FoldSketch: Enriching Garments with Physically Reproducible Folds

by

Minchen Li

B.Eng. in Computer Science and Technology, Zhejiang University, 2015

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

April 2018

Abstract

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. This thesis presents *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, the main algorithmic challenge here is that the target geometry is missing. 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; 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.

FoldSketch obtains 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; then we use these patterns to produce an estimated fold-enhanced target drape geometry that balances designer expectations against physical reproducibility. Next, we generate an initial reproducible output using the expanded patterns and the estimated target drape as input to a garment simulation engine. Then we improve the output's alignment with designer expectations by progressively refining the patterns and the estimated target drape, converging to a final fully physically reproducible fold-

enhanced garment. The experiments confirm that FoldSketch reliably converges to a desired garment geometry and corresponding patterns and drape, and works well with different physical simulators. My collaborators and I demonstrate the versatility of this approach by showcasing a collection of garments augmented with diverse fold and pleat layouts specified via the FoldSketch interface, and further validate this approach via feedback from potential users.

Preface

The ideas and algorithms described in this thesis, unless stated below, were developed by myself in consultation with Prof. Alla Sheffer and Prof. Eitan Grinspun.

The designer validation and perceptual validation mentioned in Section 8.4 and Section 8.6 were conducted by Prof. Alla Sheffer, Nicholas Vining, and myself with UBC approval (UBC BREB Number H16-02320).

The manufacturing mentioned in Section 8.5 is conducted by Mary Buckland.

The *we* and *our* used in this thesis express my acknowledgement and respect to all my collaborators.

All the ideas and algorithms described will be published as a research paper in ACM TOG and presented at a conference (SIGGRAPH) in August 2018:

- Minchen Li, Alla Sheffer, Nicholas Vining, Eitan Grinspun. FoldSketch: Enriching Garments with Physically Reproducible Folds. *Conditionally accepted to SIGGRAPH 2018.*

Table of Contents

List of Figures

Acknowledgments

I would like to express my sincere thanks to my academic advisor Prof. Alla Sheffer for her continuous support of my graduate study and research throughout the past two years. Without her guidance and encouragement, this thesis would never exist. I also thank her for providing me many oppoortunities for my career.

Furthermore, I would like to thank Prof. Eitan Grinspun for helpful discussion, insight and feedback.

Thanks to Nicholas Vining, Mary Buckland, Silver Burla, Yuan Yao, Enrique Rosales, Qingyue Wang, and Xinyu Li for the help when was most needed and for complementing my skills.

Thanks to Aric Bartle, Vladimir G. Kim, Danny M. Kaufman, Floraine Berthouzoz, and Nobuyuki Umetani for providing their code and garment data, and thanks to Damien Rohmer for generating output of previous work for comparison.

Most of all, I would like to deeply thank my parents, who encouraged and supported my choice of studying far from home.

Chapter 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 [1, 16]. We propose *FoldSketch*, a new algorithmic framework for fold and pleat generation that enables users to directly generate their desired fold-enhanced 3D garments without the need for manual pattern editing or draping (Figure 1.1). 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 1.2). 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

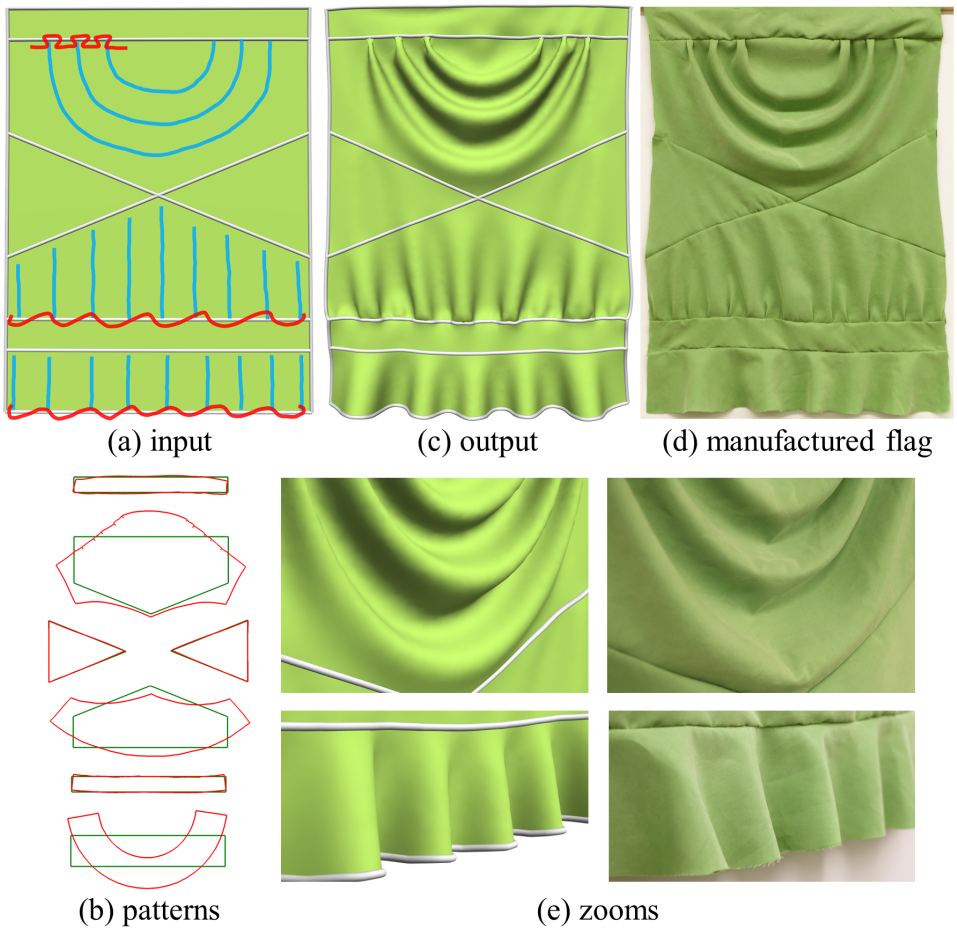

Figure 1.1: Complex physically simulated fabric folds generated using FoldSketch: Plain input flag with user sketched schematic folds (a), original (green) and modified final (red) patterns (b); final post-simulation flag augmented with user-expected folds (c); real-life replica manufactured using the produced patterns (d); zooming in highlights the complex and evolving output fold profile shapes (e).

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 1.1) that depends on a range of physical factors such as fabric stretchiness, thickness, and bending flexibility (Figure 2.1). While designers have an overall

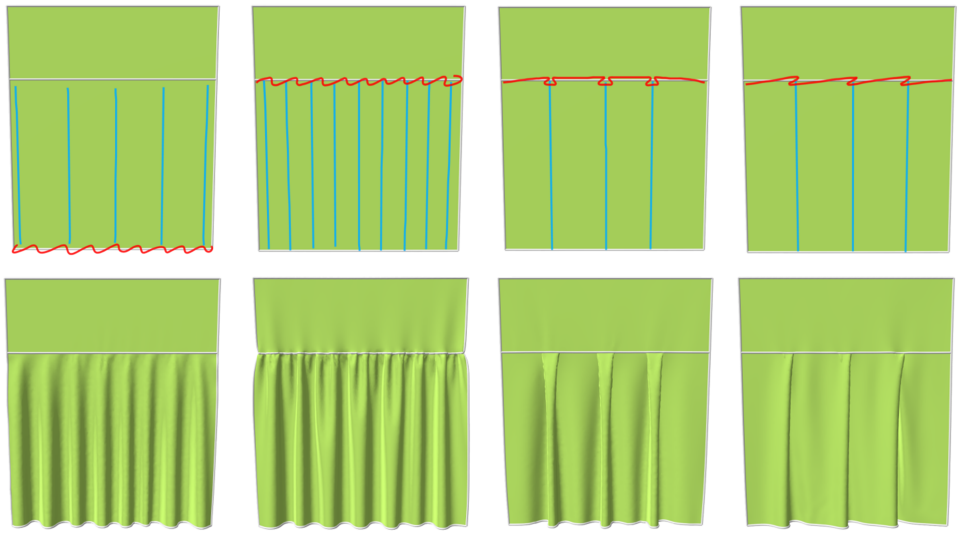

Figure 1.2: Different types of folds and pleats supported by our framework: (top) schematic user sketch; (bottom) output folds.

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 1.1a, 1.2). 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 1.1d).

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 [32, 28, 27, 33]

(Figure 2.2,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 [3] is likely to fail ((Figure 2.2,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 [3, 1]. 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 (Chapter 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 (Chapter 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 2.3). We

minimize these artifacts by progressively updating both the patterns and the target drape, reducing the difference between augmented and unconstrained simulation outputs (Chapter 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 [34] and ARCSim [21, 22], one of which is optimized for speed while the other is optimized for accuracy (Figures 8.2, 8.4). 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 1.1, 8.1).

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, 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.

Chapter 2

Related Work

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

2.1 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 1.2); 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 [1, 16]. 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

Figure 2.1: 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).

to fold design techniques [16].

Commercial garment design software, such as [8, 7, 14, 13, 35], 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.

2.2 Computational Garment Design

Recent algorithms enable procedural modeling of static virtual garments that have the appearance of clothing [36, 32, 12, 33, 27, 19, 11, 17].

Some of these methods use sketching interfaces to produce low frequency garments [36, 32, 12, 33, 27, 11]. Specifically, Wang et al. [36] propose a feature-based garment modeling method that can enable users to draw 2D sketches in 3D, using the extracted mannequin features to specify garment profiles and patterns. Tuiquin et al. [32, 33] use fine sketch inputs within multiple types of strokes, leading to a more accurate modeling of garment profiles. Decaudin et al. [12] further

Figure 2.2: Existing fold modeling methods (here [28]) 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).

enhance this sketching interface by enabling the user to draw seam-lines and darts. Moreover, to interpret sketches more consistently over different views, Robson et al. [27] incorporate context-aware considerations based on key factors that affect the shape of garments. In addition, DePaoli et al. [11] propose a sketch-based modeling system to create structures that are comprised of layered and shape interdependent 3D volumes. This method potentially inspires the emergence of tools for multi-layered garment creation.

Others support mixing of existing garment elements in 3D space [19, 17]. For instance, Li et al. [19] model new garments on individual human models by compositing 3D parts from garment examples. In the meanwhile, Kwok et al. [17] generate new and reasonable styling designs from existing garments based on evolution theory. The garments produced by them have no physics aware patterns, no physical parameters, and are often not physically reproducible; they are thus unsuitable for cloth simulation or manufacturing.

Grading methods algorithmically resize garments to fit a mannequin different from the one they were originally designed for [9, 37, 6]. Cordier et al. [9] and Wang et al. [37] resize the garment based on element correspondence between garment and mannequin. Brouet et al. [6] also explicitly define style measurement of shape, fit, and proportion to be preserved during grading. These methods can

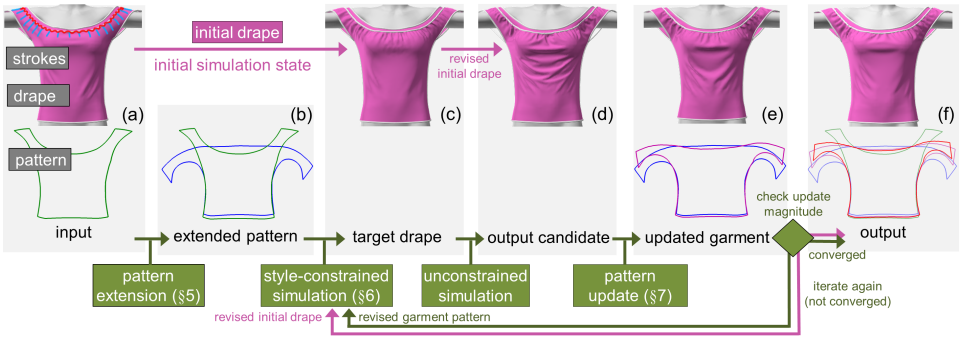

Figure 2.3: FoldSketch Algorithm: (left to right) The *input* consists of a garment pattern, a corresponding drape on a mannequin, and designer-sketched folds. Folds require additional fabric material, which is obtained by *extending the pattern*. Draping depends heavily on the *initial drape* of the 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.

potentially transfer the location and 3D shape of existing folds to a new garment, but are not designed for forming new folds.

Sensitive Couture [34] 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 2.1) and external forces.

2.3 Fold and Wrinkle Modeling

Computer animation frameworks produce dynamic folds whose location and shape are determined by the physical forces rather than the user [5]. 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 [33, 27] model folds by sweeping a cylindrical profile along a user sketched path. Turquin et al. [33] directly move the garment mesh vertices away or close to the mannequin according to user specified fold orientation, depth, width and location with their fold-sketching UI. In contrast, Robson et al. [27] model the profile by rotating the nearby triangles along the path and re-solve for surface normals of the garment.

Cylindrical arc profiles, smoothly blended with the surrounding surface, are similarly used to procedurally model dynamic folds to augment low-quality animations [24, 28]. In [24], folds are added to enhance details for garment animation capturing techinques, where the fold locations are computed from the frames. In contrast, the input in [28] is an existing garment animation, of which the stretch tensors are used to guide fold formation. Both methods ensure temporal coherence with a space-time approach allowing for smooth and natural wrinkle behavior.

However, 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 2.2a). Even with BendSketch [18] that translates user input into actual surface detail geometry, it is still hard for users to draw fold profiles that are physically reliable. More importantly, fold-enhanced garments generated using these approaches have no corresponding patterns, and often are not physically reproducible. Figure 2.2b shows the result of attempting to reproduce the geometry in Figure 2.2a by computing physics-aware patterns [3] 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 2.2,cd).

Chapter 3

Algorithm

3.1 Problem Statement

The input to our algorithm is a simulated 3D garment, draped around a character or mannequin, and its corresponding set of 2D patterns (Figure 2.3,a). Using FoldSketch, designers specify their desired fold configuration by sketching on top of this input. They draw *path strokes* (Figure 1.2, blue) to indicate the direction and length of their desired folds, and schematic *gathering* strokes (Figure 1.2, 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 (Chapter 4, Figure 1.2). 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 2.3,d).

3.2 Solution Framework

We approach this problem using an alternating 2D-3D process, inspired by traditional garment design practices (Figure 2.3). We first create an initial set of extended patterns. We then create an initial, synthetic, target drape that utilizes

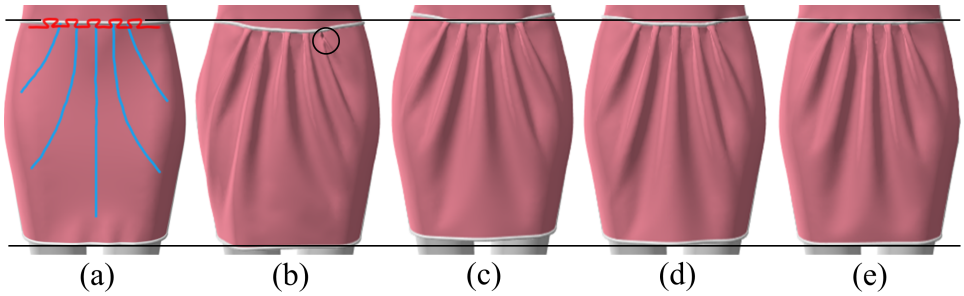

Figure 3.1: Input garment and strokes (a) and unconstrained simulation outputs produced using different initial drapes: (b) from flat panels; (c) from initial 3D garment (augmented with pleat sewing scheme); (d) from our initial 3D target drape. (e) Final result (using iteratively updated patterns and target drape). The horizontal lines highlight the garment proportions.

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 alignment with designer expectations by alternatingly refining the patterns and the target drape, and generate the final output (Figure 2.3, right) by performing one more round of unconstrained simulation using those.

3.3 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 (Chapter 5, Figure 2.3,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.

3.4 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 3.1), 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 2.3c, Chapter 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 (Chapter 6.2).

3.5 2D-3D 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 (Chapter 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.

Chapter 4

Sketch Interface

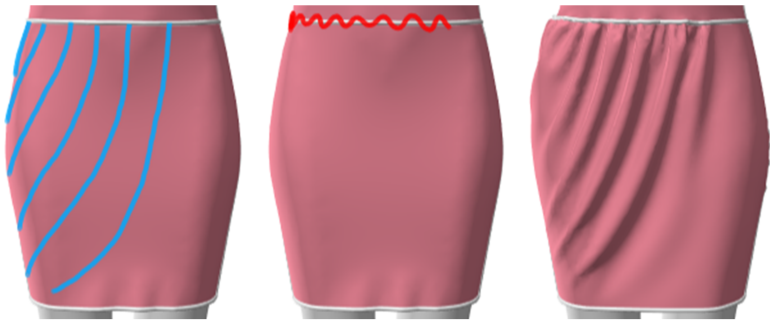

Figure 4.1: UI walkthrough example: (a) fold path strokes (blue) traced over input garment in 3D view; (b) gathering stroke; (c) output garment.

This chapter illustrates FoldSketch's UI with a worked example. Starting with a draped garment on a mannequin (Figure 4.1, 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, and adding a *gathering stroke* (Figure 4.1, b) atop of the corresponding boundary.

This stroke indicates the amount of extra material that must be incorporated into the folds; in this case, uniformly gathered folds is to be added. The system then synthesizes a new garment (Figure 4.1, 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 reason for sketching the gathering configuration and not the fold profile (cross-section) elsewhere along the fold paths is that designers know the gathering configuration they want to use; while the fold cross-section varies in shape based on fabric physics. This interface enables naturally employing schematic input to augment garments with different fabric properties.

The design tool provides four types of folds (Figure 1.2): hemline folds, uniformly gathered folds, pinched pleats, and knife pleats. In all cases, the process is the same: the designer chooses a fold type, 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. 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, fabric needs to be folded onto itself and the overlapping portions need to be stitched together. We use the sharp corners in the gathering stroke to dictate the location and magnitude of the pleats (See Figure 4.2, e and f). For knife pleats, the user can also choose between right or left foldovers (See Figure 4.2, f). Designers can specify folds that extend between two gathering boundaries (Figure 2.2). 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 [28]. Therefore designers are only expected 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 locations. In contrast, for pleats we use the stroke geometry to dictate the fabric folding and stitching pattern,

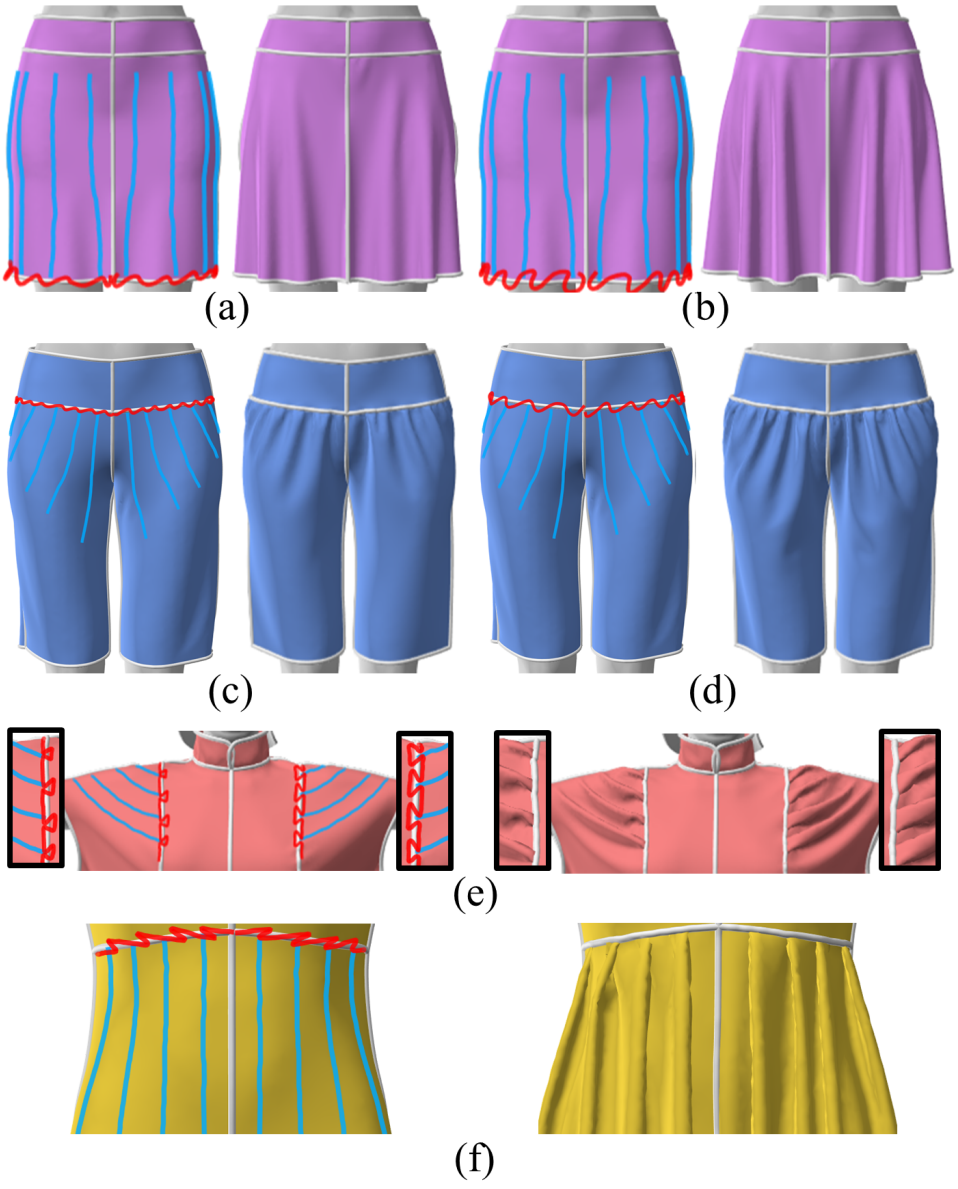

Figure 4.2: Impact of different gathering strokes on output folds: (a,b) hemline folds with different magnitude, (c,d) gathered folds with different magnitude, (e) pinched pleats with different width (left and right shoulders), (f) knife pleats with different orientation (left and right panels).

determining their location, magnitude, and orientation (Figure 4.2).

In many of the cases, the input garment contains pairs of symmetric patterns, which enables users to design symmetric folds. To support convenient symmetric fold design, FoldSketch can reflect the user strokes from one pattern to its symmetric pattern (e.g. all examples in Figure 4.2). This also proves that our system can handle fold enhancement on multiple patterns simultaneously, while in this thesis with the prototyping FoldSketch, all examples with folds designed on multiple patterns (except symmetric designs) are enhanced successively (e.g. Figure 1.1, Figure 8.1, Figure 8.2a, Figure 8.3a).

Chapter 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 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).

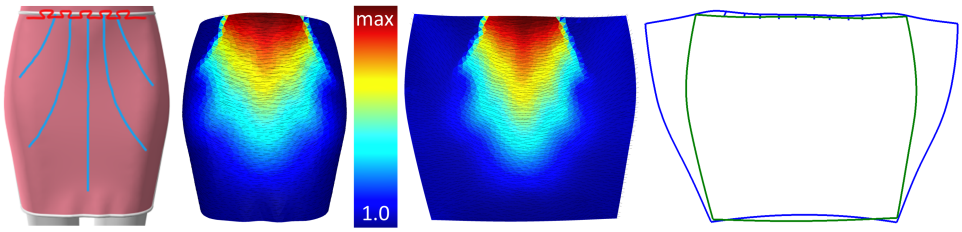

Figure 5.1: Pattern extension: (a) input garment and strokes; (b) visualized 3D space scaling field; (c) scaling field on 2D patterns; (d) extended patterns (blue) and input patterns (green).

5.1 Tensor Field Computation

We use stretch tensor field to represent the target expansion of patterns, thus we are essentially solving a sketch based tensor field design problem [38, 31]. 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 [38], 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.

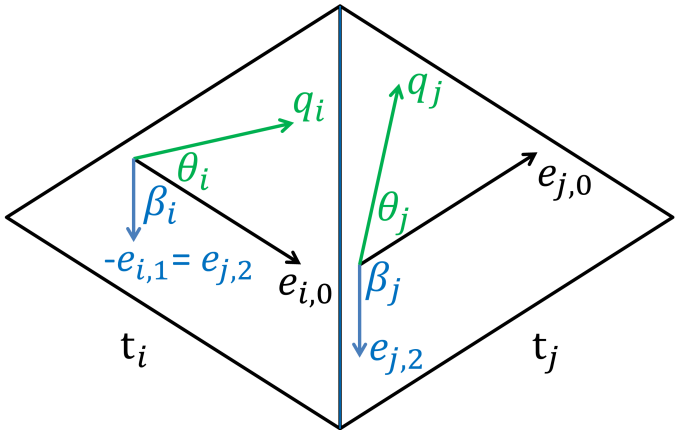

Figure 5.2: $e_{i,k}$ is the k-th edge of t_i, q_i is the vector defined in the local frame of t_i, the discrete Levi-Civita connection is defined as $r_{ij} = \beta_i - \beta_j$

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:

$$\begin{aligned} E &= E_{\text{smooth}} + E_{\text{fit}} \\ E_{\text{smooth}} &= \sum_{ij \in \mathscr{E}^*} w_{ij}^{-1} (V_i - R(2r_{ij})V_j)^2 \\ E_{\text{fit}} &= (1/\overline{|t_i|}) \sum_i |t_i| (V_i - V_i^{\text{init}})^2 \end{aligned} \tag{5.1}$$

where $\mathscr{E}^*$ is the set of all adjacent triangle pairs.

The term E_{fit} is evaluated over all triangles i crossed by fold paths. Here w_{ij} are the standard cotangent weights [23]; $r_{ij} \in \mathbb{R}$ is the discrete Levi-Civita connection [10], 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 $\theta_i = atan(V_i \cdot (0,1)/V_i \cdot (1,0))/2$. This iterative renormalization could robustly achieve acceptable accuracy with just a few more iterations compared to directly enforcing the nonlinear constraints.

One may consider directly taking the angles as optimization variable like in [25] or [10]. However, it turns out that [26] is most suitable for our settings since in [25], an iterative framework is still needed for handling integer variables in the angles for invariance under 180°, and in [10], 2-RoSy fields need to be handled by specifying singularities with fractional indices while in our case the singularities might be outside the garment surface and we really want it to be handled automatically.

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 [2] to perform the tracing. Specifically, we are solving an initial value problem of the ordinary differential equation

$$\frac{db_p}{dt} = f(b_p(t)), \quad b_p(0) = p_{e,i}$$

, 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 applied the midpoint RK2 scheme to evaluate

$$b_p(t_{n+1}) \leftarrow b_p(t_n) + \Delta t f(b_p(t_n) + 0.5\Delta t f(b_p(t_n)))$$

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:

$$\min_s E = \underbrace{\sum_{ij\in\mathscr{E}^*} w_{ij}^{-1}(s_i - s_j)^2}_{\text{Smoothness}} + \underbrace{\lambda_l E^{len}}_{\text{Boundary length preservation}} + \underbrace{\lambda_v E^{FL}}_{\text{Fold path scaling}} + \underbrace{\lambda_v/\overline{|t_i|}\sum_{i\in\mathscr{B}}|t_i|(s_i-1.0)^2}_{\text{Scale preservation outside region of interest}} \tag{5.2}$$

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 s_i 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:

$$E^{FL} = 1/\overline{|t_i|}\sum_{i\in\mathscr{F}}|t_i|(s_i - s_i')^2 \tag{5.3}$$

where $\mathscr{F}$ 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:

$$l_i(s_i) = l_i' \sqrt{s_i^2 \cos^2 \alpha_i + \sin^2 \alpha_i}. \tag{5.4}$$

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 non linear and thus hard to optimize; we therefore linearly approximate Eq.5.4 around $s_i = 1$ instead:

$$l_i^a(s_i) = l_i' + (\frac{\partial l_i}{\partial s^t})_{s^t=1}(s_i - 1) \tag{5.5}$$

We express boundary length preservation as:

$$E_j^{len} = 1/\overline{l_i'} \sum_{i \in \mathscr{E}(j)} (l_i^a(s_i) - l_i')^2 \tag{5.6}$$

Here $\overline{l_i'}$ is the average boundary edge length, and $\mathscr{E}(j)$ are the participating boundary edges. This expression is equivalent to enforcing the constraint that $s_i = 1.0$, $i \in \mathscr{E}(j)$ 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.

Linear Solver. Since both optimizations formulated for solving direction field and scalar field are linear optimization, the optimum is found by solving the linear system that enforces the energy gradient to be equal to 0 using Sparse Cholesky implementation in the Eigen library [15] as the coefficient matrix is symmetric positive-definite.

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 5.3). 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 with $R(\theta_i)S_iR(\theta_i)^T$, 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 $T_i^t\tilde{t}_i = T_i^t T^{d0}t_i = T^{d0}T_i't_i$. We can consequently compute T_i' as

$$T_i' = (T^{d0})^{-1}T_i^t T^{d0} \tag{5.7}$$

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 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 5.3 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

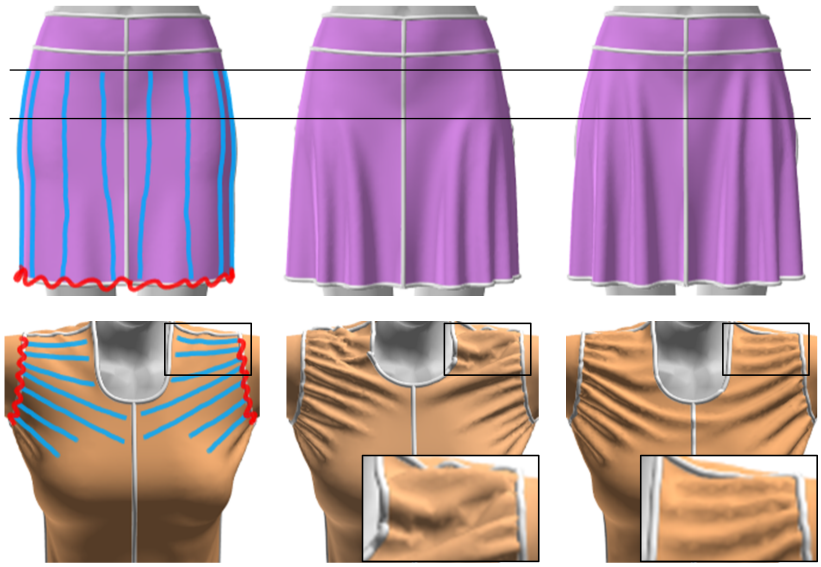

Figure 5.3: Impact of fold pre-conditions: (top) result without and with orthogonal stretch cancellation; (bottom) result without and with fold path stretching. Note the impact on fold length and shape.

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 5.3 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.5.7 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:

$$(T_i^s)^{-1} T_i^{d0} T_i^0 = T_i^t T_i^{d0} \tag{5.8}$$

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:

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

Although obtaining the 2D tensor field by first projecting the strokes onto 2D patterns and then solve for the fields in 2D would be less complicated, the mechanism described in this section is more in demand. Since cloth simulation is a nonlinear process, the 2D to 3D map T_i^{d0} is also nonlinear: computing T_i^0 by smoothly averaging directions in 3D and projecting them to 2D is not equivalent to smoothly averaging in 2D. What's more, only solving in 2D space will make it hard to do fold pre-conditions because there would be no access to direct manipulation of tensors in design space - the tangent space of 3D garment surface.

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 [29, 20]. 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:

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

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 [29], where R_i is optimized in local step using singular value decompo-

sition, and S_i is optimized in global step by solving a symmetric positive-definite linear system using the Sparse Cholesky implementation in the Eigen library [15].

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-1} 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 $e_j = v_j - v_{j-1}$ is the unknown variable.

For straight boundaries, we use 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.

Chapter 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 3.1 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 3.1c). To improve this garment's adherence to designer expectations, we update the patterns and recompute the drape (Chapter 7, 3.1d).

We introduce synthetic design-preserving forces by augmenting the energy for-

mulation 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 [34] and ArcSim [22], as discussed in Chapter 8.

6.1 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:

$$E^f_{\text{design}} = \frac{1}{2} k_f \sum_i ||T_i - R^j_i T^p_i||^2_F. \tag{6.1}$$

Here k_f is the stiffness coefficient of the energy, T_i and T^p_i are the local coordinate frames [30] 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_i^j as the product of the two matrices $R_i^o R_i^{e,j}$, where R_i^o is the rotation extracted from the deformation gradient of the original garment T_i^o using singular value decomposition, and $R_i^{e,j}$ 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 $R_i^{e,j}$ at each simulator time step as follows. We first project the current triangle normal n_i^j in simulation step j onto the plane orthogonal to the fold path to obtain $n_i^{p,j} = \frac{n_i^j - (f_i \cdot n_i^j) f_i}{|n_i^j - (f_i \cdot n_i^j) f_i|}$, where f_i is the path direction on triangle i. Then we compute the angle $\theta_i^{e,j} = acos(n_i^{p,j} \cdot n_i^o)$ between $n_i^{p,j}$ and the normal on the input garment n_i^o. $R_i^{e,j}$ is then given by the matrix that rotates n_i^o to $n_i^{p,j}$ around f_i with $\theta_i^{e,j}$.

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:

$$E^f_{\text{design}} = \frac{1}{2} k_f \sum_i ||T_i - R_i^j T_i^p||_F^2 + \frac{1}{2} k_u \sum_j \left(((v_j - v'_j) \cdot t_j^0)^2 + ((v_j - v'_j) \cdot t_j^1)^2 \right)$$

Here j iterates over the hemline vertices inside the region of interest, and t_j^0 and t_j^1 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. Chapter 8 provides the specific numbers used for the simulators we tested.

6.2 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 1.2). To allow crisp pleats, we refine the

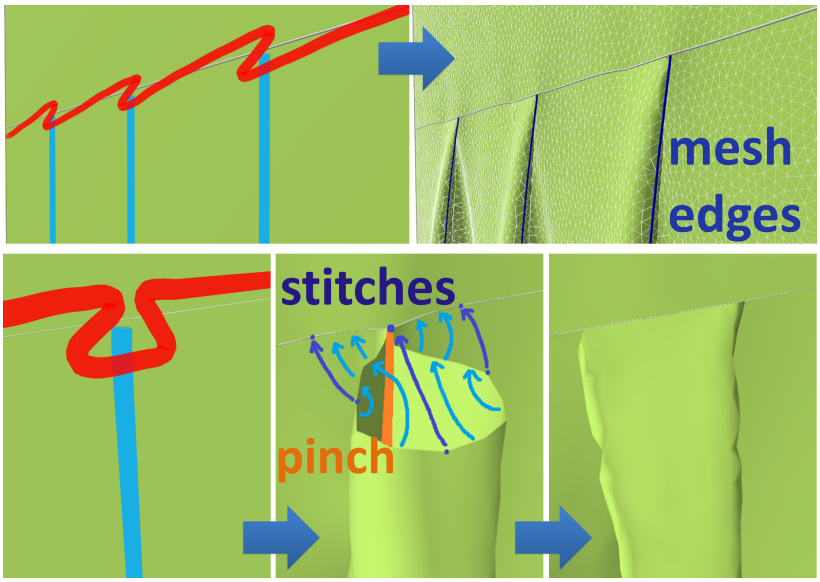

Figure 6.1: Top: crisp knife pleats; bottom: pinched pleats sewing.

mesh along the expected pleat paths starting at the sharp corners of each pleat, and following the fold-path directions (see Figure 6.1, 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 Figure 6.1, bottom).

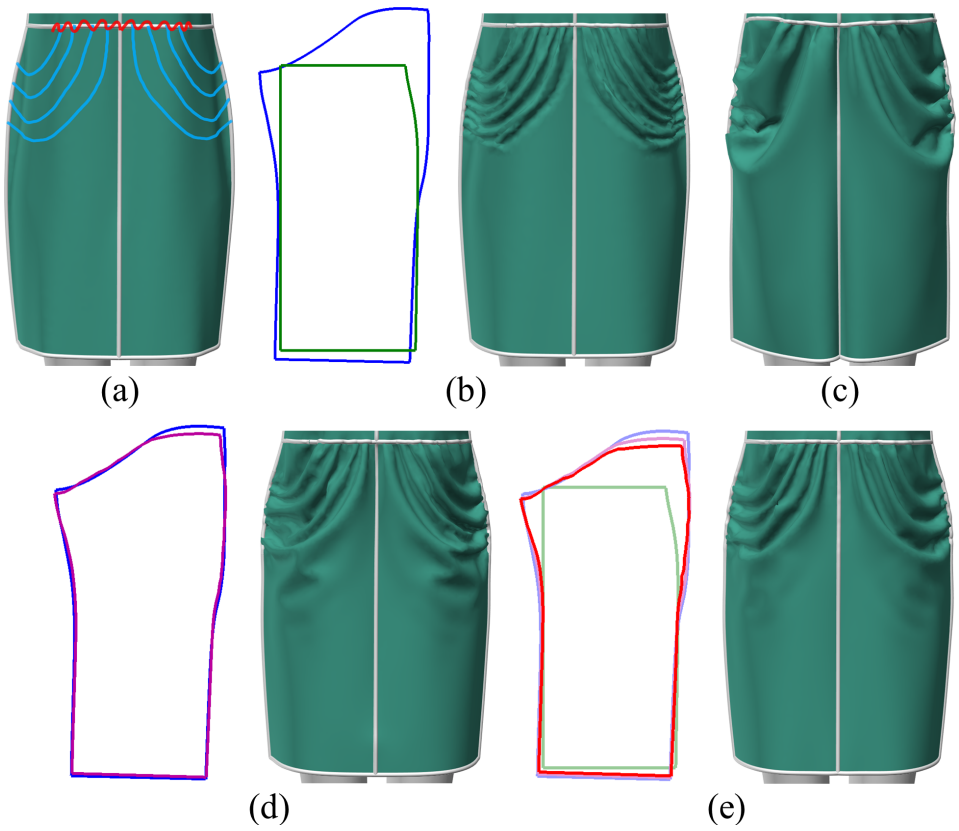

Figure 6.2: Pattern and garment update: (a) user input garment and strokes; (b) extended patterns (blue) superimposed on original (green) and initial target drape; (c) unconstrained simulation output using these patterns and drape; (d) updated patterns (purple) superimposed on extended ones (b,blue) 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.

Chapter 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 6.2). We minimize such artifacts by using a two pronged approach that modifies our patterns and target drape. First, we modify the 2D patterns to construct new patterns that, under unconstrained sumulation, 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.

7.1 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.

7.2 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 6.2c). 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 6.2) 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.

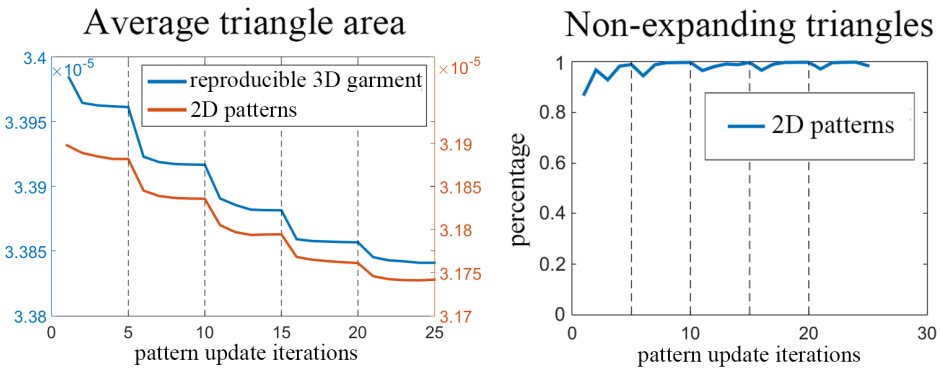

Figure 7.1: Left: average triangle area per pattern update iteration; right: percentage of non-expanding triangles per pattern update iteration.

7.3 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 9.1 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 [3], 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 [4]. 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 Figure 7.1). The graphs show consistent and convergent shrinkage behavior consistent with that of a fixed-point scheme.

Chapter 8

Results and Validation

We used FoldSketch to generate the multiple examples showcased throughout this thesis, created from a range of diverse initial garments including shirts, dresses, pants, shorts, and overalls. We also tested input on non-garment cloth objects, including a flag and a tissue box cover. The 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 2.2, and the skirts in Figures 3.1, 4.1) and loose garments (e.g dresses in Figure 8.3). We showcase all four types of folds handled by FoldSketch UI: hemline folds (e.g. Figures 1.1 and 8.1), gathered folds (e.g. skirt in 2.1, and shirt in 2.2), knife pleats (e.g. short sides of the tissue box in 8.1, and blue pants in Figure 8.2d), and pinched pleats (e.g. orange dress in 8.4 and yellow dress in 8.2). We showcase a range of fold-path orientations including vertical (Figure 1.1, pants in 8.2), diagonal (skirt in Figure 4.1, green T-shirt in Figure 8.2), and horizontal (e.g Figure 8.1, 6.2, and shirt in Figure 5.3).

Our examples were generated using four material settings: thinner (e.g. purple skirts in Figure 4.2, green dress in Figure 8.3) and thicker (e.g. yellow skirt in Figure 8.5, yellow and purple dresses in Figure 8.3) textiles in Sensitive Couture (SC) examples, SC material matching our real-world flag and cover, and the default shirt material in ARCSim. Figure 2.1 and Figure 8.6 depicts application of the same schematic input to garments created from the same initial patterns, but

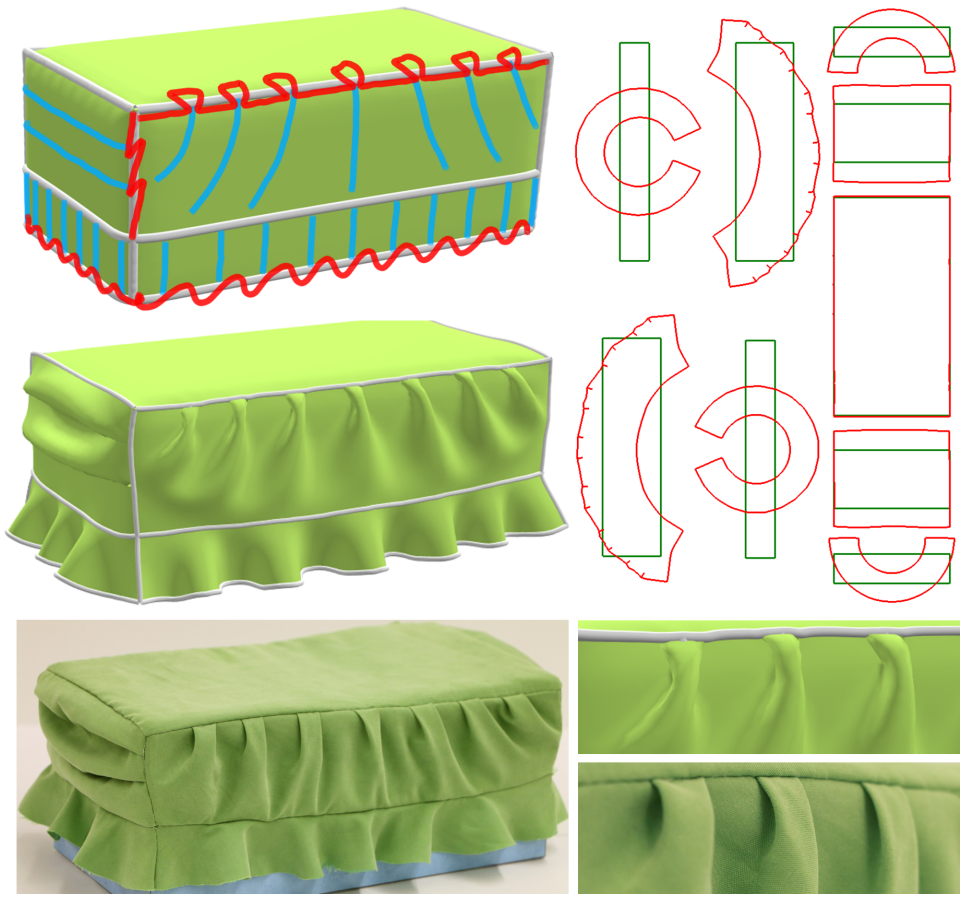

Figure 8.1: Manufactured example: (left to right) Plain input tissue box cover with user sketched schematic folds and final post-simulation result augmented with user-expected folds; original (green) and modified final (red) patterns; real-life replica manufactured using the produced patterns, with zooming in highlights the complex and evolving output fold profile shapes.

Figure 8.2: Additional results created using Sensitive Couture.

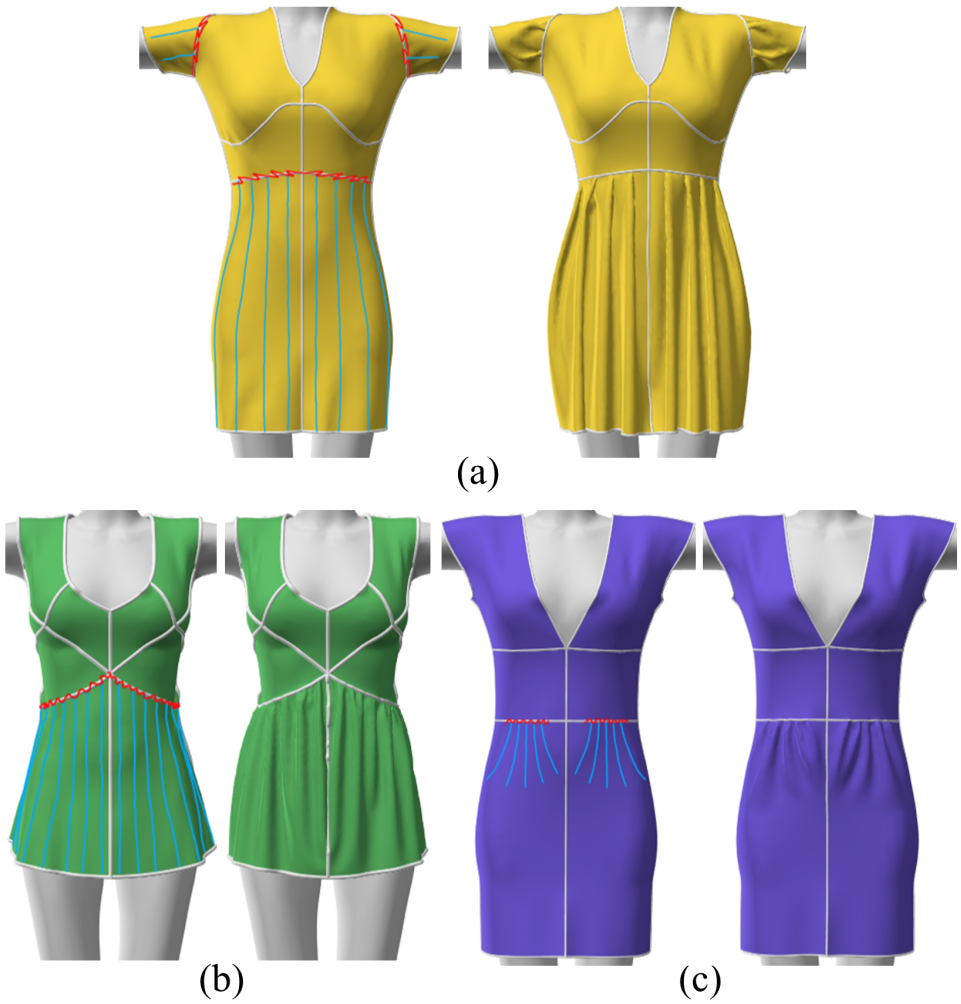

Figure 8.3: Folds created from designer annotations.

with different materials. While the resulting folds are distinctly different, they still clearly reflect designer intent if only the schematic input is reasonable for the chosen material supported by the simulator. The flag and tissue-box examples show progressive application of multiple fold types and orientations to the same input.

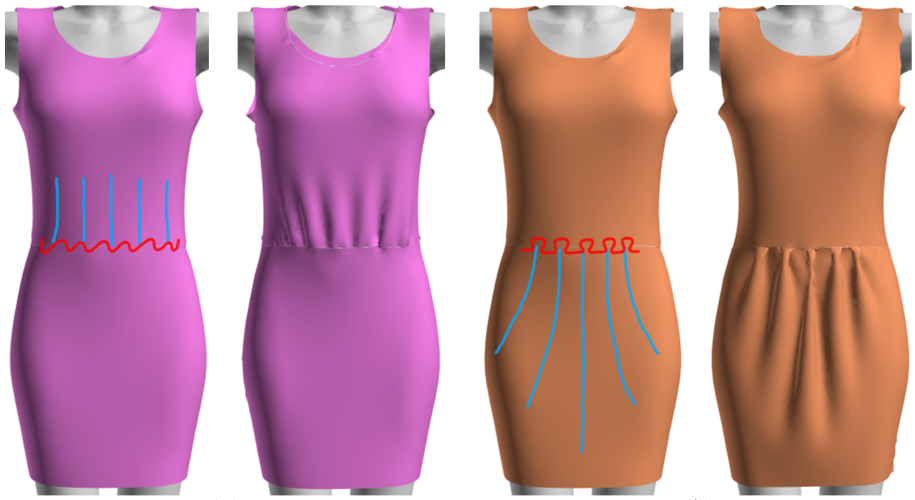

Figure 8.4: Examples created using ARCSim.

8.1 Simulators

We tested our method with two simulation engines, Sensitive Couture [34] and ARCSim [22]. Sensitive Couture is optimized for speed over accuracy, whereas ARCSim is optimized for precision at the expense of slower computation times. Results created with ARCSim are shown in Figure 8.4; the remainder of the results in the paper were generated using Sensitive Couture, which is faster and provides sufficient accuracy for this thesis.

Note that the sensitivity analysis in Sensitive Couture is not used here. Although it is developed for speeding up convergence, it does not help on converging to design preserved garment. This also means that simulators are treated as complete black box in FoldSketch, except for augmenting the style-preserving energy.

For ArcSim to support garment meshes composed of multiple connected components, the stitching scheme in Sensitive Couture is implemented into ArcSim. However, 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

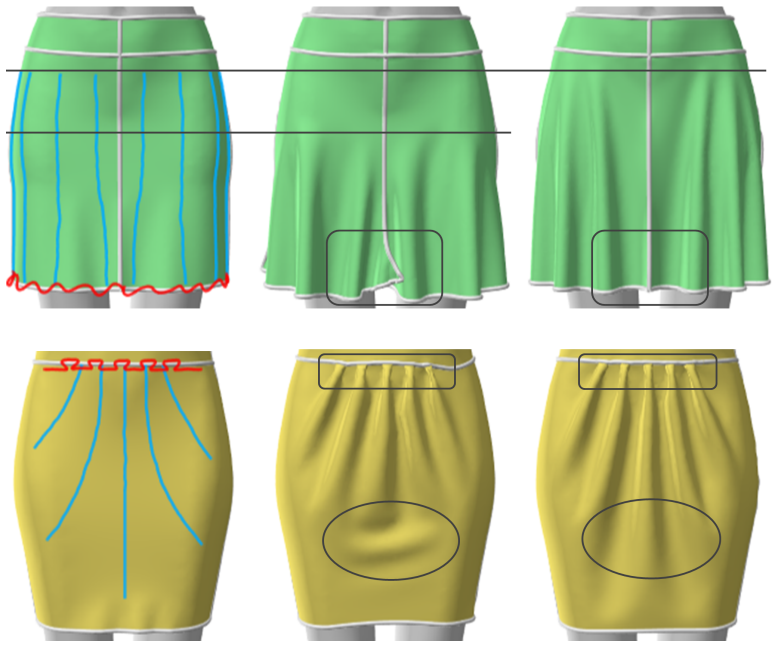

Figure 8.5: Comparison with deformation based pattern extension: (left) input, (center) garments simulated using deformation-based extended patterns, (right) Our results. The naive approach results in multiple artifacts.

Figure 8.6: 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).

shorter seam.

Using garment meshes composed of single connected component would require non-manifold vertices and edges to support pleats, where the gathering seams of uniform folds will also be non-trivial to resolve.

8.2 Timing And Parameters

The 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. The runtimes using Sensitive Couture Simulator [34] 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 over all examples generated with a specific simulator. They are chosen to match the energy scale of the simulators and set to provide enough force to manipulate the garment shape and override other simulation elements. Different simulators have different scales for their cloth energies, which is why we must use different values across simulators.

All parameters in Chapter 5 stay the same across different inputs, materials, and simulators.

To test our algorithm's scalability, we subdivided the input mesh in Figure 4.1, 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.

8.3 Algorithmic Choices

We consider three different alternatives to our algorithm, and show their failure modes. Figure 2.2, 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 mine (Section 5.1.3). As Figure 8.5 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 3.1). 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. FoldSketch's final combination of patterns and drape produces a garment that satisfies both the physical reproducibility requirement and design constraints with no unwanted artifacts (Figure 3.1e).

8.4 Designer Validation

Three of the input fold designs (Figure 8.3) 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.

8.5 Manufacturing

One of my collaborators Mary Buckland manufactured two cloth objects using patterns generated by FoldSketch - a flag (Figure 1.1) and a tissue-box cover (Figure 8.1). The resulting manufactured objects clearly contain the designer's expected folds, and were manufactured directly from FoldSketch 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.

8.6 Perceptual Validation

My collaborators and I validate the outputs of our method via feedback from ten non-experts. To collect their input, we presented 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 Appendix A for more information.

Chapter 9

Conclusions

This thesis 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 demonstrate 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. We believe our contributions will open up a new avenue for researches trying to enable direct manipulation in 3D space for various design problems.

9.1 Limitations and Future Work

Feasibility Detection. Our system follows user specifications in an effort to generate the folds they desire. While we do pre-processing of the garments to better accommodate folds in tight-fitting regions (Section 5.1.4) we cannot guarantee that folds that require more significant pattern edits will be generated. For future work, it would be interesting to explore detection of infeasible inputs, something our

Figure 9.1: FoldSketch 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).

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.

Extending Achievable Design Space. 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. Besides, supporting more sophisticated and flexible stitching schemes to allow more control along the profile is another interesting future work.

Material Modeling. 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 more practical applications.

Avoid Simulation in the Loop. In the 2D-3D Update stage, FoldSketch seeks for patterns that best reproduce the target garment geometry under simulation via pattern adjustment which requires simulation in the loop, and the reproducibility of the target garment needs to be improved in between each pattern adjustment process. For future work, on the technical side, it would be interesting to search for both the 2D patterns and 3D target/output garment in an optimization that directly minimizes the net forces exerted on the garment. Without simulation in the loop, the computational cost could be significantly improved, and the direct manipulations on 3D garment could be more flexible on improving the physical reproducibility.

Bibliography

[1] J. Arnold. *Patterns of Fashion: The cut and construction of clothes for men and women c1560-1620*. Macmillan, 1985. → pages 1, 4, 6

[2] U. M. Ascher and C. Greif. *A First Course on Numerical Methods*. SIAM, 2011. → pages 21

[3] A. Bartle, A. Sheffer, V. G. Kim, D. M. Kaufman, N. Vining, and F. Berthouzoz. Physics-driven pattern adjustment for direct 3d garment editing. *ACM Trans. Graph.*, 35(4):50:1–50:11, 2016. ISSN 0730-0301. → pages 4, 9, 10, 33, 35, 43

[4] D. P. Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999. → pages 35

[5] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proc. Symposium on Computer Animation*, pages 28–36, 2003. → pages 10

[6] R. Brouet, A. Sheffer, L. Boissieux, and M.-P. Cani. Design preserving garment transfer. *ACM Trans. Graph.*, 31(4):36:1–36:11, 2012. ISSN 0730-0301. → pages 8

[7] S. P. L. Browzwear. Browzwear, 2017. URL https://browzwear.com/. → pages 7

[8] V. F. I. CLO. Marvelous designer, 2017. URL https://www.marvelousdesigner.com/. → pages 7

[9] F. Cordier, H. Seo, and N. Magnenat-Thalmann. Made-to-measure technologies for an online clothing store. *IEEE Computer graphics and applications*, 23(1):38–48, 2003. → pages 8

[10] K. Crane, M. Desbrun, and P. Schröder. Trivial connections on discrete surfaces. *Computer Graphics Forum*, 29(5):1525–1533, 2010. ISSN 1467-8659. → pages 20, 21

[11] C. De Paoli and K. Singh. Secondskin: sketch-based construction of layered 3d models. *ACM Transactions on Graphics (TOG)*, 34(4):126, 2015. → pages 7, 8

[12] P. Decaudin, D. Julius, J. Wither, L. Boissieux, A. Sheffer, and M.-P. Cani. Virtual garments: A fully geometric approach for clothing design. In *Computer Graphics Forum*, volume 25, pages 625–634, 2006. → pages 7

[13] O. EFI. Optitex pds, 2017. URL http://optitex.com/. → pages 7

[14] M. Fontana, A. Carubelli, C. Rizzi, and U. Cugini. Clothassembler: a cad module for feature-based garment pattern assembly. *Computer-Aided Design and Applications*, 2(6):795–804, 2005. → pages 7

[15] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. → pages 23, 27

[16] K. Kiisel. *Draping: the complete course*. Laurence King Publishing, 2013. → pages 1, 6, 7

[17] T.-H. Kwok, Y.-Q. Zhang, C. C. Wang, Y.-J. Liu, and K. Tang. Styling evolution for tight-fitting garments. *IEEE transactions on visualization and computer graphics*, 22(5):1580–1591, 2016. → pages 7, 8

[18] C. Li, H. Pan, Y. Liu, A. Sheffer, and W. Wang. Bendsketch: Modeling freeform surfaces through 2d sketching. *ACM Trans. Graph. (SIGGRAPH)*, 36(4):125:1–125:14, 2017. → pages 10

[19] J. Li and G. Lu. Modeling 3d garments by examples. *Computer-Aided Design*, 49:28–41, 2014. → pages 7, 8

[20] L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. A local/global approach to mesh parameterization. *Computer Graphics Forum*, 27(5): 1495–1504, 2008. ISSN 1467-8659. → pages 26

[21] R. Narain, A. Samii, and J. F. O'Brien. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph.*, 31(6):152:1–152:10, 2012. ISSN 0730-0301. → pages 5

[22] R. Narain, T. Pfaff, and J. F. O'Brien. Folding and crumpling adaptive sheets. *ACM Trans. Graph.*, 32(4):51:1–51:8, 2013. ISSN 0730-0301. → pages 5, 29, 40

[23] U. Pinkall and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2(1):15–36, 1993. → pages 20

[24] T. Popa, Q. Zhou, D. Bradley, V. Kraevoy, H. Fu, A. Sheffer, and W. Heidrich. Wrinkling captured garments using space-time data-driven deformation. In *Computer Graphics Forum*, volume 28, pages 427–435, 2009. → pages 10

[25] N. Ray, B. Vallet, W. C. Li, and B. Lévy. N-symmetry direction field design. *ACM Trans. Graph.*, 27(2):10:1–10:13, 2008. ISSN 0730-0301. → pages 21

[26] N. Ray, B. Vallet, L. Alonso, and B. Levy. Geometry-aware direction field processing. *ACM Trans. Graph.*, 29(1):1:1–1:11, 2009. ISSN 0730-0301. → pages 20, 21

[27] C. Robson, R. Maharik, A. Sheffer, and N. Carr. Context-aware garment modeling from sketches. *Comput. Graph.*, 35(3):604–613, 2011. ISSN 0097-8493. → pages 3, 7, 8, 10

[28] D. Rohmer, T. Popa, M.-P. Cani, S. Hahmann, and A. Sheffer. Animation wrinkling: Augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph.*, 29(6):157:1–157:8, 2010. ISSN 0730-0301. → pages vii, 3, 8, 10, 15

[29] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, 2007. → pages 26

[30] R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004. ISSN 0730-0301. → pages 29

[31] K. Takayama, M. Okabe, T. Ijiri, and T. Igarashi. Lapped solid textures: filling a model with anisotropic textures. In *ACM Transactions on Graphics (TOG)*, volume 27, page 53. ACM, 2008. → pages 19

[32] E. Turquin, M.-P. Cani, and J. Hughes. Sketching garments for virtual characters. In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, Grenoble, France, 2004. Eurographics. → pages 3, 7

[33] E. Turquin, J. Wither, L. Boissieux, M.-P. Cani, and J. F. Hughes. A sketch-based interface for clothing virtual characters. *IEEE Comput. Graph. Appl.*, 27(1):72–81, 2007. ISSN 0272-1716. → pages 3, 7, 10

[34] N. Umetani, D. M. Kaufman, T. Igarashi, and E. Grinspun. Sensitive couture for interactive garment modeling and editing. *ACM Trans. Graph.*, 30(4): 90:1–90:12, 2011. ISSN 0730-0301. → pages 5, 9, 29, 40, 42

[35] P. Volino, F. Cordier, and N. Magnenat-Thalmann. From early virtual garment simulation to interactive fashion design. *Comput. Aided Des.*, 37 (6):593–608, 2005. ISSN 0010-4485. → pages 7

[36] C. C. Wang, Y. Wang, and M. M. Yuen. Feature based 3d garment design through 2d sketches. *Computer-Aided Design*, 35(7):659–672, 2003. → pages 7

[37] C. C. Wang, Y. Wang, and M. M. Yuen. Design automation for customized apparel products. *Computer-aided design*, 37(7):675–691, 2005. → pages 8

[38] E. Zhang, J. Hays, and G. Turk. Interactive tensor field design and visualization on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):94–107, 2007. ISSN 1077-2626. → pages 19

Appendix A

Perceptual Validation Material

Figure A.1: Perceptual Validation, with stats.

Figure A.2: Perceptual Validation, with stats (continued).