meshing algorithms, which first compute vector fields first and then generate discrete Chebyshev nets [SFCBCV19, LLZ*20]. Fabrication is further considered by [LLZ*20] to facilitate home users.

Length-preserving constraint Some applications aim to control the edge length deviation. To achieve this goal, [LBOK13] defines the spring potential by Hooke's law to simulate a mass-spring system, which restricts the edges physically. Several local-global algorithms are proposed to gradually approach the target edge length [BDS*12, DBD*15, LLZ*20]. There are also many methods focusing on isometric constraints, which can also preserve edge lengths. They construct energy functions to describe the isometric constraints, for example, local rigidity energy [SA07] and Ginzburg-Landau energy [SFCBCV19]. However, these methods have no control over the maximum deviation of the edge length from the target length. Our work constrains the maximum edge length deviation under a tiny threshold during editing by an energy function defined in the form of ℓ_p-norm.

Geometric optimization A variety of geometry processing tasks in computer graphics, such as mesh deformation and parameterization, are finally converted into nonlinear optimization problems [FSZ*21]. The design of suitable solvers for these complex formulations has become a hotspot in recent years. As for the first-order algorithm, the time spent on each iteration is very small, but many iterations are required for convergence, such as the local-global solver [BDS*12, LZX*08, SA07] and the block coordinate descent method [FLG15]. Some methods are developed to improve the convergence of local-global solvers [RPPSH17, PDZ*18, SFL19]. Newton's method is one of the most popular second-order methods to solve optimization problems, which uses the Hessian of the objective function to update variables iteratively [NW06]. But this method only fits the situation where the objective function is convex. When the function has a non-convex form, it is necessary to modify the Hessian to enforce its positive definiteness [SPSH*17, SGK19, GSC18]. To achieve faster convergence than second-order methods, a progressive reference is proposed [LYNF18]. Quasi-Newton algorithms [NW06] develop Newton's method, which have a superlinear convergence rate and are reliable in nonlinear optimization [LBK17, ZBK18].

The Majorization-Minimization (MM) algorithm [Lan13a] is an iterative optimization method that exploits the convexity of a function to find its minima. At each iteration, it first constructs a convex upper bound function of the objective function, and then minimizes the convex proxy function. [SPSH*17] adopts this framework to deal with the minimization of composite objectives, which is further utilized in [SYLF20] to compute bijective parameterizations. We also follow the pipeline of composite majorization and construct a novel convex majorizer for the length-preserving energy function, which is used to perform the deformation of discrete Chebyshev nets interactively. Our solver outperforms prior work in the test of a large number of examples.

3. Method

3.1. Problem and formulation

Inputs The input quad mesh is a 3D discrete Chebyshev net $\mathcal{N}$. Each edge of $\mathcal{N}$ has the same length, denoted as L. Our goal is to edit $\mathcal{N}$ to obtain a new discrete Chebyshev net $\mathcal{M}$. The sets of vertices, edges, and quad facets of $\mathcal{M}$ are denoted as $\mathcal{V}$, $\mathcal{E}$, and $\mathcal{F}$, respectively. For each edge $\mathbf{e} = \overline{\mathbf{v}_a\mathbf{v}_b}$ in $\mathcal{E}$, its length $L_\mathbf{e}$ is computed as $\|\mathbf{v}_a - \mathbf{v}_b\|_2$.

Requirements We use a handle-based interface to edit $\mathcal{N}$. The handles and their target positions are represented as $\{\mathbf{v}_h, h = 1, ..., N_h\}$ and $\{\mathbf{v}_h^\star, h = 1, ..., N_h\}$, respectively. The following requirements should be satisfied after the editing:

- $\mathcal{M}$ is still a discrete Chebyshev net.
- Each handle reaches its target position, i.e., $\mathbf{v}_h = \mathbf{v}_h^\star$, $\forall h$.
- The editing is fast enough, e.g., an interactive rate is achieved for medium-sized nets.

Approximation To satisfy the first requirement, all edges in $\mathcal{M}$ should have the same length, i.e., $L_{\mathbf{e}_i} = L_{\mathbf{e}_j}$ for all edge pairs $(\mathbf{e}_i, \mathbf{e}_j)$. As this strict requirement is too hard to meet, we use a soft constraint energy function to penalize the maximum edge length deviation $dL_{\max}$ to approach zero as an approximation in practice. To define $dL_{\max}$, we first define the edge length deviation $dL_\mathbf{e}$ for each edge $\mathbf{e}$:

$$dL_\mathbf{e} = L_\mathbf{e}/L_{\text{ref}} + L_{\text{ref}}/L_\mathbf{e} - 2, \tag{1}$$

where L_{ref} is a reference length. Then, $dL_{\max} = \|\{dL_\mathbf{e} \mid \mathbf{e} \in \mathcal{E}\}\|_\infty$. In general, to keep a high similarity between $\mathcal{M}$ and $\mathcal{N}$, we take $L_{\text{ref}} = L$.

Soft constraint energy It is difficult to directly optimize $dL_{\max}$ to approach zero due to the non C^1 continuity of the infinite norm. Since the ℓ_p-norm ($p \geq 2$) with a large p is a good proxy to approximate the infinite norm, we adopt the ℓ_p-norm to define the optimization objective E_p:

$$E_p = \|\{dL_\mathbf{e} \mid \mathbf{e} \in \mathcal{E}\}\|_p^p = \sum_{\mathbf{e}\in\mathcal{E}} (dL_\mathbf{e})^p. \tag{2}$$

However, as shown in Figure 7, we observe that directly optimizing E_p needs many iterations to obtain a small $dL_{\max}$ in our experiments. Therefore, we construct another new objective function to achieve faster convergence, which is based on the following proposition.

PROPOSITION 1 $F_{p,\mathbf{e}}^{\text{new}} = (L_\mathbf{e}/L)^p + (L/L_\mathbf{e})^p - 2 \geq (L_\mathbf{e}/L + L/L_\mathbf{e} - 2)^p$ when $L_\mathbf{e}/L > 0$, and the equality holds when $L_\mathbf{e}/L = 1$.

We prove Proposition 1 in the Appendix and show the difference in Figure 6 - Left. As shown in Figure 6-Left, $F_{p,\mathbf{e}}^{\text{new}}$ decreases much faster than $(dL_\mathbf{e})^p$ near $x = 1$. According to Proposition 1, we construct the new objective function as follows:

$$E_p^{\text{new}} = \sum_{\mathbf{e}\in\mathcal{E}} ((L_\mathbf{e}/L)^p + (L/L_\mathbf{e})^p). \tag{3}$$

In Figure 6 - right, we plot $F_{p,\mathbf{e}}^{\text{new}}$ with different ps. When p is large, our objective function can also control $\max_{\mathbf{e}\in\mathcal{E}} |L_\mathbf{e}/L - 1|$. The reason is that $L_\mathbf{e}/L + L/L_\mathbf{e} - 2$ approximates $(L_\mathbf{e}/L - 1)^2$ when $L_\mathbf{e}/L$ is close to 1.

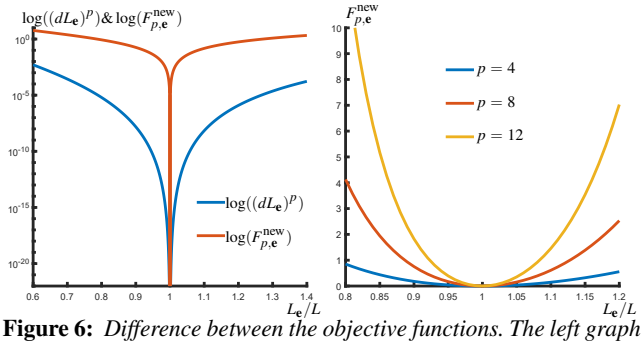

Figure 6: *Difference between the objective functions. The left graph shows difference with $(dL_{\mathbf{e}})^p$ and $F^{new}_{p,\mathbf{e}}$ when choosing $p=4$. The right graph plots $F^{new}_{p,\mathbf{e}}$ vs. $L_{\mathbf{e}}/L$ with different p values.*

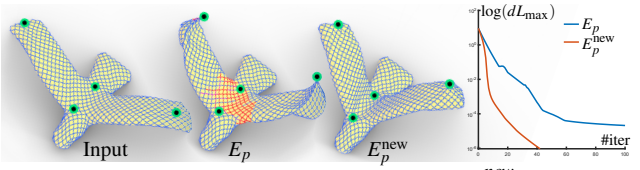

Figure 7: *Difference between optimizing E_p and E_p^{new}, when $p=4$. The rightmost graph shows $\log(dL_{\max})$ vs. the number of iterations. We show results selected at the maximum number of iterations, and this rule is applied for later figures if the horizontal axis of the diagram is the iteration count except for special notes.*

Formulation The problem of editing discrete Chebyshev nets can be formulated as a constrained optimization problem:

$$\begin{aligned} \min_{\mathcal{V}} \quad & E_p^{\text{new}}, \\ \text{s.t.} \quad & \mathbf{v}_h = \mathbf{v}_h^\star, \ \forall h. \end{aligned} \tag{4}$$

3.2. Second-order solvers

Our method relies on a second-order solver for solving (4).

Solver overview Given an initial $\mathcal{M}$, the second-order solver runs as follows:

1. Compute the Hessian matrix H of E_p^{new}, which is further modified to be a positive semi-definite matrix H^+.
2. Since the positional constraints of the handles are linear, the KKT system is used to compute the descent direction $\mathbf{d}$:
$$\begin{bmatrix} H^+ & A^T \\ A & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{d} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla E_p^{\text{new}} \\ \mathbf{0} \end{bmatrix}, \tag{5}$$
where the matrix A selects the handles for restricting them to the specified positions.
3. Perform a standard Armijo backtracking algorithm to determine the step size and update positions of vertices.
4. We terminate the solver if the relative change of E_p^{new} is less than ε_1, or $\|\nabla E_p^{\text{new}}\|_2 < \varepsilon_2$, or $dL_{\max} < \varepsilon_3$, where ε_1, ε_2, and ε_3 are three small positive thresholds. Otherwise, go to Step 1.

In our experiments, we set $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 10^{-6}$ empirically. Note that all the steps are standard except the construction of H^+.

Constructing H^+ Depending on the method of constructing H^+, the convergence rate may vary significantly. Hence, it is the key challenge to obtain fast convergence. As our objective function E_p^{new} is the sum of the energy on each edge, we construct $H^+ = \sum_{e\in\mathcal{E}} H_{\mathbf{e}}^+$, where the positive semi-definite $H_{\mathbf{e}}^+$ is the modification of the Hessian matrix of $f_{\mathbf{e}} = (L_{\mathbf{e}}/L)^p + (L/L_{\mathbf{e}})^p$. This construction method is similar to the previous geometric optimization methods [SPSH*17, GSC18, SGK19].

3.2.1. Convex majorizers

At each iteration, we use the Hessian matrix of a majorizer, which is a convex surrogate upper bound of the objective function, to compute $H_{\mathbf{e}}^+$ [Lan13a]. Next, we propose a novel convex majorizer for our objective function.

Majorizer and minorizer If a function r can be written as the sum of a convex function and a concave function, it has a convex-concave decomposition:

$$r = r^+ + r^-,$$

where r^+ is convex and r^- is concave. We define the majorizer of r [Lan13b] at $\mathbf{x}_0$ as:

$$\bar{r}(\mathbf{x};\mathbf{x}_0) = r^+(\mathbf{x}) + r^-(\mathbf{x}_0) + \nabla r^-(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0), \tag{6}$$

and the minorizer at $\mathbf{x}_0$ as:

$$\underline{r}(\mathbf{x};\mathbf{x}_0) = r^-(\mathbf{x}) + r^+(\mathbf{x}_0) + \nabla r^+(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0). \tag{7}$$

The majorizer and minorizer satisfy the following properties [Lan13b]:

1. $\bar{r}$ is convex and $\underline{r}$ is concave.
2. $r(\mathbf{x}_0) = \bar{r}(\mathbf{x}_0;\mathbf{x}_0) = \underline{r}(\mathbf{x}_0;\mathbf{x}_0)$ and $\nabla r(\mathbf{x}_0) = \nabla\bar{r}(\mathbf{x}_0;\mathbf{x}_0) = \nabla\underline{r}(\mathbf{x}_0;\mathbf{x}_0)$.
3. $\underline{r}(\mathbf{x};\mathbf{x}_0) \le r(\mathbf{x}) \le \bar{r}(\mathbf{x};\mathbf{x}_0), \forall \mathbf{x}$.

Composite functions Similar to [SPSH*17], we consider a composite function f having the following form:

$$f(\mathbf{x}) = h(\mathbf{g}(\mathbf{x})) = h(g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_k(\mathbf{x})), \tag{8}$$

where $h : \mathbb{R}^k \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R}, i = 1,2,\ldots,k$ are at least C^2 functions. Suppose h and each g_i have convex-concave decompositions:

$$h = h^+ + h^-, \quad g_i = g_i^+ + g_i^-, \quad i = 1,2,\ldots,k,$$

where h^+, g_i^+ are convex, and h^-, g_i^- are concave.

Majorizer construction for f We first construct a majorizer $\bar{h}$ of h via (6). Then, suppose $\bar{h}$ can be decomposed as a sum of an increasing function and a decreasing function which are convex:

$$\bar{h} = h_{\text{inc}} + h_{\text{dec}},$$

where h_{inc} is convex and increasing with respect to each variable when all the others are fixed, h_{dec} is convex and decreasing with respect to each variable when all the others are fixed. Different from [SPSH*17], the majorizer for f is constructed according to the following proposition.

PROPOSITION 2 The following function $\hat{f}$ is a convex majorizer of f at $\mathbf{x}_0$:

$$\hat{f}(\mathbf{x};\mathbf{x}_0) = h_{\text{inc}}(\bar{\mathbf{g}}(\mathbf{x};\mathbf{x}_0)) + h_{\text{dec}}(\underline{\mathbf{g}}(\mathbf{x};\mathbf{x}_0)), \tag{9}$$

where

$$\bar{\mathbf{g}}(\mathbf{x};\mathbf{x}_0) = (\bar{g}_1(\mathbf{x};\mathbf{x}_0), \bar{g}_2(\mathbf{x};\mathbf{x}_0), \ldots, \bar{g}_k(\mathbf{x};\mathbf{x}_0)),$$
$$\underline{\mathbf{g}}(\mathbf{x};\mathbf{x}_0) = (\underline{g}_1(\mathbf{x};\mathbf{x}_0), \underline{g}_2(\mathbf{x};\mathbf{x}_0), \ldots, \underline{g}_k(\mathbf{x};\mathbf{x}_0)).$$

The proof is provided in the supplementary materials.

Our majorizer construction To adapt the aforementioned construction method for our objective function $f_{\mathbf{e}} = (L_{\mathbf{e}}/L)^p + (L/L_{\mathbf{e}})^p$, we define h and $\mathbf{g}$ in the form of (8). Since the vector $\mathbf{g}$ contains only one element, it is denoted as g:

$$g = (L_{\mathbf{e}}/L)^p.$$

It is easy to prove that g is convex. We then define $h : \mathbb{R}^+ \to \mathbb{R}^+$ as:

$$h = x + \frac{1}{x}.$$

As h is convex, $\bar{h} = h$, $h_{\text{inc}} = x$, and $h_{\text{dec}} = \frac{1}{x}$. Hence, we can use (9) to construct the convex majorizer $\hat{f}_{\mathbf{e}}$ for $f_{\mathbf{e}}$. Then, $H^+_{\mathbf{e}}$ is computed as the Hessian matrix of $\hat{f}_{\mathbf{e}}$ (see mote details in the supplementary materials).

3.2.2. Difference from [SPSH*17]

A revisit of [SPSH*17] A prior method to construct the convex majorizer of the composite function (8) is proposed in [SPSH*17]. The constructed majorizer $\tilde{f}$ of f at $\mathbf{x}_0$ is as follows:

$$\tilde{f}(\mathbf{x};\mathbf{x}_0) = \bar{h}([\mathbf{g}](\mathbf{x};\mathbf{x}_0);\mathbf{u}_0), \tag{10}$$

where $\mathbf{u}_0 = \mathbf{g}(\mathbf{x}_0)$, and each entry of $[\mathbf{g}] : \mathbb{R}^n \to \mathbb{R}^k$ is:

$$[g_i](\mathbf{x};\mathbf{x}_0) = \begin{cases} \bar{g}_i(\mathbf{x};\mathbf{x}_0), & \text{if } \dfrac{\partial \bar{h}}{\partial u_i}(\mathbf{u}_0;\mathbf{u}_0) > 0, \\ \underline{g}_i(\mathbf{x};\mathbf{x}_0), & \text{if } \dfrac{\partial \bar{h}}{\partial u_i}(\mathbf{u}_0;\mathbf{u}_0) < 0. \end{cases} \tag{11}$$

Majorizer differences We think our majorizer is different from (10)'s, and our solver provides another way to construct a majorizer. We have the following proposition proved in the supplementary materials for the differences between the two majorizers.

PROPOSITION 3 $\tilde{f}(\mathbf{x};\mathbf{x}_0) \le \hat{f}(\mathbf{x};\mathbf{x}_0)$; $\nabla^2\tilde{f}(\mathbf{x}_0;\mathbf{x}_0) \le \nabla^2\hat{f}(\mathbf{x}_0;\mathbf{x}_0)$.

Although this proposition shows our majorizer is greater than that of [SPSH*17], we think a "tighter" majorizer cannot guarantee better global performance in theory. The reason is that "tighter" is more local and tends to be more effective when near the global optimum, but it may lead to insufficient decrease when far away from the global optimum. In our experiments, when $dL_{\max} > 10^{-6}$, optimizing the tighter majorizer of [SPSH*17] does not perform better than ours (Figure 8).

Differences for our objective functions The constructed majorizer (10) only considers the influence of $\nabla^2\bar{g}_i$ or $\nabla^2\underline{g}_i$ at the two sides of the point, where $\dfrac{\partial \bar{h}}{\partial u_i}(\mathbf{u}_0;\mathbf{u}_0) = 0$. Then for our function, since g is convex, the majorizer (10) actually does not use the Hessian of g within the range where h is decreasing (i.e., $0 < g < 1$). We propose our decomposition to take the Hessian of g into consideration in this case. Our constructed majorizer uses the decomposition of h_{inc} and h_{dec} regardless of the value of g or $\nabla\bar{h}$.

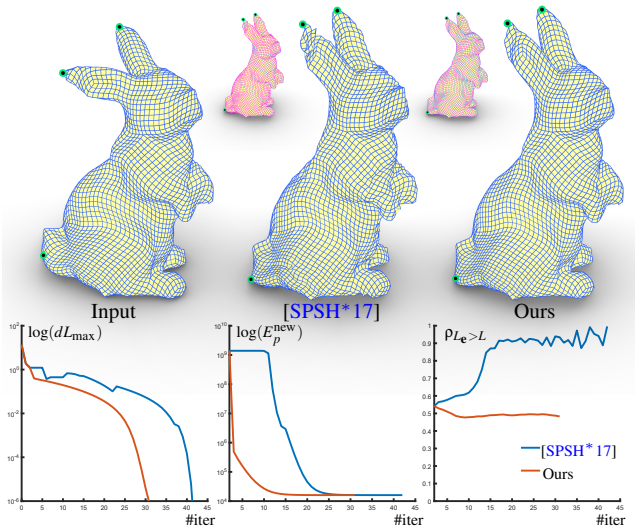

Figure 8: *Comparison with [SPSH*17] on a Rabbit model containing 4011 vertices. We define a measure $\rho_{L_{\mathbf{e}}>L} = N_{L_{\mathbf{e}}>L}/N$, where $N_{L_{\mathbf{e}}>L}$ is the number of edges that satisfy $L_{\mathbf{e}} > L$, and N is the number of edges in $\mathcal{E}$. We color those edges of the resulting meshes to be pink in the zoom-out figures. The three graphs show $\log(dL_{\max})$, $\log(E_p^{new})$, and $\rho_{L_e>L}$ as functions of the number of iterations.*

For some special functions, e.g., (12) and (13), our method can have the same majorizer as that of [SPSH*17] by choosing some special decompositions. For example, considering $h = x + 1/x$, if we choose an increasing function $h_1(x) = x + 1/x - 1$ when $x >= 1$, $h_1(x) = 1$ when $x < 1$, and a decreasing function $h_2(x) = x + 1/x - 1$ when $x < 1$, $h_2(x) = 1$ when $x >= 1$, the decomposition $h = h_1 + h_2$ leads to the same majorizer as that of [SPSH*17]. But by choosing different decompositions, our method provides a different way to construct majorizers.

Experimental observations We compare our method with [SPSH*17] without any initial smoothing, and the results are shown in Figure 8. Because of numerical issues caused by large p, we choose $p = 20$ here. As shown in Figure 8 - Bottom right, the solver of [SPSH*17] tries to make $L_e > L$ for all edges. When the length of most edges is greater than L, $\log(dL_{\max})$ and $\log(E_p^{\text{new}})$ drop suddenly (Figure 8 - Bottom left and middle). We conjecture the reason is when $g < 1$ (i.e. $L_e < L$), it uses the Hessian of $\underline{g}$ which is zero. Consequently, their solver uses many iterations to remove the influences of the lost Hessian. Besides, the process of trying to make $L_e > L$ for all edges leads to noisy results (Figure 8 - Top middle). Despite the noise, the solver of [SPSH*17] can also optimize the energy to a small level, and the result is also a discrete Chebyshev net. The quite different results are caused by the freedom of deforming quad mesh with only edge lengths constraints. Our method is faster and does not generate noise in

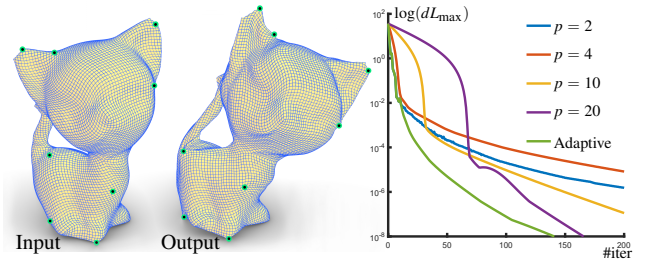

Figure 9: *Different ps. Left: the input discrete Chebyshev net. Middle: since different p values lead to similar edited results, we only show the result using the adaptive p strategy. Right: the graph plots* $\log(dL_{\max})$ *vs. number of iterations.*

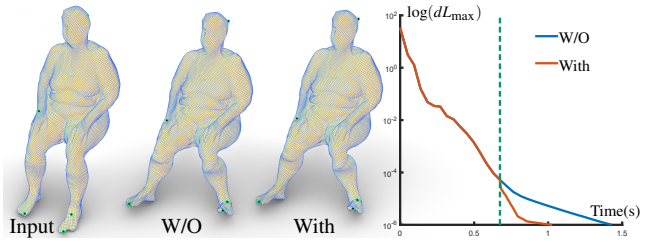

Figure 10: *Adaptive variable reduction. In the rightmost graph, the red curve after the dotted line indicates the optimization using the adaptive variable reduction strategy. This model has 17699 vertices and 35394 edges. In the adaptive variable reduction strategy, 12452 vertices (which are not in* $\mathcal{V}_{small}$*) and 26938 edges are fixed.*

practice. For comparisons with [SPSH*17], more examples are provided in the supplementary material.

3.3. Acceleration strategies

We propose two speed-up strategies to accelerate the editing of discrete Chebyshev nets.

Adaptive p strategy We observe that the value of p significantly affects the convergence rate. As shown in Figure 9, a small p leads to a rapid decrease of the maximum edge length deviation at the beginning, yet cannot penalize the maximum edge length deviation to be very small. A large p produces an opposite behavior compared with a small p. According to these observations, we propose an adaptive p strategy, which first uses a small p at the beginning and adopts a large p at the end. The switch condition is that $dL_{\max} < 10^{-2}$ is satisfied for the first time.

Adaptive variable reduction Our goal is to minimize the maximum edge length deviation. We observe that the edge length deviation of most edges will be tiny after several iterations. It indicates that these edges do not need to be optimized in the subsequent iterations, thus reducing the number of variables. Based on this observation, we propose the following strategy:

1. Optimize all vertices until $dL_{\max} < 10^{-4}$.
2. Collect the vertices of the edges that satisfy $dL_{\mathbf{e}} > \varepsilon_4$ into a set $\widehat{\mathcal{V}}$, where ε_4 is a positive threshold that is not greater than ε_3. Initialize a new set $\mathcal{V}_{\text{small}} = \widehat{\mathcal{V}}$. If a vertex does not belong to $\widehat{\mathcal{V}}$ and is in the n-rings of one vertex in $\widehat{\mathcal{V}}$, we push it into $\mathcal{V}_{\text{small}}$.
3. Optimize vertices in $\mathcal{V}_{\text{small}}$ while fixing other vertices until $dL_{\max} < \varepsilon_3$.

In the experiments, we set n-rings as 2-rings and $\varepsilon_4 = \varepsilon_3/4$. More discussions for ε_4 are provided in Section 4. Using this strategy reduces the optimization time and generates almost the same result compared to not using this strategy (Figure 10).

Our optimization workflow With these two acceleration strategies, our optimization procedure contains three stages: (1) optimizing all vertices using a small p until $dL_{\max} < \varepsilon_{dL1} = 10^{-2}$ is met for the first time, (2) updating all vertices using a large p until $dL_{\max} < \varepsilon_{dL2} = 10^{-4}$ is satisfied, and (3) optimizing the selected vertices with a large p until $dL_{\max} < \varepsilon_3 = 10^{-6}$. An example of our optimization pipeline is shown in Figure 11.

4. Experiments

We have tested our algorithm on various discrete Chebyshev nets to evaluate its performance. Our method is implemented in C++, and all the experiments are performed on a desktop PC with a 3.80 GHz Intel Core i7-10700 and 16 GB of RAM. The linear systems are solved using the Intel® Math Kernel Library. The color bar in Figure 5 is used by all figures. The used discrete Chebyshev nets are kindly provided by the authors of [LLZ*20].

Termination thresholds With the two acceleration strategies, we terminate the optimization process until $dL_{\max} < \varepsilon_3$. Three values for ε_3 are tested in Figure 12. From the experimental results, we are able to achieve a very small $dL_{\max}$. Besides, there is almost no difference in the results obtained using different ε_3s. In our interactive editing and experiments, we set $\varepsilon_3 = 10^{-6}$ by default.

Adaptive p strategy In the adaptive p strategy, the values of ps affect the convergence rate. In general, the small p is set to 2. Different large p values are tested in Figure 13. We observe that the larger p leads to faster convergence. However, a very large p, e.g., $p = 75$, usually causes numerical issues in practice. Hence, the large p is set to 50 by default.

Adaptive variable reduction The threshold ε_4 determines the number of fixed vertices, thereby affecting the optimization time. In Figure 14, three different ε_4s, including $\varepsilon_4 = \varepsilon_3$, $\varepsilon_4 = \varepsilon_3/4$, and $\varepsilon_4 = \varepsilon_3/25$, are tested. Almost the same results are produced. A smaller ε_4 generally indicates fewer fixed vertices, causing more optimization time. Although a larger ε_4 leads to fewer optimization variables, the degrees of freedom to achieve the termination condition $dL_{\max} < \varepsilon_3 = 10^{-6}$ become fewer, resulting in more time to terminate. Hence, we practically set $\varepsilon_4 = \varepsilon_3/4$ to achieve a tradeoff.

More parameter testing We test three parameters (i.e., ε_{dL1}, ε_{dL2}, and ε_4) in our workflow with more options. The results are in the supplementary material. Since the results are similar for most parameters, we choose the parameters to be $\varepsilon_{dL1} = 10^{-2}$, $\varepsilon_{dL2} = 10^{-4}$, and $\varepsilon_4 = 2.5 \times 10^{-7}$ as above.

Ablation study Our method contains two acceleration strategies. Here we perform an ablation study to show how each strategy improves the convergence rate. Four methods are studied: (1) without

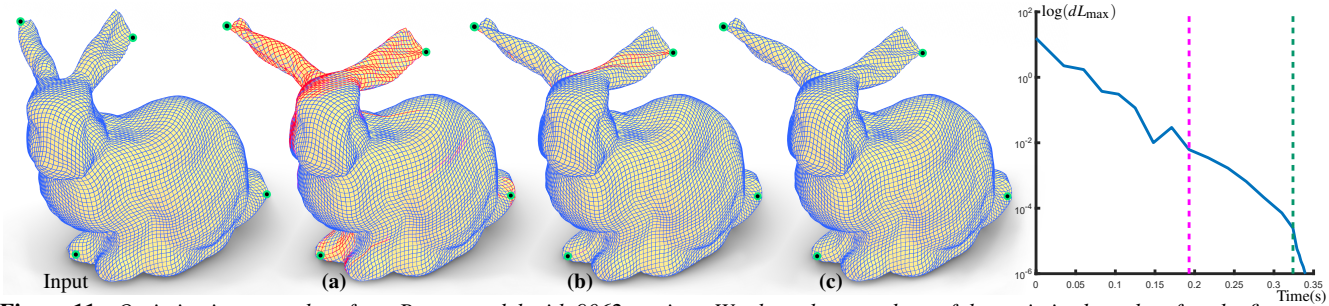

Figure 11: *Optimization procedure for a Bunny model with 9062 vertices. We show the snapshots of the optimized results after the first stage (a), after the second stage (b), after the third stage (final result) (c). In the rightmost graph the left pink dotted line means that our method changes from the first stage to the second stage. The right green dotted line indicates the change from the second stage to the third stage.*

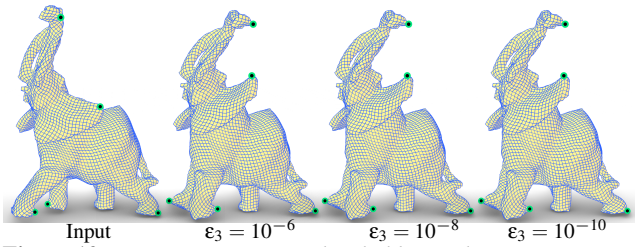

Figure 12: *Various termination thresholds ε_3. The running timings to terminate are 0.250s for $\varepsilon_3 = 10^{-6}$, 0.343s for $\varepsilon_3 = 10^{-8}$, and 0.436s for $\varepsilon_3 = 10^{-10}$.*

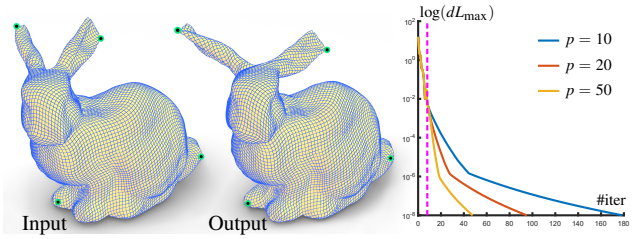

Figure 13: *Testing different large ps for the first acceleration strategy. Since the final results using these three ps are similar, we show the edited net with $p = 50$ as output.*

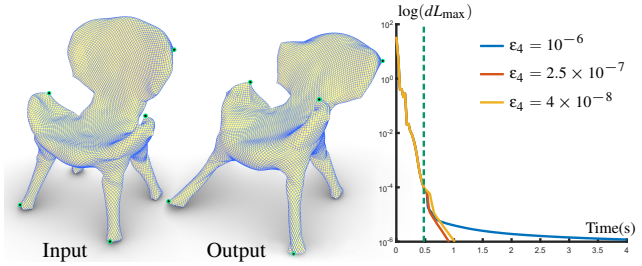

Figure 14: *Testing length deviation thresholds ε_4 for the second acceleration strategy. The output is generated using $\varepsilon_4 = 10^{-6}$, which is similar to the other two results using $\varepsilon_4 = 2.5 \times 10^{-7}$ and $\varepsilon_4 = 4 \times 10^{-8}$.*

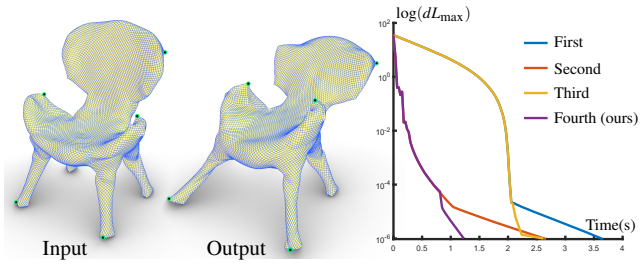

Figure 15: *An ablation study on the two acceleration strategies. All the results are similar, and we show the edited net generated by the fourth method in the middle.*

two acceleration strategies; (2) with adaptive p strategy and without adaptive variable reduction; (3) without adaptive p strategy and with adaptive variable reduction; and (4) with two acceleration strategies (the choice of our method).

Figure 15 shows an example. Here we choose adaptive large $p = 20$. From the comparisons between the first and second methods as well as the third and fourth methods, the adaptive p strategy saves the optimization time. The acceleration effects of the adaptive variable reduction are shown by the comparisons between the first and third methods as well as the second and fourth methods. Therefore, both strategies contribute to optimization acceleration.

Mesh resolutions Shapes with three different resolutions are tested (Figure 16). We specify similar target positions of the handles on meshes with different resolutions. The time cost increases when the mesh resolution increases, and our method can still preserve edge lengths nicely.

Comparisons We select four methods as competitors. As rigid transformations are isometric, the first one is an as-rigid-as-possible editing technique [SA07]. The ARAP energy of [SA07] can be directly used for quad meshes to preserve the edge lengths, and we use the uniform weights. Since [LBOK13] and [BDS*12] optimize the edge lengths to their targets using the ℓ_2-norm, it is selected as the second competitor. We use the solver in [BDS*12] for optimization. The third competitor optimizes the following ℓ_p-norm energy:

$$E_{\text{diff}} = \sum_{\mathbf{e}\in\mathcal{E}} (L_{\mathbf{e}}/L - 1)^p, \tag{12}$$

which is the most natural form for preserving edge lengths. In the experiments, we choose $p = 4$ and $p = 8$, and we call them D_{p4} and D_{p8}, respectively. We optimize E_{diff} by setting $g = L_{\mathbf{e}}/L - 1$ and

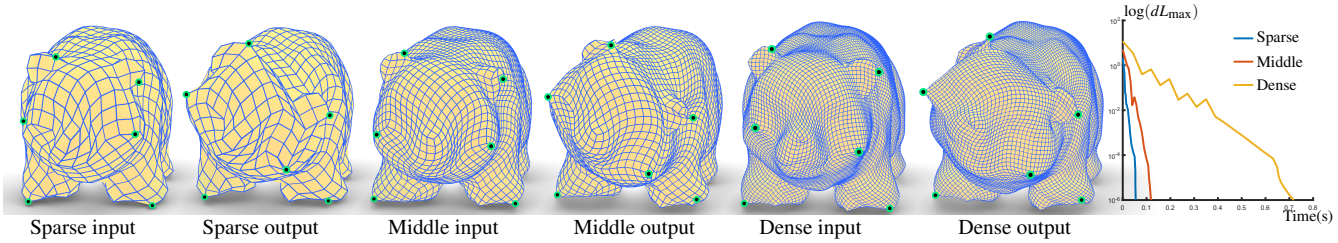

Figure 16: *Various resolutions. We choose three meshes which have similar shapes but different resolutions, and we specify the target positions of the handles. According to their resolutions, they are called sparse (972 vertices), middle (3895 vertices) and dense (15477 vertices).*

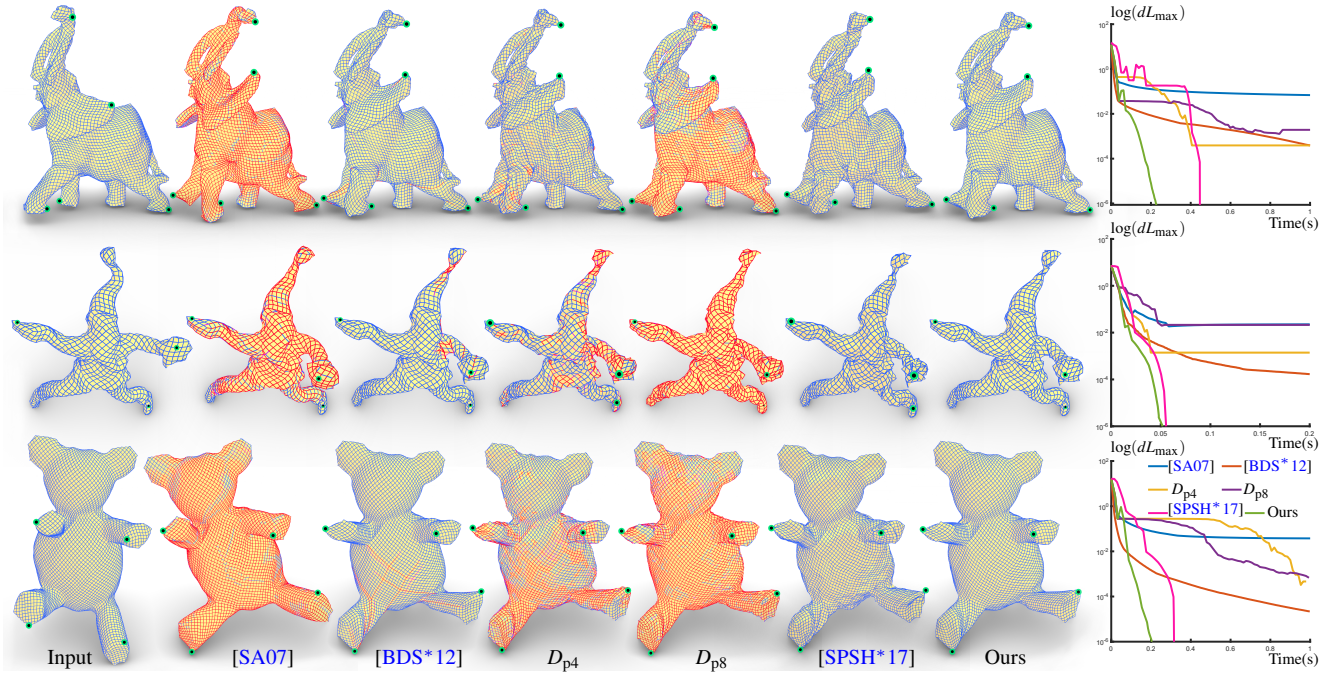

Figure 17: *Comparisons with [SA07], [BDS*12], D_{p4}, D_{p8} and [SPSH*17] on the Elephant model (6055 vertices), the Santa model (1488 vertices), and the Teddy model (6777 vertices).*

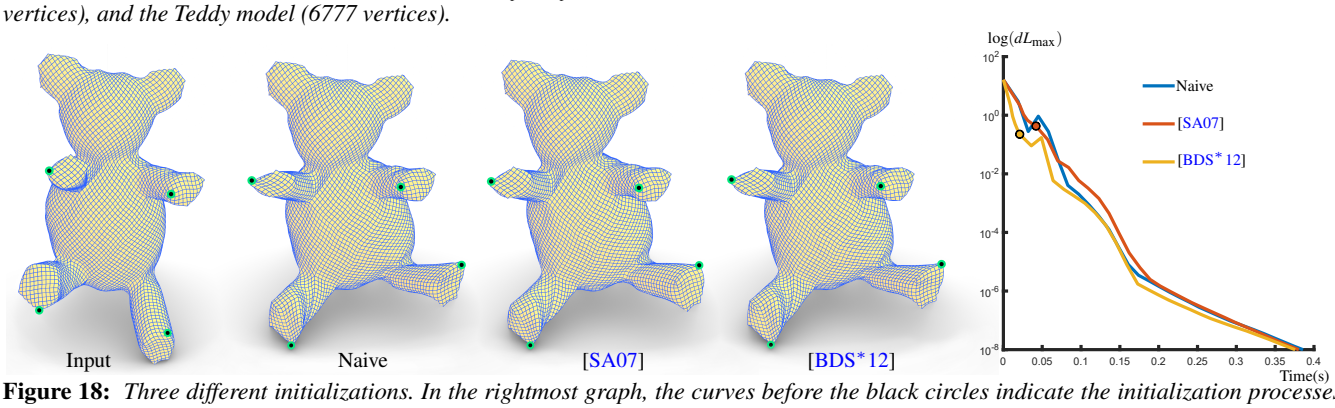

Figure 18: *Three different initializations. In the rightmost graph, the curves before the black circles indicate the initialization processes of [SA07] and [BDS*12].*

$h = x^p$ that is decomposed as follows:

$$h_{\text{inc}}(x) = \begin{cases} x^p, & \text{if } x \geq 0, \\ 0, & \text{if } x < 0, \end{cases} \text{ and } h_{\text{dec}}(x) = \begin{cases} 0, & \text{if } x \geq 0, \\ x^p, & \text{if } x < 0. \end{cases} \quad (13)$$

Since we need to guarantee $(L_{\mathbf{e}}/L - 1)^p \geq 0$, we choose p to be even. Then, our second-order solver is used to minimize E_{diff}, and the corresponding adaptive p strategy is used. The fourth competitor is solver of [SPSH*17]. In this part we use both acceleration strategies for the solver of [SPSH*17]. The optimization procedure and parameters are the same as our method.

As shown in Figure 17, we compare with these competitors on three examples. From the comparisons, they cannot minimize the

maximum edge length deviation to be small enough, i.e., $dL_{\max}$ values are greater than $\varepsilon_3 = 10^{-6}$, or they are slower. Our method succeeds in achieving a much smaller $dL_{\max}$ using less time. Besides, more comparisons are provided in the supplementary material.

In the fourth and fifth columns, we optimize E_{diff} with different ps. As shown in Figure 17, increasing p cannot contribute to improving the convergence rate. When $(L_{\mathbf{e}}/L-1)$ is close to 0, i.e., $\varepsilon = |L_{\mathbf{e}}/L-1| \to 0$, minimizing $(L_{\mathbf{e}}/L-1)^p$ is actually to decrease ε^p which is less than ε. Thus, increasing p cannot reduce $|L_{\mathbf{e}}/L-1|$ or $dL_{\max}$ effectively.

Although some of them cannot optimize $dL_{\max}$ to be small enough, we can use their results as the initializations of our method. In Figure 18, we test three different initializations. First, the naïve initialization is generated by dragging the handles directly to their target positions, leaving other vertices in the rest poses. The results after running [SA07] and [LBOK13] for five iterations are used as the second initialization and the third initialization, respectively. We observe that there is almost no difference in convergence rate using these different initializations. Thus, we use the naïve initialization by default. In the interaction process, the resulting net in the last frame is used as an initialization for the next frame.

More examples Our tool is able to edit discrete Chebyshev nets interactively. In practice, real-time editing (about 25 FPS) can be realized for the discrete Chebyshev nets with around 4000 vertices. If 10 FPS is allowed during the editing process, nets with around 8000 vertices are supported. More editing results are shown in Figure 19. We also provide the reference implementation of our algorithm and an interactive demo in the supplementary materials.

Extension If L_{ref} is set as the input length for each edge $\mathbf{e}$ in the objective function, our method can solve length-preserved deformation for general meshes instead of only discrete Chebyshev nets. An example is shown in Figure 20.

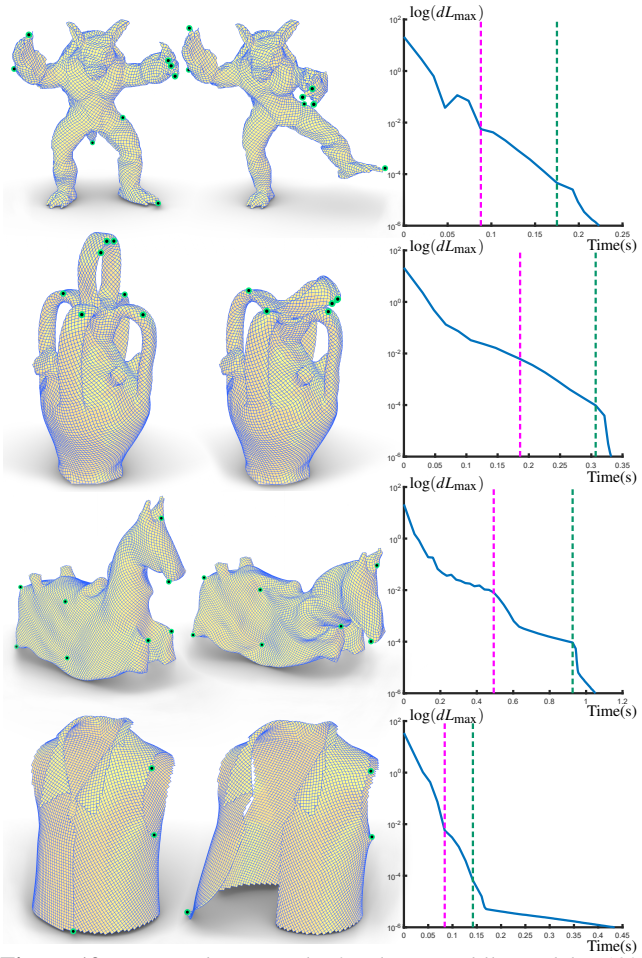

Figure 19: *More editing results for the Armadillo model (7192 vertices), Botijo model (9224 vertices), Rocking horse model (11153 vertices), and Sleeveless shirt model (8592 vertices).*

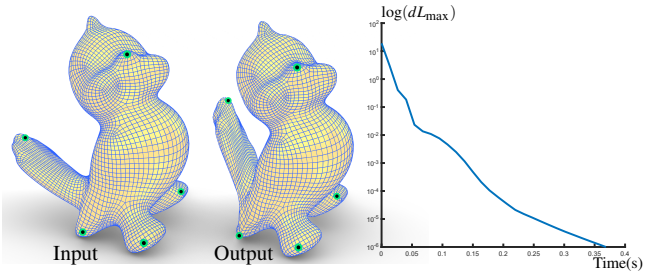

Figure 20: *Extension. Our method can solve length-preserving deformation for general meshes.*

5. Conclusion

Our method provides a novel approach to edit discrete Chebyshev nets interactively. To make the maximum edge length deviation drop to be very small, we use ℓ_p-norm to define an objective function. Then, the objective function is minimized by a second-order solver with a novel majorizer. Consequently, our optimization method is able to reduce the maximum edge length deviation quickly. The feasibility and practicality of our editing tool are demonstrated by successfully deforming various discrete Chebyshev nets.

Failure cases We have no theoretical guarantee that the resulting mesh is always a discrete Chebyshev net. Specifically, large deformation or hard positional constraints of handles may cause the algorithm to fail (Figure 21), and the quads may collapse as we circle in Figure 21 since we can't guarantee isometric deformation. To accommodate large deformations, one possible solution is to adaptively change the target edge length L. For the hard positional constraints, relaxing them as soft constraints is a potential solution. We consider these attempts as future work.

Global effect When moving a point handle, it appears that the shape is globally affected even if the vertices are far away from the handle. To control this, we plan to add an extra energy term defined as the sum of the squared distance from the new position to the original position of each vertex weighted by the geodesic distances to the handles. This is an interesting future work.

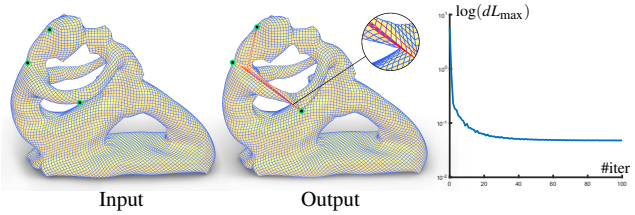

Figure 21: *The hard constraints that move the handles to the specified positions cause a failure to obtain a discrete Chebyshev net. The collapsed quads caused by non-isometric deformation are circled out.*

Acknowledgments

We would like to thank the anonymous reviewers for their constructive suggestions and comments. This work is supported by the National Natural Science Foundation of China (62025207), the USTC Research Funds of the Double First-Class Initiative (YD0010002003), and the Zhejiang Lab (NO. 2019NB0AB03).

References

[ABW01] AONO M., BREEN D. E., WOZNY M. J.: Modeling methods for the design of 3d broadcloth composite parts. *Computer Aided Design 33*, 13 (2001), 989–1007. 3

[ADBW96] AONO M., DENTI P., BREEN D. E., WOZNY M. J.: Fitting a woven cloth model to a curved surface: dart insertion. *IEEE Computer Graphics and Applications 16*, 5 (1996), 60–70. 2, 3

[Aon94] AONO M.: *Computer-aided geometric design for forming woven cloth composites*. PhD thesis, Rensselaer Polytechnic Institute, 1994. 1, 2, 3

[BBC10] BOUHAYA L., BAVEREL O., CARON J.-F.: Mapping two-way continuous elastic grid on an imposed surface: Application to grid shells. In *Symposium of the International Association for Shell and Spatial Structures (50th. 2009. Valencia). Evolution and Trends in Design, Analysis and Construction of Shell and Spatial Structures: Proceedings* (2010), Editorial Universitat Politècnica de València. 2

[BBC14] BOUHAYA L., BAVEREL O., CARON J.-F.: Optimization of gridshell bar orientation using a simplified genetic approach. *Structural and Multidisciplinary Optimization 50*, 5 (2014), 839–848. 2, 3

[BDS*12] BOUAZIZ S., DEUSS M., SCHWARTZBURG Y., WEISE T., PAULY M.: Shape-up: Shaping discrete geometry with projections. *Comput. Graph. Forum 31*, 5 (2012), 1657–1667. 2, 3, 7, 8

[BP*96] BOBENKO A., PINKALL U., ET AL.: Discrete surfaces with constant negative gaussian curvature and the hirota equation. *Journal of Differential Geometry 43*, 3 (1996), 527–611. 2

[BSFJR18] BAEK C., SAGEMAN-FURNAS A. O., JAWED M. K., REIS P. M.: Form finding in elastic gridshells. *Proceedings of the National Academy of Sciences 115*, 1 (2018), 75–80. 2

[DBD*15] DENG B., BOUAZIZ S., DEUSS M., KASPAR A., SCHWARTZBURG Y., PAULY M.: Interactive design exploration for constrained meshes. *Computer Aided Design 61* (2015), 13–23. 2, 3

[FLG15] FU X.-M., LIU Y., GUO B.: Computing locally injective mappings by advanced mips. *ACM Trans. Graph. 34*, 4 (2015). 3

[FSZ*21] FU X.-M., SU J.-P., ZHAO Z.-Y., FANG Q., YE C., LIU L.: Inversion-free geometric mapping construction: A survey. *Computational Visual Media 7*, 3 (2021), 289–318. 3

[GSC18] GOLLA B., SEIDEL H.-P., CHEN R.: Piecewise Linear Mapping Optimization Based on the Complex View. *Computer Graphics Forum* (2018). 3, 4

[GSFD*14] GARG A., SAGEMAN-FURNAS A. O., DENG B., YUE Y., GRINSPUN E., PAULY M., WARDETZKY M.: Wire mesh design. *ACM Trans. Graph. 33*, 4 (2014). 1, 2, 3

[HSRG12] HERNÁNDEZ E. L., SECHELMANN S., RÖRIG T., GENGNAGEL C.: Topology optimisation of regular and irregular elastic gridshells by means of a non-linear variational method. In *AAG* (2012), pp. 147–160. 2

[Lan13a] LANGE K.: The mm algorithm. In *Optimization*. Springer, 2013, pp. 185–219. 3, 4

[Lan13b] LANGE K.: *Optimization*. Springer Science & Business Media, 2013. 4

[LBK17] LIU T., BOUAZIZ S., KAVAN L.: Quasi-newton methods for real-time simulation of hyperelastic materials. *ACM Trans. Graph. 36*, 3 (2017), 23. 3

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Trans. Graph. 32*, 6 (2013). 2, 3, 7, 9

[LLZ*20] LIU H.-Y., LIU Z.-Y., ZHAO Z.-Y., LIU L., FU X.-M.: Practical fabrication of discrete chebyshev nets. *Comput. Graph. Forum 39*, 7 (2020), 13–26. 1, 2, 3, 6

[LYNF18] LIU L., YE C., NI R., FU X.-M.: Progressive parameterizations. *ACM Trans. Graph. 37*, 4 (2018). 3

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S. J.: A local/global approach to mesh parameterization. *Comput. Graph. Forum 27*, 5 (2008), 1495–1504. 3

[Mas17] MASSON Y.: *Existence and construction of Chebyshev nets with conical singularities and application to gridshells*. PhD thesis, Ph. D. Dissertation. ENPC, University Paris-Est, 2017. 1, 3

[NW06] NOCEDAL J., WRIGHT S.: *Numerical optimization*. Springer Science & Business Media, 2006. 2, 3

[PDZ*18] PENG Y., DENG B., ZHANG J., GENG F., QIN W., LIU L.: Anderson acceleration for geometry optimization and physics simulation. *ACM Trans. Graph. 37*, 4 (2018). 3

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Trans. Graph. 36*, 4 (2017). 3

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Symposium on Geometry processing* (2007), vol. 4, pp. 109–116. 2, 3, 7, 8, 9

[SD95] SAMELSON S. L., DAYAWANSA W.: On the existence of global tchebychev nets. *Transactions of the American Mathematical Society 347*, 2 (1995), 651–660. 2

[SFCBCV19] SAGEMAN-FURNAS A. O., CHERN A., BEN-CHEN M., VAXMAN A.: Chebyshev nets from commuting polyvector fields. *ACM Trans. Graph. 38*, 6 (2019), 1–16. 1, 3

[SFL19] SU J.-P., FU X.-M., LIU L.: Practical foldover-free volumetric mapping construction. *Comput. Graph. Forum 38*, 7 (2019), 287–297. 3

[SGK19] SMITH B., GOES F., KIM T.: Analytic eigensystems for isotropic distortion energies. *ACM Trans. Graph. 38* (2019). 3, 4

[SPSH*17] SHTENGEL A., PORANNE R., SORKINE-HORNUNG O., KOVALSKY S. Z., LIPMAN Y.: Geometric optimization via composite majorization. *ACM Trans. Graph. 36*, 4 (2017), 1–11. 2, 3, 4, 5, 6, 8

[SYLF20] SU J.-P., YE C., LIU L., FU X.-M.: Efficient bijective parameterizations. *ACM Trans. Graph. 39*, 4 (2020). 3

[ZBK18] ZHU Y., BRIDSON R., KAUFMAN D. M.: Blended cured quasi-newton for distortion optimization. *ACM Trans. Graph.* (2018). 3