Digital Three-dimensional Smocking Design

JING REN, AVIV SEGALL, and OLGA SORKINE-HORNUNG, ETH Zurich, Switzerland

We develop an optimization-based method to model *smocking*, a surface embroidery technique that provides decorative geometric texturing while maintaining stretch properties of the fabric. During smocking, multiple pairs of points on the fabric are stitched together, creating non-manifold geometric features and visually pleasing textures. Designing smocking patterns is challenging, because the outcome of stitching is unpredictable: The final texture is often revealed only when the whole smocking process is completed, necessitating painstaking physical fabrication and time consuming trial-and-error experimentation. This motivates us to seek a digital smocking design method. Straightforward attempts to compute smocked fabric geometry using surface deformation or cloth simulation methods fail to produce realistic results, likely due to the intricate structure of the designs, the large number of contacts and high-curvature folds. We instead formulate smocking as a graph embedding and shape deformation problem. We extract a coarse graph representing the fabric and the stitching constraints and then derive the graph structure of the smocked result. We solve for the three-dimensional embedding of this graph, which in turn reliably guides the deformation of the high-resolution fabric mesh. Our optimization based method is simple, efficient, and flexible, which allows us to build an interactive system for smocking pattern exploration. To demonstrate the accuracy of our method, we compare our results to real fabrications on a large set of smocking patterns.

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

Additional Key Words and Phrases: Smocking, embroidery, shape deformation, graph embedding

ACM Reference format:
Jing Ren, Aviv Segall, and Olga Sorkine-Hornung. 2024. Digital Three-dimensional Smocking Design. *ACM Trans. Graph.* 43, 2, Article 14 (January 2024), 17 pages.
https://doi.org/10.1145/3631945

1 INTRODUCTION

Smocking is a surface embroidery technique used in textile design that serves two main purposes: it is highly decorative and provides ornamentation, and it also has the practical benefit of allowing a close fit of the garment while maintaining a certain degree of stretch. Consequently, smocking is an artistic means of controlling a garment's fullness, thereby creating more shape for the wearer [Banner 2022; Durand 1979]. Moreover, smocking can act as reinforcement and insulation, padding over areas such as shoulders and chest to add durability to the garment [Toplis 2021].

This work was supported in part by the ERC Consolidator Grant No. 101003104 (MYCLOTH).
Authors' address: J. Ren, A. Segall, and O. Sorkine-Hornung, ETH Zurich, Department of Computer Science, Institute of Visual Computing, Universitätstrasse 6, 8092 Zurich, Switzerland; e-mails: {jing.ren, aviv.segall, sorkine}@inf.ethz.ch.

0730-0301/2024/01-ART14
https://doi.org/10.1145/3631945

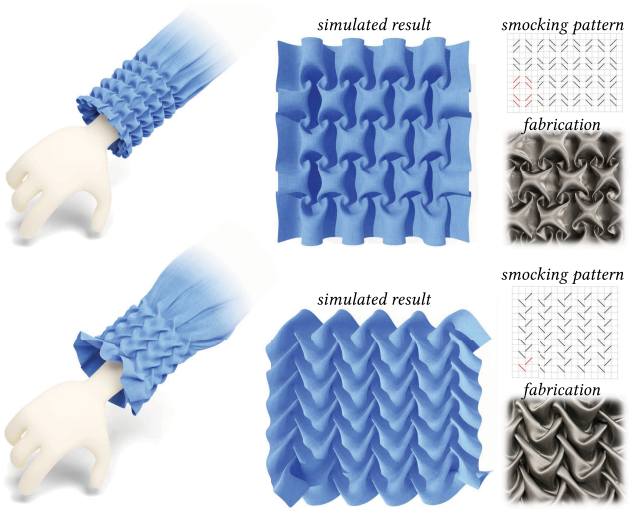
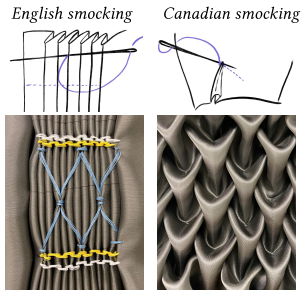


Fig. 1. We model *smocking*, a decorative geometric cloth texturing technique, where pairs of points are stitched together to form a pleated pattern. Here we show examples of smocked sleeves produced using our method, where the volumetric smocked textures create natural folds. The geometry of the smocked fabric computed with our method based on the input pattern closely matches the physically fabricated counterpart (photos in grey).

Garments made with this technique are called *smock-frocks* or *smocks*.

Smocking can be roughly categorized into two styles according to the embroidering process: *English smocking*, where the pleating and stitching are done sequentially, and *Canadian smocking*, where the stitching generates the pleating simultaneously. In traditional English smocking, the fabric is first folded into close and *uniform* pleats, and then the gathered threads are used as a guide to embroider rows of stitches through the pleats. The stitches remain visible and play the main decorative role, akin to standard two-dimensional (2D) embroidery, whereas the pleating serves mainly to create a thicker base medium and generate folds when transitioning between smocked and non-smocked parts of the fabric. The final appearance of an English smocking pattern is predictable, since the pleats are pre-folded and the embroidery patterns are determined by the alignment of stitches. In Canadian smocking, the fabric is pleated by stitching it locally, connecting or "pinching" pairs of points in a special pattern. The stitches are invisible in the final result, and the decorative, *geometric* texture



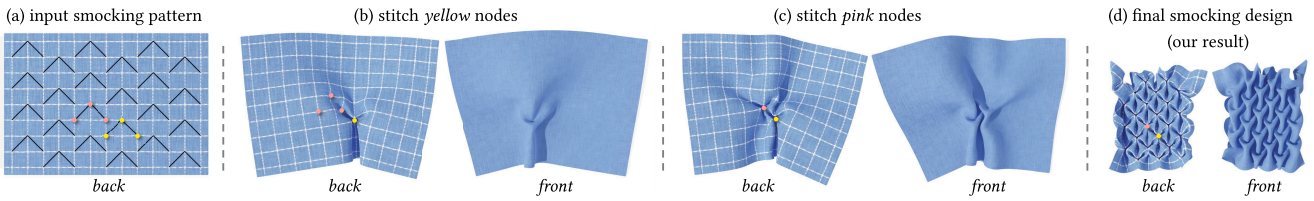


Fig. 2. The smocking process. A smocking pattern (a) consists of stitching lines (the polylines in black). Each stitching line needs to be contracted into a point by gathering and sewing together its nodes ((b) and (c)). Sewing all stitching lines reveals the final geometric texture (d). All pieces of fabric are shown in scale; note that smocking shrinks the starting piece of cloth significantly, since multiple points are pinched together.

is formed by the pleats themselves. The final appearance of a Canadian smocking pattern is much harder to predict based on the given stitch pattern; the geometric texture is often revealed only when the whole smocking process is completed, making its design a challenging trial-and-error process [Efrat et al. 2016].

In this work, we therefore focus on *Canadian smocking*, with the goal of creating a digital framework for design and preview, where users can explore and experiment with various stitching patterns and visualize the smocking results without having to sew them physically (Figure 1). We investigate a mathematical formalization of the smocking problem and design an automatic and efficient algorithm to compute smocked fabric geometry based on input patterns. Surprisingly, approaching smocking modeling in a straightforward way as a constrained surface deformation or cloth simulation problem generally fails to satisfy all point-to-point stitching constraints and deliver faithful results, likely due to the intricate, essentially non-manifold structure of the design, the very large number of contacts, and the high-curvature folds. We instead formulate smocking as a graph embedding optimization problem that guides the cloth deformation. We extract a coarse graph representing the fabric and the stitching constraints and then derive the graph structure of the smocked result. We solve for the 3D embedding of this graph, which in turn reliably guides the deformation of the high-resolution fabric mesh. To demonstrate the accuracy of our method, we compare our results to real fabrications on a variety of smocking patterns.

Contributions. In this work we propose (1) the first formalization of smocking design as a graph embedding and shape deformation problem and (2) an efficient algorithm to compute the smocked fabric geometry from a given pattern, enabling (3) an interactive tool for designing smocking patterns.

2 RELATED WORK

Smocking. The word *smock* comes from the Anglo-Saxon word *smocc*, the name of an outer sack-like garment, later called *smock frock*, which was worn over a farmer's other clothes to protect them from getting soiled [Durand 1979; Spufford and Mee 2017; Toplis 2021]. We refer the interested readers to a recent book, *The Hidden History of the Smock Frock* [Toplis 2021], for details.

Existing research on smocking is related to adult education [Bauer and Elsey 1992], psychology [Elbyaly and Elfeky 2022], or bedroom decorations marketing [Joseph et al. 2011]. Efrat et al. [2016] propose a digital design tool for smocking, where users can tile predefined unit smocking patterns and then print them on fabric. Their system does not visualize the result: The smocking itself needs to be completed manually by sewing the physical fabric. The authors note that the complexity of smocking makes it difficult to automate and that predicting the smocking result of a given pattern is challenging.

Lind [2019] explores the design of colorful jacquard woven patterns that serve as templates for the smocking stitches, such that the woven pattern shapes the fabric. The jacquard patterns are customized for different smocking patterns. During the design and experimentation, the patterned fabrics have to be produced on a jacquard machine and then smocked manually in each design iteration. Kim [2020] emulates smocking in a step-by-step manner in a commercial virtual clothing software [CLO 2023], manually creating each stitch by simulating a tacking and folding step. Online creators post similar manual techniques to model smocking details in digital garments [CLO 2020]. In contrast, our method is fully automatic and efficiently simulates the entire smocked shape.

Physically based cloth simulation. Following the pioneering work of Terzopoulos et al. [1987], different elastic models have been studied for representing cloth dynamics, including finite element methods [Baraff and Witkin 1998; Narain et al. 2012], mass-spring systems [Choi and Ko 2002; Liu et al. 2013], and yarn-level cloth simulation [Cirio et al. 2014; Kaldor et al. 2010]. To accurately model folding or wrinkling of cloth, different collision handling techniques have been proposed [Bridson et al. 2005; Li et al. 2021; Tang et al. 2018; Wang 2021]. Chen et al. [2021] propose a new model based on thin shells to model fine-scale wrinkling. However, general-purpose cloth simulators struggle with the smocking task, because the pleats are mainly formed by stitches, whose pinching effect is challenging to capture by simulated wrinkles from cloth dynamics alone (see Figure 6 for an example). FoldSketch [Li et al. 2018] is a dedicated inverse modeling system for folds and pleats, where the user sketches the desired folds on the draped 3D garment, and the algorithm adjusts the sewing pattern to reproduce them. While very effective for pleats and gathers that extend along *one-dimensional* curved paths, this system is not suitable for sketching smocked pleats, which are arranged in a *two-dimensional* pattern with many occlusions and overlaps. The smocked appearance is not entirely independent of the fabric type, but the dominant factor that governs the geometry and the regularity of the pleats is the structural stitching pattern, as opposed to the cloth parameters. In this work, we focus on the geometric formulation of smocking and assume the fabric to be roughly inextensible [Goldenthal et al. 2007].

Shape deformation. Instead of dynamically simulating cloth, smocking can be seen as an end state of a draping process that

can be computed via surface deformation with positional constraints [Sorkine and Botsch 2009]. Among the many surface deformation methods, as-rigid-as-possible deformation (ARAP) [Sorkine and Alexa 2007] models least-squares isometric deformations, which can be used as a stand-in for inextensible cloth. Deformation methods generally do not consider self collisions and contacts and do not do well on the smocking task when applied directly (Figure 5). In our approach, we encapsulate the contacts, i.e., the sewing constraints, in the smocked graph structure, which guides the subsequent fine-grained deformation to a feasible and faithful configuration.

Digital design. The subject of our work fits into digital design—algorithms and systems that assist users in creating digital artifacts before physically fabricating them. Recent examples in this space include origami [Dudte et al. 2016], kirigami [Castle et al. 2014, 2016; Jiang et al. 2020], knittable meshes [Wu et al. 2019], 3D weaving [Ren et al. 2021], and quilting [Carlson et al. 2015; Igarashi and Mitani 2015; Leake et al. 2021], among many others. Here we focus on kirigami and quilting, which are more closely related to smocking.

Kirigami is a generalized origami technique where cutting out holes is allowed. It is often employed for regular tessellation patterns, similar to Canadian smocking, which are visually appealing and/or achieve particular mechanical behaviors [An et al. 2020; Wang et al. 2017]. Castle et al. [2014, 2016] explore rules for cutting and folding kirigami, while Jiang et al. [2020] investigate the inverse problem of designing a kirigami pattern such that the deployed result is similar to a given 3D shape. The main difference between kirigami and smocking is the material: Kirigami uses paper, which can neither stretch nor shear. In contrast, smocking is intended for woven fabric, where a certain degree of shearing is possible even if the warp and weft yarns are inextensible. Fabric has a much richer set of degrees of freedom when deforming, so that smocking geometry is smoother and generally more varied compared to kirigami.

Leake et al. [2021] formalize the foundation paper piecing process, which is popular for constructing textile patchwork quilts based on printed patterns. This work encodes the pattern geometry via a dual hypergraph and investigates whether a given pattern is valid, i.e., pieceable. The challenge is to solve for the order of placing the fabric pieces to meet the constraints posed by *known* geometry. In contrast, the challenge of formulating smocking is that the final 3D geometry is *unknown* before the fabrication process is completed. We therefore need to build a graph that can capture the unknown structure information.

3 PRELIMINARIES

Canadian smocking consists of the following steps: (1) preparing a smocking pattern by drawing a grid and designing stitching lines on a piece of fabric and (2) gathering all grid vertices of one stitching line and sewing them together. The sewing is repeated for all stitching lines. Optionally, one can (3) fold the pleats formed during the stitching in a nicer way and iron the smocked pattern if necessary. See also Figure 2 and the accompanying video. In Section 3.1, we formalize each step, and in Sections 3.2 and 3.3 we discuss conceivable straightforward approaches.

3.1 Notation and Problem Formulation

A classic smocking pattern consists of a piece of fabric with a 2D grid drawn on top of it and a set of *stitching lines*, each containing a list of grid nodes. A *pleat* is formed when the nodes of one stitching line are gathered and stitched together into a single point. In practice, a stitching line is annotated by a set of connected line segments to visually separate different stitching lines from each other. The overview of the smocking process is illustrated in Figure 2.

Definition 3.1. A *smocking pattern* $\mathcal{P} = (\mathcal{G}, \mathcal{L})$ is a piece of fabric, represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$, annotated with a set of stitching lines $\mathcal{L} = \{\ell_i\}$. A *stitching line* ℓ is a subset of vertices in $\mathcal{V}$ that are to be stitched together.

Figure 3 shows a simple smocking pattern, represented by a grid, where we denote the vertices as $\mathcal{V} = \{v_{0,0}, \ldots, v_{i,j}, \ldots, v_{n,m}\}$. Then we can read the annotated stitching lines $\mathcal{L} = \{\ell_i\}$ as $\ell_1 = (v_{0,1}, v_{1,2})$, $\ell_2 = (v_{1,1}, v_{2,0})$, $\ell_3 = (v_{3,2}, v_{4,1})$, $\ell_4 = (v_{4,0}, v_{5,1})$, $\ell_5 = (v_{6,1}, v_{7,2}), \ldots$. In practice, a smocking pattern is obtained by tiling a *unit* smocking pattern regularly on the fabric. We delineate the unit smocking pattern by a pink rectangle, and the stitching lines of the unit pattern are marked in red. Note that we include the diagonals of the grid quads into the graph edges $\mathcal{E}$, since they play a role in the subsequent graph embedding.

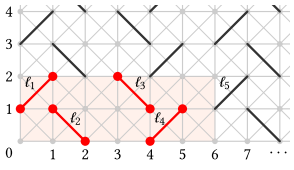


Fig. 3. A smocking pattern $\mathcal{P}$.

During the smocking process, the vertices belonging to the same stitching line (e.g., $v_{0,1}$ and $v_{1,2}$) are gathered and stitched together. A stitching line can consist of multiple line segments, in which case more than two points need to be stitched at the same time (see Figure 2(a) for such an example). Our goal is to compute the *smocking design*, i.e., the 3D geometric texture shape resulting from any given smocking pattern. For this purpose, we use a higher-resolution representation of the fabric, $\widetilde{\mathcal{P}} = (\widetilde{\mathcal{G}}, \mathcal{L})$, where $\widetilde{\mathcal{G}} = (\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}})$ and $\mathcal{V} \subset \widetilde{\mathcal{V}}$, see Figure 4.

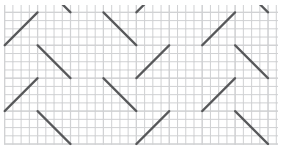


Fig. 4. Finer discretization $\widetilde{\mathcal{P}}$.

Definition 3.2. The *smocking design* from a pattern $\mathcal{P}$ is a mesh $\widetilde{\mathcal{M}} = (\widetilde{\mathbf{X}}, \widetilde{\mathcal{G}})$ embedded in 3D, where $\widetilde{\mathbf{X}} \in \mathbb{R}^{|\widetilde{\mathcal{V}}| \times 3}$ stores the 3D positions $\mathbf{x}_p$ of all nodes $v_p \in \widetilde{\mathcal{V}}$ and satisfies $\mathbf{x}_p = \mathbf{x}_q$, $\forall v_p, v_q \in \ell_i$, $\forall \ell_i \in \mathcal{L}$. We can extract a non-manifold mesh representation $\mathcal{M}'$ from $\widetilde{\mathcal{M}}$ by removing the duplicated vertices and updating the topology of $\widetilde{\mathcal{G}}$ accordingly.

The above definition seems to imply that the smocking design can be computed using shape deformation or cloth simulation, but these approaches fall short.

3.2 Shape Deformation using ARAP

We can cast the smocking design computation as a shape deformation problem and easily adapt as-rigid-as-possible deformation

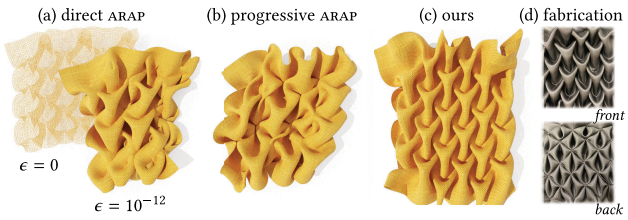


Fig. 5. Straightforward application of ARAP deformation [Sorkine and Alexa 2007] to the smocking pattern shown in Figure 2 fails to recover the expected smocked geometry (a). The value of ϵ stands for the maximum allowed distance between stitched nodes; when $\epsilon = 0$, the deformed mesh stays planar. In (b) we show the result of gradually decreasing ϵ from half of the initial length to 0. Our method successfully computes the smocking design (c), closely matching its physical fabrication (d).

(ARAP) [Sorkine and Alexa 2007] to obtain $\widetilde{\mathbf{X}}$,

$$\min_{\widetilde{\mathbf{X}}\in\mathbb{R}^{|\widetilde{\mathcal{V}}|\times 3}} \sum_i \min_{R_i\in SO(3)} \sum_{j\in\mathcal{N}(i)} w_{ij} \left\| \left(\mathbf{x}_i - \mathbf{x}_j\right) - R_i \left(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\right) \right\|_2^2,$$
$$\text{s.t.} \quad \left\| \mathbf{x}_p - \mathbf{x}_q \right\|_2 = \epsilon, \quad \forall (v_p, v_q) \in \ell_k,\ \forall \ell_k \in \mathcal{L}, \tag{1}$$

where $\bar{\mathbf{x}}_i$ denotes the known starting position of vertex $\mathbf{x}_i$ in the flat fabric piece, $\mathcal{N}(i)$ is the one-ring neighborhood of the ith node in $\widetilde{\mathcal{G}}$, and w_{ij} are the cotangent weights [Meyer et al. 2003]. The ARAP energy encourages the edges in $\widetilde{\mathcal{G}}$ to stay rigid and maintain their length. The deformation occurs due to stitching, which is modeled via the constraints. For two nodes in the same stitching line, we allow ϵ distance in the deformed state; ϵ can either be set to the thickness of the fabric or zero for simplicity.

We note that the constraints in Equation (1) are nonlinear and non-convex for $\epsilon \neq 0$, so we simplify the constraints by linearization. In Figure 5, we show the smocking design results of ARAP obtained using three different settings for the same pattern illustrated in Figure 2(a). (i) When $\epsilon = 0$, we get planar positional constraints $\mathbf{x}_p = \mathbf{x}_q$. Since the initial mesh is planar, the ARAP deformation does not manage to get out of the planar configuration and produces a planar self-intersecting mesh (Figure 5(a), left). (ii) Softening the stitching constraints by setting ϵ to a small non-zero value allows the optimization to find a non-planar local minimum, but the result is irregularly wrinkled (Figure 5(a), right), likely because the stitching constraints overpower the optimization, not letting the surface relax. (iii) In Figure 5(b) we attempt a progressive strategy, where we iteratively reduce ϵ from half of the initial length of the stitching lines to 0. The result is better but still not sufficiently regular.

3.3 Cloth Simulation

Computing the smocking design can naturally be formulated as a cloth simulation problem. We use a popular simulator, CLOTH, implemented in Blender [2023], which uses the point-based dynamics of a mass spring system [Bridson et al. 2002] and incorporates contacts and friction. To simulate smocking, we add virtual linear springs of rest length 0, connecting each pair of nodes in each stitching line. In Figure 6, we show the simulated result of the pattern in Figure 2 over iterations. We observe a similar effect as with ARAP: As the sewing lines become shorter, the textile

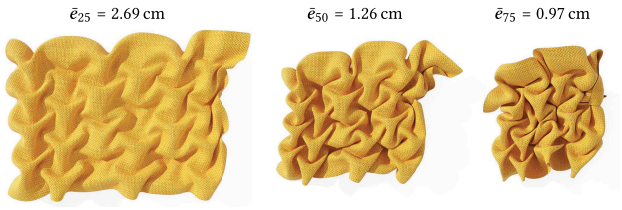


Fig. 6. Simulated smocking design using Blender [2023]. We report the average length in centimeters $\bar{e}_k$ of all sewing lines after k iterations, where $k = 25, 50, 75$ (converged), for a smocking pattern of size 50 cm × 70 cm. The stitching lines have initial length of 5.5 cm and are expected to reach zero length after stitching.

becomes bunched up in an irregular fashion, because the simulator is not aware of the high-level regularity of the smocking pattern. As a comparison, our result shown in Figure 5(c) achieves regular and realistic smocking with zero-length sewing lines. See Figure 9 for more examples.

3.4 Observations and Challenges

Through our experiments, we have discovered that while it is possible to find many smocking designs that meet the criteria outlined in Definition 3.2, the definition itself falls short of adequately describing the desired voluminous and regular geometric texture preferred by artists. In practice, a smocking pattern is usually obtained by evenly tiling a unit pattern onto the fabric. As a result, one would anticipate achieving regular pleats with visually repetitive and consistent patterns. Prior knowledge of regularity is crucial, as the absence of such knowledge causes both state-of-the-art shape deformation methods and cloth simulators to struggle with avoiding visually unpleasant or degenerated local minima. At the same time, formulating regularity in smocking is quite challenging, given that the geometry remains unknown until the fabrication process is completed. Additionally, imposing 3D geometry priors on a 2D input pattern is nontrivial.

4 METHOD

The experiments above reveal that to model smocking, we need to somehow impose a global regular structure on the fabric, because the deformation energies that are based on purely local differential properties have abundant local minima that lack symmetry and yield undesirable results. To tackle this challenge, we solve the smocking design problem in two steps: We consider the input smocking pattern $\mathcal{P}$, defined on a *coarse* representation of the fabric (see Section 3.1) and optimize its 3D graph embedding. We then apply ARAP, guided by the computed 3D embedding, on a finer representation of the fabric, $\widetilde{\mathcal{P}}$, to compute the final smocking design. In the following, we explain our method in detail.

4.1 Smocked Graph Extraction

We define the *smocked graph* from the input smocking pattern $\mathcal{P} = (\mathcal{G} = (\mathcal{V}, \mathcal{E}), \mathcal{L} = \{\ell_i\})$ to represent the non-manifold structure of the resulting smocking design. We first categorize the vertices $v \in \mathcal{V}$ and edges $e \in \mathcal{E}$ as follows.

Definition 4.1. A vertex $v \in \mathcal{V}$ in a smocking pattern $\mathcal{P}$ is called an *underlay vertex* if it belongs to a stitching line, i.e., $\exists \ell_i \in \mathcal{L}$ s.t. $v \in \ell_i$, and it is called a *pleat vertex* otherwise.

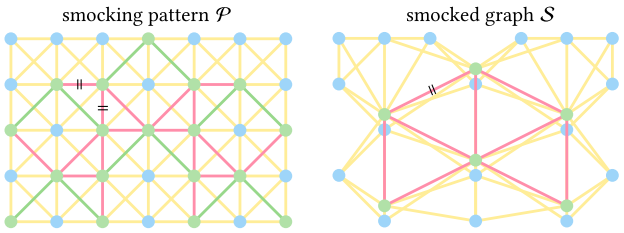


Fig. 7. *Left*: For the smocking pattern $\mathcal{P}$, we color the underlay nodes (respectively, pleat nodes) in green (respectively, blue), and the underlay edges (respectively, pleat edges) in pink (respectively, yellow). The stitching lines and the degenerated edges are colored in green. *Right*: We show the corresponding smocked graph $\mathcal{S}$.

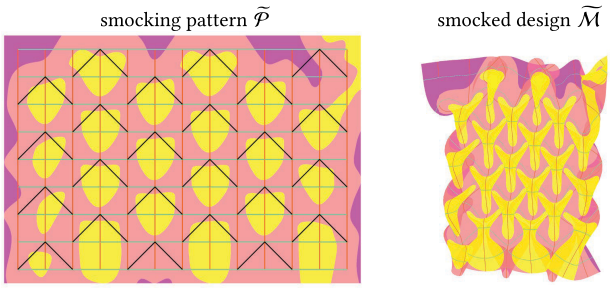


Fig. 8. Inspired by Lind [2019], we color the smocking pattern w.r.t. height after smocking: Yellow highlights the regions that form the pleats, while pink highlights the regions that are almost hidden in the smocked result and form the underlay layer that supports the pleats.

Definition 4.2. An edge $e \in \mathcal{E}$ in a smocking pattern $\mathcal{P}$ is called a *degenerated edge* if its two endpoints belong to the same stitching line, an *underlay edge* if its two endpoints belong to two different stitching lines, and a *pleat edge* otherwise.

For example, in Figure 7 we construct the *smocked graph* $\mathcal{S} = (\mathcal{V}_\mathcal{S}, \mathcal{E}_\mathcal{S})$ from pattern $\mathcal{P}$ by fusing all underlay vertices sharing the same stitching line into one, deleting *degenerated* edges and removing edges that become duplicate as a result of the fusing of underlay vertices. An example of such duplicate edges is marked with "=" in Figure 7 (left); they correspond to a single edge in the smocked graph.

The smocked graph $\mathcal{S}$ is a subgraph of $\mathcal{P}$ that encodes the structure of the final smocked design; the vertices and edges of $\mathcal{S}$ inherit the pleat/underlay attributes (the colors in Figure 7) from $\mathcal{P}$. We denote the set of underlay (pleat) nodes in $\mathcal{S}$ as $\mathcal{V}_u$ ($\mathcal{V}_p$), and the set of underlay (pleat) edges in $\mathcal{S}$ as $\mathcal{E}_u$ ($\mathcal{E}_p$). We have

$$\mathcal{V}_\mathcal{S} = \mathcal{V}_u \cup \mathcal{V}_p, \quad \mathcal{E}_\mathcal{S} = \mathcal{E}_u \cup \mathcal{E}_p. \tag{2}$$

Note each vertex in $\mathcal{V}_u$ represents a single stitching line in $\mathcal{P}$; therefore, $|\mathcal{V}_u| = |\mathcal{L}|$. We also define two important subgraphs of $\mathcal{S}$:

Definition 4.3. The subgraph of the smocked graph $\mathcal{S}$ induced by the underlay edges is termed the *underlay graph*, denoted as $\mathcal{S}_u$. It contains all underlay edges $\mathcal{E}_u$ and their incident underlay vertices $\mathcal{V}_u$. The subgraph of $\mathcal{S}$ induced by the pleat edges is termed the *pleat graph*, denoted $\mathcal{S}_p$. It contains all pleat edges $\mathcal{E}_p$ and their incident vertices, including all pleat vertices $\mathcal{V}_p$ and incident underlay vertices.

We can see that $\mathcal{S} = \mathcal{S}_u \cup \mathcal{S}_p$. Figure 8 provides an intuition for the smocked graph: We color the smocking pattern by height after smocking, where yellow corresponds to large height where a pleat pops up (encoded by the pleat graph $\mathcal{S}_p$) and pink signifies the underlay with low height that forms the base layer of the smocked design (encoded by the underlay graph $\mathcal{S}_u$).

4.2 Smocked Graph Embedding

The smocked graph is a distilled abstract representation of the smocking pattern, with the stitching constraints already satisfied. Our goal is to find a proper embedding of the smocked graph $\mathcal{S}$, i.e., assign a 3D position for each vertex $v \in \mathcal{V}_\mathcal{S}$, such that the embedded smocked graph forms a realistic 3D structure. We formulate this graph embedding problem as an optimization and design appropriate energies and constraints.

4.2.1 Embedding Distance Constraint. We observe that the nodes in the smocked graph $\mathcal{S}$ are constrained by the underlying fabric and cannot move completely freely in space. For example, consider two vertices in the underlay graph $v_{\ell_i}, v_{\ell_j} \in \mathcal{V}_u$ (which originated from stitching lines ℓ_i and ℓ_j in $\mathcal{P}$) with embedded 3D coordinates $\mathbf{x}_{\ell_i}$ and $\mathbf{x}_{\ell_j}$, respectively. Denote by $d(\cdot,\cdot)$ the geodesic distance on the fabric between two vertices, which is approximately equal to their Euclidean distance on the flat fabric. The Euclidean distance between the embedded underlay vertices is constrained by

$$\left\|\mathbf{x}_{\ell_i} - \mathbf{x}_{\ell_j}\right\|_2 \le \min_{v_p \in \ell_i,\, v_q \in \ell_j} d(v_p, v_q), \tag{3}$$

i.e., the *shortest geodesic* distance on the fabric among any pair of stitching vertices on ℓ_i and ℓ_j. For simplicity of exposition, here we assume the fabric cannot stretch.

For example, in Figure 10 we can see that the constraint for the pair of stitching lines is $d_{i,j} = 1$. If the embedded positions for the two corresponding underlay nodes had a distance larger than 1, and assuming infinite stiffness, then the fabric would tear.

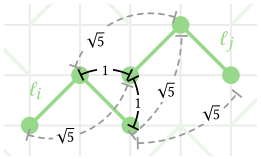


Fig. 10. The $d_{i,j}$ constraint.

We can compute such an *embedding distance constraint*, denoted as $\|\mathbf{x}_i - \mathbf{x}_j\|_2 \le d_{i,j}$, for any pair of vertices $(v_i, v_j) \in \mathcal{V}_\mathcal{S} \times \mathcal{V}_\mathcal{S}$. We have $d_{j,i} = d_{i,j}$ and

$$d_{i,j} = \begin{cases} \min\limits_{v_r \in \ell_{v_i},\, v_q \in \ell_{v_j}} d\left(v_r, v_q\right), & \text{if } v_i, v_j \in \mathcal{V}_u, \\ \min\limits_{v_r \in \ell_{v_i}} d\left(v_r, v_j\right), & \text{if } v_i \in \mathcal{V}_u,\ v_j \in \mathcal{V}_p, \\ d\left(v_i, v_j\right), & \text{if } v_i, v_j \in \mathcal{V}_p, \end{cases} \tag{4}$$

where ℓ_{v_i} denotes the stitching line in $\mathcal{P}$ that corresponds to the underlay node v_i in the smocked graph $\mathcal{V}_\mathcal{S}$.

Ideally, we wish to find a valid embedding of the smocked graph such that all vertex pairs satisfy the embedding distance constraints. Intuitively, it means that if we physically pin the vertices of the pattern annotated on real fabric to their embedding

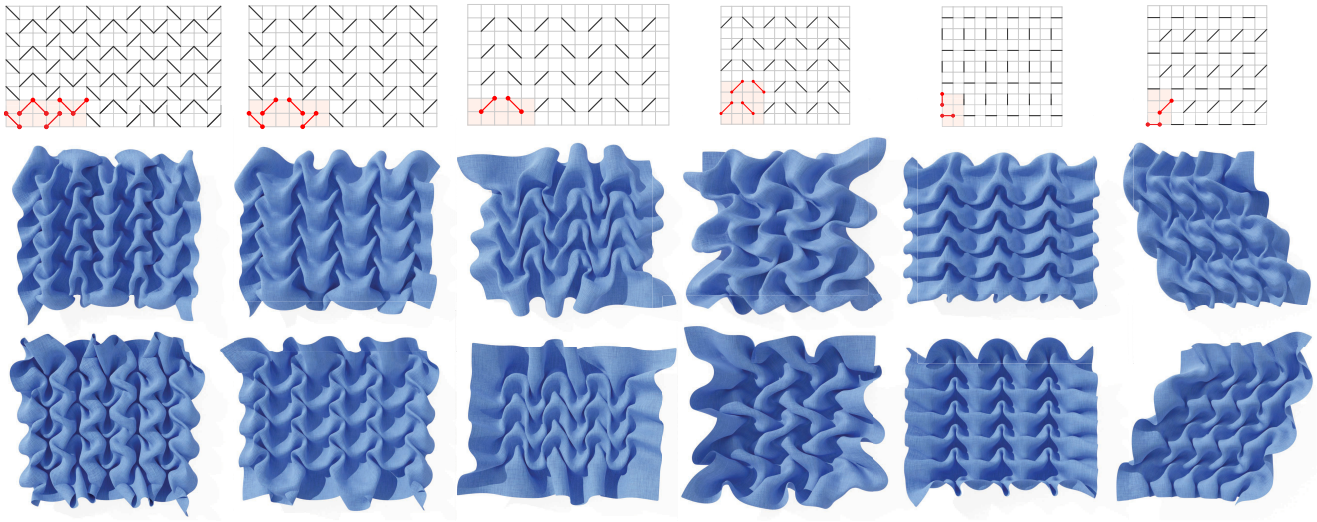


Fig. 9. For different smocking patterns (top row), we show the computed smocking designs' front side (middle row) and back side (bottom row).

coordinates, then there is no risk of the fabric tearing. Note that such a valid embedding always exists, since we can simply put all vertices in one point, and all constraints are satisfied in this trivial solution.

4.2.2 Maximizing Embedding Energy. While the distance constraints determine the *search space* of valid embeddings, we need an objective function to find a desirable embedding and avoid the trivial solution where all nodes get assigned the same location. We wish to encourage all nodes to stay as far from each other as possible. Let $\mathbf{x}_i \in \mathbb{R}^3$ be the embedded position of vertex $v_i \in \mathcal{V}_S$, and $\mathbf{X}$ the stacking of all these positions. We can formulate the following optimization problem for the embedding of $\mathcal{S}$:

$$\begin{aligned} \max_{\mathbf{X}\in\mathbb{R}^{|\mathcal{V}_S|\times 3}} \quad & \sum_{\forall i\neq j} \|\mathbf{x}_i - \mathbf{x}_j\|_2 \\ \text{s.t.} \quad & \|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq d_{i,j} \quad \forall i \neq j. \end{aligned} \tag{5}$$

Despite the simple formulation, this is a difficult, non-convex problem with $\frac{1}{2}n(n-1)$ hard non-convex inequality constraints defined on every pair of $n = |\mathcal{V}_S|$ vertices in the smocked graph.

4.2.3 Simpler Formulation as Graph Embedding. Our solution is to relax the optimization problem in Equation (5) into an easier to solve form, where the inequality constraints are replaced by a significantly smaller set of (possibly soft) equality constraints, leading to a classical graph embedding problem. For simplicity, here we assume the smocking pattern $\mathcal{P}$ is such that the underlay graph $\mathcal{S}_u$, as well as the pleat graph $\mathcal{S}_p$, is non-empty and has exactly one single connected component. We discuss other cases in Section 4.4. Moreover, we are particularly interested in *well-constrained* smocking patterns that produce pleasant patterns when fabricated. These patterns have balanced and structured underlay region, such that the pleats are constrained to be regular. See Section 5 for further discussion.

We observe that the pleats that form the geometric textures are constrained by the underlay (see Figure 8), while the underlay graph encodes the overall structural information and determines the final appearance. The distance bounds in Equation (4) hint that the local geometry around the stitching lines gets significantly changed by smocking, since they are pinched together. The underlay nodes are heavily constrained by each other and determine the overall smocking structure. However, the pleat nodes have more freedom to move in 3D, since they are not stitched to any other points on the fabric, and they are expected to form the volumetric 3D textures. This inspires us to split the embedding problem into two sub-problems: the embedding of the underlay and the pleat graphs in two separate steps, where the embedding of the underlay is employed to constrain the embedding of the pleat nodes.

4.2.4 Embedding the Underlay Graph. We first try to find the embedding $\mathbf{X}_u$ for the underlay graph $\mathcal{S}_u = (\mathcal{V}_u, \mathcal{E}_u)$. We observe that for a well-constrained smocking pattern, the underlay graph is *planar* (see the pink subgraph in Figure 7 (right)) and therefore can be embedded in 2D. The maximizing embedding energy encourages large distances between nodes, while the distance constraints bound them by $d_{i,j}$, so we propose to find the 2D embedding of the underlay graph by relaxing Equation (5) as

$$\min_{\mathbf{X}\in\mathbb{R}^{|\mathcal{V}_u|\times 2}} \quad \sum_{(v_i,v_j)\in\mathcal{E}_u} \left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2 - d_{i,j}\right)^2. \tag{6}$$

The relaxation is justified by the fact that in reality the distance constraints are not as stringent, as even the stiffest fabric can stretch a bit. Instead of considering every pair of underlay nodes, here we only consider the adjacent ones. Recall that the distance constraint $d_{i,j}$ is derived from the smocking pattern $\mathcal{P}$, which represents a flat fabric, a connected 2-manifold. Thus it is reasonable to only consider the constraints in local neighborhoods; the constraints beyond 1-ring neighbors should fall in line due to the metric structure and therefore can be ignored. See Appendix A for more detailed discussions.

4.2.5 Embedding the Pleat Graph. We find the 3D embedding $\mathbf{X}_p$ for the pleat nodes $\mathcal{V}_p$ in the pleat graph $\mathcal{S}_p$ using a similar

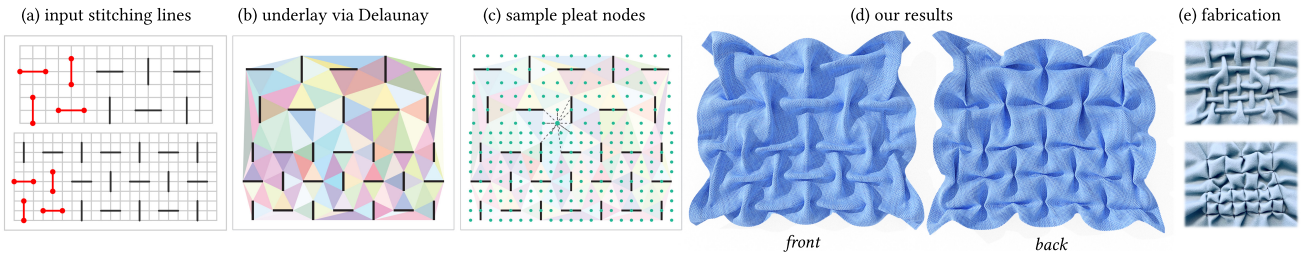


Fig. 11. Grid-free smocking design. (a) We combine two smocking patterns of different scales. It is challenging to design a single regular grid to accommodate all stitching lines simultaneously. (b) Instead of constructing a new grid, we compute a Delaunay triangulation conditioned on the input stitching lines, which gives us the underlay graph. (c) We then sample pleat nodes w.r.t. the input stitching lines (green dots) and add connectivity between the pleat and the underlay nodes via local Delaunay triangulation (see the dashed lines for some examples), which gives us the pleat graph. (d) We can then embed the smocked graph and solve for the smocking design. (e) Physical fabrication of the pattern in (a).

formulation:

$$\min_{\mathbf{X}\in\mathbb{R}^{|\mathcal{V}_p|\times 3}} \sum_{(v_i,v_j)\in\mathcal{E}_p} \left(\left\|\mathbf{x}_i-\mathbf{x}_j\right\|_2 - d_{i,j}\right)^2, \tag{7}$$

where we want to stretch each pleat edge to its upper bound $d_{i,j}$ to maximally spread the overall embedding. Recall that some of the pleat edges in $\mathcal{E}_p$ connect a pleat node and an underlay node. For these edges, the underlay nodes are fixed to the previously solved positions $\mathbf{X}_u$, and only the positions of the pleat nodes are involved in the optimization step in Equation (7). The pleat vertices are initialized with the same height to help break the symmetry ambiguity.

4.3 Smocking Design from Embedded Smocked Graph

Having solved for the embedding $\mathbf{X}_u \cup \mathbf{X}_p$ of the smocked graph $\mathcal{S}$, we immediately deduce the geometry of the coarse smocking pattern $\mathcal{P}$: All vertices in a stitching line ℓ_i have the same location as the embedded position in $\mathbf{X}_u$ of the respective underlay vertex v_{ℓ_i}, and all remaining (pleat) vertices in $\mathcal{P}$ have their locations in $\mathbf{X}_p$, corresponding to the vertices in the pleat graph. To compute the smocking design in finer resolution, we run ARAP on the high-resolution smocking pattern $\widetilde{\mathcal{P}}$, constraining the positions of the vertices of $\mathcal{P}$ to their embedded locations. Figure 9 shows further results on several interesting smocking patterns.

4.4 Generalizations: Non-regular Smocking Patterns

4.4.1 Grid-free Smocking Design. We can further generalize our algorithm to more challenging cases where the stitching lines are distributed non-uniformly, making it hard to extract a regular grid to abstract the smocking pattern. In this case, we can construct a graph from the input stitching lines based on Delaunay triangulation [Delaunay et al. 1934; Lee and Schachter 1980] and use it to compute the smocking design as before. See Figure 11 for an overview and Algorithm 1 in Appendix C for further details.

4.4.2 Honeycomb Grid. Our formulation does not depend on the exact shape of the grid; we just need to construct the graph $\mathcal{G}$ of the grid, so we can easily apply our algorithm to different types of grids. See Figure 12 for an example, where the smocking pattern is defined on a hexagonal grid, and the unit pattern (in red) is tiled in a cyclic fashion.

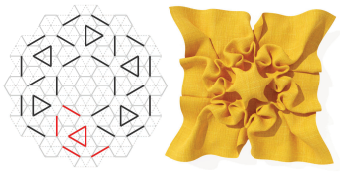


Fig. 12. Honeycomb grid.

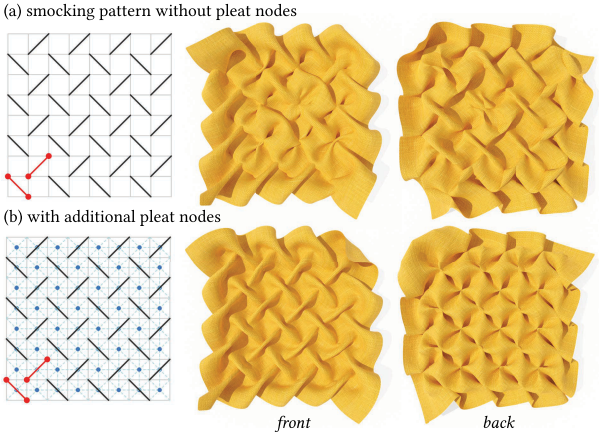


Fig. 13. For a smocking pattern that does not have any pleat nodes (except some free boundary nodes) as shown in (a), our algorithm can still produce a reasonable result, but the geometric texture features are less pleasing, since no constraints on the pleats are considered during the optimization. An easy fix is to insert additional pleat nodes (colored blue) and pleat edges (dashed lines), leading to a more regular and realistic result.

4.4.3 Empty Pleat Graph. It is unlikely to have an empty underlay graph, unless the set of stitching lines is empty ($\mathcal{V}_u = \emptyset$), or the smocking pattern $\mathcal{P}$ is not coarse enough, such that the underlay nodes are not connected to each other ($\mathcal{E}_u = \emptyset$), which can be easily fixed by making $\mathcal{P}$ coarser (e.g., removing the grid lines that do not contain stitching points or using Delaunay triangulation, as discussed in Section 4.4.1, to find $\mathcal{E}_u$). However, it is possible to have stitching lines so densely defined that the pleat node set is empty (see Figure 13). In this case, we can insert additional pleat nodes to the smocking pattern and then apply our algorithm.

5 WELL-CONSTRAINED SMOCKING PATTERN

Most online tutorials discuss how to smock a pre-designed pattern without providing any heuristics on how to design a good pattern that leads to satisfactory textures in the first place. Here we discuss some observations made during our experiments. In general, the

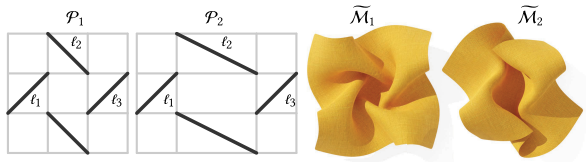


Fig. 14. We compare $\mathcal{P}_2$, an *under*-constrained smocking pattern, to $\mathcal{P}_1$, a well-constrained pattern. $\widetilde{M}_i$ visualizes our modeling result of $\mathcal{P}_i$.

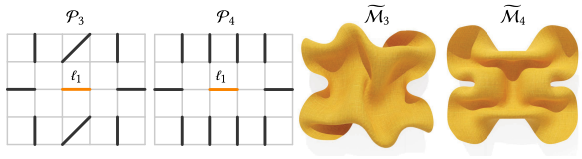


Fig. 15. We compare $\mathcal{P}_4$, an *over*-constrained smocking pattern to $\mathcal{P}_3$, a well-constrained pattern. $\widetilde{M}_i$ visualizes our modeling result of $\mathcal{P}_i$.

stitching lines of a good smocking pattern should yield an underlay graph that is well constrained: If the underlay graph is *underconstrained*, then it means that the smocked result is "loose," and its underlay nodes have excess freedom to move in the 2D plane during the embedding, which makes the pleats on top of them less deterministic. However, if the underlay graph is *overconstrained*, then it means we added too many equality constraints to some underlay nodes, making it impossible to embed the whole underlay graph in 2D. Embedding in 3D would introduce more degrees of freedom and make it harder to obtain regular, visually pleasing textures.

Underconstrained underlay. In Figure 14, we show two patterns $\mathcal{P}_1$ and $\mathcal{P}_2$, where we increase the width of the middle grid cells in $\mathcal{P}_2$ from 1 to 2. We now consider the embedding distance constraints (defined in Equation (4)) between the three stitching lines. For $\mathcal{P}_1$, we have $d_{1,2} = 1, d_{2,3} = 1, d_{1,3} = \sqrt{2}$. We can embed these three underlay nodes in 2D, forming a right triangle, and we call this underlay graph well constrained, since none of the underlay nodes can move locally (only rigid motion of the embedding as a whole is possible). However, the underlay graph of $\mathcal{P}_2$ is underconstrained. Specifically, we have $d_{1,2} = 1$, $d_{2,3} = 1$, and $d_{1,3} = \sqrt{5}$. According to triangle inequality, $\|\mathbf{x}_1 - \mathbf{x}_3\| \leq d_{1,2} + d_{2,3} = 2 < \sqrt{5} = d_{1,3}$. In other words, $d_{1,3}$ can be removed from Equation (5), since this inequality can never be violated. Therefore, during the embedding of the underlay graph, we would only consider $d_{1,2}$ and $d_{2,3}$, which allows the underlay nodes of ℓ_1, ℓ_3 to move around the node of ℓ_2.

Overconstrained underlay. $\mathcal{P}_4$ in Figure 15 shows an example of an overconstrained underlay graph. The underlay node that corresponds to ℓ_1 (colored in orange) is connected to 10 underlay nodes with maximum embedding distance 1 or $\sqrt{2}$. One can check that, when all the 10 neighboring underlay nodes are coplanar, it is impossible to embed the underlay node of ℓ_1 on the same plane such that the maximum embedding distance is reached. In this case, the underlay graph is overconstrained, and the embedding of the underlay by our method cannot achieve zero energy as defined in Equation (6). As a comparison, $\mathcal{P}_3$ shows a similar but well-constrained pattern.

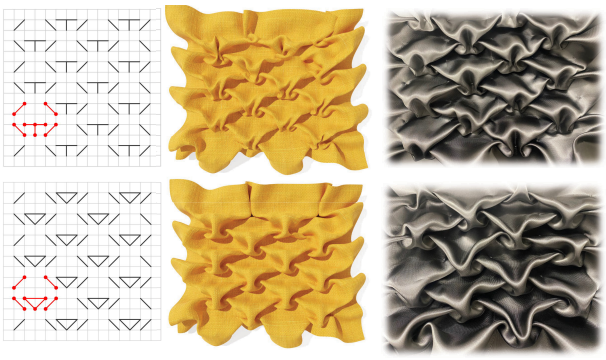


Fig. 16. We compare the smocking designs of two similar smocking patterns, where the top pattern has an extra stitching point on the longest stitching line compared to the bottom pattern.

Note that our optimization-based formulation works in both cases and produces reasonable smocked results, as shown in Figures 14 and 15. We observe that usually the well-constrained smocking patterns can produce more regular and visually pleasant textures. Based on these observations, we independently designed the patterns in Figure 9 (2th, 5th), Figure 12, Figure 16 (top), and Figure 17 (bottom).

6 RESULTS

We demonstrate that our algorithm can produce faithful results that match physical fabrication for different types of smocking patterns, as can be seen in the figures throughout the article and in the accompanying video. We also provide an interactive UI for smocking pattern exploration. The full implementation can be found at https://github.com/llorz/SmockingDesign.

6.1 Smocking Design

Folded smocking design. During the experiments, we observe that there are roughly two different styles of designing stitching lines. The first is *conflicting* stitching lines, where if extended, pairs of stitching lines would intersect with each other; such stitching lines create concave features after stitching (see, e.g., Figure 18). The second kind is *parallel* stitching lines, where after stitching, the in-between fabric is folded flatly, leading to less voluminous textures (see Figure 17). Our method can handle both cases.

Local modification. Our method is intuitive and predictable with respect to local changes of the *unit* smocking pattern. As shown in Figure 16, when we modify a stitching line, the final smocking design does not change drastically. Instead, the final results differ locally, as intuitively expected.

Irregular grids. Our method is not limited to uniform square grids. We can handle hexagonal grids (Figure 12), radial grids (Figure 18), combinations of grids (Figure 11) and other irregular grids (see Section 6.2).

Long stitching lines. Computing smocking designs with long stitching lines can be quite challenging. Stitching lines that stretch across multiple grid cells, as in Figure 19(a), (b), and (d), can potentially create large protruding features and allow the pleat nodes to

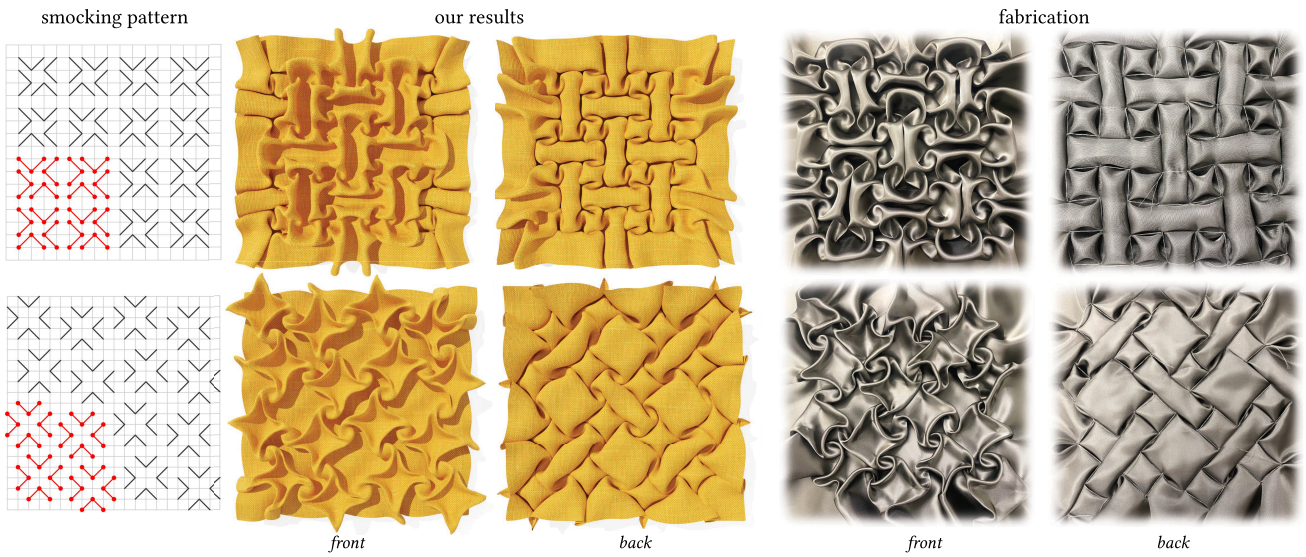


Fig. 17. Folded design. Our method can handle parallel stitching lines, which lead to folded pleats and sharper, less voluminous textures.

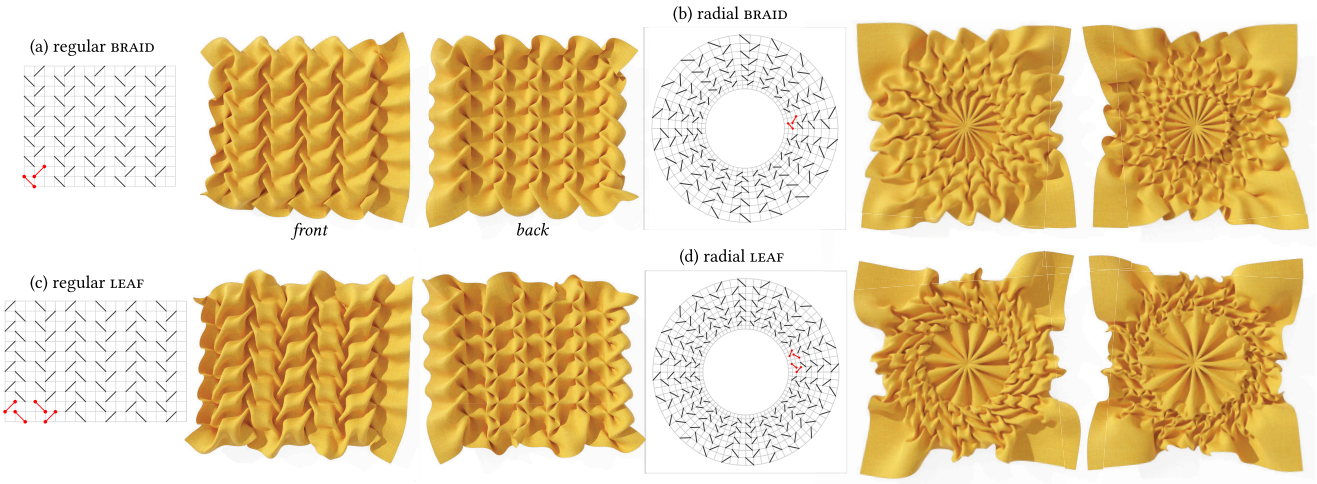


Fig. 18. Radial grids. We show the smocked shapes from regular (*left*) and radial (*right*) grids for the BRAID (*top*) and LEAF (*bottom*) patterns.

have more freedom to move during optimization. Stitching lines that connect multiple nodes in a single grid cell, as in Figures 19(c) and 16, can lead to complicated texture in a small region, which is difficult to model in general. Our method can produce reasonable and visually pleasing results in both cases.

6.2 Interactive UI

We integrate our method into Blender as an add-on with an interactive interface that allows users to design and modify smocking patterns, as well as visualize the computed smocking designs. Efrat et al. [2016] also provide a UI for smocking pattern design that allows users to tile five known patterns with different spacing or rotation. The tiled smocking pattern needs to be printed out for fabrication to see the resulting smocking design. In comparison, our UI is more flexible, it supports mesh-level modifications (see Figure 20) and allows the user to design stylish patterns by drawing stitching lines freely. Our method can also be used to explore variations of existing patterns. For example, in Figure 21 we show smocking designs with modified grids using our UI. Since the smocking design computation and visualization are integrated into the UI, it becomes much easier and more efficient for casual users to explore different patterns. To stress this point: It can take a few *hours* to smock a piece of physical fabric, including drawing grids, annotating stitching lines, and sewing every single stitching line with pleating and knotting of the threads. In contrast, our algorithm demonstrates the computed smocking design in *seconds*. As a proof of concept, we prototype smocked sleeve designs, as shown in Figure 1, by computing the smocking designs with extra margins, which leads to natural folds on the boundary. We then deform the smocked shape w.r.t. a hand model using the "bend"

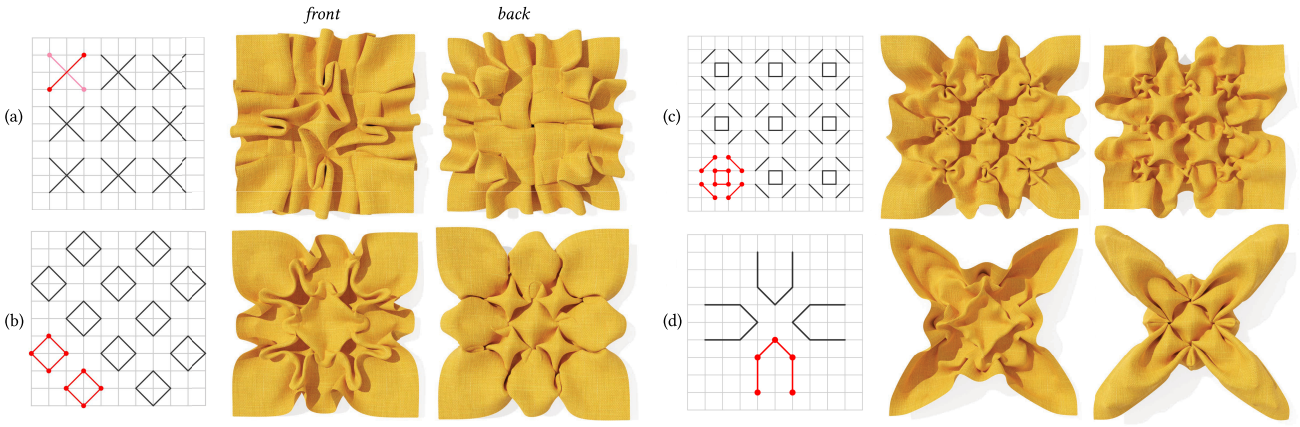


Fig. 19. Long stitching lines. We show examples of long stitching lines that cross multiple grid cells ((a), (b), and (d)) and connect many nodes in a small region (c). Note that the unit smocking pattern in (a) contains two separate stitching lines.

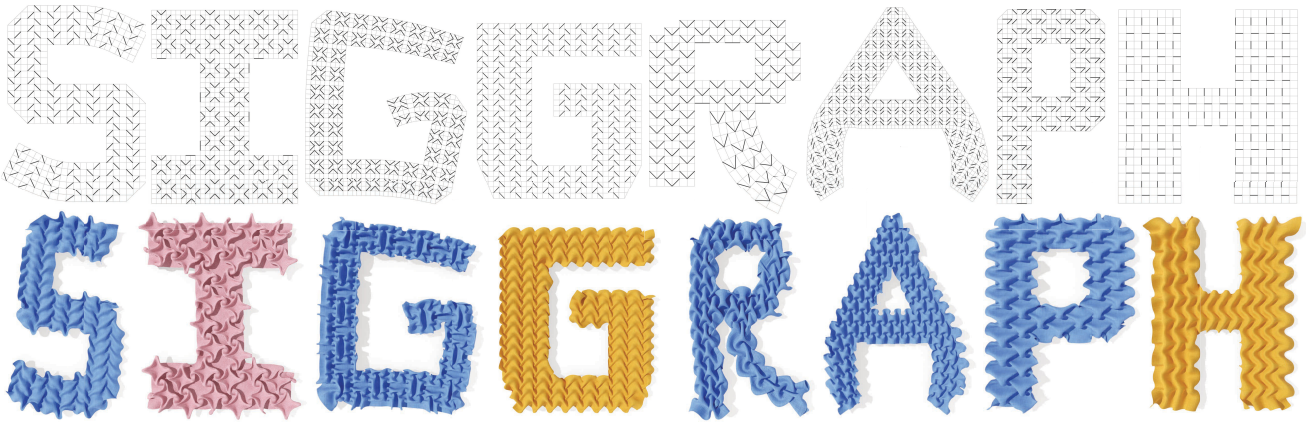


Fig. 20. We use our UI to edit different smocking patterns to decorate letters, including editing operators of cutting and warping grids, removing and adding stitching lines. We show the edited smocking pattern in the top row and the corresponding smocked results in the bottom row. Note that different smocking patterns shrink the fabric in different ratios.

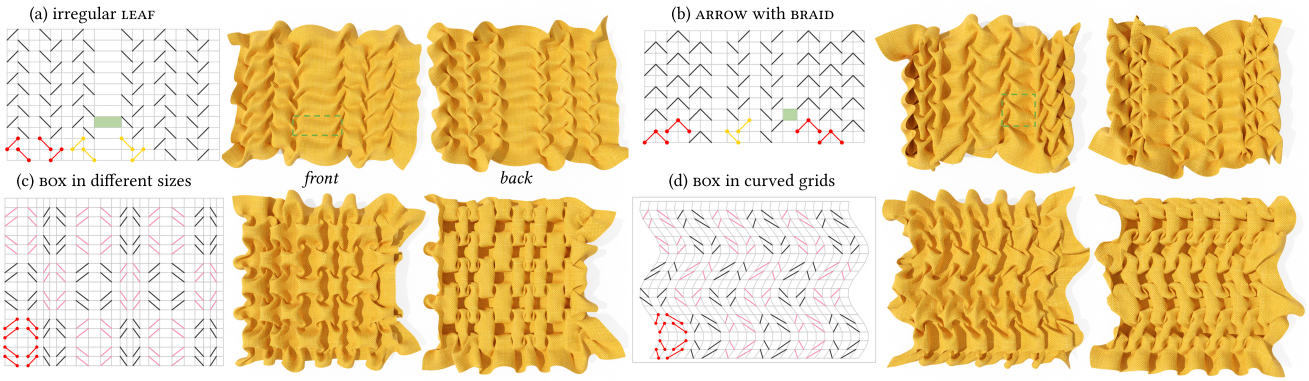


Fig. 21. Irregular grids. (a) We increase the space inside the BRAID pattern, as highlighted in green. (b) We mix the ARROW and the LEAF patterns, with the gap in-between highlighted in green. ((c) and (d)) We non-linearly deform the BOX pattern (adjacent unit patterns are colored in different colors for better visualization).

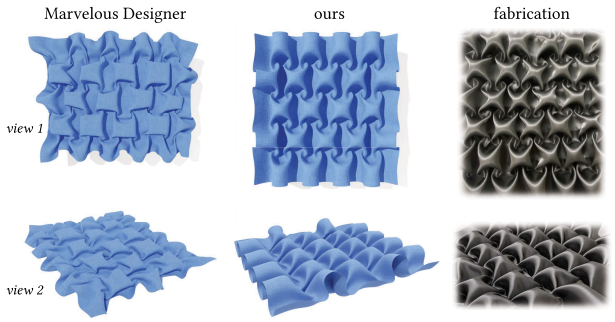


Fig. 22. Comparison to the commercial software [MarvelousDesigner 2023].

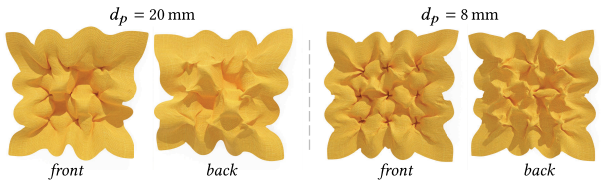


Fig. 23. Using Marvelous Designer to simulate the pattern in Figure 19 (c) with particle distance d_p set to 20 mm (left) and 8 mm (right).

function in Blender. These preliminary results suggest that our algorithm can be potentially used for digital garment design. See Appendix B and the supplementary video for a more detailed exposition of the functionalities of our UI.

6.3 Comparison to Baselines

In Figure 6 and the supplementary material, we show comparisons to the cloth simulator of the open source software Blender [Foundation and Community 2023]. In this section, we provide additional comparisons to the state-of-the-art cloth simulators, ArcSim [Narain et al. 2012] and C-IPC [Li et al. 2021], and the commercial software [MarvelousDesigner 2023], which is closed source. For all the comparisons to baselines, we use the fabric in the same resolution as ours. In particular, for ArcSim and C-IPC we additionally provide the *non-planar* initial configuration for the fabric, where all stitching points are *offset* out of the fabric plane in the same direction (see Figure 25(e)). If starting from a planar configuration, then ArcSim gets stuck in the first iteration, and C-IPC produces a cluttered result with irregular pleats.

Comparison to Marvelous Designer [2023]. To prepare the input for the commercial software **Marvelous Designer (MD)**, each stitching line in the fabric needs to be specified using the "tack" tool, which adds extra complexity for smocking simulation in MD. In Figure 22, we report the best result obtained from this software, where we experimented with different parameters such as stiffness, damping, pressure, sewing distance, and so on. Please see the supplementary materials for full simulations with different parameter settings. We observe that the "solidify" function, which is designed to maintain the desired draping state per pattern unit, is the key factor in helping Marvelous Designer achieve the expected box-like geometric features. However, the simulated smocking details

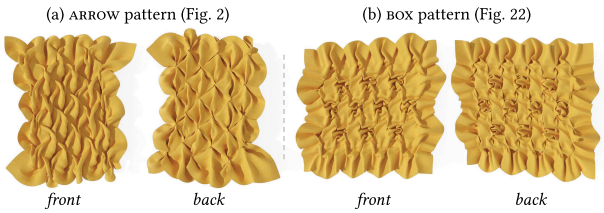


Fig. 24. Results of ArcSim. We provide two examples of using [Narain et al. 2012], starting from initial configurations with offsets to break the symmetry. It takes 3 min and 5 min to obtain the results for the ARROW and BOX pattern, respectively. Without a geometric prior, the simulated pleats are not voluminous and do not realistically reflect the physical fabrications.

in Marvelous Designer are less regular and do not match the physically fabricated result as well as our approach does. In Figure 23 we show the results of Marvelous Designer on a much more complicated pattern, depicted in Figure 19(c), containing long stitching lines. With the "solidify" function and high-enough resolution, Marvelous Designer struggles to produce meaningful results, while our method produces a faithful preview of the fine details of the smocking results.

Comparison to ArcSim. ArcSim [Narain et al. 2012] is a powerful method for simulating fine features, such as wrinkles and creases for cloth deformations. We adapt the more advanced implementation[1] of ArcSim [Sperl et al. 2020] for smocking, where the to-be-stitched vertex pairs are specified using the "glue" constraints. In Figure 24 we show the best results we attained in consultation with the authors of the method. We ran the simulation from the initial configuration where all stitching points are offset. We experimented with the parameters of repulsion thickness, collision stiffness and different fabric materials. We also disabled the "remeshing" option to preserve the glue constraints. We can see that the shrinking ratio of the smocked fabric is more accurate than Blender and Marvelous Designer. However, the lack of volume and realism in the simulated pleats suggests that this method may not be suitable for use as a direct preview tool for artists designing smocking patterns.

Comparison to C-IPC. Co-dimensional incremental potential contact (C-IPC) [Li et al. 2021] is a current state-of-the-art method for cloth simulation that can model thickness and handle collision and frictional contact. To run C-IPC,[2] we rescale the smocking pattern to its intended dimensions in centimeters and offset all stitching points to guide the simulation. In Figure 25, we show the best attained results on four examples in consultation with the authors of the method. We experimented with various values of bending, stretching, stitching force, timestep size, offset value, and so on. We also tried using both the static solver and the dynamic solver with various timesteps. When using the dynamic solver, we found that using a large timestep (e.g., $dt = 10$ s) can achieve much less wrinkled and more realistic results than the default timestep ($dt = 0.01$ s). The dynamic solver without collision handling is much more efficient than the static solver. However, finding a

[1] https://git.ista.ac.at/gsperl/ARCSim-HYLC
[2] https://github.com/ipc-sim/Codim-IPC

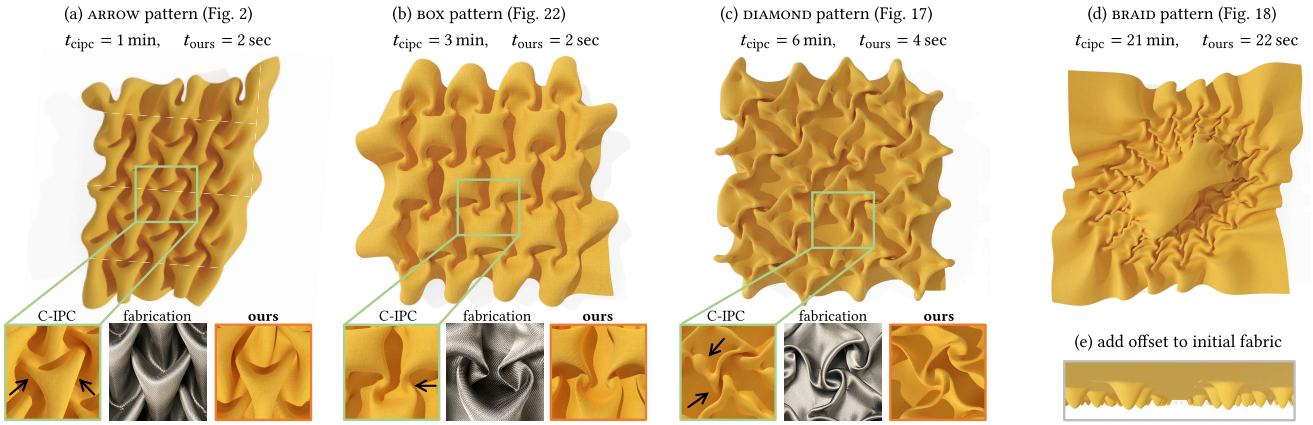


Fig. 25. Comparison to C-IPC. We show four smocked results ((a)–(d)) using C-IPC [Li et al. 2021], where all stitching points in the initial configuration are offset along the z-axis, as shown in (e), to obtain more regular results. In comparison, our method provides a more realistic preview of the smocked fabric in a much shorter time, which allows the artists to iterate the smocking design *interactively*. For example, for the pattern of (d), it takes our method 22 seconds, while C-IPC takes 21 minutes.

Table 1. Different Solutions to (Pre-)visualize a Smocking Pattern

Properties \ Solutions	Fabric.	Blender	MD	ArcSim	C-IPC	**Ours**
***Easy** to prepare input?*	✘	✔	✘	✔	✔	✔
***Easy** to use (fabricate)?*	✘	✔	✔	✘	✘	✔
*Are the pleats **accurate**?*	✔	✘	✘	✘	✔	✔
*Fabric shrinks **realistically**?*	✔	✘	✘	✔	✔	✔
***Efficient** for preview?*	✘	✔	✔	✔	✔	✔

MD stands for Marvelous Designer, and "fabric." stands for manual physical fabrication.

suitable equilibrium state for the dynamic solver is challenging. For example, running the dynamic solver until convergence, where the change of the vertex positions is smaller than a threshold while setting the vertex velocity to zero at each iteration, leads to a cluttered configuration. The best intermediate results are similar to the ones shown in Figure 25, where the static solver is used. Overall, C-IPC achieves better and more realistic results than the other baselines. However, the whole fabric gets sheared, and the geometric shape of the pleats is not as accurate as in our method. For example, as highlighted in Figure 25, the transition regions between the box shapes are wrong in example (b) and the bumps along the edges of the diamond shapes are unnatural in example (c). The method takes minutes to execute. Moreover, expertise in cloth simulation is needed to tune the parameters to obtain reasonable results, as the default values did not work out of the box.

Summary. In comparison, our method is much simpler to use, requiring no domain knowledge, and it is more efficient for previewing purposes, taking only a few seconds. This enables interactive design iterations for artists. See Table 1 for a comparison summary.

6.4 Ablation Study and Justifications

Geometric appearance. In this work, we aim to preview the shape of a smocked pattern solely based on its geometric features, disregarding the impact of different fabric materials. Indeed, the final outcome of smocking can be influenced by the type of

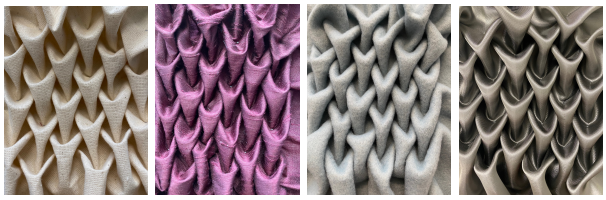


Fig. 26. Fabricating the ARROW pattern using different fabric materials including canvas, polyester (crisp, thin), polyester (soft, thick), and satin, from left to right respectively.

fabric used, which may possess varying levels of stretchiness. However, as evidenced by the multitude of examples available online, the *geometric appearance* of a smocked pattern remains very similar regardless of the fabric used, including our experiments with canvas, satin, and polyester (see Figure 26), as well as numerous examples found on YouTube and Pinterest featuring silk, leather, wool, cotton, lace, denim fabrics, and so on. It is, in fact, the stitch structure, not so much the specific material, that ultimately determines the geometric structure of the pattern. Therefore, it is reasonable to model the geometric appearance for preview purposes and delegate the material-dependent characteristics, such as bending stiffness, to cloth simulators.

Smocked graph. The key component of our method is the formulation of the *smocked graph*, extracted from the smocking pattern, which explicitly encodes the modified geometry after stitching (as discussed in Section 4.2.1). To check that our embedded smocked graph is indeed critical for the successful computation of the smocking design, we try applying ARAP on

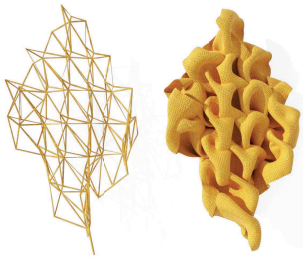


Fig. 27. Coarse-to-fine ARAP.

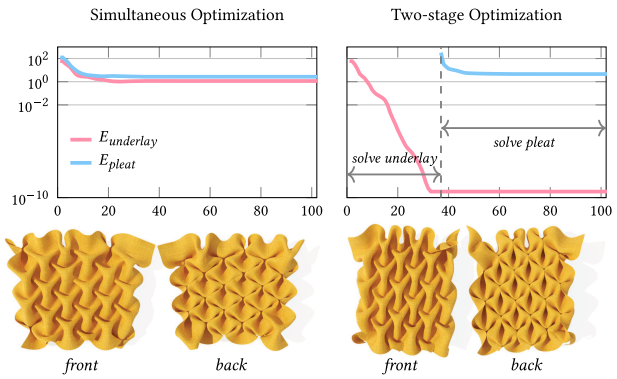


Fig. 29. Ablation on solver. We compare our two-stage optimization scheme to a simultaneous setting, where the underlay and the pleat graph are embedded at the same time. We show the energies over iterations on the top and the final results on the bottom. The spatial energy distributions are visualized in Figure 35.

the coarse smocking pattern $\mathcal{P}$ instead of utilizing the smocked graph. We optimize Equation (1) on the coarse smocking pattern $\mathcal{P}$ and use the result to deform the finer fabric discretization $\widetilde{\mathcal{P}}$, as discussed in Section 4.3. Figure 27 shows the resulting $\mathcal{P}$ and $\widetilde{\mathcal{P}}$. We can see that the result is more regular than applying ARAP to the fine grid $\widetilde{\mathcal{P}}$ directly (cf. Figure 5). However, the overall geometric texture is still not as well structured as ours. The reason is that the pleat vertices have too many degrees of freedom and are not sufficiently regularized in this approach, whereas our smocked graph encodes the global structural information and firmly sets the relationship between the underlay and the pleat nodes, yielding more regular results.

Pleat graph embedding. Our method embeds both the underlay graph and the pleat graph to guide the fabric deformation. To justify that the pleat graph embedding is indeed helpful, we try using only the optimized embedding of the *underlay* graph to guide the ARAP deformation, see the left part of Figure 28. For completeness, we also show the result of only using the optimized embedding of the *pleat* graph to guide the deformation on the right of Figure 28. We can see that the optimized embedding of the underlay graph can help to guide the deformation to achieve a less cluttered result compared to the other ARAP-baselines. However, without guidance from the pleat graph to reduce the search space, the pleats exhibit inconsistent orientations and irregular shapes, resulting in an unpleasant (but still feasible) preview. This ablation justifies that both the underlay and the pleat graphs contribute to form regular and faithful appearance of the geometric texture.

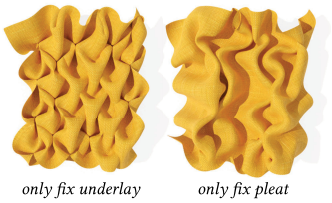


Fig. 28. Ablation on pleat graph.

Two-stage optimization. It is natural to have a two-stage optimization of embedding the underlay and pleat graph separately, since the pleats are induced by the fixed underlay graph. To further justify it, we compare to the setting where the underlay and pleat graphs are solved simultaneously by minimizing the sum of energies in Equation (6) and Equation (7). We show the corresponding energy over iterations in Figure 29. We can see that the simultaneous optimization still produces reasonable results, since the proposed distance constraints $d_{i,j}$ properly encode the modified local structure after smocking. However, our two-stage optimization leads to a more faithful result w.r.t. the real fabrications shown in Figure 5(d), in 3 times shorter computation time due to the smaller number of variables to optimize in each stage. More specifically, in our two-stage optimization process, we first focus on embedding the underlay graph accurately, then use it to constrain and embed the pleat graph. Solving the two embeddings simultaneously is more likely to land in an undesirable local minimum.

Table 2. Smocking Pattern Complexity and Modeling Runtime

smocking pattern	*smocked graph complexity* $\lvert\mathcal{V}_u\rvert$	$\lvert\mathcal{E}_u\rvert$	$\lvert\mathcal{V}_p\rvert$	$\lvert\mathcal{E}_p\rvert$	$\lvert\mathcal{V}_{\widetilde{\mathcal{P}}}\rvert$	*optimization (sec.)* $\mathcal{S}_u$	$\mathcal{S}_p$	$\widetilde{\mathcal{P}}$
Figure 2	24	53	45	186	5,074	0.0015	0.130	1.920
Figure 12	30	66	121	360	8,613	0.0012	1.161	3.370
Figure 17(a)	64	210	97	382	12,769	0.0017	0.714	3.768
Figure 17(b)	64	98	249	1,038	19,321	0.0014	1.972	4.088
Figure 16(a)	49	106	130	537	14,994	0.0016	0.852	3.269
Figure 16(b)	49	106	144	621	11,236	0.0014	0.836	3.215
Figure 18(a)	60	149	88	418	9,116	0.0015	0.402	3.044
Figure 18(b)	144	353	262	1,222	67,600	0.0023	2.705	21.25
Figure 18(c)	72	153	103	525	10,836	0.0007	0.364	2.882
Figure 18(d)	192	392	346	1705	81,796	0.0076	7.527	25.89

We report the topology of the smocked graph, including the number of underlay/pleat vertices and edges, and the resolution of the fine grid $\lvert\mathcal{V}_{\widetilde{\mathcal{P}}}\rvert$. The runtimes of embedding the underlay graph $\mathcal{S}_u$, the pleat graph $\mathcal{S}_p$, and solving for the full smocking design $\widetilde{\mathcal{P}}$ are reported in seconds.

6.5 Implementation

Implementation and runtime. We design the GUI as an addon in Blender and implement our algorithm in Python where the projected Newton solver is used for optimization. Recall that our algorithm has three main steps: (1) embedding the underlay graph $\mathcal{S}_u$, (2) embedding the pleat graph $\mathcal{S}_p$ with fixed underlay graph, and (3) solving for the smocking design on a finer grid $\widetilde{\mathcal{P}}$ based on the embedded smocked graph. In Table 2, we report the runtime of each step of our method on multiple smocking patterns with different complexities. Note that the number of stitching lines equals to the number of underlay nodes, $\lvert\mathcal{L}\rvert = \lvert\mathcal{V}_u\rvert$, so we do not report it separately in the table. Our method takes a few seconds on the medium-sized smocking patterns, and up to half a minute on the large ones. As a comparison, it usually takes up to a few *hours* to smock a pattern, including drawing the grid on the fabric, annotating all the stitching lines and sewing them. Sewing and making knots for all the stitching points are the most time-consuming parts of the process. Usually it takes about 2 to 3 minutes to finish a *single* stitching line for an experienced maker. We can see that our method is more efficient, convenient, and error tolerant.

Regularizers. When optimizing for the embedding of the pleat graph as discussed in Section 4.2.5, we can add extra regularizers

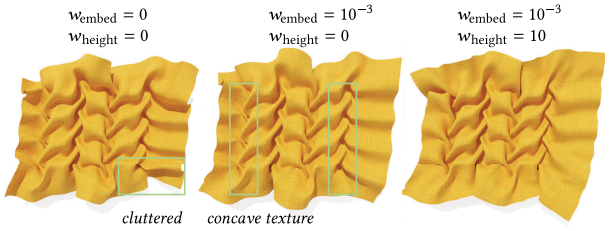


Fig. 30. The LEAF pattern with different parameters in Equation (8).

to make the pleats more regular,

$$\min_{\mathbf{X}\in\mathbb{R}^{|\mathcal{V}_p|\times 3}} \quad \sum_{(v_i,v_j)\in\mathcal{E}_p} \left(\left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2 - d_{i,j}\right)^2 - w_{\text{embed}} \sum_{\forall i\neq j} \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2 + w_{\text{height}}\,\text{Var}[h], \tag{8}$$

where $\text{Var}[h]$ is the variance of the heights (the z coordinates) of the pleat nodes. Here we add the maximizing embedding energy with a negative sign to make the underconstrained pleat nodes (e.g., boundary pleats) stay away from each other. We also encourage the geometric texture to keep a uniform height distribution by penalizing its variance. In Figure 30 we show a simple ablation study. We can see that adding the maximizing embedding term leads to a less cluttered boundary. Meanwhile, adding the pleat height regularizer can push the concave pleats (with negative height) upwards to form a more regular pattern. We observe that even without these regularizers our method produces good results away from the fabric boundary, and we use very small weights $w_{\text{embed}} = w_{\text{height}} = 10^{-3}$ to make the boundary pleats more attractive.

Initialization and parameters. We initialize each underlay node v_{ℓ_i} by the average position on the flat fabric of all the stitching points in ℓ_i, with zero height. We initialize each pleat node by its original position on the fabric, with the initial height set to 1. The weights w_{embed} and w_{height} in Equation (8) are both set to 10^{-3} for all the experiments. We run the embedding optimization until convergence. The underlay embedding energy at convergence is smaller than 10^{-8} for a well-constrained pattern.

7 CONCLUSION, LIMITATIONS, AND FUTURE WORK

In this article, we discuss how to mathematically formulate Canadian smocking, a decorative and practically beneficial surface embroidery technique. We introduce a simple yet effective method that solves for the smocking design with 3D geometric textures based on an input smocking pattern, which is represented by a set of stitching lines drawn on top of the fabric. We first extract the smocked graph from the input pattern, where the points in the same stitching lines are merged into a single vertex, and the degenerated or redundant edges are removed. This smocked graph encodes the geometric features of the final smocking design. To obtain the smocking design, we first embed the smocked graph in 3D, where we embed the underlay graph and the pleat graph in two steps. We then use the embedded smocked graph to guide the deformation of the fabric represented via a finer grid using ARAP. Our method is efficient and accurate, and our computed smocking designs are very similar to real fabrications for a large set of patterns, which allows us to design a user interface for smocking design exploration.

In this work, we formulate smocking as a pure shape modeling problem without considering cloth dynamics and collision response. Though the self-intersections do not significantly affect the visual appearance of the digital smocking design, it would be interesting to take them into consideration during modeling. We also wish to investigate smocking from the perspective of cloth simulation, as we can see that the state-of-the-art simulators cannot tackle this problem directly. We leave it for future work to investigate how to use our computed results as an initial guess for cloth simulators while incorporating additional material-dependent parameters, such as bending stiffness, to generate more realistic results at fine scales. Another limitation of our current approach is that we do not fully explore all possible smocking designs from an input pattern. For some complicated smocking patterns (such as Figure 19(d)), multiple visually appealing local minima (i.e., multiple final smocking designs) are possible. These results can guide the user or designer to iron or steam the smocked fabric into different shapes. We leave this shape space exploration as future work. Another interesting direction is to investigate the inverse problem of smocking, i.e., finding the arrangement of stitching lines such that the final result is close to an expected 3D texture or shape. Since smocking is a popular embroidery technique used by high-end fashion designers, we wish to explore smocking design directly on surfaces in 3D, so that it could be integrated with garment design. In our experiments, we notice that the smocked shapes can serve as meta-materials, since the pleats on top of the rigid underlay create extra thickness and elastic cushioning. In future work, it would be interesting to design smocking patterns with particular physical properties.

APPENDICES

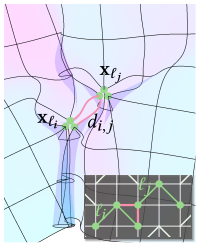
A SMOCKED GRAPH EMBEDDING

In this section, we discuss the intuition behind our relaxation of the problem defined in Equation (5) for the embedding of the smocked graph, presented in Sections 4.2.4 and 4.2.5.

Embedding the underlay. We embed the underlay graph in 2D by fully stretching the underlay edges to their upper bound $d_{i,j}$ as defined in Equation (4). Intuitively, gathering the underlay nodes in a stitching line is equivalent to translating all these nodes in the xy-plane (the initial fabric plane) to the same position. After smocking (translation), the underlay nodes still stay on the xy-plane, i.e., they remain co-planar. This co-planarity property allows us to solve the embedding of the underlay in 2D, which significantly reduces the search space. Indeed, we observe that the underlay regions of the fabricated results are co-planar, which validates our 2D search space.

Our reformulation is based on the fact that the optimum to Equation (5) is at the *exact* boundary of some inequality constraints. We can picture the inequality constraints in Equation (5) as follows: imagine we have a set of tiny balls or beads placed on the xy-plane, and each bead represents an underlay node. For each pair of beads, e.g., the beads that represent the i, jth underlay node, we use a string with length $d_{i,j}$ to connect the two

beads (see the inset figure). We then move the beads on the plane such that they are far from each other and none of the strings are broken. We can imagine that at the optimal situation, some of the strings become *tight*, which means if we move the attached beads any further, then these strings will break. At the same time, there are other strings that stay *loose*, which means their extreme can never be reached and they are useless in constraining the beads. In other words, if we simply remove the loose strings, and do the above experiment again, then we will end up with the same configuration (up to translation and rotation). Moreover, the short strings are more likely to be tight at the optimum compared to the long strings. Since the underlay graph is planar and the $d_{i,j}$ values are derived from the fabric, which is a connected and presumably inextensible 2-manifold, we can conclude that the underlay edges (e.g., pink edges/strings highlighted in the inset figure) become tight at the optimum. Therefore, in Equation (6) we only consider the underlay nodes that are connected to each other.



Embedding the pleats. We embed the pleat nodes in 3D on top of the solved underlay graph. Our reformulation in Equation (7) is based on the observation that the 3D embedding of a pleat node is only constrained by the embeddings of the *neighboring* nodes. Specifically, the embedding distance constraint between a pleat node and a faraway node can be ignored. The intuition behind this is that the distance constraint between a pleat node v_1 and a faraway node v_2 is satisfied automatically according to the triangle inequality if the distance constraint between v_2 and a neighboring node v_3, and the distance constraint between v_3 and v_1 are satisfied. Thus, we only need to consider the pleat nodes and their neighboring nodes, i.e., the node pairs in the pleat edge set $\mathcal{E}_p$.

B INTERACTIVE USER INTERFACE

We implement an interactive user interface in Blender as an add-on (see Figure 31) including the following functionalities:

- *define a unit smocking pattern* by creating a 2D grid and drawing stitching lines
- *define a full smocking pattern* by
  - tiling the loaded unit smocking pattern (with user-defined repetition and shift of the unit pattern)
  - drawing stitching lines directly on a square or hexagonal grid to define the full pattern
- *modify a full smocking pattern* by
  - deforming the square grid into a *radial* grid (with user-defined radius)
  - adding margins to the pattern
  - combining it with another smocking pattern (along user-specified axis and space)
  - deleting/adding stitching lines from/to the pattern
- *simulate the smocked pattern* with intermediate steps including
  - extracting the smocked graph from the pattern
  - embedding the underlay and pleat subgraphs of the smocked graph
  - applying ARAP to compute the smocking design
- *render the smocking design*
- *run cloth simulator* implemented in Blender on the fine-resolution smocking pattern

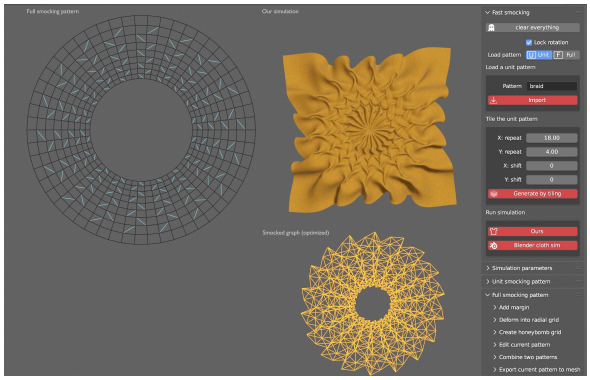


Fig. 31. We introduce an interactive user interface for smocking design, implemented in Blender as an add-on.

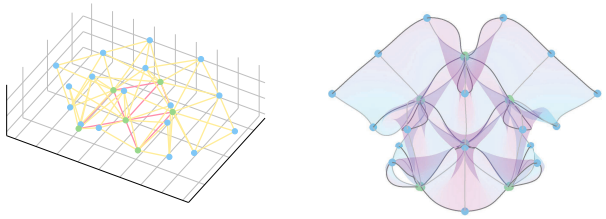
ALGORITHM 1: Smocking Pattern from Stitching Lines

Input : A set of stitching lines $\mathcal{L} = \{\ell_i\}$
Output: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to complete the smocking pattern
$\mathcal{V} \leftarrow \{v \in \ell_i \,|\, \forall \ell_i \in \mathcal{L}\}$
// create underlay edges
$\mathcal{E} \leftarrow$ DelaunayTriangulation $(\mathcal{V})$
regularly sample a set of pleat nodes $\mathcal{V}'$ inside the bounding box of $\mathcal{L}$
foreach $v \in \mathcal{V}'$ **do**
  // create pleat edges for v
  $\mathcal{E}' \leftarrow$ DelaunayTriangulation $(\mathcal{V} \bigcup \{v\})$
  $\mathcal{E} \leftarrow \mathcal{E} \bigcup \{e \in \mathcal{E}' \,|\, v \in e\}$
end
$\mathcal{E}' \leftarrow$ DelaunayTriangulation $(\mathcal{V}')$
$\mathcal{E} \leftarrow \mathcal{E} \bigcup \mathcal{E}'$
$\mathcal{V} \leftarrow \mathcal{V} \bigcup \mathcal{V}'$

Please see the supplementary videos for the real-time demonstrations of our UI. The Blender add-on can be found at https://github.com/llorz/SmockingDesign/python_implementation.

C ALGORITHMIC DETAILS

Algorithm 1 shows the pseudo-code for grid-free smocking design discussed in Section 4.4.1, which gives instructions on how to construct a graph from input stitching lines without given grids. Specifically, given a set of stitching lines as input, we first extract the underlay nodes $\mathcal{V}$ from the endpoints of the stitching lines. We can then construct the underlay graph by computing a Delaunay triangulation [Delaunay et al. 1934; Lee and Schachter 1980] conditioned on the input stitching lines. We then sample a set of pleat nodes and construct the pleat graph. As discussed in Section 4 and demonstrated by Figure 8, the pleat region pops up from the base layer (underlay region) to form the texture. We therefore focus on connecting the sampled pleat nodes to the underlay graph to construct the pleat graph. For each sampled pleat node v, we



(a) A 3D embedding of $\mathcal{S}$ (b) Smocking design

Fig. 32. Embedding of the coarser (left) and the finer (right) discretization of the smocking pattern shown in Figure 7.

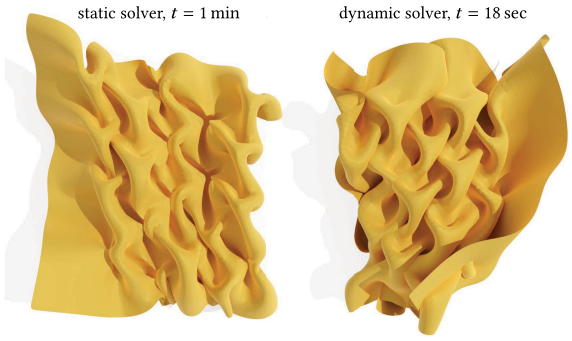


Fig. 33. C-IPC on ARROW pattern *without* collision handling.

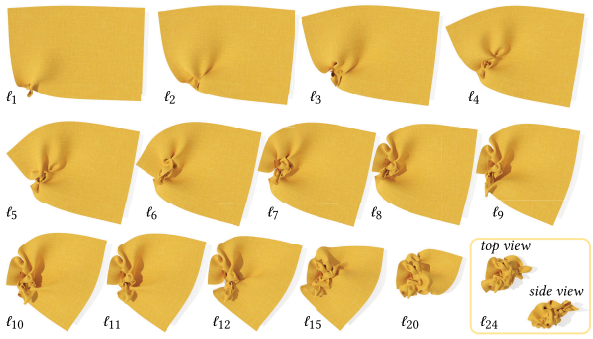


Fig. 34. Step-by-step ARAP. We run ARAP to sew each stitching line ℓ_i, $i = 1, \ldots, 24$ in the smocking pattern shown in Figure 2. Note that all visualizations are in a consistent scale.

compute the Delaunay triangulation again on the underlay nodes and this pleat node, i.e., $\mathcal{V} \bigcup \{v\}$, from which we can extract the pleat edges between v and the neighboring underlay vertices. We can additionally connect the pleat nodes to the neighboring pleat nodes by Delaunay triangulating all the pleat nodes only. In this way, we can construct the underlay graph and the pleat graph for smocking design computation from stitching lines alone. One important observation is that the pleat nodes need to be evenly sampled w.r.t. the input stitching lines. For example, one can take the middle points of the stitching lines as the pleat nodes, as shown in Figure 13. In this way, the regularity encoded in the input stitching lines is kept during the pleat graph construction, and therefore leads to desirable simulated results.

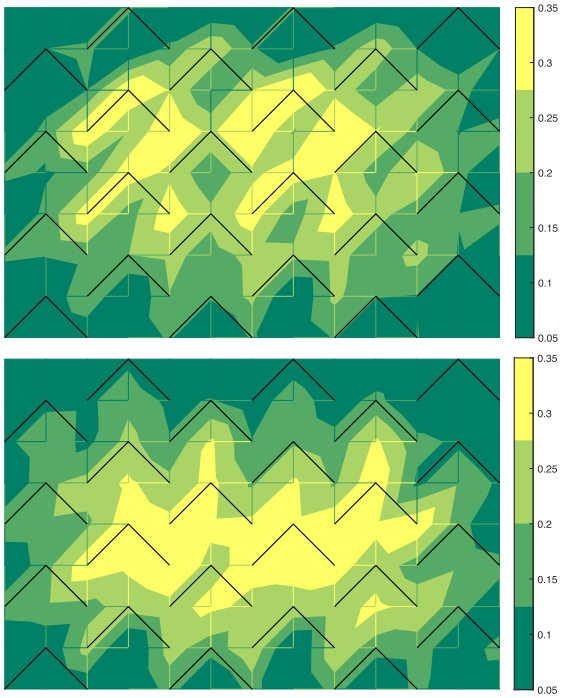


Fig. 35. Optimized energy distribution after simultaneous optimization (*top*) and our two-stage optimization (*bottom*), as discussed in Figure 29. We also draw the stitching lines in black.

D ADDITIONAL RESULTS

C-IPC without collision. Figure 33 shows the results of C-IPC [Li et al. 2021] on the ARROW pattern without collision handling. Disabling self-collision handling for the static solver does not improve computational efficiency; instead, it leads to worse results compared to Figure 25. Using the dynamic solver without self-collision handling results in faster iterations. However, running the dynamic solver until convergence leads to a cluttered configuration. Here, we select an intermediate iteration where the pleats are sufficiently formed and the overall shape starts to show signs of clumping.

Step-by-step ARAP. Another straightforward solution to model smocking is by applying ARAP to each stitching line separately to mimic the manufacturing process. In Figure 34, we adopt this strategy to model the smocking pattern depicted in Figure 2, following a left-to-right and bottom-to-top sequence of stitching. However, we observe that this approach causes the fabric to bend inward instead of shrinking towards the center as it happens during actual fabrication (illustrated in Figure 5(d)). As a result, the final outcome exhibits a messy appearance with extremely cluttered pleats with many self-intersections, as shown in Figure 34 of the final result when viewed from the side. As a comparison, our progressive ARAP simultaneously stitches all the stitching lines, ensuring more uniform "stitching forces" to promote fabric shrinkage in more accurate directions. This results in a superior baseline compared to the step-by-step ARAP approach.

Optimized energy distributions. In Figure 35, we visualize the energy (the sum of Equation (6) and Equation (7)) of the computed smocked design shown in Figure 29 after optimization using simultaneous solver and our two-stage solver. For easier visual comparison, we visualize the per-vertex errors on the original smocking pattern. We can see that the result from simultaneous optimization shows more prominent error in the underlay region (edges that connect two different stitching lines), while the result of the two-stage optimization has a smoother error distribution in the pleat region.

ACKNOWLEDGMENTS

The authors express gratitude to the anonymous reviewers for their valuable feedback. Special thanks to Minchen Li for his help with the comparison to C-IPC, Georg Sperl and Rahul Narain for their help with the comparison to ARCSim, and Libo Huang and Jiong Chen for helpful discussions. Appreciation goes to Danielle Luterbacher and Sigrid Carl for their sewing advice. The authors also extend their thanks to all IGL members for their time and support.

REFERENCES

Ning An, August G. Domel, Jinxiong Zhou, Ahmad Rafsanjani, and Katia Bertoldi. 2020. Programmable hierarchical kirigami. *Adv. Funct. Mater.* 30, 6 (2020), 1906711.

Bernadette Banner. 2022. *Make, Sew and Mend: Traditional Techniques to Sustainably Maintain and Refashion Your Clothes.* Page Street Publishing.

David Baraff and Andrew Witkin. 1998. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, Annual Conference Series, 43–54. https://www.siggraph.org/preparing-your-content/citing-siggraph-publications/

Margie Bauer and Barry Elsey. 1992. Smocking: Traditional craft as the expression of personal needs and adult community education in Australia. *Austr. J. Adult Commun. Educ.* 32, 2 (1992), 84–89.

Robert Bridson, Ronald Fedkiw, and John Anderson. 2002. Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques.* 594–603.

Robert Bridson, Sebastian Marino, and Ronald Fedkiw. 2005. Simulation of clothing with folds and wrinkles. In *ACM SIGGRAPH 2005 Courses.* 3–es.

Christopher Carlson, Nina Paley, and Theodore Gray. 2015. Algorithmic quilting. In *Proceedings of Bridges: Mathematics, Music, Art, Architecture, Culture.* 231–238.

Toen Castle, Yigil Cho, Xingting Gong, Euiyeon Jung, Daniel M. Sussman, Shu Yang, and Randall D. Kamien. 2014. Making the cut: Lattice kirigami rules. *Phys. Rev. Lett.* 113, 24 (2014), 245502.

Toen Castle, Daniel M. Sussman, Michael Tanis, and Randall D. Kamien. 2016. Additive lattice kirigami. *Sci. Adv.* 2, 9 (2016), e1601258.

Zhen Chen, Hsiao-Yu Chen, Danny M. Kaufman, Mélina Skouras, and Etienne Vouga. 2021. Fine wrinkling on coarsely meshed thin shells. *ACM Trans. Graph.* 40, 5 (2021), 1–32.

Kwang-Jin Choi and Hyeong-Seok Ko. 2002. Stable but responsive cloth. *ACM Trans. Graph.* 21, 3 (Jul. 2002), 604–611.

Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A. Otaduy. 2014. Yarn-level simulation of woven cloth. *ACM Trans. Graph.* 33, 6 (2014), 1–11.

CLO. 2020. Garment Details: Expressing Smocking Detail (w/track Tool). Retrieved from https://www.youtube.com/watch?v=GwG4gxlMC1o

CLO. 2023. clo3d.com. Retrieved from https://www.clo3d.com

Boris Delaunay et al. 1934. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdel. Mat. Estestven. Nauk* 7, 793-800 (1934), 1–2.

Levi H. Dudte, Etienne Vouga, Tomohiro Tachi, and Lakshminarayanan Mahadevan. 2016. Programming curvature using origami tessellations. *Nat. Mater.* 15, 5 (2016), 583–588.

Dianne Durand. 1979. *Smocking: Techniques, Projects and Designs.* Courier Corporation.

Tamara Anna Efrat, Moran Mizrahi, and Amit Zoran. 2016. The hybrid bricolage: Bridging parametric design with craft through algorithmic modularity. In *Proceedings of the CHI Conference on Human Factors in Computing Systems.* 5984–5995.

Marwa Yasien Helmy Elbyaly and Abdellah Ibrahim Mohammed Elfeky. 2022. Investigating the effect of vodcast to enhance the skills of the Canadian smocking and complex problem solving. *Curr. Psychol.* 41, 11 (2022), 8010–8020.

Blender Foundation and Community. 2023. Blender. Retrieved from https://docs.blender.org/manual/en/latest/physics/cloth/index.html

Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. 2007. Efficient simulation of inextensible cloth. *ACM Trans. Graph.* 26, 3 (Jul. 2007), 49–es. https://doi.org/10.1145/1276377.1276438

Yuki Igarashi and Jun Mitani. 2015. Patchy: An interactive patchwork design system. In *ACM SIGGRAPH 2015 Posters.* 1–1.

Caigui Jiang, Florian Rist, Helmut Pottmann, and Johannes Wallner. 2020. Freeform quad-based kirigami. *ACM Trans. Graph.* 39, 6 (2020), 1–11.

Ruby Joseph, Kaur Prabhjot, Mehtab Shazia, et al. 2011. Lattice smocking techniques: An innovative approach to smocking. *As. J. Home Sci.* 6, 1 (2011), 5–11.

Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2010. Efficient yarn-based cloth with adaptive contact linearization. *ACM Trans. Graph.* 29, 4, Article 105 (2010), 1–10.

Minkyoung Kim. 2020. A study on reproductions of North American smocking design using a 3D virtual clothing system. *J. Fashion Bus.* 24, 5 (2020), 106–124.

Mackenzie Leake, Gilbert Bernstein, Abe Davis, and Maneesh Agrawala. 2021. A mathematical foundation for foundation paper pieceable quilts. *ACM Trans. Graph.* 40, 4, Article 65 (Jul. 2021), 14 pages.

Der-Tsai Lee and Bruce J. Schachter. 1980. Two algorithms for constructing a Delaunay triangulation. *Int. J. Comput. Inf. Sci.* 9, 3 (1980), 219–242.

Minchen Li, Danny M. Kaufman, and Chenfanfu Jiang. 2021. Codimensional incremental potential contact. *ACM Trans. Graph.* 40, 4, Article 170 (2021).

Minchen Li, Alla Sheffer, Eitan Grinspun, and Nicholas Vining. 2018. Foldsketch: Enriching garments with physically reproducible folds. *ACM Trans. Graph.* 37, 4 (2018), 133:1–133:13.

Malin Lind. 2019. Smocked Patterns: An Exploration of Jacquard Woven Patterns and Smocking Techniques for a Spatial Textile Design Context. Bachelor thesis, University of Borås.

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Trans. Graph.* 32, 6 (Nov. 2013), 209:1–7.

MarvelousDesigner. 2023. Retrieved from https://marvelousdesigner.com/

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. 2003. Discrete differential geometry operators for triangulated 2-Manifolds. In *Visualization and Mathematics III.* Springer, 35–57.

Rahul Narain, Armin Samii, and James F. O'brien. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.* 31, 6 (2012), 1–10.

Yingying Ren, Julian Panetta, Tian Chen, Florin Isvoranu, Samuel Poincloux, Christopher Brandt, Alison Martin, and Mark Pauly. 2021. 3D weaving with curved ribbons. *ACM Trans. Graph.* 40, 4 (2021), 127.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Computer Graphics Forum*, Vol. 4. Wiley Online Library, 109–116.

Olga Sorkine and Mario Botsch. 2009. Tutorial: Interactive shape modeling and deformation. In *Proceedings of the Annual Conference of the European Association for Computer Graphics (EUROGRAPHICS '09).*

Georg Sperl, Rahul Narain, and Chris Wojtan. 2020. Homogenized yarn-level cloth. *ACM Trans. Graph.* 39, 4 (2020), 48–1.

Margaret Spufford and Susan Mee. 2017. *The Clothing of the Common Sort:1570–1700.* Oxford University Press.

Min Tang, Tongtong Wang, Zhongyuan Liu, Ruofeng Tong, and Dinesh Manocha. 2018. I-Cloth: Incremental collision handling for GPU-based interactive cloth simulation. *ACM Trans. Graph.* 37, 6 (2018), 1–10.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically deformable models. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques.* 205–214.

Alison Toplis. 2021. *The Hidden History of the Smock Frock.* Bloomsbury Publishing.

Fei Wang, Xiaogang Guo, Jingxian Xu, Yihui Zhang, and CQ Chen. 2017. Patterning curved three-dimensional structures with programmable kirigami designs. *J. Appl. Mech.* 84, 6 (2017).

Huamin Wang. 2021. GPU-based simulation of cloth wrinkles at submillimeter levels. *ACM Trans. Graph.* 40, 4 (2021), 1–14.

Kui Wu, Hannah Swan, and Cem Yuksel. 2019. Knittable stitch meshes. *ACM Trans. Graph.* 38, 1 (2019).

Received 2 May 2023; revised 21 August 2023; accepted 26 October 2023