Pacific Graphics 2013
B. Levy, X. Tong, and K. Yin
(Guest Editors)

Volume 32 (2013), Number 7

Soft Folding

L. Zhu[1,3] T. Igarashi[1,3] J. Mitani[2,3]

[1]The University of Tokyo [2]University of Tsukuba [3]JST ERATO

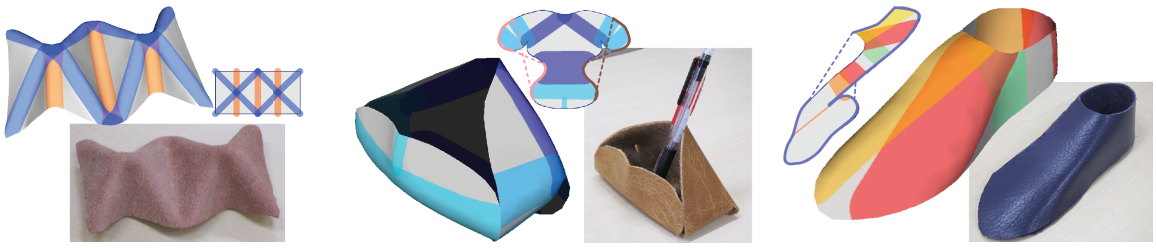


Figure 1: *Thin-plate forms modeled by soft folding. Subfigures visualize their corresponding fold and sewing patterns. Color and thickness of a fold line indicates its fold magnitude and sharpness(see Fig. 5). Seams are shown as pairs of thick curves on the border. From left to right: Water-bomb origami pattern applied on a leather sheet, a leather penholder and shoe.*

Abstract

We introduce soft folding, a new interactive method for designing and exploring thin-plate forms. A user specifies sharp and soft folds as two-dimensional(2D) curves on a flat sheet, along with the fold magnitude and sharpness of each. Then, based on the soft folds, the system computes the three-dimensional(3D) folded shape. Internally, the system first computes a fold field, which defines local folding operations on a flat sheet. A fold field is a generalization of a discrete fold graph in origami, replacing a graph with sharp folds with a continuous field with soft folds. Next, local patches are folded independently according to the fold field. Finally, a globally folded 3D shape is obtained by assembling the locally folded patches. This algorithm computes an approximation of 3D developable surfaces with user-defined soft folds at an interactive speed. The user can later apply nonlinear physical simulation to generate more realistic results. Experimental results demonstrated that soft folding is effective for producing complex folded shapes with controllable sharpness.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.6]: Methodology and Techniques—Interaction Techniques;Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems;

1. Introduction

Thin-plate forms are obtained by deforming flat sheets. When deformations of a planar sheet are restricted to be isometric, special thin-plate forms called developable surfaces are produced. Since different degrees of stretching are allowed by various real-world materials, thin-plate forms are ubiquitous in the real world and are easy to fabricate. Design of thin-plate forms has received considerable attention in the field of geometric modeling. For example, designers may design thin-plate forms to produce bags from leather materials, and architects may explore thin-plate forms to design architectural structures.

Because thin-plate forms are deformed from flat shapes, the traditional interface for designing thin-plate forms is based on deformation interfaces such as click-and-drag with physical simulation [BSG12] or geometric optimization [BK04]. However, these methods require the user to specify handles and drag them one-by-one to create designs, which is a tedious process.

Reverse-engineering [KFC*08] is another way to obtain

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

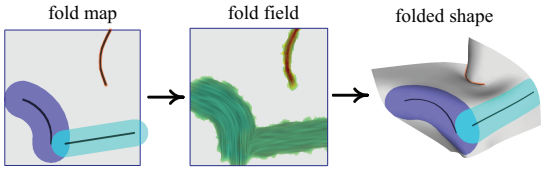


Figure 2: *Work flow of soft folding.*

thin-plate forms by optimizing acquired geometries. Sketch-based interfaces based on sketched contours have also been introduced to model developable surfaces [RSW*07]. However, these systems have a low degree of flexibility for exploring potential variants of thin-plate forms.

In origami, three-dimensional (3D) forms are designed using fold pattern [SVWG12] [DO07]. A fold pattern in origami is represented as a graph of lines (representing folded edges), where connected edges form patterns called peaks ($\wedge$) and valleys ($\vee$). Inspired by fold patterns in origami, we herein propose a new interactive method, called *soft folding*, for the design of thin-plate forms. Our method generalizes sharp folds into soft, rounded folds. The user specifies the location and sharpness of folds on a two-dimensional (2D) sheet and the system computes a 3D folded shape based on these user-defined soft folds.

The workflow of our method is illustrated in Fig. 2. We organize user inputs as a *fold map* (as illustrated by Fig. 2 left) consisting of a set of fold curves with magnitude and sharpness information(Sec. 3). The key idea of our technique is to introduce an intermediate representation of the fold pattern, called a *fold field*(Sec. 2). A fold field defines a fold direction and a fold angle at each point. It generalizes discrete fold graphs limited to sharp folds(as those used in rigid origami), to continuous fields with folds of variable sharpness. Our current implementation discretizes the continuous fold field on a 2D mesh (Sec. 4.1). We locally fold the one-ring of each vertex according to the fold field, and obtain the globally folded shape by assembling the locally folded patches using linear rotation-invariant coordinates (LRIC; Sec. 4.2) [LSLCO05]. We carried out an experiment to test the efficiency of our proposed soft folding method (Sec. 5).

Our goal is to compute a folded 3D geometry that satisfies user-defined fold patterns by deforming a planar shape. Simply applying physics is not enough, because it is not clear what force and constraints need to be applied to reproduce the user-specified folds. Thus, we derive the folded shape in a geometric way, to help users interactively explore thin-plate forms. The developability is implied in the fold field representation, but not explicitly formulated as a hard constraint in the workflow, and perfect developability is not guaranteed. To obtain a more realistic result, we run physical simulation based on shape matching [MHTG05] as a post-processing tool for soft folding (Sec. 4.3).

Contributions The present study makes the following main contributions:

- A new method is proposed for interactively exploring thin-plate forms including both sharp and soft folds.
- A computational workflow is presented that computes a 3D folded geometry from a fold map via a novel intermediate representation called a fold field.
- An implementation of the workflow combining appropriate known techniques is provided; We study local folding operations for computing fold fields and use LRIC for computing 3D geometry.

Related Work Shape modeling is a well-studied subject in the field of graphic design. Various modeling methods have been proposed for form design, such as parametric surface design systems, sculpting systems [PIX] and sketching systems [IMT07]. Various deformation methods are also available, including cage-based methods [JSW05], skeleton-based methods [YBS07] and point-based methods [BK04]. However, all of these are designed for general freeform models and are not readily useful for creating or deforming thin-plate structures. What is missing is a method for users to quickly explore various forms with sharp and soft folds while keeping isometrically close to a planar shape. Recently, new design interfaces have been developed to explore forms under various physical constraints [UIM12] or geometric constraints [YYPM11]. However, in the latter system, users are only able to explore local variants of the input shape.

Developable surfaces have been studied mathematically in differential geometry for years [DC76]. Algorithms have been developed for computer-aided geometric modeling, and these enable various features for developable surface modeling. For example, users can design developable spline surfaces [PF95] and developable surfaces with curved folds [KFC*08]. Users can also explore developable surfaces with interpolated constraints [RSW*07] [BW08] and complex layers [IM10]. Developable surfaces can also be obtained by optimizing curved surfaces [Wan08] [PHD*10] into (piecewise) developable surfaces, or optimizing face planarity of their control meshes [LPW*06]. A comprehensive review can be found in [SVWG12] and references therein. Since perfect developability is hard to achieve, those modeling techniques are either of low degree-of-flexibility or not fast enough for exploring shape variants.

Algorithms for smoothly deforming shapes can be classified into two categories: those for physical simulations and those for geometric processing . Physical simulation methods for deformable objects (e.g., shape matching [MHTG05]), as well as cloth simulation (e.g., [EB08]), can be used to explore thin-plate forms. Various geometric processing algorithms are also available for designing thin-plate forms via deformation of a flat mesh (for more details, see [BS08] and

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

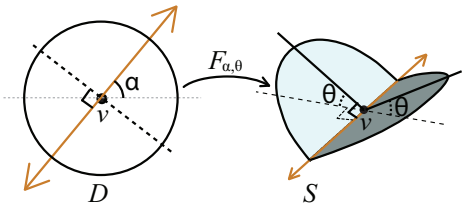


Figure 3: *A folding operation $F_{\alpha,\theta}$.*

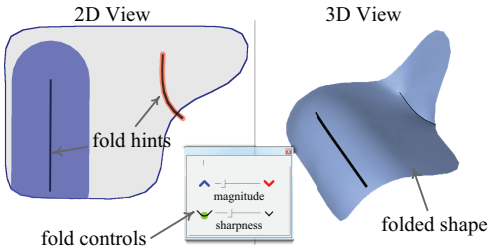


Figure 4: *The user interface. Blue curves represent the boundaries of the input planar shape, and black curves represent the fold hints on it. The magnitude and sharpness of each fold hint is mapped to the stroke's thickness and color respectively.*

[SA07]). However, these methods do not provide a convenient user interface for specifying shapes with soft folds.

Origami is the practice of folding paper to make forms, and the method consists of making sharp folds. The design rules of origami have been reported elsewhere [DO07]. Recently, physical simulations are introduced to origami [BWGO06] or paper folding [NPO13]. Physically accurate results are obtained at a cost of expensive computations.

Soft folds also appear on cloth. They can be automatically generated in physical simulation [EB08], especially at a high resolution. In order to improve the performance, geometric methods are introduced to synthesize wrinkles on a coarse simulated mesh [DJW*06] [RPC*10]. However, in those works, soft folds are created by locally projecting or offsetting vertices with a height map. Instead of bumping planar sheets, our method is completely different in that we derive global shape taking user-specified folds as inputs.

Our geometric computing is inspired by algorithms that reconstruct 3D geometry from differential quantities. 3D geometry is generated from a 2D sketching or painting by solving higher-order Laplacian equations in [AJC11] or by controlling mean curvature half-density in [CPS11]. Our operation is different in that we are folding planar sheets instead of inflating it. Eigensatz and Pauly use nonlinear optimization to control surfaces geometry by prescribing directional curvatures [EP09]. However, curvatures are not well-defined across sharp edges. We instead use local folding operations to help dealing with sharp and smooth folds in a uniform way. Winkler et.al [WDAH10] also locally interpolate edges and their dihedral angles and use shape matching to recover an interpolated mesh. We go a step further by defining a fold field for designing and exploring thin-plate forms.

2. Thin-plate Forms and Fold Field

In this section, we introduce *fold field*, an intermediate representation for thin-plate form design. A fold field defines a folding operation $F : D \in \mathbb{R}^2 \mapsto S \in \mathbb{R}^3$ at each vertex $\mathbf{v}$ on a planar shape. A folding operation can be parameterized by a fold direction $\alpha \in [0, \pi)$ and a fold angle $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, as illustrated in Fig. 3. Note that α ranges in $[0, \pi)$ since a fold direction is two-symmetric. When $\theta > 0$, the folding operation F forms a valley around vertex $\mathbf{v}$; otherwise, a peak forms.

Thin-plate forms are obtained by smoothly deforming planar shapes. When the deformation is isometric, we get developable surfaces. Looking into this ideal case, a developable surface has vanishing Gauß curvature, i.e., $K = k_1 k_2 = 0$, where k_1 and k_2 are principal curvatures. This indicates that a nonplanar vertex on a developable surface can be locally approximated by a fold along the principle direction corresponding to the vanishing principal curvature. Though general thin-plate forms are not perfectly developable, they are still nearly single-curved. Therefore, we use a fold field to approximately encode its geometry locally. In the following, we define operators on folding operations which we will use in the computation of a fold field.

Fold Difference Given two folding operations $F_{\alpha,\theta}$ and $F_{\bar{\alpha},\bar{\theta}}$ at vertex $\mathbf{v}$, the rest shape around $\mathbf{v}$ is mapped into $S(u,v)$ and $\bar{S}(u,v)$ respectively. We measure the difference of $F_{\alpha,\theta}$ and $F_{\bar{\alpha},\bar{\theta}}$ by the L^2 difference of the folded shapes S and $\bar{S}$. By expressing the difference in terms of α and θ, we define fold difference

$$\begin{aligned} d\left(F_{\alpha,\theta}, F_{\bar{\alpha},\bar{\theta}}\right) = \pi - \\ \pi\left((\sin\Delta\alpha)^2\frac{\cos\theta+\cos\bar{\theta}}{2} + (\cos\Delta\alpha)^2\frac{1+\cos\theta\cos\bar{\theta}}{2}\right) \\ -\sin\theta\sin\bar{\theta}\left(\sin\Delta\alpha + (\frac{\pi}{2}-\Delta\alpha)\cos\Delta\alpha\right) \end{aligned} \quad (1)$$

where $\Delta\alpha = |\alpha - \bar{\alpha}|$. Please see Appendix for details.

Fold Summation We define fold summation as the process of obtaining a folding operation that minimizes its difference between summing folds. Specifically, for a fold summation $F_{\alpha,\theta} = \omega_i \sum_i F_{\alpha_i,\theta_i}$, its fold direction α and fold angle θ are defined as

$$(\alpha,\theta) = \underset{\alpha,\theta}{\operatorname{argmin}}\, w_i \sum_i d\left(F_{\alpha,\theta}, F_{\alpha_i,\theta_i}\right) \quad (2)$$

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

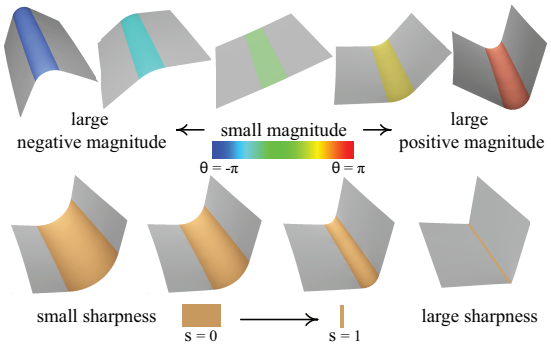


Figure 5: *Folded shapes created from fold hints with varying fold magnitudes and sharpness.*

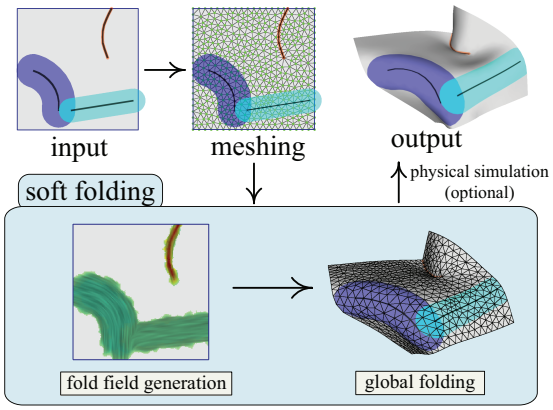


Figure 6: *Algorithm pipeline of soft folding. The fold field is visualized as a flow field here.*

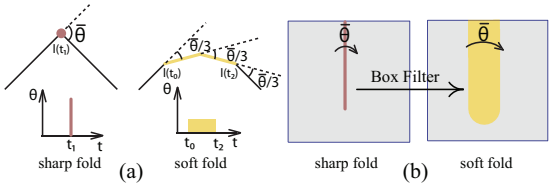


Figure 7: *Diffusing a sharp fold hint's fold angle leads to a soft fold hint. (a) A line folding illustration. (b) A box filter is applied to large fold angles (show in red) and the diffused fold angles (shown in yellow) leads to a soft fold of a plane.*

3. User Interface

Fold Map When people fold planar sheets, they manipulate the sheet by touching, holding, bending, and pinching it. Simulating this process by incorporating physics involves complex computation of external forces and thin-shell deformation. Note that the design intention behind this complex behavior is to specify the location and shape of the fold creases together with their angles and sharpness. Therefore, the input of our system is designed to be a *fold map* with the rest shape of the material and fold hints specifying the individual local folds. The geometry of the flat sheet is stored as its freeform boundary curve, shown in blue in Fig. 4; Fold hints define the geometry of each fold on the planar sheet (i.e., the *fold hint*) with their magnitude ($\theta(\theta \in [-\pi, \pi]$), as shown in Fig. 5 top) and sharpness ($s(s \in [0,1]$, as shown in Fig. 5 bottom), where $s = 1$ represents a sharp fold and $s = 0$ represents a soft fold. Fold hint geometry is represented as a 2D curve and is shown in black in Fig. 4.

Interface Our proposed system has a sketching interface that allows a user to specify the rest shape of the planar sheet and the fold hints on it. The magnitude and sharpness of a fold hint are mapped to the stroke color and thickness: red represents a large positive magnitude, blue represents a large negative magnitude, and green represents a small magnitude. A thin stroke represents a sharp fold and a thick stroke represents a soft fold. The user can control the magnitude and sharpness using slider bars, while the folded shape is updated in a 3D view at an interactive speed.

4. Algorithm

The pipeline of our algorithm is illustrated in Fig. 6. We implemented the workflow in Fig. 2 by discretizing the input planar sheet into a triangular mesh, and computing the fold field and folded shape for the mesh.

First, the planar shape is triangulated using 2D conforming-constrained Delaunay triangulation, in which the border curves of the planar shape and the fold hints are set as the constraints of the triangulation. Vertices on the user-specified fold curves are called *constraint vertices*, and *unconstraint vertices* otherwise. After meshing, the algorithm for soft folding is broken down into two parts: fold field generation and global folding.

4.1. Fold Field Generation

The goal of this stage is to find a folding operation field on the input planar mesh that reflects the fold map. Folding operations are stored on mesh vertices in our discretization. We first compute a fold field for each fold hint and then blend them together as the fold field of the input fold map.

Defining a fold field from a sharp fold hint is straightforward. If vertex i is on the sharp fold hint, its fold direction is initialized as its tangent on the fold hint and its fold angle is initialized as half of the user-specified fold magnitude. Otherwise, its fold angle is set to zero.

We diffuse the fold angle when dealing with a soft fold hint. As illustrated in Fig. 7a, by diffusing a fold angle distribution $\theta(t) = \bar{\theta}\chi_{\{t=t_1\}}(t)$ ($\chi_A(t) = 1$ when A holds, otherwise $\chi_A(t) = 0$) into a distribution $\theta(t)$ which is supported in $[t_0, t_2]$ such that $\int_{t_0}^{t_2} \theta(t)dt = \bar{\theta}$ holds, we are expected to get a soft fold with a fold magnitude $\bar{\theta}$ along the curve seg-

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

ment $l(t) : t \in [t_0, t_2]$. In our implementation, a box filter is applied to smooth the fold angles from a sharp fold hint. When folding a planar sheet, we run a box filter to compute the fold angle on each vertex. The fold direction of vertex i is set as the tangent direction of its projection on the fold hint.

Having got fold fields from each fold hint, we sum them up to obtain a fold field corresponding to the input fold map. We apply fold summation if an unconstrained vertex is shared by more than one fold hints. Fold summation is local and fast because the nonlinear optimization solved in Eqn. 2 only has two variables. Trust region method is used in our system. We uniformly sample points in $[0, \pi) \times [-\frac{\pi}{2}, \frac{\pi}{2}]$ as initial guesses, and pick the optimized result with minimal summed fold difference as the blended folding operation. In our implementation, we use $n \times n$ samples when summing up n folding operations.

Implementation Details By setting a maximum width d, we map the sharpness s of a fold hint into a distance $(1-s)d$. When implementing a box filter on mesh, we first get a fold region by offsetting the fold hint curve with distance $(1-s)d$. A breadth-first search from the fold hint is run to find triangles covered by this region. Suppose the area of triangle j is A_j and the area of its overlap with the fold region is A'_j, we weight vertex i in the fold region by $\omega_i = \sum_{j \in N_i} A'_i / \sum_{j \in N_i} A_i$, where N_i is the neighboring triangles of vertex i. Since the integral of the fold angles Θ should be invariant after applying a box filter, we compute it in advance by summing up fold angles on the fold hint. The fold angle of vertex i is finally determined by $\frac{\omega_i \Theta}{\sum_i \omega_i}$. Other vertices' fold angles are set to zero, keeping the fold field faithful to the user input. After fold field summation, we also tried smoothing the folding operations in overlapped fold regions by using the fold difference (Eqn. 1). But in our experiments, we find folded shapes are quite similar before and after this smoothing. Therefore, only fold summations are executed in our implementation.

4.2. Global Folding

The proposed system defines a local folding operation F_{α_i, θ_i} for each vertex i. In our discrete setting, we use the one-ring neighborhood of vertex i as the domain of the fold operation F_{α_i, θ_i}. The image of F_{α_i, θ_i} is called the *local fold* of vertex i. Note that fold directions are not required to be aligned with mesh edges. The meshed structure is only used to find neighboring vertices, whose geometry is transformed by local folding and used to reconstruct their global position.

The objective at this stage is to find a shape such that for each vertex i, its neighborhood approximates its local fold. This is a local-to-global deformation problem. We adapt LRIC-based mesh deformation [LSLCO05] to efficiently assemble the local folds for a global folding. Fig. 8 illustrates global folding.

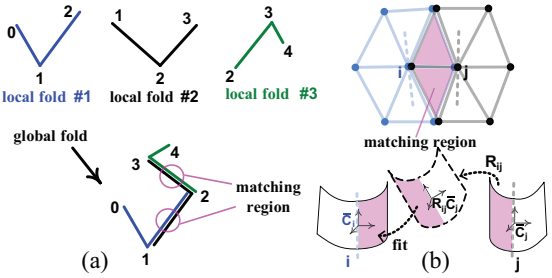


Figure 8: *Global folding. (a) A 2D illustration. (b) Locally assembling two 3D local folds. The overlapped wedge region (shown in pink) fits two neighboring local folds.*

Global folding involves two steps: encoding and reconstruction. Suppose each vertex $\mathbf{v}_i$ is equipped with a frame $C_i = (\mathbf{X}_i, \mathbf{Y}_i, \mathbf{Z}_i)$. Let $\bar{\mathbf{v}}_i$ and $\bar{C}_i$ denote its position and frame on the rest shape. In [LSLCO05], surface geometry is first locally encoded into relative local coordinates $\mathbf{c}_{ij}$ and relative rotations Q_{ij} of neighboring frames, s.t $\bar{\mathbf{v}}_i - \bar{\mathbf{v}}_j = \bar{C}_j \mathbf{c}_{ij}$ and $\bar{C}_i Q_{ij} = \bar{C}_j$. In the reconstruction step, after the user specify positions and frames of several fixed vertices, the system successively solves linear equations $C_i Q_{ij} = C_j$ for frames and $\mathbf{v}_i - \mathbf{v}_j = C_j \mathbf{c}_{ij}$ for positions of free vertices. In our global folding stage, we particularly adapt the encoding step. Local folds are used in our system. Assume the folding operation on vertex i is F_i, the encoded local coordinates are computed as $\mathbf{c}_{ij} = \bar{C}_j^T (F_j(\bar{\mathbf{v}}_i) - \bar{\mathbf{v}}_j)$. For computing relative rotations, we first locally assemble neighboring folded patches, as shown in Fig. 8b. Specifically, shape matching from the overlapped wedge region in the local fold of vertex j to the local fold of vertex i is applied, see [MHTG05] for more details on the shape-matching operation. After the best-fit rotation matrix R_{ij} is obtained, relative rotations are computed as $Q_{ij} = \bar{C}_i^T R_{ij} \bar{C}_j$. While folding planar shapes, we simply set frame $\bar{C}_i$ on the rest planar shape to be the identity matrix. Therefore, $Q_{ij} = R_{ij}$. Note that $Q_{ij} = Q_{ji}^T$ does not always holds, since the folding operations F_i and F_j may be different.

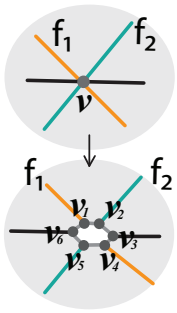
Because we globally solve for the folded geometry, the user-defined magnitude and sharpness may not exactly match with the folds appearing in the final 3D geometry. Complex interactions occur among all of the fold hints.

Implementation Details In the reconstruction step of global folding, we can obtain the folded geometry by solving two sparse linear systems. Given that the sparse matrix in the vertex-solving step (solve $\mathbf{v}_i - \mathbf{v}_j = C_j \mathbf{c}_{ij}$ for all vertices) is computed from the triangulation of the rest shape, we can pre-compute its inverse after meshing. As for the matrix in the frame-solving step (solve $C_i Q_{ij} = C_j$ for all frames), its sparse structure is only dependent on the triangulation, while its nonzero entry is computed from the fold field. Therefore, we can pre-compute its symbolic factorization after meshing, which is reused to accelerate the numerical factorization

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

when the fold parameters are adjusted. We also fix the position and frame of the vertex at the center of the material in global folding and eliminate it from linear systems to make them have unique least-square solutions.

Intersected Fold Hints When two or more fold hints intersect at vertex **v**, we make an infinite small hole around this vertex, as show in the inset. A similar method is used in [AJC11]. Assume the degree of vertex **v** is d after meshing. We duplicate the intersected vertex into d copies and add d virtual edges (shown in grey). After this duplication, they will have unique folding operations according to their own fold hints. In the global folding stage, though the duplicated vertices share the same geometry in the rest sheet, they may have different frames and folded positions since they do not share the same neighbors. We average the positions of those duplicated vertices as the position of **v** after global folding.



4.3. Post-processing

A notable characteristic of physical developable surface behavior is stress concentration. High stress appears at very limited regions and exhibits fold lines. Our algorithm is essentially a diffusion process and cannot produce new stress concentration, which might be inaccurate if we see it as a physical simulation process. However, in our framework, we expect that users explicitly specify stress concentration as fold lines. So it is desirable not to produce new stress concentration in other places if we see it as a design tool.

If users need to obtain more realistic results, an optional physical simulation can be applied as a post-processing step. However, it is not trivial to compute forces that can reproduce user-specified folds. Therefore, we developed a variant of physical simulation based on shape matching [MHTG05] to refine the folded shape. Simply applying a simulation based on shape matching will make the folded result flat again. Our solution is to reuse the local fold derived from the fold field as the rest shape. Specifically, we minimize

$$\sum_k \sum_{i \in N(k)} ||\mathbf{v}_i - R_k F_k(\bar{\mathbf{v}}_i)||^2,$$

where $N(k)$ is the one-ring neighborhood of vertex k, R_k is the best transformation which fits the local fold of $N(k)$ to its deformed shape, and F_K is the folding operation at vertex k. We use standard dynamics in [MHTG05] to optimize this nonlinear energy.

Note that the physical simulation is nonlinear and too slow to globally fold a planar sheet from the beginning. It iteratively updates vertex positions by computing local forces and it takes time to obtain a folded shape especially when the fold magnitude is large. We therefore use a linear geometric algorithm to quickly obtain a good approximation of the globally folded shape to support interactive exploration of thin-plate forms. A physical simulation can be completed at a higher resolution to optimize the result when users are satisfied with the designed form.

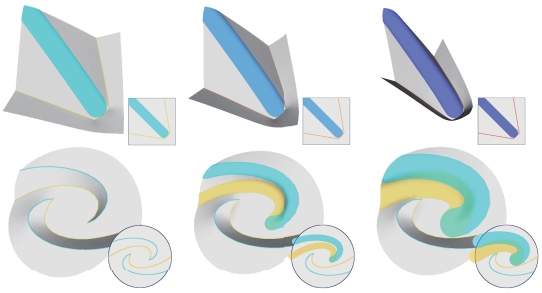


Figure 9: *Top row: From left to right, thin plate forms from increased fold magnitudes. Bottom row: From left to right, thin plate forms (inspired by David Huffman's design) from decreased fold sharpness.*

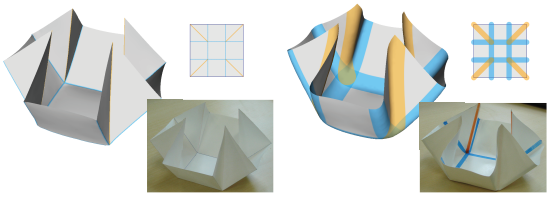


Figure 10: *Folding sheets with different materials can be modeled by varying fold sharpness.*

5. Results

We show thin-plate forms designed by soft folding in this section. Subfigures embedded in each figure visualize the input fold maps. Note that traditional click-and-drag interface is ineffective to model most of these shapes.

5.1. Experimental Results and Comparisons

Both straight folds and curved folds that are either soft or sharp are supported by soft folding, as shown in Fig. 9. Besides, users can interactively explore thin-plate forms' variants by adjusting fold magnitude (Fig. 9 top) or fold sharpness (Fig. 9 bottom).

Fig. 10 shows an example with intersected fold hints. By constraining maximum fold sharpness, soft folding can be used to design shapes with materials that are hard to bend.

Traditional origami intensively studies fold patterns' layout. In addition to their layouts, soft folding also provides an interactive tool for studying fold angles in origami. Fig. 11 left

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

Figure	#V	#F	#CV	pre-processing	fold field generation	global folding
9 top left	487	908	56	70ms	1ms	74ms
9 bottom right	407	749	79	61ms	3ms	65ms
10 left	1187	2252	207	140ms	0.2ms	161ms
10 right	1187	2252	207	140ms	1ms	163ms
11 top right	970	1842	171	134ms	0.1ms	167ms
11 bottom right	970	1842	171	187ms	7ms	274ms
11 left	1960	3834	672	322ms	0.4ms	496ms

Table 1: *Model statistics and timings of soft folding. #V, #F, and #CV represent the number of vertices, faces, and constraint vertices, respectively. All timings were measured on a laptop with a 2.8GHz CPU, 4Gb RAM.*

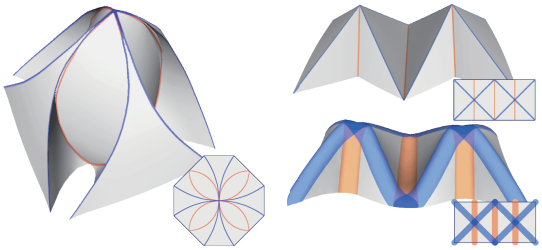


Figure 11: *Two origami results modeled by soft folding.*

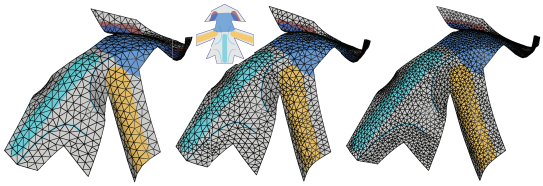


Figure 12: *Thin-plate forms designed by soft folding with varying mesh resolutions.*

shows an origami bud design. We use the 2D layout designed by Jeannine Mosely and generate a folded shape with specified fold magnitudes. When a fold map in soft folding is a valid fold pattern in origami, plausible folded shapes can be created. The top right shape in Fig. 11 gives a thin-plate form designed from the water-bomb pattern in origami. Besides, we can further estimate the folded shape if the same pattern with soft folds is applied, as shown in the bottom right in Fig. 11. By gradually changing fold magnitudes, soft folding can also be used to generate animations for non-rigid origami.

Soft folding is not sensitive to varying resolutions, as shown in Fig. 12. Therefore, we can use soft folding on a low resolution mesh to quickly explore shape variants. Table. 1 lists statistics of examples modeled by soft folding in Fig. 9, 10 and 11.

As for post-processing, the nonlinear physical simulation proposed in Sec. 4.3 on a higher resolution mesh can be applied if more realistic and detailed results are required.

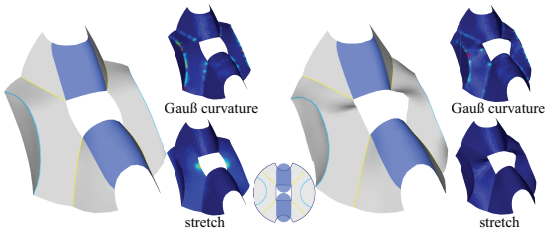


Figure 13: *A thin-plate form designed by soft folding without (left) and with (right) post-processing.*

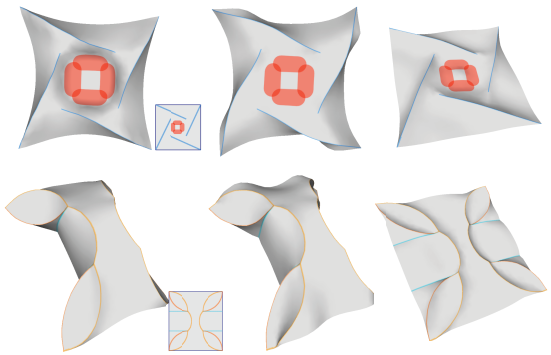


Figure 14: *Thin-plate forms designed by soft folding(left). Their nonlinear physical simulation results with (middle) and without (right) soft folding as initialization.*

Fig. 13 shows the effect of physical simulation. We visualize the stretch and developablity as a color map on the shapes. Stretch is measured based on an area-weighted average of ARAP distortion [MZ12], and developability error is measured based on an area-weighted average of discrete Gauß curvature [Gri06]. From Fig. 13, we can see that the designed forms from soft folding are almost developable except on the fold curves. Physical simulation introduces extra wrinkles on the folded shape to drastically reduce the stretch of the folded shape. Fig. 13 also shows that soft folding works on planar sheets with holes.

Soft folding provides good initialization for nonlinear phys-

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

Figure	#V	#F	#CV	averK	maxK	averS	maxS	timing
13	926	1725	116	6.3e-3(5.2e-3)	0.12(0.11)	3.9e-4(0.9e-3)	6.0e-3(1.4e-1)	4.0s
14 top	520	953	71	2.4e-2(5.2e-2)	0.38(0.92)	2.28e-3(3.75e-2)	7.9e-2(2.4e-1)	1.7s
14 bottom	2471	4697	511	9.1e-3(1.0e-2)	0.23(0.26)	5.1e-4(6.0e-3)	1.8e-2(8.2e-1)	8.3s

Table 2: *Statistics of post-processing. Average absolute Guaß curvature(averK), maximum absolute Guaß curvature(maxK), average stretch(averS) and maximum sketch(maxS) after and before(data in bracket) post-processing are given. Timings for post-processing are measured in seconds.*

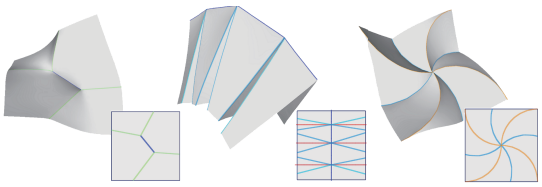


Figure 15: *Several shapes from [SVWG12] and [KFC*08] are reproduced by soft folding.*

ical simulation as shown in Fig. 14 middle. The global folding effects disappear if simply using unfolded 2D shapes as initialization (Fig. 14 right). A correct way to apply nonlinear physical simulation on unfolded 2D shapes is to divide the folding operation into small time steps. However, it is computationally expensive and not suitable for design exploration. Statistics on the effects of post-processing are listed in Table 2.

Several results in [SVWG12] and [KFC*08] are reproduced by soft folding in Fig. 15. Though soft folding does not guarantee generating developable surfaces, similar shapes are quickly produced at a tradeoff of perfect developablity. The nonlinear optimization techniques in [KFC*08] can be used to optimize the results from soft folding if piecewise developable surfaces are desired. Besides, the fold pattern regularization introduced in [SVWG12] can be considered as a pre-processing tool to correct invalid input fold maps when all fold hints are sharp and connected.

5.2. Extensions

Adding Linear Constraints Two linear systems are solved to recover the geometry of folded shapes in the global folding stage (Sec.4.2). Therefore, we can further add linear constraints in the linear systems to model other effects. For example, snapping two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ in the folded shape can be modeled by adding a hard constraint $\mathbf{v}_i = \mathbf{v}_j$. After eliminating variables from those linear constraints, we can get folded shapes with stitched vertices, as shown in Fig. 16.

Folding Curved Surfaces Though our fold operations are defined on planar shapes, we tested them on curved surfaces. Specifically, we first import a curved shape and parameterize it. The input fold map is then drawn on its parametric domain. After generating a fold field on the parametric domain, the folded shape is finally computed by replacing the flat unfolded shape with the predefined curved surface in the global folding stage(sec.4.2). Soft folding shows its possibility in folding curved surfaces, as shown in Fig. 17. By using the folded shape in Fig. 16 left as the rest shape and a complex fold pattern reproduced by referring Roy Iwaki's design [Iwa00], we generate a mask model in the left. In the right figure, buckling effects are modeled by drawing soft curved folds on a cylindric rest shape.

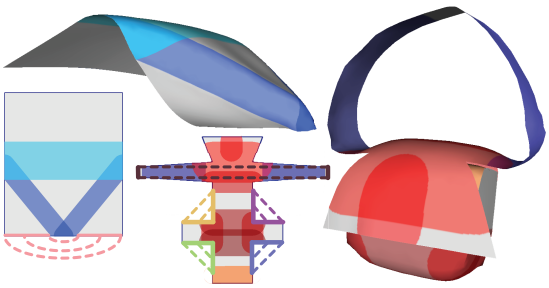


Figure 16: *Thin-plate forms designed by adding linear constraints in soft folding. Stitching edges are color-coded on the 2D patch boundary.*

Figure 17: *A mask model (left) and buckling effects on a trouser leg (right) designed by soft folding on curved surfaces.*

6. Limitation and Future Work

As a design and exploration tool for thin-plate forms, soft folding does not guarantee perfect developability. At fine scale, soft folding tends to smooth out sharp features, while

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

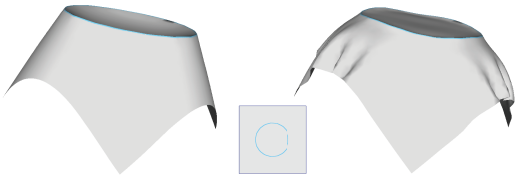


Figure 18: *Left: A thin-plate form designed by soft folding from an inappropriate fold pattern. Right: Extra wrinkles appear after physical simulation.*

in physics they're unavoidable due to stress concentration. Though designed forms from soft folding can be further optimized to piecewise developable surfaces, it remains challenging to effectively model developable surfaces. Besides, our computational model is essentially based on a heuristic though various experiments verifies the proposed system. We are also going to theoretically analyze our computational model in the future.

In the current implementation, folded shapes are directly reconstructed from fold fields without considering collisions, making it possible to cause self-intersections. Generating a collision-free fold field will be an interesting research topic for future work.

Soft folding bridges origami and thin-shell simulation. Another research avenue will be developing algorithms incorporating origami theory. A study from origami theory may further improve thin-plate form design by correcting input fold maps. Our current user interface does not provide mechanisms to prevent users from drawing inappropriate inputs, such as the one shown in Fig. 18. Though soft folding strives to create a folded shape that has relative lower stretch energy and better developability even if bad inputs are provided, unrealistic shapes are still unavoidable because soft folding keeps faithful to the user inputs. A study from the wrinkles generated from physical simulation (Fig. 18 right) may also benefit researches in origami.

Finally, instead of using constrained Delaunay triangulation, anisotropic Delaunay meshing guided by the fold field is expected to further improve the modeling system [NPO13]. One challenge here is to design fast anisotropic meshing algorithms for this interactive system. Another future line of research will be a theoretical study of fold fields on curved surfaces. Our experiments show the possibility to extend soft folding for folding curved surfaces. It may be even possible to extend our system to simulate general deformation with a new user interface.

References

[AJC11] ANDREWS J., JOSHI P., CARR N.: A linear variational system for modeling from curves. *Computer Graphics Forum 30*, 6 (2011), 1850–1861. Presented at SGP 2012. 3, 6

[BK04] BOTSCH M., KOBBELT L.: An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 630–634. 1, 2

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (Jan. 2008), 213–230. 2

[BSG12] BARBIČ J., SIN F., GRINSPUN E.: Interactive editing of deformable simulations. *ACM Trans. on Graphics (SIGGRAPH 2012) 31*, 4 (2012). 1

[BW08] BO P., WANG W.: Geodesic-controlled developable surfaces for modeling paper bending. *Comput. Graph. Forum 26* (2008), 365–374. 2

[BWGO06] BURGOON R., WOOD Z. J., GRINSPUN E., OBISPO S. L.: Discrete shells origami. In *In Proceedings of the 21st International Conference on Computers and Their Applications* (2006), pp. 180–187. 3

[CPS11] CRANE K., PINKALL U., SCHRÖDER P.: Spin transformations of discrete surfaces. *ACM Trans. Graph. 40* (2011). 3

[DC76] DO CARMO M.: *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976. 2

[DJW*06] DECAUDIN P., JULIUS D., WITHER J., BOISSIEUX L., SHEFFER A., CANI M.-P.: Virtual garments: A fully geometric approach for clothing design, sep 2006. 3

[DO07] DEMAINE E. D., O'ROURKE J.: *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge Univ. Press, 2007. 2, 3

[EB08] ENGLISH E., BRIDSON R.: Animating developable surfaces using nonconforming elements. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 66:1–66:5. 2, 3

[EP09] EIGENSATZ M., PAULY M.: Positional, metric, and curvature control for constraint-based surface deformation. *Computer Graphics Forum 28*, 2 (2009), 551–558. 3

[Gri06] GRINSPUN E.: Discrete differential geometry: an applied introduction. In *ACM SIGGRAPH Course* (2006). 7

[IM10] IGARASHI T., MITANI J.: Apparent layer operations for the manipulation of deformable objects. *ACM Trans. Graph. 29*, 4 (July 2010), 110:1–110:7. 2

[IMT07] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: a sketching interface for 3d freeform design. In *ACM SIGGRAPH 2007 courses* (2007), SIGGRAPH '07. 2

[Iwa00] IWAKI R. T.: *Cavex Round Folding: The Mask Unfolds*. 2000. 8

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3 (July 2005), 561–566. 2

[KFC*08] KILIAN M., FLÖRY S., CHEN Z., MITRA N. J., SHEFFER A., POTTMANN H.: Curved folding. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 75:1–75:9. 1, 2, 8

[LPW*06] LIU Y., POTTMANN H., WALLNER J., YANG Y.-L., WANG W.: Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph. (SIGGRAPH) 25* (2006), 681–689. 2

[LSLCO05] LIPMAN Y., SORKINE O., LEVIN D., COHEN-OR D.: Linear rotation-invariant coordinates for meshes. In *Proceedings of ACM SIGGRAPH 2005* (2005), ACM Press, pp. 479–487. 2, 5

[MHTG05] MÜLLER M., HEIDELBERGER B., TESCHNER M., GROSS M.: Meshless deformations based on shape matching. *ACM Trans. Graph. 24*, 3 (July 2005), 471–478. 2, 5, 6

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd

[MZ12] MYLES A., ZORIN D.: Global parametrization by incremental flattening. *ACM Trans. Graph. 31*, 4 (July 2012), 109:1–109:11. 7

[NPO13] NARAIN R., PFAFF T., O'BRIEN J. F.: Folding and crumpling adaptive sheets. *ACM Transactions on Graphics 32*, 4 (July 2013). Proceedings of ACM SIGGRAPH 2013, Anaheim. 3, 9

[PF95] POTTMANN H., FARIN G. E.: Developable rational bézier and b-spline surfaces. *Computer Aided Geometric Design 12*, 5 (1995), 513 –531. 2

[PHD*10] POTTMANN H., HUANG Q., DENG B., SCHIFTNER A., KILIAN M., GUIBAS L., WALLNER J.: Geodesic patterns. In *ACM SIGGRAPH 2010 papers* (New York, NY, USA, 2010), SIGGRAPH '10, ACM, pp. 43:1–43:10. 2

[PIX] PIXOLOGIC: Zbrush. http://www.pixologic.com . 2

[RPC*10] ROHMER D., POPA T., CANI M.-P., HAHMANN S., SHEFFER A.: Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Trans. Graph. 29*, 6 (Dec. 2010), 157:1–157:8. 3

[RSW*07] ROSE K., SHEFFER A., WITHER J., CANI M.-P., THIBERT B.: Developable surfaces from arbitrary sketched boundaries, 2007. 2

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing* (2007), pp. 109–116. 3

[SVWG12] SOLOMON J., VOUGA E., WARDETZKY M., GRINSPUN E.: Flexible developable surfaces. *Comp. Graph. Forum 31*, 5 (Aug. 2012), 1567–1576. 2, 8

[UIM12] UMETANI N., IGARASHI T., MITRA N. J.: Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph. 31*, 4 (July 2012), 86:1–86:11. 2

[Wan08] WANG C. C. L.: Towards flattenable mesh surfaces. *Comput. Aided Des. 40*, 1 (2008), 109–122. 2

[WDAH10] WINKLER T., DRIESEBERG J., ALEXA M., HORMANN K.: Multi-scale geometry interpolation. *Computer Graphics Forum 29*, 2 (May 2010), 309–318. 3

[YBS07] YOSHIZAWA S., BELYAEV A. G., SEIDEL H.-P.: Skeleton-based variational mesh deformations. *Computer Graphics Forum 26*, 3 (2007), 255–264. EUROGRAPHICS'07. 2

[YYPM11] YANG Y.-L., YANG Y.-J., POTTMANN H., MITRA N. J.: Shape space exploration of constrained meshes. In *Proceedings of the 2011 SIGGRAPH Asia Conference* (New York, NY, USA, 2011), SA '11, ACM, pp. 124:1–124:12. 2

Appendix A:

Here we derive the fold difference of two folding operations $F_{\alpha,\theta}$ and $F_{\bar{\alpha},\bar{\theta}}$ on vertex $\mathbf{v}$. We make a disk centered at $\mathbf{v}$ as the domain D of the folding operations. Using a polar coordinate system with $\mathbf{v}$ as its origin, the folded shape $S_{\alpha,\theta}(r,\beta)$ from $F_{\alpha,\theta}$ has a parametric form

$$S_{\alpha,\theta}(r,\beta) = r \begin{pmatrix} \cos\alpha\cos(\beta-\alpha) - \sin\alpha\sin(\beta-\alpha)\cos\theta \\ \sin\alpha\cos(\beta-\alpha) + \cos\alpha\sin(\beta-\alpha)\cos\theta \\ |\sin(\beta-\alpha)|\sin\theta \end{pmatrix}$$

Considering a neighboring patch with radius ε, the L^2 distance between $S_{\alpha,\theta}(r,\beta)$ and $S_{\bar{\alpha},\bar{\theta}}(r,\beta)$ is

$$\int_{-\pi}^{\pi}\int_{0}^{\varepsilon} ||S_{\alpha,\theta}(r,\beta) - S_{\bar{\alpha},\bar{\theta}}(r,\beta)||^2 dr d\beta.$$

Substituting the parametric equation of $S_{\alpha,\theta}(r,\beta)$, this L^2 distance can be represented in an explicit form

$$\frac{4}{3}\varepsilon^3 d\left(F_{\alpha,\theta}, F_{\bar{\alpha},\bar{\theta}}\right)$$

where $d(F_{\alpha,\theta}, F_{\bar{\alpha},\bar{\theta}})$ is written in Eqn. 1.

Having got the formulation Eqn. 1, the following properties of our proposed formulations for fold difference and fold summation can be verified:

- $d(F_{\alpha,\theta}, F_{\bar{\alpha},\bar{\theta}}) \geq 0$. when $\theta \neq 0$, $d(F_{\alpha,\theta},\ F_{\bar{\alpha},\bar{\theta}}) = 0$ if and only if $\alpha = \bar{\alpha}$ and $\theta = \bar{\theta}$.
- $d(F_{\alpha,\theta}, F_{\bar{\alpha},\bar{\theta}})$ is independent of α and $\bar{\alpha}$ when $\bar{\theta} = 0$.
- When $\alpha = \bar{\alpha}$, it is true that $F_{\alpha,\theta} + F_{\alpha,\bar{\theta}} = F_{\alpha,(\bar{\theta}+\theta)/2}$; When $\bar{\theta} = 0$, $F_{\alpha,\theta} + F_{\bar{\alpha},0} = F_{\alpha,\theta/2}$ holds.

Since a folding operation can be decomposed into two rotations of angle $\pm\theta$ along fold direction α, blending rotations might be another choice to sum up folding operations. We tested quaternion blending techniques by treating $F_{\alpha,\theta}$ as a rotation $R_{\alpha,\theta}$. Similar summed folding operations are obtained except when the difference of two folding directions are near π. For example, $R_{0°,90°} + R_{170°,90°} = R_{85°,10°}$, while $F_{0°,90°} + F_{170°,90°} = F_{175°,89.6°}$. The blended folding better preserves the user input by using the proposed fold summation. Though our formulation for fold summation works well, it is still possible to derive other formulations from a study of blending rotations. To help understanding fold summation, two fold field summations are supplemented in Fig. 19.

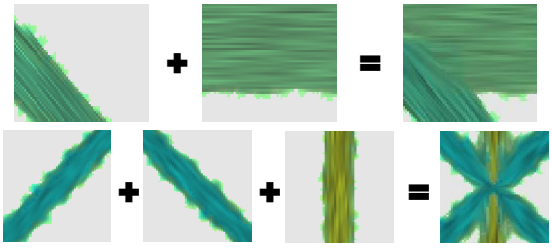


Figure 19: *From up to bottom: close-ups of fold field summations in Fig. 16 left and 11 bottom right . Fold fields are visualized as flow fields, in which the line integral convolution shows fold directions and the background color visualizes fold angles.*

© 2013 The Author(s)
Computer Graphics Forum © 2013 The Eurographics Association and John Wiley & Sons Ltd