FoldSketch: Enriching Garments with Physically Reproducible Folds

MINCHEN LI, The University of British Columbia
ALLA SHEFFER, The University of British Columbia
EITAN GRINSPUN, Columbia University
NICHOLAS VINING, The University of British Columbia

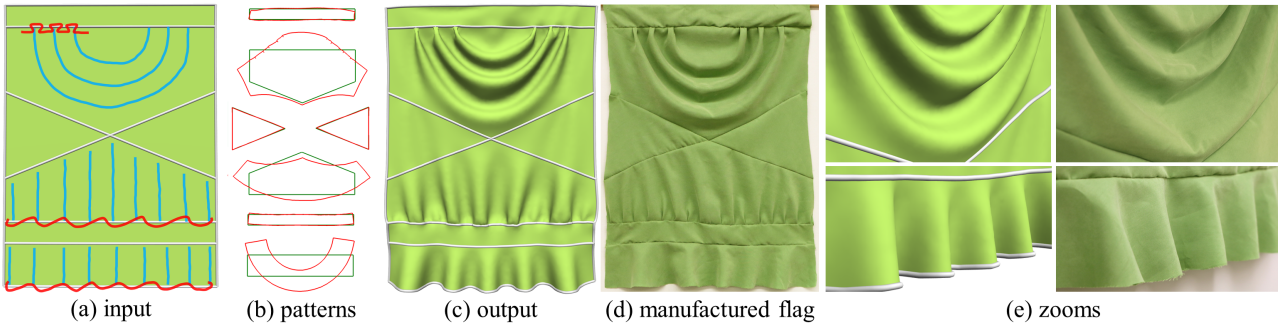


Fig. 1. Complex physically simulated fabric folds generated using FoldSketch: (left to right) Plain input flag with user sketched schematic folds, original (green) and modified final (red) patterns; final post-simulation flag augmented with user-expected folds; real-life replica manufactured using the produced patterns; zooming in highlights the complex and evolving output fold profile shapes.

While folds and pleats add interest to garments and cloth objects, incorporating them into an existing design manually or using existing software requires expertise and time. We present *FoldSketch*, a new system that supports simple and intuitive fold and pleat design. FoldSketch users specify the fold or pleat configuration they seek using a simple schematic sketching interface; the system then algorithmically generates both the fold-enhanced 3D garment geometry that conforms to user specifications, and the corresponding 2D patterns that reproduce this geometry within a simulation engine. While previous work aspired to compute the desired patterns for a given *target* 3D garment geometry, our main algorithmic challenge is that we do not have target geometry to start with. Real-life garment folds have complex profile shapes, and their exact geometry and location on a garment are intricately linked to a range of physical factors such as fabric properties and the garment's interaction with the wearer's body; it is therefore virtually impossible to predict the 3D shape of a fold-enhanced garment using purely geometric means. At the same time, using physical simulation to model folds requires appropriate 2D patterns and initial drape, neither of which can be easily provided by the user. We obtain both the 3D fold-enhanced garment and its corresponding patterns and initial drape via an alternating 2D-3D algorithm. We first expand the input patterns by allocating excess material for the expected fold formation; we then use these patterns to produce an estimated fold-enhanced drape geometry that balances designer expectations against physical reproducibility. We use the patterns and the estimated drape as input to a simulation generating an initial reproducible output. We improve the output's alignment with designer expectations by progressively refining the patterns and the estimated drape, converging to a final fully physically reproducible fold-enhanced garment. Our experiments confirm that FoldSketch reliably converges to a desired garment geometry and corresponding patterns and drape, and works well with different physical simulators. We demonstrate the versatility of our approach by showcasing a collection of garments augmented with diverse fold and pleat layouts specified via the FoldSketch interface, and further validate our approach via comparisons to alternative solutions and feedback from potential users.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**; *Physical simulation*;

Additional Key Words and Phrases: garment modeling, sketch-based modeling, folds

ACM Reference Format:
Minchen Li, Alla Sheffer, Eitan Grinspun, and Nicholas Vining. 2018. FoldSketch: Enriching Garments with Physically Reproducible Folds. *ACM Trans. Graph.* 37, 4, Article 133 (August 2018), 13 pages. https://doi.org/10.1145/3197517.3201310

Authors' addresses: Minchen Li, The University of British Columbia, minchernl@gmail.com; Alla Sheffer, The University of British Columbia, sheffa@cs.ubc.ca; Eitan Grinspun, Columbia University, eitan@cs.columbia.edu; Nicholas Vining, The University of British Columbia, nvining@cs.ubc.ca.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2018/8-ART133 $15.00
https://doi.org/10.1145/3197517.3201310

1 INTRODUCTION

Fashion designers frequently use strategically placed folds and pleats to add interest to garments and other cloth objects to increase their visual appeal. Pleats and folds are typically formed by gathering fabric along a seamline and stitching the gathered fabric to hold it in place. In both traditional garment design software and manual workflows, these features are incorporated into an existing design by first skillfully modifying the underlying 2D patterns, and then *draping* the garment atop a mannequin or a dress form by meticulously positioning it to achieve the desired fold look. This workflow requires expertise and time, since it often takes multiple trial and error iterations to successfully incorporate the envisioned folds or pleats into an existing design [Arnold 1985; Kiisel 2013]. We propose *FoldSketch*, a new algorithmic framework for fold and pleat

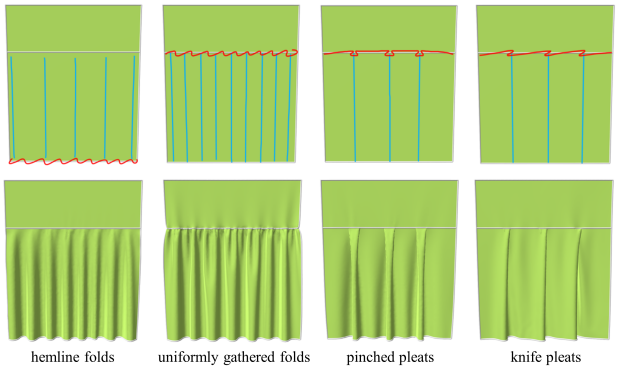


Fig. 2. Different types of folds and pleats supported by our framework: (top) schematic user sketch; (bottom) output folds.

generation that enables users to directly generate their desired fold-enhanced 3D garments without the need for manual pattern editing or draping (Figure 1). Our output garments are physically reproducible - they can be generated using physical simulation from a set of patterns and an initial drape we compute. Our method applies to the design of both virtual and real cloth objects, and enables experts and amateurs alike to quickly generate sophisticated fold and pleat configurations.

Garment design literature distinguishes between folds and pleats based on how they are formed (Figure 2). While folds are formed by uniformly gathering fabric along a seam or a hemline, pleats are formed by doubling fabric back on itself and securing it in place. We will use the term *folds* in this paper to describe both folds and pleats, for simplicity's sake, and we will only refer to pleats when addressing pleat specific processing.

Real-life folds have complex and smoothly changing cross-section geometry (Figure 1) that depends on a range of physical factors such as fabric stretchiness, thickness, and bending flexibility (Figure 3). While designers have an overall sense of how the folds they envision will look, they do not mentally account for all these factors. It is therefore unreasonable to expect designers to communicate an exact, detailed description of the folds they envision. Instead, using *FoldSketch*, designers schematically provide their anticipated fold configuration, namely their paths, magnitudes, and stitching patterns (Figures 1a, 2). Our underlying algorithm then generates a detailed fold-augmented garment consistent with this schematic input, and a set of corresponding 2D patterns and initial drape that reproduce this garment under physical simulation (Figure 1d).

Translating the user's input into detailed folds is a challenging task. Fold geometry is highly dependent on physical factors such as fabric properties and external forces, thus it is impossible to predict physically achievable fold shapes without a physical simulation context. At the same time, applying a physical simulation to generate a fold-enhanced garment requires correctly extended patterns capable of supporting the desired folds and an initial drape configuration, neither of which is available. Previous frameworks that attempted to generate garment folds in 3D space used highly simplified, groove-like fold shapes [Robson et al. 2011; Rohmer et al. 2010; Turquin et al. 2004, 2007] (Figure 4,a) which do not reflect the complexity of real-life folds. More importantly, the grooves they create are purely geometric and have no basis in real-world physics; they are not physically realizable, and any attempt to resimulate these geometric folds using state-of-the-art physics-aware pattern computation [Bartle et al. 2016] is likely to fail ((Figure 4,b).

Instead, we focus on computing patterns and the initial *target* drape, and obtain the output garment geometry via actual simulation that uses these as input. Our computation employs two key observations. First, we note that while we have no exact 3D fold geometry to begin with, we can use the designer-specified schematic input to estimate the changes in 2D patterns necessary to accommodate their intended folds. We also observe that while we do not *a priori* know the shape of the final fold-augmented garment, we can distinguish between changes to the 3D garment which are consistent with the designer's intended fold formation, and those which are not: when editing existing garments, designers attempt to introduce the modifications they envision locally while maintaining the garment shape elsewhere [Arnold 1985; Bartle et al. 2016]. In the context of fold generation, designers expect to see no garment geometry changes in areas away from the folds, while in the *region of interest* near the folds, they expect the fabric to bend along the specified fold direction.

We use these two observations to compute the target drape and patterns, and use those to obtain the output garment. Starting with an input simulated garment, its corresponding patterns, and a schematic indicating the user's desired fold placement, we compute an initial new set of 2D patterns which have sufficient material to incorporate these folds (Section 5). These new patterns are optimized by extending the original patterns orthogonally to the fold paths to facilitate fold formation, while minimizing any secondary changes in pattern shape away from the user specified folds. We use the new patterns to obtain a target 3D drape that balances physical reproducibility and designer intent; this is achieved by using a simulation framework which augments standard physical forces that reflect real-life phenomena with new synthetic forces that reflect our style preservation and fold alignment constraints (Section 6). While the resulting drape conforms to designer expectations, using it as starting point for simulation without these synthetic forces may produce undesirable artifacts in the output garment, such as local sagging and bulging that violates our expectation of style preservation (Figure 5). We minimize these artifacts by progressively updating both the patterns and the target drape, reducing the difference between augmented and unconstrained simulation outputs (Section 7).

We demonstrate our method's capabilities by applying a range of diverse fold and pleat patterns to different input garments and other cloth-made objects with varying material properties. In all cases, the resimulated outputs reflect the designer's intended fold configurations while preserving the original style of the input, and are accompanied by corresponding 2D patterns. Our framework is not specific to a particular simulation engine; it works equally well with two distinctly different simulation engines: Sensitive Couture [Umetani et al. 2011] and ARCSim [Narain et al. 2013, 2012], one of which is optimized for speed while the other is optimized for accuracy (Figures 14, 16). We validate our approach via expert critique, and by comparisons to alternative solutions. Finally, we generate two real cloth objects using the patterns produced by our system, confirming its applicability to real-life fashion design (Figures 1, 13).

Our overall contribution is a novel framework that allows expert and amateur fashion designers to enhance existing cloth-made objects with complex fold and pleat patterns via a simple to use interface that successfully replaces the currently used cumbersome,

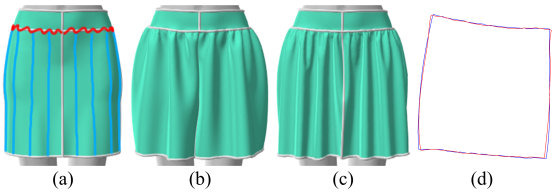


Fig. 3. Given the same schematic fold notations and visually identical input garments with different fabric properties (a) our framework produces reproducible fold-augmented garments (b,c) that reflect the different bending parameters of the inputs and the corresponding sets of patterns (d) - red for the thicker fabric (b) and blue for the thinner one (c).

and time consuming, iterative workflow for adding folds to garments. Key to our method are the pattern extension and target drape computation procedures that incorporate cues provided by physical simulation into the geometry optimization process.

2 RELATED WORK

Our work builds upon traditional fashion design methodologies for fold creation, algorithms for fold generation, and modern computational fashion design tools.

Garment Construction and Traditional Fold Design. Garments and other cloth objects are traditionally constructed by first cutting 2D fabric panels following a given pattern and stitching these panels together along shared *seams*. To incorporate folds into an existing design, tailors add additional material to panels along a hemline (open boundary) or a seam. While hemline folds form due to gravity, seam folds are formed by gathering the excess material to match a shorter opposite panel boundary and stitching the two. Pleats are added by folding the material sideways and stitching it in place (Figure 2); commonly pleats are then ironed to keep them in position.

Designers employ a mixture of 2D and 3D fabric manipulation to form their desired fold arrangements [Arnold 1985; Kiisel 2013]. They typically start with approximate 2D patterns, then drape them around a mannequin, using pins to gather and hold the desired fold configuration in place. Designers then trim the patterns, eliminating redundant material, or alternatively repeat the cycle with wider initial patterns when the original are insufficient to achieve the desired fold magnitude. They subsequently adjust and repin the drape, and iterate. Successful fold design requires significant time and skill; design courses and tutorials dedicate multiple lectures and chapters to fold design techniques [Kiisel 2013].

Commercial garment design software, such as [Browzwear 2017; CLO 2017; EFI 2017; Fontana et al. 2005; Volino et al. 2005], employs a 2D-to-3D design framework where users manually specify both 2D pattern geometry and an initial draping configuration. These systems then use physics-based simulation to generate the resulting 3D garment shapes. To add folds to a garment using such systems users need to employ the knowledge intensive and time consuming workflow described above - they need to manually edit the patterns, specify the initial drape, and then use the simulation output to iterate on both.

Computational Garment Design. Recent algorithms enable procedural modeling of static virtual garments that have the appearance of clothing [De Paoli and Singh 2015; Decaudin et al. 2006; Kwok et al. 2016; Li and Lu 2014; Robson et al. 2011; Turquin et al. 2004, 2007; Wang et al. 2003]. Some of these methods use sketching interfaces to produce low frequency garments [De Paoli and Singh 2015; Decaudin et al. 2006; Robson et al. 2011; Turquin et al. 2004, 2007; Wang et al. 2003]. Others support mixing of existing garment elements in 3D space [Kwok et al. 2016; Li and Lu 2014]. The garments they produce have no patterns, no physical parameters, and are often not physically reproducible; they are thus unsuitable for cloth simulation or manufacturing.

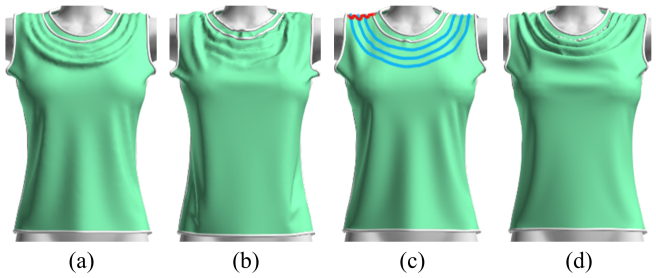


Fig. 4. Existing fold modeling methods (here [Rohmer et al. 2010]) generate simplistic fold shapes (a); which cannot be reproduced via subsequent pattern computation and resimulation (b); given similar input fold-paths (c) we compute complex reproducible folds (d).

Grading methods algorithmicaly resize garments to fit a mannequin different from the one they were originally designed for [Brouet et al. 2012; Cordier et al. 2003; Meng et al. 2010; Wang et al. 2005]. These methods can potentially transfer the location and 3D shape of existing folds to a new garment, but are not designed for forming new folds.

Sensitive Couture [Umetani et al. 2011] supports a limited set of 3D to 2D edits where simple 3D changes, such as elongating or widening a garment, are propagated to 2D space. It does not provide the fine control necessary for detailed edits such as fold addition. Bartle et al. [2016] enable coarse scale 3D garment edits, such as garment mixing, length and fit changes. They use a geometric approach to produce a target 3D shape encapsulating both the original design and user-specified changes, and then reverse-engineer 2D patterns whose draped result is isometric to the target. Our focus on 3D fold design requires a target shape computation that, as opposed to being purely geometric, is closely linked to garment material's constitutive laws (Figure 3) and external forces.

Fold and Wrinkle Modeling. Computer animation frameworks produce dynamic folds whose location and shape are determined by the physical forces rather than the user [Bridson et al. 2003]. The computation they employ relies on input patterns and draped geometry to guide fold formation. In our setting we have neither. Sketch-based static garment modeling tools such as [Robson et al. 2011; Turquin et al. 2007] model folds by sweeping a cylindrical profile along a user sketched path. Cylindrical arc profiles, smoothly blended with the surrounding surface, are similarly used to procedurally model dynamic folds to augment low-quality animations [Popa et al. 2009; Rohmer et al. 2010]. Real-life folds have complex profile shapes that can significantly differ at different points along the path; thus the results produced by such methods provide only a coarse unrealistic-looking approximation (Figure 4a). Purely geometric sketch-based surface bending methods such as BendSketch [Li et al. 2017] translate user input into surface detail geometry but cannot guarantee that the resulting folds are physically plausible. More importantly,

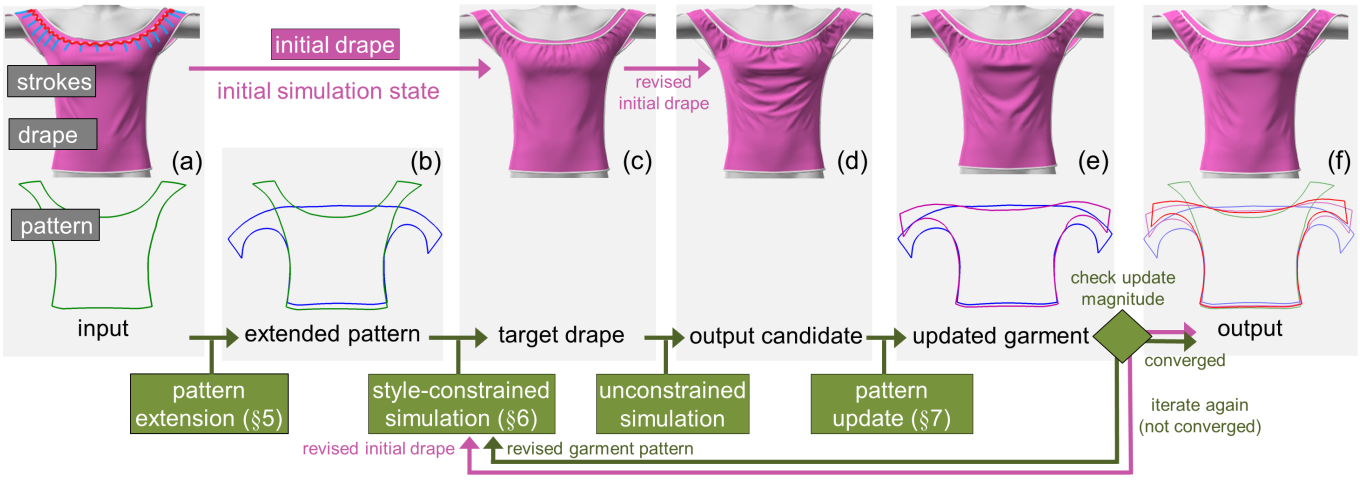


Fig. 5. FoldSketch Algorithm: (left to right) The *input* consists of a garment pattern, a corresponding drape on a mannequin, and designer-sketched folds. Folds require additional fabric material, which is obtained by *extending the pattern*. Draping depends heavily on the *initial drape* of the simulation; a poor initial drape does not lead to the desired fold arrangement. We add fictitious *style constraining* forces to induce the fold arrangement, obtaining a *target drape*. We iterate, seeking an initial drape that yields a similar fold arrangement *without* fictitious forces. Starting from the target drape, an unadulterated draping simulation produces a *candidate output drape*. The candidate is acceptable if we cannot improve upon it. We attempt to improve the garment pattern to reduce differences between the target and output drapes. If we make a substantial update to the pattern, we iterate, using our output candidate as the new initial drape.

fold-enhanced garments generated using all these approaches have no corresponding patterns, and often are not physically reproducible. Figure 4b shows the result of attempting to reproduce the geometry in Figure 4a by computing physics-aware patterns [Bartle et al. 2016] that match this geometry and using those patterns and the fold-enhanced geometry as input to an actual simulator. Our framework successfully generates complex reproducible fold geometries and their corresponding patterns using sketched paths as guidance (Figure 4,cd).

3 ALGORITHM

The input to our algorithm is a simulated 3D garment, draped around a character or mannequin, and its corresponding set of 2D patterns (Figure 5,a). Using FoldSketch, designers specify their desired fold configuration by sketching on top of this input. They draw *path strokes* (Figure 2, blue) to indicate the direction and length of their desired folds, and schematic *gathering* strokes (Figure 2, red), whose label indicates the type of fold they want to form (e.g. "knife pleats" or "gathered folds") and whose geometry encodes fold properties such as the pattern boundary location where new material should be added, the stitching scheme, and the amount of extra material that should be added (Section 4, Figure 2). Given this input, our goal is to generate a new garment that has both the desired fold geometry within the *region of interest* surrounding the input strokes and the input garment geometry away from the strokes, and to produce a corresponding set of 2D patterns (Figure 5,f).

We approach this problem using an alternating 2D-3D process, inspired by traditional garment design practices (Figure 5). We first create an initial set of extended patterns. We then create an initial, synthetic, target drape that utilizes these patterns and conforms to designer expectations. We use the patterns and this drape as input to a standard simulator to obtain a reproducible output, which may or may not align with designer expectations. We optimize output

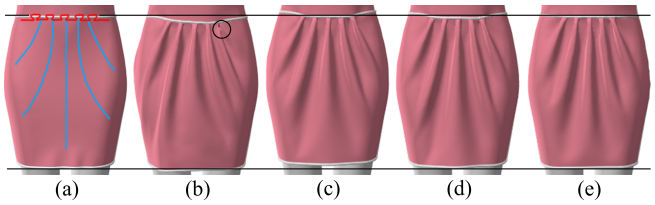


Fig. 6. Input garment and strokes (a) and unconstrained simulation outputs produced using different initial drapes: (b) from flat panels; (c) from initial 3D garment (augmented with pleat sewing scheme); (d) from our initial 3D target drape. (e) Final result (using iteratively updated patterns and target drape). The horizontal lines highlight the garment proportions.

alignment with designer expectations by alternatingly refining the patterns and the target drape, and generate the final output (Figure 5, right) by performing one more round of unconstrained simulation using those.

Pattern Extension. Adding folds to an existing garment requires extending the patterns in the region of interest, in the direction orthogonal to the fold paths direction, to allow for buckling. We compute the extension direction and the amount of extension necessary to accommodate the user anticipated fold magnitudes for each triangle contained in the region of interest. We extend the input patterns by scaling these triangles using the specified directions and scaling factors, while retaining triangle shape and scale everywhere else (Section 5, Figure 5,b). In our pattern extension computation, we explicitly account for sewing and pattern making constraints. We enforce seam compatibility, ensuring that shared boundaries between panels that had the same length in the original garment retain this property after extension. We also minimize changes in pattern boundary shape in order to prevent high curvature oscillations, which complicate pattern cutting and sewing.

3D Target Drape Estimation. FoldSketch's desired output is a 3D garment that is physically reproducible from an input set of patterns, and which conforms to the designer's expectations as expressed in the sketched input. The shape of a fold-enhanced garment is highly dependent on an initial drape (Figure 6), as loose fabric can be easily arranged to have different forms. One of our core challenges is to obtain a suitable initial drape that, when combined with the patterns, will produce the desired simulation output. We would like a drape that reflects designer expectations and is also close to the final output, in order to minimize the changes induced by the simulation. We generate a *target* drape geometry that balances these two considerations by employing a constrained garment simulation. We augment the standard physical forces with synthetic forces that serve two roles: they explicitly enforce designer expectations of preserving input garment geometry outside the region of interest, and of purely fold-aligned buckling inside it (Figure 5c, Section 6). We initialize this simulation using the extended patterns and an initial 3D garment drape which is based on the original garment geometry, but which adds the additional synthetic constraints that are necessary to form the designer's envisioned folds (Section 6.1).

Update. Using the obtained estimated patterns and drape as inputs to a simulation is unlikely to produce a garment that fully aligns with designer expectations. We optimize the resulting garment using an alternating 2D-3D process that updates the patterns and the drape (Section 7). The output of this final stage consists of a set of patterns, an initial drape, and a final output garment produced via simulation that uses the patterns and the drape as input.

4 SKETCH INTERFACE

We illustrate FoldSketch's UI with a worked example. Starting with a draped garment on a mannequin (Figure 7, a) a designer adds one or more *path strokes* (blue); the path strokes describe the locations, length and direction of the folds that they wish to add. As explained earlier designers typically form folds by elongating one or more garment panel boundaries. While the choice of the boundary can be deduced from the path strokes end-points, designers need the means to define the stitching pattern and the expected amount of boundary elongation. The designer provides this information by specifying a fold type via a dropdown button; in this case, we are adding uniformly gathered folds. They then add a *gathering stroke* (Figure 7, b) atop of the corresponding gathered boundary. This stroke indicates the amount of extra material that must be incorporated into the folds; for pleats it also encodes the pleat stitching pattern. Our system then synthesizes a new garment (Figure 7, c)), incorporating the designer's desired folds. The designer may then add other folds as desired, or revert their folds and experiment with new designs.

The design tool provides four types of folds (Figure 2): hemline folds, uniformly gathered folds, pinched pleats, and knife pleats. In all cases, the process is the same: the designer chooses a fold type from a dropdown menu, draws multiple *path strokes* which define the direction and length of the folds, and concludes with a single *gathering stroke* which defines the amount of boundary elongation and the stitching pattern for pleats. FoldSketch asks users to specify the gathering configuration, and not the fold profile cross section elsewhere along the fold paths, because designers know the gathering configuration they want to use, whereas the fold cross section varies in shape based on fabric physics. This interface allows us to naturally employ schematic input to augment garments with different fabric properties.

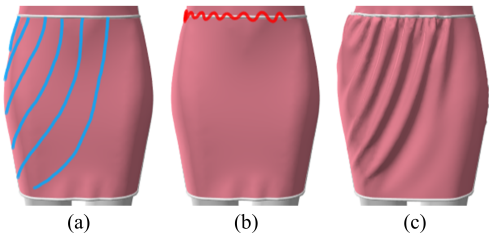


Fig. 7. UI walkthrough example: (a) fold path strokes (blue) traced over input garment in 3D view; (b) gathering stroke; (c) output garment.

The amount of extra material needed to form the folds is then computed by comparing the length of this stroke to the portion of the corresponding, gathering, boundary in-between the stroke end-points. To form pleats, we need to fold fabric onto itself and stitch the overlapping portions together. We use the sharp corners in the gathering stroke to dictate the location and magnitude of the pleats (See Figure 8, e and f). For knife pleats, the user can also choose between right or left foldovers (See Figure 8, f). Designers can specify folds that extend between two gathering boundaries (Figure 4). To communicate their intent rather than drawing two gathering strokes, they simply need to draw path strokes that start and end at two boundaries, and provide a gathering stroke on one of these boundaries. We interpret such strokes as *two-sided* and mirror the gathering pattern along the gathering boundary onto its opposite boundary.

For hemline and uniformly gathered folds the frequency and magnitude of the formed folds directly depend on the properties of the fabric used [Rohmer et al. 2010]. We therefore only expect designers to encode the amount of the extra material, rather than the fold magnitude, when drawing the gathering stroke, and to use the path strokes to dictate fold direction rather than frequency or exact locations. For pleats we use the stroke geometry to dictate the fabric folding and stitching pattern, determining their location, magnitude, and orientation (Figure 8). Additionally, many input garments contain pairs of symmetric panels. To support convenient symmetric fold design, FoldSketch allows the user to reflect strokes from one panel to its symmetric counterpart. Examples are shown in Figure 8.

5 PATTERN EXTENSION

After the designer has marked up their desired changes in FoldSketch, we estimate the shape of new garment patterns that are appropriately extended to allow the designer's target folds to form. We have as inputs the garment patterns, draped on the mannequin; the extended gathering seam length, computed from the user annotation; and the designer specified fold paths.

We observe that, in order to facilitate the desired behavior, our extended patterns have to satisfy the following conditions. Inside the folds' *region of influence* we expect each triangle to be elongated orthogonally to the fold paths direction to facilitate subsequent buckling. We expect the elongation to be maximal next to the gathering seam, and to smoothly decrease further away from it; we also expect these triangles to retain their original length along the direction of the fold paths. Conversely, we expect pattern triangles

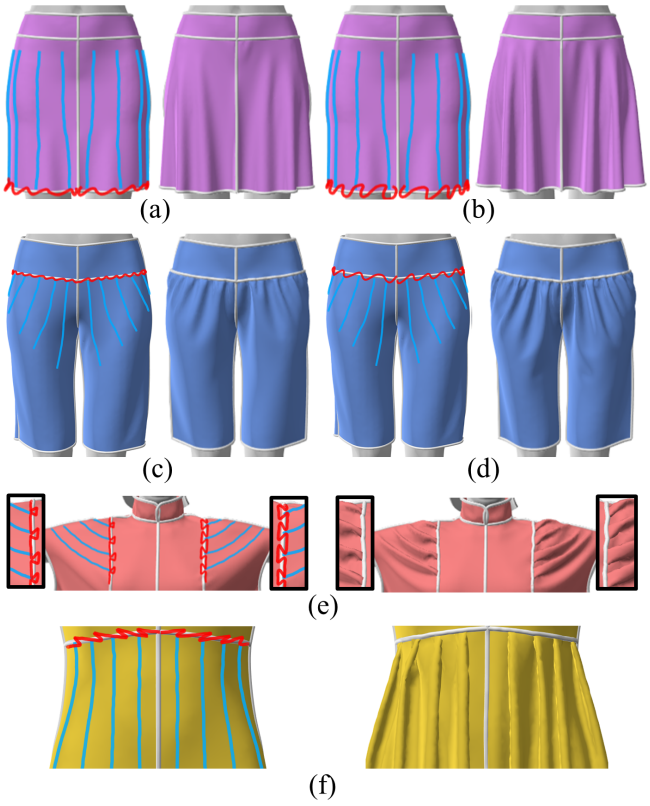


Fig. 8. Impact of different gathering strokes on output folds: (a,b) hemline folds with different magnitude, (c,d) gathered folds with different magnitude, (e) pinched pleats with different width (left and right shoulders), (f) knife pleats with different orientation (left and right panels). The strokes on the right side of the garment mirror those on the left.

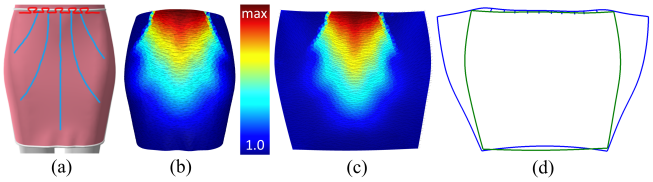


Fig. 9. Pattern extension: (a) input garment and strokes; (b) visualized 3D space scaling field; (c) scaling field on 2D patterns; (d) extended patterns (blue) and input patterns (green).

away from the folds to retain their original shape and size. To avoid sewing artifacts, we must pay special attention to panel boundaries: we need to strictly preserve the lengths of all panel boundaries except the gathering seam. We also need to avoid high curvature oscillations along them, as those make sewing and cutting panels more challenging.

We solve for our initial extended patterns using a two step process. We first compute the direction and amount of extension for each garment pattern triangle (Section 5.1). We then deform the existing 2D patterns to incorporate this desired expansion (Section 5.2).

5.1 Tensor Field Computation

We express the problem of finding expansion magnitudes and directions for patterns as the computation of an appropriate field of stretch tensors. In computing the field we face a chicken-and-egg problem, where we must know the folds' region of influence in order to compute where scale should be preserved, while at the same time we cannot determine the exact boundaries of this region of influence without knowing the amount of scaling required to accommodate the folds. We compute expansion directions first and use them to compute preliminary region of influence boundaries; we then use this set of boundaries to compute expansion magnitudes and finalize the region of influence.

We compute target pattern expansion directions and magnitudes in 3D space first as this is where the user input is provided. We then project them to the actual patterns accounting for the deformation these patterns undergo during the garment simulation. To define our tensor we use a 2-Rotational Symmetry tensor field [Zhang et al. 2007], as stretch should be invariant under 180° rotation. We express each tensor T_i^t as a 2x2 matrix using the singular value decomposition $T_i^t = R(\theta_i)S_iR(\theta_i)^T$, where $R(\theta_i) = \begin{bmatrix}\cos(\theta_i) & -\sin(\theta_i)\\ \sin(\theta_i) & \cos(\theta_i)\end{bmatrix}$ is a rotation matrix, and S_i is the diagonal skew matrix $S_i = \begin{bmatrix}s_i & 0\\ 0 & 1\end{bmatrix}$, where $s_i \geq 1$. Both T_i^t and θ_i are defined on the 2D local frame of triangle i. This allows us to decompose the problem of finding the tensor field into separately finding the per-triangle expansion directions θ_i and the per-triangle expansion amounts s_i, facilitating the two stage region-of-influence computation.

5.1.1 Computation of 3D Expansion Direction. For computing the expansion directions θ of the tensor field, we follow the framework and nomenclature of Ray et al. [2009]. Rather than computing directional angles θ_i, we employ their method and instead compute the representative vectors $V_i = (\cos(2\cdot\theta_i), \sin(2\cdot\theta_i))$, and extract θ_i from them. To compute V_i, we minimize an energy function that balances smoothness of the tensor directions against the requirements for these tensors to be orthogonal to the fold paths:

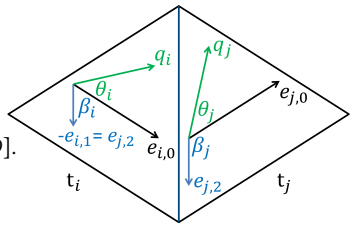


Fig. 10. $e_{i,k}$ is the k-th edge of t_i, q_i is the vector defined in the local frame of t_i, the discrete Levi-Civita connection is defined as $r_{ij} = \beta_i - \beta_j$

$$\begin{aligned} E &= E_{\text{smooth}} + E_{\text{fit}} \\ E_{\text{smooth}} &= \sum_{ij\in\mathcal{E}^*} w_{ij}^{-1}(V_i - R(2r_{ij})V_j)^2 \\ E_{\text{fit}} &= (1/\overline{|t_i|})\sum_i |t_i|(V_i - V_i^{\text{init}})^2 \end{aligned} \tag{1}$$

where $\mathcal{E}^*$ is the set of all adjacent triangle pairs, sharing a common edge.

The term E_{fit} is evaluated over all triangles i crossed by fold paths. Here w_{ij} are the standard cotangent weights [Pinkall and Polthier 1993]; $r_{ij} \in \mathbb{R}$ is the discrete Levi-Civita connection [Crane et al. 2010], which is necessary as θ_i and θ_j exist in different local tangent spaces; V_i^{init} is the orthogonal direction of the path strokes

on the triangle i; $|t_i|$ is the area of triangle i; and $\overline{|t_i|}$ is the average triangle area. Since V_i must be unit length, we use an iterative minimization: we first minimize $E_{\text{smooth}}+E_{\text{fit}}$ without normalization constraints, then normalize the computed V_i and proceed to iterate renormalizing them after every iteration. We then compute $\theta_i = \text{atan}(V_i \cdot (0,1)/V_i \cdot (1,0))/2$.

5.1.2 Preliminary Region of Influence. We use the computed directions to extract an approximate set of boundaries for the folds' region of influence. We trace the approximate boundaries by starting from the end points of the gathering seam, and following the direction orthogonal to our computed extension direction. Tracing terminates when the boundary of the current pattern is reached. We use the Runge-Kutta method [Ascher and Greif 2011] to perform the tracing. We solve an initial value problem for the ordinary differential equation $db_p/dt = f(b_p(t))$, $b_p(0) = p_{e,i}$, where b_p is the boundary to be traced, t is the integration parameter, f is the vector field orthogonal to the extension directions θ, and $p_{e,i}$ is the i-th end point of the gathering seam. Starting from each $p_{e,i}$, we apply the midpoint RK2 scheme to evaluate $b_p(t_{n+1}) \leftarrow b_p(t_n) + \Delta t f(b_p(t_n) + \frac{1}{2}\Delta t f(b_p(t_n)))$ in each step n.

5.1.3 Computation of Expansion Magnitudes. To obtain the expansion magnitudes s_i, we compute a smoothly varying scalar field that propagates the anticipated expansion along the fold paths throughout the triangles in the region of influence on the 3D garment mesh. Our formulation accounts for four considerations: magnitude smoothness, accommodation of the anticipated scaling along fold paths, preservation of original scale outside the region of influence, and preservation of the lengths of all pattern boundaries except the gathering seam:

$$\min_s E = \underbrace{\sum_{ij\in\mathcal{E}^*} w_{ij}^{-1}(s_i - s_j)^2}_{\text{Smoothness}} + \underbrace{\lambda_l E^{len}}_{\text{Boundary length preservation}} + \underbrace{\lambda_v E^{FL}}_{\text{Fold path scaling}} + \underbrace{\lambda_v/\overline{|t_i|}\sum_{i\in\mathcal{B}}|t_i|(s_i-1.0)^2}_{\text{Scale preservation outside region of interest}} \quad (2)$$

We empirically set $\lambda_l = 100$, $\lambda_v = 10$.

Fold-Driven Scaling. To scale garment material around fold paths, we first smoothly interpolate the scale terms along each fold path. We use the ratio between the lengths of the gathering stroke and its corresponding pattern boundary segment as the scale at the start of the path, and set the value to 1 at the end of the path furthest from the gathering boundary. We then constrain the scales within triangles intersecting the fold path to scale by the average scale factor s'_i along the intersected fold path segment:

$$E^{FL} = 1/\overline{|t_i|}\sum_{i\in\mathcal{F}}|t_i|(s_i - s'_i)^2 \quad (3)$$

where $\mathcal{F}$ contains triangles intersecting fold paths.

Pattern Boundary Length Preservation. We seek to strictly preserve the lengths of non gathering seam panel boundaries. We can explicitly express triangle edge length as a function of the stretch s_i in a given direction:

$$l_i(s_i) = l'_i\sqrt{s_i^2\cos^2\alpha_i + \sin^2\alpha_i}. \quad (4)$$

Here l'_i and l_i are the lengths of the boundary edge before and after expansion, and α_i is the angle between the stretch direction and the boundary edge. This function is nonlinear and thus hard to optimize; we therefore linearly approximate Eq.4 around $s_i = 1$ instead:

$$l_i^a(s_i) = l'_i + (\frac{\partial l_i}{\partial s^t})_{s^t=1}(s_i - 1) \quad (5)$$

We express boundary length preservation as:

$$E_j^{len} = 1/\bar{l}'_i\sum_{i\in\mathcal{E}(j)}(l_i^a(s_i) - l'_i)^2 \quad (6)$$

Here $\overline{l'_i}$ is the average boundary edge length, and $\mathcal{E}(j)$ are the participating boundary edges. This expression is equivalent to enforcing the constraint that $s_i = 1.0$, $i \in \mathcal{E}(j)$ weighted by $l'_i\cos^2\alpha_i$. Note that if the extension direction is orthogonal to the edge segment, this constraint vanishes as desired.

Final Region of Influence. At this point, we can now trivially define the final region of influence: a triangle t_i is contained in the region of influence if its scalar component $s_i \geq 1.0 + \varepsilon$.

5.1.4 Tensor Field Projection onto 2D Patterns. To actually expand the patterns, we must transfer the expansion tensor field, computed on and expressed with respect to the 3D draped input garment, onto the set of 2D patterns. Since fabric often stretches under simulation this stretch must be factored into our computations: failing to account for stretch incurred by the draping process would cause a naive algorithm to assume sufficient material already exists to form folds, and hence the amount of required 2D pattern extension may be underestimated (Figure 11). We employ a transformation scaling approach to correctly account for stretch during 2D pattern extension.

Recall that we have constructed the per-triangle symmetric tensors $T_i^t = R(\theta_i)S_iR(\theta_i)^T$, which specify how much each triangle must expand and in what direction. However, this deformation is computed with respect to the *simulated* 3D garment. To obtain the corresponding change for garment patterns, we need to compute a transformation for each of the 2D pattern triangles t_i that, after simulation, will extend their corresponding 3D triangle t'_i in the direction and by the amount specified by the tensor field. We first compute a common coordinate frame for the 2D pattern and 3D garment triangles by rotating them to the xy plane and co-aligning them, so that both triangles are placed at the origin and have a designated common edge u that is aligned with the x-axis. We refer to the transformed replica of the triangle t' as $\tilde{t}$. The 2D intrinsic action of the garment simulation per triangle is then described by the 2×2 matrix T^{d0}, $\tilde{t}_i = T^{d0}t_i$. Assuming the impact of the simulation on the deformed triangle is the same as on the undeformed one, we must find a transformation T'_i such that $T_i^t\tilde{t}_i = T_i^tT^{d0}t_i = T^{d0}T'_it_i$. We can consequently compute T'_i as

$$T'_i = (T^{d0})^{-1}T_i^tT^{d0} \quad (7)$$

Fold Pre-Conditions. In order for the designer's specified folds to form, we must take into account that the original simulation may have stretched the fabric when draping it on the mannequin, and that this stretch is not explicitly accounted for in the computation

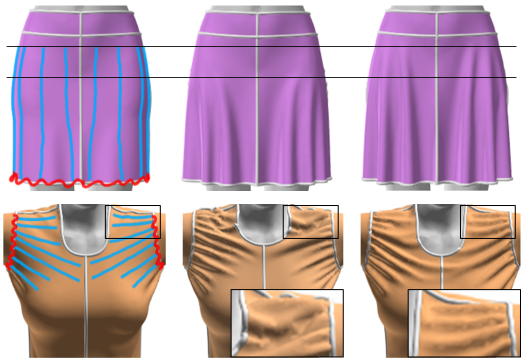


Fig. 11. Impact of fold pre-conditions: (top) result without and with orthogonal stretch cancellation; (bottom) result without and with fold path stretching. Note the impact on fold length and shape.

of the per-triangle stretch tensors T_i^t. If the extension that we have computed is of similar or smaller magnitude to the stretch of the draped garment, then adding the amount of material specified by T_i' to the garment pattern is sufficient to release the stretched fabric, but may not be sufficient to form folds (Figure 11 top).To actually add visible folds, the extension needs to be increased to fully cancel out the stretch *and* to locally extend the 3D garment to reflect the designer's expected fold magnitude.

In addition to accounting for stretch orthogonal to the fold direction in the original drape, we also consider fold feasibility. While vertical folds are consistent with gravity, horizontal and near-horizontal folds can only form if the fabric is either very stiff or is stretched along the fold path (Figure 11 bottom). If the user placed folds are on a loose part of the garment, there is no way to guarantee that they show up without major changes to the garment style. However if the garment is locally tightly fitting, we can improve fold feasibility by ensuring that the garment is, at least, weakly stretched along the fold direction.

We therefore account for draped garment stretching and fold feasibility by simultaneously canceling the stretch in the original deformation gradient of the draping on the left-hand side of Eq.7 to approximate the deformation gradient when draping the new patterns, and weakly shrinking each input triangle t_i when T_i^{d0} has no stretch along the fold path:

$$(T_i^s)^{-1} T_i^{d0} T_i^0 = T_i^t T_i^{d0} \tag{8}$$

Here $T_i^s = R(\theta_i) \begin{bmatrix} \max(1, T_{i,(2,2)}^{d0,R}) & 0 \\ 0 & k \end{bmatrix} R(\theta_i)^T$ is the tensor with the part of stretch and compression to be cancelled in $T_i^{d_0}$, where $k = 1$ if $T_{i,(1,1)}^{d0,R} > \frac{1}{0.95}$ and $k = 0.95 \cdot T_{i,(1,1)}^{d0,R}$ otherwise, and $T_i^{d0,R} = R(\theta_i)^T T_i^{d0} R(\theta_i)$. The final transformation T_i^0 is then:

$$T_i^0 = (T_i^{d0})^{-1} T_i^s T_i^t T_i^{d0}$$

5.2 Pattern Deformation

The collection of transformations T^0 describes the intrinsic change that each triangle on the input 2D pattern P_0 is expected to undergo. These can be applied to the patterns using standard local-global deformation approaches [Liu et al. 2008; Sorkine and Alexa 2007]. However, as previously mentioned, the shape of the resulting pattern boundaries affects both cutting and sewing - more curved boundaries, especially those with high curvature variation are harder to process. We thus augment our formulation with a boundary-shape preserving term, and minimize:

$$\sum_{i \in \mathcal{T}} \|S_i - R_i T_i^0 S_i'\|_F^2 + \lambda_b \sum_{j \in \mathcal{B}} E_j^{boundary}$$

Here R_i is a per-triangle rotation matrix; $S_i = [v_{i,1} - v_{i,0}, v_{i,2} - v_{i,0}] \in R^{2\times 2}$ is the local coordinate frame of the deformed triangles, and $S_i' \in R^{2\times 2}$ is the local coordinate frame of the original triangles. $E_j^{boundary}$ encodes boundary shape. We empirically set $\lambda_b = 10$. We minimize this energy using a standard local-global approach, [Sorkine and Alexa 2007]. We optimize R_i in the local step using singular value decomposition, and optimize S_i in the global step using the Sparse Cholesky implementation in the Eigen library [Guennebaud et al. 2010].

Boundary Shape Preservation. When preserving boundary shape, we differentiate between vertices that are located on straight boundary sections and those that are located on curved boundary sections. Let e_j' be the vector between the two original vertices v_j' and v_{j-1}' on a boundary; that is: $e_j' = v_j' - v_{j-1}'$. We distinguish between straight and curved vertices by thresholding $|\frac{|e_j' \cdot e_{j+1}'|}{|e_j'||e_{j+1}'|} - 1|$ with the threshold set to 10^{-3}. For each curved boundary vertex v_j' we express its curvature normal

$$n_j' = (-e_{j,y}' - e_{j+1,y}', e_{j,x}' + e_{j+1,x}')$$

as a linear combination of e_j' and e_{j+1}':

$$n_j' = \alpha_j e_j' + \alpha_{j+1} e_{j+1}'.$$

We then define

$$E_j^{boundary} = \|n_j' - (\alpha_j e_j + \alpha_{j+1} e_{j+1})\|^2$$

where $e_j = v_j - v_{j-1}$. For straight boundaries, we use the line Laplacian as the respective energy:

$$E_j^{boundary} = \|v_j - (w_{j-1} v_{j-1} + (1 - w_{j-1}) v_{j+1})\|^2$$

where $w_{j-1} = |e_{j+1}'|/(|e_j'| + |e_{j+1}'|)$.

To avoid simulation artifacts due to poor triangulations, we remesh the obtained 2D patterns P_e, and resample the tensor fields defined on them for later use.

6 TARGET DRAPE COMPUTATION

Our final goal is a garment which can be reproduced via an unconstrained simulation from a set of appropriate patterns and a suitable initial drape. However, draping a garment to achieve a desired fold look often requires careful initial placement. As Figure 6 demonstrates, simulation using the extended patterns alone and a range of standard initial drape configurations can result in unappealing outputs that do not conform to the expected garment look. We thus require a principled way to compute an initial drape that, under simulation, will produce the designer's expected output. Our drape computation is based on two observations that follow traditional fold design practices. We note the shape of the output garment is more strongly correlated with the shape of the input drape if this

drape itself is reproducible - that is, if the drape is, by itself, an output of a simulation using the current patterns. While we have no direct control on the shape of the output garment, we note that we can indirectly control its shape by computing a new target drape that is as close as possible to reproducible, but that also aligns with designer expectations. We balance reproducibility and design preservation by computing the drape via a constrained simulation that augments the cloth simulation energy with synthetic *design preserving* forces. We can use the resulting drape and current patterns as an input to an unconstrained simulation that computes a reproducible garment (Figure 6c). To improve this garment's adherence to designer expectations, we update the patterns and recompute the drape (Section 7, 6d).

We introduce synthetic design-preserving forces by augmenting the energy formulation used by the cloth simulator of interest E_{cloth} with an additional design term E_{design}. We integrate the synthetic term into the formulation, augmenting the energy gradients and Hessians, and optimize

$$E_{\text{PAS}} = E_{\text{cloth}} + E_{\text{design}}$$

using the standard time-stepping process. We tested our framework with Sensitive Couture [Umetani et al. 2011] and ArcSim [Narain et al. 2013], as discussed in Section 8.

Design Preservation Energy. Our design preserving energy consists of two terms, one applied within the region of interest and one applied outside it. Outside the region of influence, we preserve the original garment shape by aligning every vertex to its positions on the original 3D garment G_o using spring forces:

$$E^u_{\text{design}} = \frac{1}{2} k_u \sum_i ||v_i - v'_i||^2$$

Here k_u is the stiffness coefficient of the energy, and v_i and v'_i are vertex coordinates on the target garment G_t and original garment G_o respectively.

Within the region of influence, we expect the garment shape to significantly change compared to the original. We expect the fabric to bend while forming the desired folds; we therefore anticipate changes in surface normals, as well as some tangential and normal vertex displacement. Consequently, the main phenomenon that we seek to penalize is buckling or bending along an undesirable direction. This requirement can be cast as a restriction that any change in the normal should be orthogonal to the path direction:

$$E^f_{\text{design}} = \frac{1}{2} k_f \sum_i ||T_i - R^j_i T^p_i||^2_F. \quad (9)$$

Here k_f is the stiffness coefficient of the energy, T_i and T^p_i are the local coordinate frames [Sumner and Popović 2004] of the corresponding triangles on the target garment G_t and the 2D patterns P_e, and R^j_i is a 3×3 rotation matrix recomputed at every simulation time step j. We compute R^j_i as the product of the two matrices $R^o_i R^{e,j}_i$, where R^o_i is the rotation extracted from the deformation gradient of the original garment T^o_i using singular value decomposition, and $R^{e,j}_i$ is the projection of the transformation between the original and current time step drapes to the valid space of rotations around the fold line direction. We compute $R^{e,j}_i$ at each simulator time step as follows. We first project the current triangle normal n^j_i in simulation step j onto the plane orthogonal to the fold path to obtain $n^{p,j}_i = \frac{n^j_i - (f_i \cdot n^j_i) f_i}{|n^j_i - (f_i \cdot n^j_i) f_i|}$, where f_i is the path direction on triangle i. Then we compute the angle $\theta^{e,j}_i = acos(n^{p,j}_i \cdot n^o_i)$ between $n^{p,j}_i$ and the normal on the input garment n^o_i. $R^{e,j}_i$ is then given by the matrix that rotates n^o_i to $n^{p,j}_i$ around f_i with $\theta^{e,j}_i$.

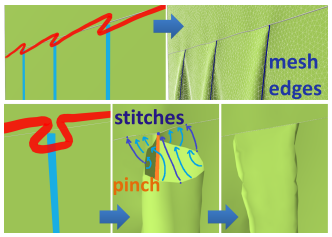
In addition to undesired buckling, we seek to minimize tangental displacements of garment boundaries with respect to the body within the region of interest. This constraint is enforced implicitly for all seam between panels inside and outside the region of influence. For hemlines, we enforce this constraint by augmenting the energy with a term that penalizes tangential shifts:

$$E^f_{\text{design}} = \frac{1}{2} k_f \sum_i ||T_i - R^j_i T^p_i||^2_F + \frac{1}{2} k_u \sum_j \left(((v_j - v'_j) \cdot t^0_j)^2 + ((v_j - v'_j) \cdot t^1_j)^2 \right)$$

Here j iterates over the hemline vertices inside the region of interest, and t^0_j and t^1_j are two orthogonal tangential vectors at vertex j.

To incorporate this term into the cloth simulator, we apply damping to E^u_{design} and E^f_{design} that is similar to the one employed by the simulator when minimizing E_{cloth}, and adjust the magnitudes k_u and k_f of the staging "forces" to bring them to the same scale with the cloth forces. Section 8 provides the specific numbers used for the simulators we tested.

6.1 Initial Drape



When running the augmented simulation for the first time, we need a suitable initial drape that can accommodate our target folds. While the initial garment provides a reasonable starting point for most fold types, pleats require special processing. In particular, we want fabric to fold sharply along pleats, and we want the folding order along them to be preserved (Figure 2). To allow crisp pleats, we refine the mesh along the expected pleat paths starting at the sharp corners of each pleat, and following the fold-path directions (see inset, top). We introduce the correct folding order by stitching the pleat boundary segments in an in-to-out order, instead of all at once, starting from the layer closest to the mannequin. To further penalize interpenetrations, we apply weak spring forces to pull the segments in the outer layer along their normal direction away from the mannequin. This term helps separate the layers, guiding the fabric towards the desired folding order that we want while simultaneously avoiding collisions. The stiffness of these springs is set to be proportional to the gap between the inner stitch pairs, so that they won't pull the outer layered fabrics after the inner layer has been stitched. We emphasize pinched pleats by introducing short 2cm seams orthogonal to the gathering seam at the pinching point (see inset, bottom).

7 2D AND 3D UPDATE

At this stage in the process we have extended patterns and an initial drape that we can use to compute a *simulated garment*, via simple unconstrained simulation. While clearly reproducible, this garment may exhibit undesirable artifacts (Figure 12). We minimize such artifacts by using a two pronged approach that modifies our patterns

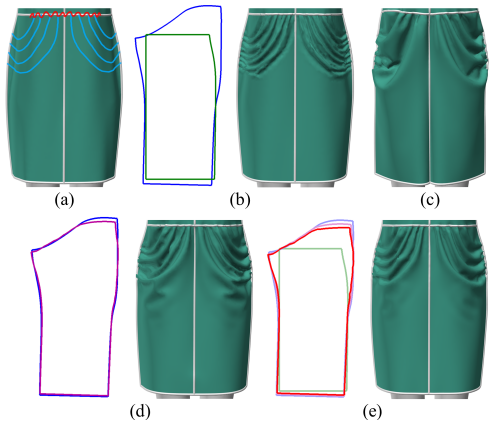


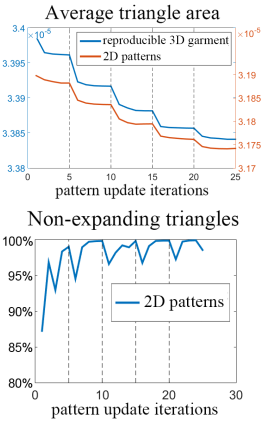
Fig. 12. Pattern and garment update: (a) user input garment and strokes; (b) extended patterns (blue) superimposed on original (green) and initial target drape; (c) unconstrained simulation output using these patterns and drape; (d) updated patterns (purple) superimposed on extended ones (b,blue) and unconstrained simulation output using these new patterns and initial drape; (e) final patterns (red) superimposed on updated (purple) and extended (blue) and final unconstrained drape.

and target drape. First, we modify the 2D patterns to construct new patterns that, under unconstrained simulation, result in an output that is closer to the initial drape. Second, if the result is not sufficiently close, we repeat the constrained simulation with the new patterns generating a new drape, which we can expect to be more physically reproducible. We then iterate until the two steps converge.

Pattern Update. We first look for a new set of 2D patterns that, in an unconstrained simulator without synthetic forces, will produce an output garment as similar as possible to the target drape. We achieve this goal by following the pattern update framework of Bartle et al. [2016]. Specifically, we measure the intrinsic difference between the target drape and the current simulated garment and update the patterns in a manner designed to minimize this difference; we repeat the simulation and update steps until convergence. The output of this stage is a set of patterns and a new simulated garment.

Garment Update. The obtained simulated garments are close to, but not necessarily identical to, the target drape generated using synthetic forces. In particular they may exhibit fine-level deviations such as secondary folds, local sagging, or bulging (Figure 12c). These artifacts are typically due to the fact that the constrained simulation result is not fully physically reproducible, and has material held in place by the synthetic draping forces; without these synthetic forces, this material sags or slides due to gravity. We rectify this problem by computing a new target drape that is more reproducible. We repeat the constrained simulation (Section 6.1) using the new patterns and the simulated garment as the initial drape. The input to this constrained simulation consists of a set of more accurate patterns, and an initial drape (simulated garment) that is both reproducible and better aligned with the synthetic design energy we use. We thus expect the resulting target drape to be much closer to the simulated one, and thus more reproducible. We repeat the pattern and target drape update steps until the simulated garment geometry no longer changes. Each iteration simultaneously improves the physical feasibility of the target garment and the visual similarity between the target and simulated garments by reducing undesirable artifacts on the latter.

Convergence. While there is no theoretical guarantee that the final, unconstrained simulation output garment fully conforms to designer expectations, in our experience this is the case for all inputs where the user specified folds can be feasibly achieved; see Figure 19 for some examples where they cannot. In our experiments, five update iterations were sufficient for convergence. While we similarly have no theoretical guarantees of algorithm convergence, we note that, starting from the initial extended patterns, each pattern update step we perform reduces the area of pattern triangles [Bartle et al. 2016], and each constrained simulation reduces the difference between the target drape and the prior unconstrained simulation output. Consequently, one can see our framework as an example of a fixed-point iteration scheme [Bertsekas 1999]. We validate our convergence claim empirically by measuring the evolution of areas of the triangles on the patterns and the simulated garment, as well as the percentage of pattern triangles that at each iteration either shrink or remain the same with with respect to the previous iteration (see inset). The graphs show consistent and convergent shrinkage behavior consistent with that of a fixed-point scheme.



8 RESULTS AND VALIDATION

We used FoldSketch to generate the multiple examples showcased throughout the paper, created from a range of diverse initial garments including shirts, dresses, pants, shorts, and overalls. We also tested our input on non-garment cloth objects, including a flag and a tissue box cover. Our outputs contain the designer's expected folds, and preserve the input look in regions away from the folds.

The inputs we tested on are representative of a large spectrum of cloth objects, including both tight fitting (e.g. the shirt in Figure 4, and the skirts in Figures 6, 7) and loose garments (e.g dresses in Figure 15). We showcase all four types of folds handled by our UI: hemline folds (e.g. Figures 1 and 13), gathered folds (e.g. skirt in 3, and shirt in 4), knife pleats (e.g. short sides of the tissue box in 13, and blue pants in Figure 14d), and pinched pleats (e.g. orange dress in 16 and yellow dress in 14). We showcase a range of fold-path orientations including vertical (Figure 1, pants in 14), diagonal (skirt in Figure 7, green T-shirt in Figure 14), and horizontal (e.g Figure 13, 12, and shirt in Figure 11).

We generated our examples with four different material settings: a thinner material (e.g. purple skirts in Figure 8, green dress in Figure 15) and a thicker material (e.g. yellow skirt in Figure 17, yellow and purple dresses in Figure 15) in our Sensitive Couture examples; a Sensitive Couture material matching our real-world flag and cover; and the default shirt material supplied by ARCSim.

We further validate the performance of FoldSketch on different materials, by showing examples of the same schematic input applied to garments created from the same initial patterns, but with different, more extreme materials such as silk and leather (Figures 3

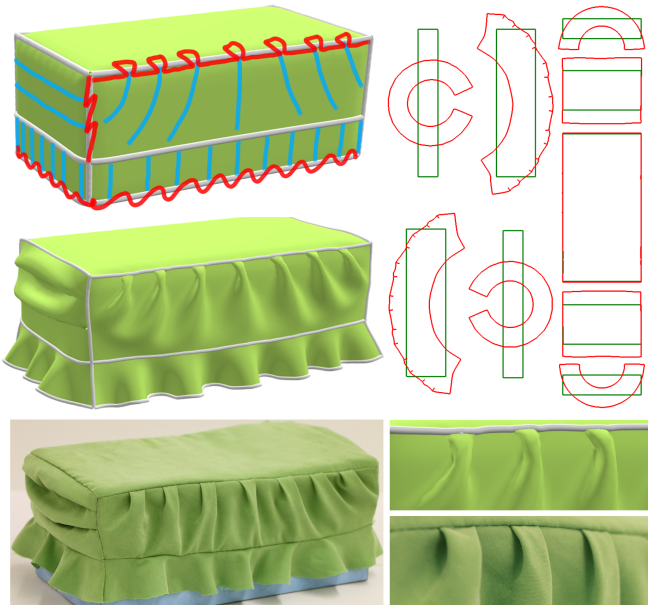


Fig. 13. Manufactured example: (left to right) Plain input tissue box cover with user sketched schematic folds and final post-simulation result augmented with user-expected folds; original (green) and modified final (red) patterns; real-life replica manufactured using the produced patterns, with zooming in highlights the complex and evolving output fold profile shapes.

and 18). While the resulting folds are distinctly different, they still clearly reflect designer intent. This shows our ability to support multiple materials with the same schematic input, provided that the schematic input is reasonable for the chosen material supported by the simulator. The flag and tissue-box examples also show progressive application of multiple fold types and orientations to the same input.

Simulators. We tested our method with two simulation engines, Sensitive Couture [Umetani et al. 2011] and ARCSim [Narain et al. 2013]. Sensitive Couture is optimized for speed over accuracy, whereas ARCSim is optimized for precision at the expense of slower computation times. We show results created with ARCSim in Figure 16; the remainder of the results in the paper were generated using Sensitive Couture, which is faster and provides sufficient accuracy for our needs. To implement multi-component garment meshes in ArcSim, we reimplemented the stitching scheme used by Sensitive Couture. In order to avoid numerical instability, stitches are only defined per vertex of the shorter seam, which allows small holes between seams if the two sides have very different lengths; this occurs when our system produces gathered folds. After simulation concludes, we postprocess these seams for rendering purposes by moving vertices on the longer seam towards their corresponding positions on the shorter seam.

Timing And Parameters. Our input meshes range in size from 10K to 20K triangles. This number is consistent with those used in commercial garment design softwares such as Marvelous Designer, and was chosen to provide a reasonable trade-off between speed and accuracy. Our runtimes using Sensitive Couture Simulator [Umetani et al. 2011] range from 8 to 20 minutes, with the vast majority of the time spent inside the simulation engine. For Sensitive Couture, we set $k_u = 0.08$ and $k_f = 0.1$ for E_{design}, and apply damping with coefficient 0.01. For ARCSim, we set $k_u = 2.0$ and $k_f = 2.5$ for E_{design} with damping coefficient 0.25. These design-preserving energy parameters remain constant for all examples generated with a specific simulator. As different simulators have different scales for their cloth energies, we must scale our parameters to ensure that they have enough force to allow FoldSketch to modify the garment shape and override other simulation elements.

To test our algorithm's scalability, we subdivided the input mesh in Figure 7, creating a new mesh with 54K triangles. Running this model throughout the system took significantly longer - 110 minutes total - but required the same number of iterations to achieve the same error bound and converged to a visually similar result.

Algorithmic Choices. We consider three different alternatives to our algorithm, and show their failure modes. Figure 4, left shows the effect of producing a 3D target garment using purely geometric folds created by folding circular grooves around the designer specified strokes, and then using the physics-aware pattern computation of Bartle et al [2016] to create the patterns. As this figure shows, the intended folds simply collapse during subsequent simulation.

We compare our pattern extension technique to one which scales patterns along the gathering seam and fold paths using standard ARAP deformation, where scales for along the gathering seam and paths are computed using the same process as ours (Section 5.1.3). As Figure 17 shows, the results generated by this approach fail to capture the designer folds while introducing additional unwanted folds and changing the garment style.

Finally, we show the effects of sidestepping the 3D target computation stage and using the initial extended patterns directly within a simulation engine (Figure 6). As shown, simulating a new garment using the extended patterns and a range of standard initial drape configurations results in unappealing garments that do not correspond to the expected garment look. Our final combination of patterns and drape produces a garment that satisfies both our physical reproducibility requirement and design constraints with no unwanted artifacts (Figure 6e).

Designer Validation. Three of our input fold designs (Figure 15) were traced by a professional designer who found our tracing interface easy to use. He commented that the results were "exactly what I expected!", and that "the software you are working on really delivers" and would be very helpful for experts and amateurs alike.

Manufacturing. We manufactured two cloth objects using patterns generated by FoldSketch - a flag (Figure 1) and a tissue-box cover (Figure 13). The resulting manufactured objects clearly contain the designer's expected folds, and were manufactured directly from our generated patterns without further modifications. To evaluate the efficiency of FoldSketch, we asked a professional designer to modify the initial patterns for the tissue-box cover to contain our desired folds. The designer produced an initial set of extended patterns for a fold-enhanced tissue box that would still require several stages of revision in order to be acceptable; this first set of patterns took five hours to complete. The designer was very impressed to note that our complete set of patterns was created in under half an hour, including both design and computation time.

Perceptual Validation. We validate the outputs of our method via feedback from ten non-experts. To collect their input, we presented

Fig. 14. Additional results created using Sensitive Couture.

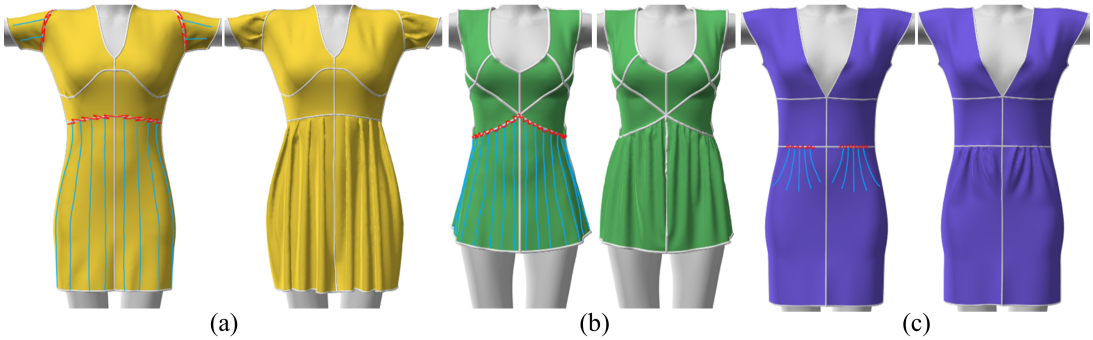


Fig. 15. Folds created from designer annotations.

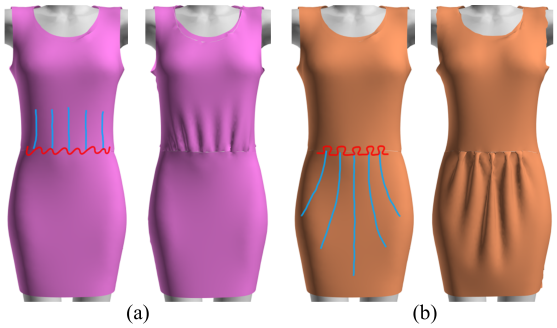


Fig. 16. Examples created using ARCSim.

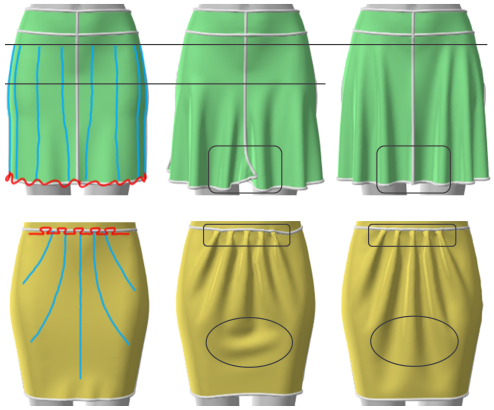


Fig. 17. Comparison with deformation based pattern extension: (left) input, (center) garments simulated using deformation-based extended patterns, (right) our results. The naive approach results in multiple artifacts.

them with a series of pictures of garments, consisting of an input garment with FoldSketch annotations, and two outputs: one of which was generated by FoldSketch, and one of which was generated by one of our design alternatives. We asked the question: "Which of the garments below “B” or “C” is more reflective of the user input “A” above?" Survey participants selected our results as being more representative of the input annotations 95% of the time; please see the supplemental material for more information.

9 CONCLUSIONS

We presented FoldSketch, the first framework that allows designers of both virtual and real garments to create physically reproducible folds and pleats via a simple 3D sketching interface, eliminating the need for tedious and unintuitive 2D pattern manipulation. We demonstrated the applicability of our framework on a large range of fold and pleat configurations, and confirmed its ability to operate in conjunction with different simulation engines. The key novel technical components of FoldSketch are a 3D to 2D pattern extension algorithm that translates user sketched schematic folds into per-triangle deformation gradients applied to the triangles of the original patterns, a constrained simulation framework that incorporates design constraints into off-the shelf garment simulators, and an update mechanism that enables the correction of secondary simulation artifacts.

Limitations and Future Work. Our system follows user specifications in an effort to generate the folds they desire. While we do pre-process the garments to better accommodate folds in tight-fitting regions (Section 5.1.4) we cannot guarantee that folds that

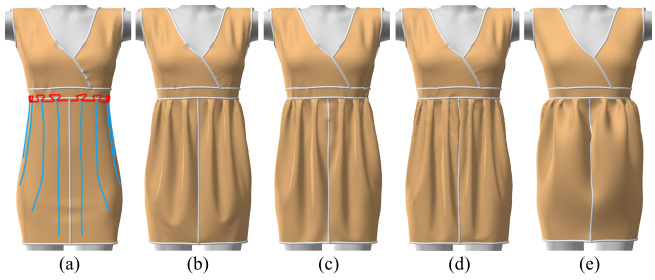


Fig. 18. Extreme material experiment: user input garment and strokes (a), final unconstrained drape with thinner textiles (b), stretchy knits (c), silk (d), and thick leather (e).

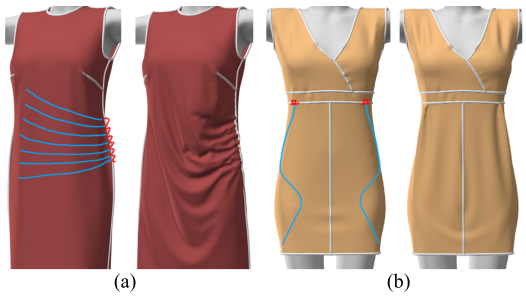


Fig. 19. Our method fails to detect infeasible inputs such as long horizontal fold-paths in loose garment regions (a), or highly curved independent fold-path that would need more stitches along the path, or special fabrics to support (b).

require more significant pattern edits will be generated. For future work, it would be interesting to explore detection of infeasible inputs, something our current system does not do, and to provide feedback to the designer. This would be particularly useful for amateur designers who have no sense of what folds are feasible. Our system is designed for creating folds achievable through changes in pattern shape alone. Future work on incorporating topological changes, such as the introduction of darts and gussets, can extend the range of achievable designs. Future work on material modeling, such as searching for a space of reasonable material parameters for a specific design, or applying adaptive stitching to achieve consistent design across different materials, would benefit many practical applications.

ACKNOWLEDGEMENTS

This work has been funded by NSERC and NSF under Grant No. 1409286. We are grateful to Damien Rohmer for generating comparison data, Mary Buckland for manufacturing the flag and tissue box cover, and Alain Chavez for providing designer inputs and feedback. We would like to thank Silver Burla, Yuan Yao, Enrique Rosales, Qingyue Wang, and Xinyu Li for their help with the project. We would also like to thank Aric Bartle, Vladimir G. Kim, Danny M. Kaufman, and Floraine Berthouzoz for providing their code and garment data.

REFERENCES

Janet Arnold. 1985. *Patterns of Fashion: The cut and construction of clothes for men and women c1560-1620.* Macmillan.
Uri M Ascher and Chen Greif. 2011. *A First Course on Numerical Methods.* SIAM.
Aric Bartle, Alla Sheffer, Vladimir G. Kim, Danny M. Kaufman, Nicholas Vining, and Floraine Berthouzoz. 2016. Physics-driven Pattern Adjustment for Direct 3D Garment Editing. *ACM Trans. Graph.* 35, 4, Article 50 (2016), 50:1–50:11 pages.
Dimitri P Bertsekas. 1999. *Nonlinear programming.* Athena scientific Belmont.
R. Bridson, S. Marino, and R. Fedkiw. 2003. Simulation of Clothing with Folds and Wrinkles. In *Proc. Symposium on Computer Animation.* 28–36.
Remi Brouet, Alla Sheffer, Laurence Boissieux, and Marie-Paule Cani. 2012. Design Preserving Garment Transfer. *ACM Trans. Graph.* 31, 4, Article 36 (2012), 36:1–36:11 pages.
Solutions Pte Ltd. Browzwear. 2017. Browzwear. https://browzwear.com/
Virtual Fashion Inc. CLO. 2017. Marvelous Designer. https://www.marvelousdesigner.com/
Frederic Cordier, Hyewon Seo, and Nadia Magnenat-Thalmann. 2003. Made-to-measure technologies for an online clothing store. *IEEE Computer graphics and applications* 23, 1 (2003), 38–48.
Keenan Crane, Mathieu Desbrun, and Peter Schröder. 2010. Trivial Connections on Discrete Surfaces. *Computer Graphics Forum* 29, 5 (2010), 1525–1533.
Chris De Paoli and Karan Singh. 2015. SecondSkin: sketch-based construction of layered 3D models. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 126.
Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. 2006. Virtual garments: A fully geometric approach for clothing design. In *Computer Graphics Forum*, Vol. 25. 625–634.
Optitex EFI. 2017. OptiTex PDS. http://optitex.com/
Marzia Fontana, Alberto Carubelli, Caterina Rizzi, and Umberto Cugini. 2005. ClothAssembler: a CAD module for feature-based garment pattern assembly. *Computer-Aided Design and Applications* 2, 6 (2005), 795–804.
Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.
Karolyn Kiisel. 2013. *Draping: the complete course.* Laurence King Publishing.
Tsz-Ho Kwok, Yan-Qiu Zhang, Charlie CL Wang, Yong-Jin Liu, and Kai Tang. 2016. Styling evolution for tight-fitting garments. *IEEE transactions on visualization and computer graphics* 22, 5 (2016), 1580–1591.
Changjian Li, Hao Pan, Yang Liu, Alla Sheffer, and Wenping Wang. 2017. BendSketch: Modeling Freeform Surfaces Through 2D Sketching. *ACM Trans. Graph. (SIGGRAPH)* 36, 4 (2017), 125:1–125:14.
Jituo Li and Guodong Lu. 2014. Modeling 3D garments by examples. *Computer-Aided Design* 49 (2014), 28–41.
Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A Local/Global Approach to Mesh Parameterization. *Computer Graphics Forum* 27, 5 (2008), 1495–1504.
Yuwei Meng, PY Mok, and Xiaogang Jin. 2010. Interactive virtual try-on clothing design systems. *Computer-Aided Design* 42, 4 (2010), 310–321.
Rahul Narain, Tobias Pfaff, and James F. O'Brien. 2013. Folding and Crumpling Adaptive Sheets. *ACM Trans. Graph.* 32, 4, Article 51 (2013), 51:1–51:8 pages.
Rahul Narain, Armin Samii, and James F. O'Brien. 2012. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Trans. Graph.* 31, 6, Article 152 (2012), 152:1–152:10 pages.
Ulrich Pinkall and Konrad Polthier. 1993. Computing Discrete Minimal Surfaces and Their Conjugates. *Experimental Mathematics* 2, 1 (1993), 15–36.
Tiberiu Popa, Quan Zhou, Derek Bradley, Vladislav Kraevoy, Hongbo Fu, Alla Sheffer, and Wolfgang Heidrich. 2009. Wrinkling Captured Garments Using Space-Time Data-Driven Deformation. In *Computer Graphics Forum*, Vol. 28. 427–435.
Nicolas Ray, Bruno Vallet, Laurent Alonso, and Bruno Levy. 2009. Geometry-aware Direction Field Processing. *ACM Trans. Graph.* 29, 1, Article 1 (2009), 1:1–1:11 pages.
Cody Robson, Ron Maharik, Alla Sheffer, and Nathan Carr. 2011. Context-aware Garment Modeling from Sketches. *Comput. Graph.* 35, 3 (2011), 604–613.
Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. 2010. Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-looking Wrinkles. *ACM Trans. Graph.* 29, 6, Article 157 (2010), 157:1–157:8 pages.
Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4.
Robert W. Sumner and Jovan Popović. 2004. Deformation Transfer for Triangle Meshes. *ACM Trans. Graph.* 23, 3 (2004), 399–405.
Emmanuel Turquin, Marie-Paule Cani, and John Hughes. 2004. Sketching garments for virtual characters. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling.* Eurographics, Grenoble, France.
Emmanuel Turquin, Jamie Wither, Laurence Boissieux, Marie-Paule Cani, and John F. Hughes. 2007. A Sketch-Based Interface for Clothing Virtual Characters. *IEEE Comput. Graph. Appl.* 27, 1 (2007), 72–81.
Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. 2011. Sensitive Couture for Interactive Garment Modeling and Editing. *ACM Trans. Graph.* 30, 4, Article 90 (2011), 90:1–90:12 pages.
Pascal Volino, Frederic Cordier, and Nadia Magnenat-Thalmann. 2005. From Early Virtual Garment Simulation to Interactive Fashion Design. *Comput. Aided Des.* 37, 6 (2005), 593–608.
Charlie CL Wang, Yu Wang, and Matthew MF Yuen. 2003. Feature based 3D garment design through 2D sketches. *Computer-Aided Design* 35, 7 (2003), 659–672.
Charlie CL Wang, Yu Wang, and Matthew MF Yuen. 2005. Design automation for customized apparel products. *Computer-aided design* 37, 7 (2005), 675–691.
E. Zhang, J. Hays, and G. Turk. 2007. Interactive Tensor Field Design and Visualization on Surfaces. *IEEE Transactions on Visualization and Computer Graphics* 13, 1 (2007), 94–107.