# Non-smooth developable geometry for interactively animating paper crumpling

Camille Schreck, Damien Rohmer, Stefanie Hahmann, Marie-Paule Cani, Shuo Jin, Charlie C.L. Wang, Jean-Francis Bloch

# Non-smooth developable geometry for interactively animating paper crumpling

Camille SCHRECK,
Univ. Grenoble-Alpes, CNRS (Laboratoire Jean Kuntzmann), Inria, France
and
Damien ROHMER,
Univ. Grenoble-Alpes & Lyon, Inria, CNRS (Laboratoire Jean Kuntzmann), CPE Lyon, France
and
Stefanie HAHMANN, Marie-Paule CANI,
Univ. Grenoble-Alpes, CNRS (Laboratoire Jean Kuntzmann), Inria, France
and
Shuo JIN, Charlie C.L. WANG,
The Chinese University of Hong Kong, China
and
Jean-Francis BLOCH
Univ. Grenoble-Alpes, CNRS (Laboratoire Génie des Procédés Papetiers), France

We present the first method to animate sheets of paper at interactive rates, while automatically generating a plausible set of sharp features when the sheet is crumpled. The key idea is to interleave standard physically-based simulation steps with procedural generation of a piecewise continuous developable surface. The resulting hybrid surface model captures new singular points dynamically appearing during the crumpling process, mimicking the effect of paper fiber fracture. Although the model evolves over time to take these irreversible damages into account, the mesh used for simulation is kept coarse throughout the animation, leading to efficient computations. Meanwhile, the geometric layer ensures that the surface stays almost isometric to its original 2D pattern. We validate our model through measurements and visual comparison with real paper manipulation, and show results on a variety of crumpled paper configurations.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

General Terms: Paper, Developable Surface, Interactive Deformation, paper crumpling, isometric deformation

Additional Key Words and Phrases: Sharp Crease

## 1. INTRODUCTION

Paper is a very common material in everyday life: we generally use it when reading, writing or drawing, as well as for wrapping things up. In contrast, animations of this material are still almost absent from virtual environments, movies, and games. This is due to the extreme difficulty of efficiently capturing the dynamic, non-smooth developable geometry of paper sheets. The latter undergo irreversible damages upon deformation, causing most of them to end up in crumpled form. While state of the art physically-based models are able to generate visually compelling animations of paper crumpling [Narain et al. 2013], these have a high computational cost: computing a few minutes animation sequence may take several hours.

The goal of this work is to achieve the animation of paper crumpling processes at interactive rates. The macroscopic geometry of the paper depends on the microscopical fiber structure. Our insight is to enforce these features through a dedicated geometric model. This reduces most of the time usually spent in physically-based simulation of a fine mesh with high stiffness. Let us list the macroscopic features that need to be captured in order to crumple a sheet of paper in a visually plausible way:

**Length preservation.** Paper has a high in-plane stiffness, i.e. it neither stretches nor squeezes under classical conditions of usage. The assumption can be made that the surface of a sheet of paper preserves lengths upon deformation. In particular, although acceptable for animating cloth, the small elastic deformations between neighboring vertices typically allowed by deformable models should be prevented while modeling paper material. Because of length preservation, a sheet of paper can always be isometrically developed onto the flat pattern representing its original state. This leads to the well known developable property of paper.

**Sharp features.** Crumpled sheets of paper exhibit singular points, and possibly some sharp creases. Standard physically-based simulation methods for thin sheets usually assume that the limit surface approximated by a discrete mesh is smooth, and therefore do not handle discontinuities of tangential plane around singular vertices. In contrast, our model explicitly handles these sharp features.

**Plastic behavior and shape memory.** In real life, the microscopic fibers of paper or the bonds between them may break upon deformation, causing irreversible damages to the structure. Therefore, all singularities and creases are persistent over time, even after trying to flatten back the sheet. A dynamic model of paper must incorporate this persistence property, causing the number of singularities to always increase.

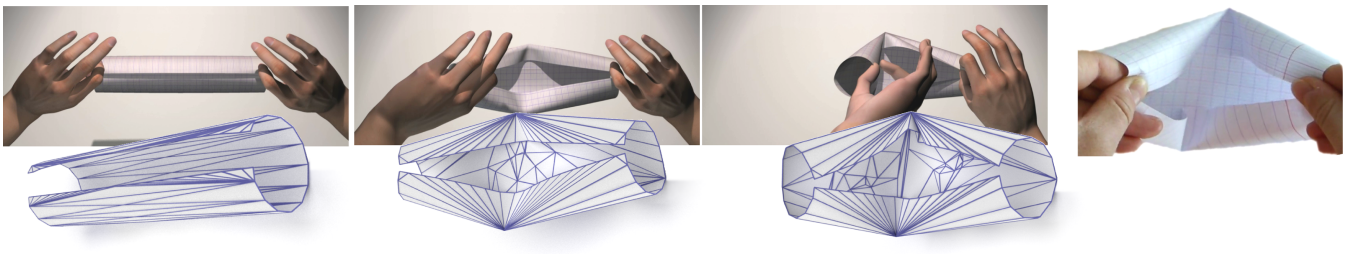

Fig. 1. Interactive folding of a virtual sheet of paper, after a cylindrical wrap. The mesh triangulation is kept coarse and dynamically adapts to the deformation. Our method automatically inserts sharp features where and when needed. A photo of a similar, real deformation is shown at the right.

In this work, we introduce a new surface model for virtual paper which allows for plausible, time-coherent deformations at interactive rates while incorporating the three characteristics we just listed. This model, namely a piecewise developable surface made of generalized cones, is designed to capture the sudden occurrence of singular points during the deformation process. It is deformed based on both physical constraints and geometrical laws. The fact that only a small set of conical patches is used for representing geometry reduces computational cost of several orders of magnitude, since it enables simulation to be performed on a very coarse mesh.

Our contributions include:

**A hybrid physically-based and geometric model.** The specific combination of geometric and physically-based layers we are using is the key feature of our approach: made of generalized conical patches that fit an adaptive set of singular points, our geometric layer ensures length preservation while visually capturing tangent discontinuities. The animation is guided by an underlying coarse simulation, where stiffness only needs to be of medium range. This is the key towards efficiency, since no ill-conditioned system needs to be solved.

**Realistic modeling of singular points.** We propose a new approach to detect when and where singularities should appear, on top of a standard coarse simulation. The set of sharp features is explicitly stored throughout the animation and used to model physical plastic behavior such as non-planar rest poses.

**Optimal, adaptive isometric meshing.** The surface geometry is defined by an adaptive mesh that aligns its edges along the rulings of the generalized conical patches and along the sharp edges of the folds. Meanwhile, these edges efficiently maintain their initial length. This brings an accurate representation of isometric surfaces with sharp features using only a small number of triangles instead of dense mesh subdivision.

**Validation through comparison with real paper.** We use experiments with real paper in a set of simple crumpling configurations test cases to validate our method: we compare the kinetics needed for the appearance of surface singularities. In addition to these statistical results, we provide visual comparison with videos of real paper crumpling, enabling us to show that our method generates the same first folds when a sheet of paper is crumpled.

## 2. RELATED WORK

**The phenomenon of paper crumpling** appears due to localization of mechanical stress, leading to permanent deformation of the initial flat piece of paper. When the stress exceeds the limit of resistance of the bonds between fibers, these bonds break leading to plastic deformations. Mechanical behavior of paper plays consequently a major role, and has been extensively studied [Steenberg 1947; Thorpe 1981; Makela and Ostlund 2003; Coffin 2009]. More specifically for crumpled paper, Amar and Pomeau [1997] have underlined the relationship between the mechanical behavior (elasticity theory) and the geometry of developable surfaces.

**Physically based modeling** is the most standard approach to animate thin sheets of material. As simulating paper material using fully non-linear 3D finite elements would be too expensive for computer graphics applications [Bronkhorst 2003], thin sheet materials are typically modeled in our field using linear elastic behavior. Therefore, elements are allowed to stretch a little. This approach has been successfully used to model cloth [Provot 1996; English and Bridson 2008; Wang et al. 2010], and was extended to bend sheets of paper using the thin plate theory [Grinspun et al. 2003; Burgoon et al. 2006]. However, this method cannot handle the dynamic formation of singular points and sharp edges: the latter need to be manually inserted within the mesh.

Modeling the plastic behavior of the paper, with the progressive formation of sharp features, was already addressed in the graphics literature. Kang *et al.* [2009] propose a dedicated mass spring simulation where edges of the underlying triangulation can be split into two pieces, while the midpoint is modeling a sharp vertex on the surface. The approach enables real-time deformation but suffers from visual artifacts as edge-split and singularities can only occur along preexisting coarse triangle edges.

Gingold *et al.* [2004] proposes a discrete model for inelastic deformations based on bending strain. Recently, Simnett *et al.* [2009] and Narain *et al.* [2012; 2013] combine this inelastic deformation model with an adaptive remeshing method. When the surface deforms, the mesh is locally subdivided in highly curved regions, while the low curvature parts stay more sparsely sampled. This results into high-quality visual simulation of paper. Yet, this approach is computationally expensive due to the large number of triangles in the folded regions, so it cannot be used in interactive applications.

**Geometric modeling** has also been used to create a variety of developable surfaces. Frey [2004] proposes a buckled developable surface construction using triangulation for 2D height fields. A general algorithm to generate developable surfaces from 3D boundary curves is introduced in [Rose et al. 2007], while other works address developable surfaces based on simple shapes (e.g., strips) [Pottmann and Wallner 2001; Chu and Séquin 2002; Liu et al. 2006; Bo and Wang 2007; Pottmann et al. 2010]. These approaches only study smooth developable surfaces and none of them considers how to form and retain sharp features during the modeling process. [Decaudin et al. 2006] proposes a method for modeling sharp features using a minimal mesh, but the latter is restricted to specific cylindrical buckling patterns.

Another family of approaches to achieve developability is to progressively improve it using constrained optimization of input surfaces. Wang *et al.* [2004; 2008b] minimize an objective in terms of discrete Gaussian curvature; A local/global optimization method is proposed in [Kilian et al. 2008] to model a discrete developable surface with curved folds, by fitting a set of 2D quadrilateral pieces to the original surface. These approaches are computationally intensive. Moreover, although they create developable surfaces, they cannot explicitly formulate length preservation with respect to an original 2D pattern.

Bending and creasing virtual paper can also be expressed through geometric deformation [Kergosien et al. 1994]. An interactive tool is presented in [Zhu et al. 2013] to fold thin sheet of materials. Other methods addressed interactive origami modeling [Tachi 2010; Mitani and Igarashi 2011; Solomon et al. 2012] when the folds are explicitly defined by the user. A geometric way to create a virtual sheet of paper with sharp features from a 3D boundary curves and a 2D pattern is proposed in [Rohmer et al. 2011] but cannot be extended to animation.

In this work, we achieve temporal coherence, isometry to a pattern, and the modeling of sharp features with developable surfaces. In contrast with prior work on origami, we are only interested in the sharp features that naturally appear during arbitrary paper manipulation. Therefore, we do not model creases - typically formed by pressing along a fold with finger or with an instrument - but singular points, such as those shown on the photo at the top of Figure 2.

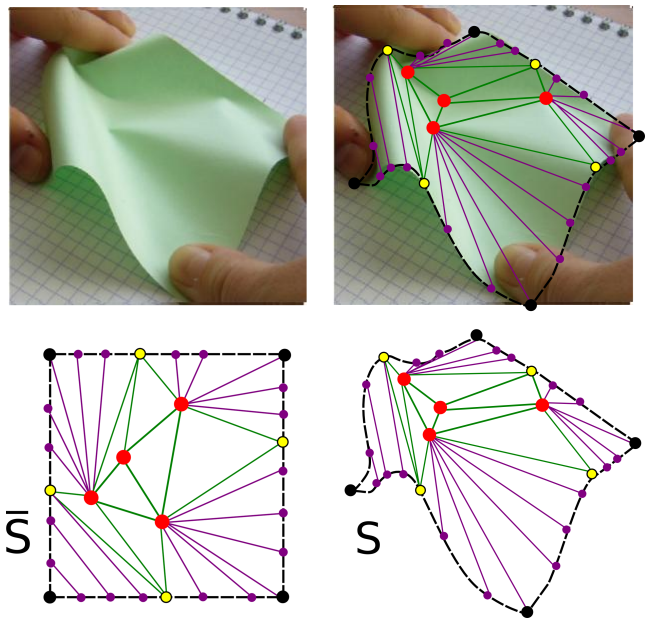


Fig. 2. Singular points appear when a sheet of paper is bent in several directions (photograph at the top left). Our geometric surface representation (pattern on the bottom left, and 3D surface on the bottom right) includes *flat regions* (green triangles) and *curved regions* made of generalized cones (magenta triangles and quadrangles). Singular vertices (red dots) are the apices of some of the generalized cones. Flat regions are defined between singular vertices and junction vertices (yellow dots) which separate flat and curved regions on the boundary. The user directly manipulates the model through handles (black dots) on the boundary.

## 3. A NEW HYBRID MODEL FOR PAPER MATERIAL

In this section, we introduce our new model, dedicated to allow the interactive deformation and crumpling of sheets of paper. We then give an overview of the associated animation algorithm.

### 3.1 Geometry and physics of paper sheets

The surface model used to represent the shape of a sheet of paper must be highly deformable, as sharp features may appear anywhere during deformation. This usually leads to the use of dense meshes, which provide a large number of degrees of freedom and reduce visual artifacts due to badly oriented edges. Yet, only a limited number of triangles can be handled at interactive rates.

A key feature of our surface model is to represent sharp features and smooth surface parts separately. While sharp features are explicitly modeled as positional parameters – enabling the creation of an arbitrary number of singular points, located anywhere – smooth parts are modeled using constrained parametric surfaces. This enables us to reduce the number of triangles of the adaptive mesh used for animation and display, while providing all the necessary degrees of freedom. This geometrical model is interwoven with a physical simulation with a coarse mesh in order to guide shape deformations. The choice of this hybrid approach for paper crumpling is motivated, not only by efficiency, but also by specific geometrical and physical properties of paper material discussed next.

As already mentioned, sheets of paper can be considered as developable surfaces. They can be unfolded into the plane without distortion by preserving in-plane distances, i.e. by remaining isometric to their 2D pattern. A developable surface is a ruled surface with zero Gaussian curvature (product of the principal curvatures) everywhere. Gauss' Theorema Egregium states that the Gaussian curvature is an intrinsic invariant of a surface. As a consequence, if a smooth part of the paper is bent such that one principal curvature is not zero, then the other principal curvature in the direction of the fold is necessarily zero. Thus the surface becomes rigid in the direction of zero-curvature, causing discontinuities to appear instead of having the surface bend if a compression is applied in the orthogonal direction. In consequence, forcing some general bending on a sheet of paper, as in Figure 2, leads to the appearance of singularities in the surface geometry.

The appearance of singularities has also been studied in the mechanical literature: Due to the fact that the bending rigidity of

thin plates is much smaller than their stretching rigidity, the paper mechanical transformations are favorable to bending [Cambou and Menon 2011]. In addition, thin sheets have the specific behavior of concentrating elastic energy when being constrained [Witten 2007]. When the thickness tends to 0, this energy concentrates into singular points, which are called **d-cones** [Amar and Pomeau 1997; Mahadevan and Cerda 1998; Cerda and Mahadevan 1998]. These d-cones denote developable generalized cones defined by a fixed point called the *apex* and a one-parameter family of straight lines (rulings) passing through the apex. Outside of the influence of the d-cones, the surface is smooth and exhibits a low distribution of elastic energy [Schroll et al. 2011]: it can be deformed accordingly to standard linear models for continuous mechanics laws. Figure 2 shows a photograph of a crumpled sheet of paper (left) enhanced by sketches of the singular points and the rulings of the corresponding d-cones (right).

These observations give us the main requirements for a model to capture simplified, yet visually realistic paper behavior:

• The surface should be a piecewise continuous developable surface composed of generalized cones (see the magenta rulings in Figure 2) and planar pieces (in green in Figure 2). Note this structure is similar to the one used by Frey [2004].

• The parts composed of generalized cones should be able to smoothly bend only in the direction orthogonal to their rulings, while planar pieces should be able to bend in any direction. Bending conflicts should be solved by the introduction of apices of d-cones. In our solution, these d-cones are modeled as specific vertices that we call *singular vertices* (red dots in Figure 2). We record the position of these vertices on the 2D-pattern and on the 3D shape of the deformed sheet.

• The motion of the surface should be guided by continuous mechanics in between plasticity events, i.e. when some paper fibers or their bonds break leading to the creation of a new singular point.

Animating such a structure raises specific challenges such as simultaneously bending smooth surface parts and maintaining singular points, while preserving isometry with the 2D pattern and therefore preserving developability. In addition, new singularities have to be integrated incrementally. Our model, presented next, is especially well suited to handle these constraints, since the singular vertices are stored explicitly and govern the generation of the smooth surface parts.

## 3.2 Our geometric and physical model

During the whole animation process, we maintain an isometric mapping between the paper surface $S$ and its pattern $\overline{S}$ defined in the 2D parameter domain. $\partial S$ denotes the sheet's boundary.

In this work, we use two representations of the surface $S$: a geometric representation $S_G$ which explicitly approximates $S$ by a set of generalized cones and planar parts, enabling developability enforcement and the explicit handling of singular points, and a physical representation $S_P$ which is a triangular mesh used for the physical simulation steps.

The geometric model $S_G = \{\mathscr{C}, \mathscr{F}\}$ is a coarse mesh defined as a set of curved regions $\mathscr{C} = \{C_i\}$ (in purple in Figure 2) and a set of flat regions $\mathscr{F} = \{F_i\}$ (in green in Figure 2).
Each *curved region* $C_i$, that we simply call $C$ for the sake of simplicity when there is no ambiguity, is segmented into a set of generalized cones defined by a one-parameter family of rulings $\mathscr{R}_C$. Each ruling of $\mathscr{R}_C$, $r = \{p_1, p_2\}$, is defined and delimited by two vertices $p_1$ and $p_2$. We distinguish two cases. First, the two points are two vertices lying on the sheet's boundary $\partial S$, in which case their corresponding cone's apex is either outside $S$ or $\in \partial S$ (magenta quadrangles in Figure 2). Second, one vertex lies on $\partial S$ and the other is an interior singular vertex (magenta triangles in Figure 2), in which case their corresponding cone's apex is the singular vertex. The generalized cones are coarsely sampled on the boundary $\partial S$. The sampling density of the boundary $\rho_{\text{bound}}$ is a fixed parameter defined by the user.

Each *flat region* $F_i$, that we similarly call $F$, is defined by a set of connected triangles $\mathscr{T}_F = \{T_j\}$ that triangulate a planar region of the surface. Each of those triangles is delimited only by singular vertices or vertices from $\partial S$.

The interior vertices of $S_G$ (red in Figure 2) are singular points of the surface $S$. They are therefore the apices of developable cones. The yellow vertices of $S_G$ in Figure 2 are another kind of singular vertices. They lie on $\partial S$ and belong to at least two different regions (curved or flat). Note that all the coarse triangles of the pattern $\overline{S}$ whose vertices are singular (shown as green triangles in Figure 2) are necessarily mapped onto triangles in the 3D space and not to a more general curved surface: indeed, as all green edges from $\overline{S}$ are by definition rulings of the surface $S$, they are necessarily straight line-segments, yielding a flat surface in-between (since developable surfaces with planar boundaries are planar), i.e. a triangle.

The triangle mesh $S_P$ is created by adding ghost vertices to $S_G$ to ensure a regular and symmetrical sampling. $S_P$ is thus ready to be used by a simulator. The process for constructing $S_P$ from $S_G$ will be detailed in Section 4. The set of vertices of $S_G$ is a subset of the vertices of $S_P$, so the displacement of the surface computed through the simulation can be easily mapped to the vertices of $S_G$.

All the notations are summarized in Table I.

Table I. Summary of symbols used

| Symbol | |
|---|---|
| $S$ | paper's surface |
| $\overline{S}$ | 2D pattern corresponding to $S$ |
| $\partial S$ | boundary of the paper's surface |
| $S_G$ | geometric structure of $S$ composed of $\mathscr{C}$ and $\mathscr{F}$ |
| $S_P$ | physical triangle mesh of $S$ |
| $\mathscr{C}$ | set of curved regions of $S_G$ |
| $\mathscr{F}$ | set of flat regions of $S_G$ |
| $\mathscr{R}_C$ | one-parameter family of rulings of the curved region $C$ |
| $\mathscr{T}_F$ | set of triangles of the flat region $F$ |

## 3.3 Overview of the animation algorithm

**Initialization.**
The input is a developable surface $S$ defined as a mesh made of triangles and/or quadrangles, and its isometric mapping to a 2D pattern $\overline{S}$, meshed with the same mesh connectivity. Typically, $S$ is initially a flat mesh, although other input shapes could be used. Additionally, some *handles*, i.e. a finite set of points that the user can manipulate, are defined. In our current implementation, these

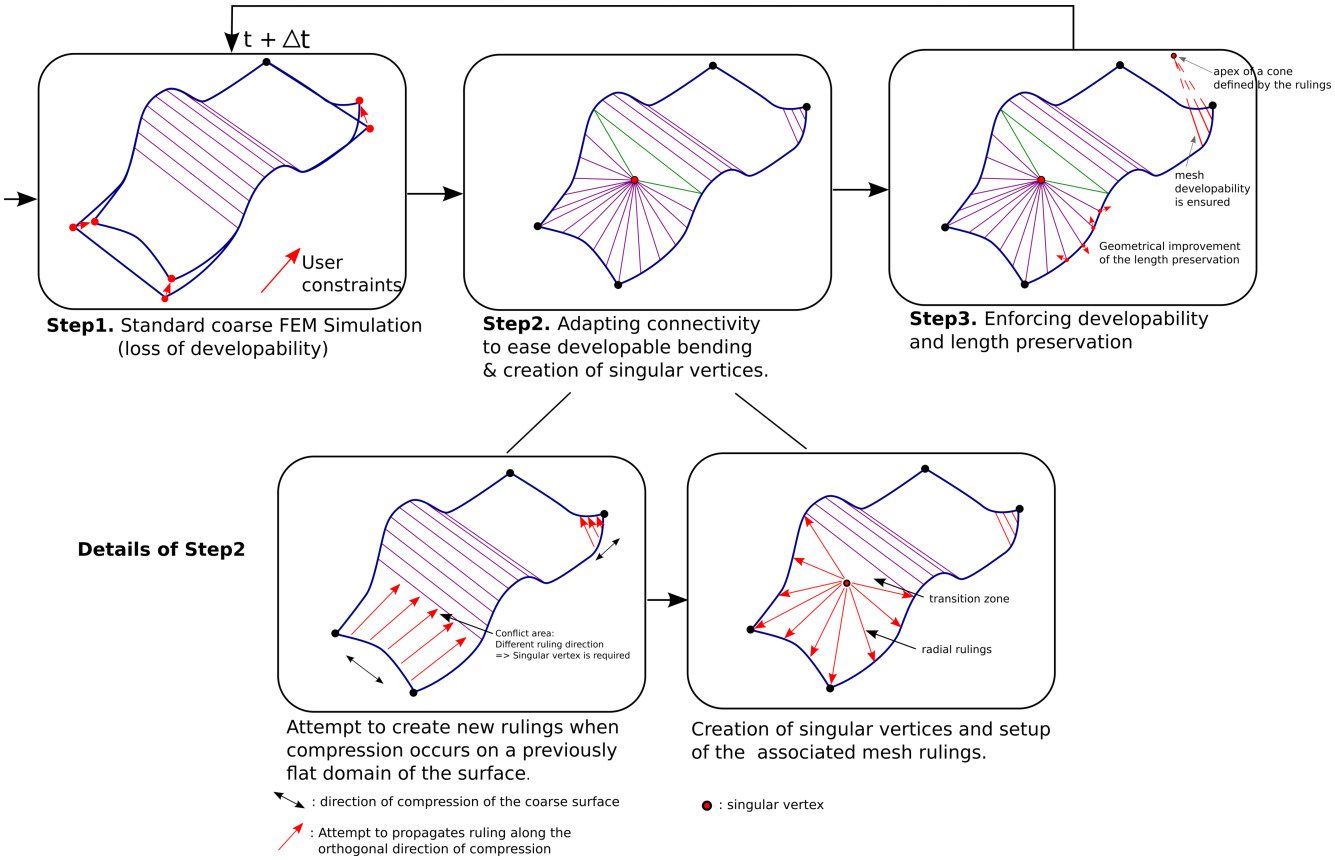


Fig. 3. Overview of the algorithm: each animation step combines a standard FEM simulation (step 1) with a remeshing step (step 2) that approximate the surfaces using generalized cones and mesh it accordingly, notably by finding the position of singular points and enforcing them as apex of cones. Developability is improved in step 3.

handles are located on the boundary $\partial S$ of the sheet. They act as hard constraints and are used to govern the deformation.

**Animation loop.**
As the handles move, the paper deforms and may crumple. At each time step, the algorithm interweaves a physically-based simulation step to guide the primary smooth deformation of $S$ with geometrical steps (see Figure 3):

Step 1: **Elastic deformation** (Section 4)
A state-of-the-art physically-based simulation is applied for efficiently deforming the surface mesh $S_P$ in a plausible way, but without taking care of paper plasticity at this stage. Isometry to $\overline{S}$, and therefore developability, may thus get lost. This step outputs new positions of the vertices of $S_G$.

The two following steps use a geometric analysis approach to adapt $S_G$ to the surface computed through elastic deformation while maintaining specific paper properties.

Step 2: **Modeling bending and crumpling** (Section 5)

- *Remeshing of the compressed planar parts.* Some of the flat regions are remeshed according to locally measured compressions in order to ease subsequent bending.
- *Singularity generation.* A geometric analysis of $S_G$ indicates if and where new singular vertices have to be generated. $S_G$ is remeshed accordingly, see Figure 3-middle.

Step 3: **Developable and isometric tracking** (Section 6)
Firstly, $S_G$ is segmented into quasi developable regions by locally computing the best approximation by generalized cones and developability is geometrically enforced by aligning the mesh edges along these rulings. Secondly, isometry preservation is optimized by constraining edge lengths. The structure of $S_G$ is finally used to update the mesh $S_P$ before displaying it and performing the next iteration of the animation loop, see Figure 3-right.

Throughout the description of our method, we will refer to Figure 16, which shows another general example case spanning two successive steps of the algorithm loop.

## 4. ELASTIC DEFORMATION

The first part of each animation step consists in computing the deformation of the sheet of paper using a standard physically-based simulation method, based on elastic energy.

Our implementation makes use of the Finite Element Method (FEM) code provided by Narain *et al.* [2012]. It relies on Green

strain computation for stretching forces, while bending forces are computed using *discrete flexural energy* [Bridson et al. 2003; Grinspun et al. 2003]. Time integration is performed using implicit integration and we usually obtain convergence after 10 to 100 iterations. The simulation also integrates collision detection based on a hierarchy of bounding volumes [Tang et al. 2010] for implicit time integration. Collisions are solved using the sparse Cholesky method and response is computed using a combination of non-rigid impact zones [Harmon et al. 2008] and repulsion springs [Bridson et al. 2002].

Handle positions, interactively controlled by the user, are enforced as hard constraints on the corresponding vertices of $S_P$ while the physical simulation relaxes the deformed mesh to a smooth rest state. We also provide control on the tangent plane at the handles through the control of 2 neighboring vertices. This enables us to achieve more realistic movements, mimicking the tangent plane control due to the pressure of fingers. Arrows (a) and (b) in Figure 16 illustrate the user-defined deformation followed by a physical-based simulation.

Yet, contrary to standard elastic simulation, our triangular mesh is made of a very few triangles, which are moreover dynamically adapted during the deformation. This brings specific constraints when setting up the simulation, described next.

**Enriching the coarse mesh before simulation.**
Let us first note that the coarse mesh $S_G$ composed of triangles from $\mathscr{F}$, and triangles and quadrangles from $\mathscr{C}$ (see Figure 4-left) brings most of the necessary degrees of freedom for surface deformation, while allowing very efficient computation. The edges from $\mathscr{C}$ are oriented along the rulings of the surface which enables to maintain the rigidity of the bent region, and the vertex positions of $\partial S$ adequately sample the degrees of freedom needed for manipulating the boundary of the surface.

Still, $S_G$ represents the planar regions from $\mathscr{F}$ without using any interior vertex. This coarse representation does not provide the necessary degrees of freedom for the simulation to locally deform these parts of the surface. To enable a more flexible configuration, we insert extra vertices in $S_P$, interior to the surface and on the boundary $\partial S$, such that flat domains are uniformly and isotropically sampled. The number of additional vertices is controlled through user-defined density values, $\rho_{\text{bound}}$ for the boundary and $\rho_{\text{int}}$for the interior. The new vertices are connected using Delaunay triangulation.

Similarly, the quadrangles of the curved regions $\mathscr{C}$ from $S_G$ need to be triangulated in $S_P$. To preserve the original symmetry of the quad, an extra vertex is inserted at its barycenter and connected to the four vertices. See Figure 4.

In practice, $S_P$ is computed just before display, at the end of the animation loop, enabling us to use it as well as the visual representation of the paper surface. See the arrow (g) in Figure 16. $S_P$ is then used for simulation at the next animation step (arrows (a) and (b)). Finally, the new position of its vertices are used to update the position of the $S_G$ vertices (arrow (c)).

**Modeling fiber damage.**
In a smooth bent configuration, i.e. without interior singular points, bending forces tend to make the sheet flat when no constraint is applied, since the bending force equals zero when the dihedral angle between triangles is also zero. But when the sheet is crumpled and contains singular vertices, the surface should not come back to such a flat configuration by itself, as paper fibers or their bonds have been damaged. We model this plastic behavior using a non-zero rest angle for the bending force in the presence of singular vertices. We call $\theta_{\text{rest}}$ the rest dihedral angle, meaning that the bending force tends to generate triangles with dihedral angle $\theta = \theta_{\text{rest}}$.
Let us call $N_i^{\text{sing}}$ the number of edges around the vertex $i$. For each vertex $i$ we define:

$$\theta_i = \begin{cases} \pi/(8N_i^{\text{sing}}) & \text{if the vertex } i \text{ is a singular vertex} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

We set the dihedral rest angle of edge $(i, j)$ to

$$\theta_{\text{rest}} = \theta_i + \theta_j.$$

This way, the rest angle is distributed around the singular point and the singularity will remain visible even when no constraint is applied. An edge between two singular points will be marked more than others. Note that the value $\pi/8$ was set experimentally while trying to match the behavior of standard sheets of paper (70 $g/m^2$). This value could be tuned to model thinner or thicker paper material.

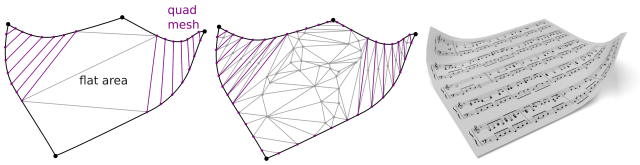


Fig. 4. Creation of the physical triangle mesh structure $S_P$ (middle) from the geometrical mesh structure $S_G$ (left). The added vertices and edges in $S_P$ are shown in gray.

## 5. MODELING BENDING AND CRUMPLING

Thanks to our hybrid model taking into account the specific paper's developable properties, restricting our surface $S_G$ to deform with bending and crumpling instead of stretching can be achieved by adapting its connectivity while still keeping a very coarse triangulation. We first describe how we adapt the flat regions of $S_G$ so that they can be bend easily, and then describe how singular points are added so that the surface is allowed to crumple.

### 5.1 Analyzing mesh compression to generate bent surfaces

In the following, we consider that two adjacent triangles are coplanar if the dihedral angle between them is smaller than a user-defined threshold $\theta_0$. Using this approximation, we define a *flat region* $F$ as a maximal connected set of pair-wise coplanar triangles $\mathscr{T}_F = \{T_j\}$ (the dihedral angle between any pair of adjacent triangles should be $\leq \theta_0$).

A flat region $F$ should be able to easily bend in any direction. The enriched triangulation $S_P$ used for simulation enables $F$ to slightly bend, but the corresponding coarse triangles in $S_G$ can only become compressed. To make it possible for this region to further bend in subsequent steps, we modify the connectivity of $S_G$ and insert a new curved region, as follows:

When a flat region bends in $S_P$, leading to compression for the coarse triangles in $S_G$, new rulings (i.e. directions of zero

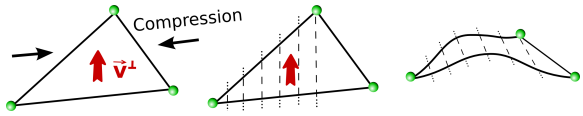


Fig. 5. Remeshing of a compressed triangle from SG using rulings in the direction of minimal compression $\mathbf{v}^{\perp}$, to favor uni-directional bending in subsequent simulation steps.

curvature) are inserted into $S_G$ along the direction of minimal compression (see Figure 5). Therefore, bending in the direction of maximal compression will be favored, whereas the surface will be more rigid in the direction of the rulings and more likely to avoid future compression or bending along this orthogonal direction.

This specific local remeshing is performed in the 2D pattern space before applying it to $S_G$ in 3D. It is applied to each flat region $F \in \mathscr{F}$ and consists of 3 steps:

**(a)** Computing the most compressed triangle in $F$ and the direction of minimal compression;

**(b)** Computing a triangle strip spanning $F$, and aligned with the direction of minimal compression;

**(c)** Remeshing $F$ locally, either:

(c1) without inserting new singular vertices

(c2) with insertion of new singular vertices (see Section 5.2).

**(a) Analyzing local compression**
For each triangle in $F$, the direction in which the triangle is the most compressed is computed using the standard method based on stretch tensors described in Rohmer *et al.* [2010] which computes the principal directions of strain. We denote $\mathbf{v}$ the direction of maximal compression, and $\mathbf{v}^{\perp}$ the direction of minimal compression, orthogonal to $\mathbf{v}$. We also compute the ratio of compression of the triangle in $\mathbf{v}$ direction and consider that the flat region $F$ is compressed if it contains at least one triangle with a compression ratio greater than the user-defined threshold $\varepsilon_c$.

**(b) Computing a triangle strip aligned with $\mathbf{v}^{\perp}$**
We call $T_{\text{comp}}$ the triangle of $\mathscr{T}_F$ (set of triangles of $F$) of largest compression ratio (red triangle in Figures 7,8,9(a)). This triangle should actually not belong to a flat region anymore, but should instead be part of a bent surface. To compute the region affected by this bending, we start with $T_{\text{comp}}$ and collect the adjacent triangles by propagating the endpoints of imaginary rulings aligned along the minimal direction of compression in both directions until reaching the boundary of $F$, see Figure 6. This propagation procedure, detailed in the Appendix A, computes a triangle strip (the red triangle and the two adjacent ones in Figure 7(ab)) and returns the *borders* of the triangle strip. We call *border* the edges of triangles in $\mathscr{T}_F$ where the triangle strip ends. This *border* is either a portion of $\partial S$ (Figure 6(b)), or an internal ruling (Figure 6(c,d)).

**(c) Remeshing $F$ locally.**
Depending on the type of border, we decide whether the rulings to be inserted can remain parallel or need to be conical with a singular vertex as common apex. This procedure is explained with more details in Appendix B. In the case (c1) where the border is part of $\partial S$ (Figure 6(b)), this border is not constrained and can thus freely bend. The current set of rulings is then used to tessellate the triangle strip into quads (see Figures 7(c,d)). The remaining parts of $F$ are triangulated and stay planar in $S_G$. But in the case (c2) where the border is an edge common to either another flat region

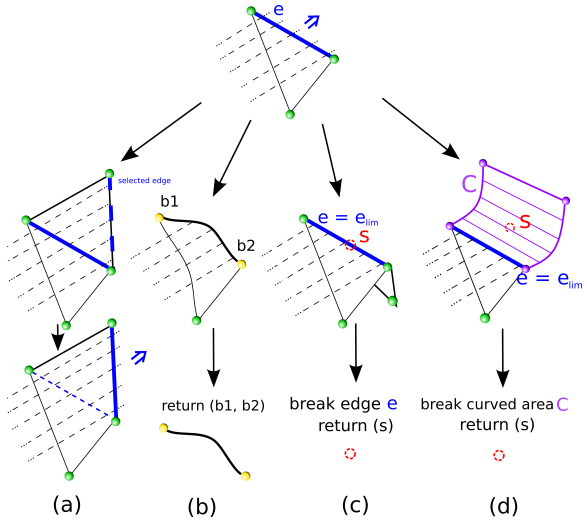


Fig. 6. The four possible cases of the imaginary rulings propagation algorithm. The edges met during propagation may respectively belong to: another triangle of the same flat region (a), $\partial S$ (b), another flat region (c), a curved region (d).

or a curved region (Figure 6(c,d)), this border is in fact a ruling of this other region and thus constrained to be rigid.

The second case (c2) leads to the main contribution in this section, namely the insertion of new singular points: the conflict between the rulings we intend to place in $\mathscr{T}_F$ and the already existing curved part they cross is solved by generating a new singular point (see Section 5.2). This point will be the apex of a generalized cone, so the rulings we actually insert have to be conical instead of parallel to the direction of minimum compression.

Figures 7 to 9 show the remeshing of a flat region in the three cases that can occur for borders: the imaginary rulings illustrated as dashed lines reach the non constrained boundary $\partial S$ respectively at both ends (Fig. 7), at only one end (Fig. 8), and never (Fig. 9). The whole process is also represented in Figure 16(d), where the first row shows the case of inserting rulings into a non-constrained flat region, whereas in the second row a singular point needs to be created.

## 5.2 Singularity generation

As stated previously, a piece of paper that starts to bend generates rulings that propagate throughout the whole surface. When these rulings intersect a preexisting ruling (see Figure 6(c,d)), we introduce a new singular point to accommodate for the two non-compatible constraints. In the following, we call $\mathbf{e}_{\text{lim}}$ the edge in the mesh which coincides with the preexisting ruling.

**Finding the singularity position.**
The apparition of new singularities when real pieces of paper are compressed in two opposite directions depends on various parameters such as paper mechanical properties and external mechanical constraints. We can also note that the position of the singularity can largely vary even when crumpling paper sheets of the same thickness under similar constraints.

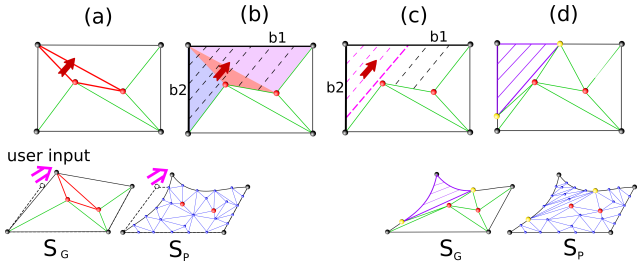


Fig. 7. Generation of rulings to favor further bending when the propagation reaches the boundary $\partial S$ at both ends. (a) The triangle of maximal compression is in red. (b) Propagation of triangle strip with imaginary rulings as dashed lines. Both borders belong to $\partial S$. (c) Rulings inserted (purple lines). (d) Final meshing of $\overline{S}$ and $S_G$.
Top: the steps in the 2D pattern $\overline{S}$.
Bottom: the 3D structure of $S_G$ and the mesh $S_P$ used by the physical simulation at the beginning (left) and at the end (right) of the process.

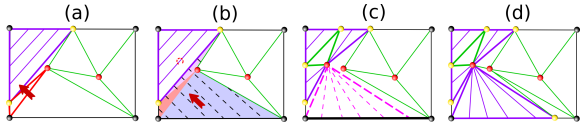


Fig. 8. Insertion of a singular vertex and generation of rulings. (b) The imaginary rulings (dashed lines) of the triangle strip encounter a non-constrained border (fat black) at one end and a constrained existing ruling (fat purple) at the other end. (c) A singular vertex is created and (d) conical rulings are inserted in $S_G$.

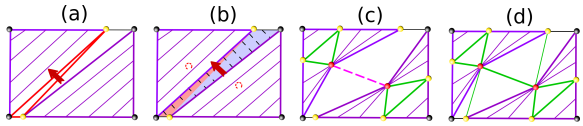


Fig. 9. Insertion of 2 singular vertices.(a) compressed triangle in flat region. (b) Imaginary rulings and 2 constrained borders (fat purple lines). (c) 2 singular points and a ruling between them are then inserted. (d) Final meshing of $S_G$.

As easily verified by experiments, changing the direction of zero curvature (direction of the rulings) of a piece of paper in order to bend it smoothly in an arbitrary direction is easy as long as the surface is flat, or has a sufficiently low curvature along its current rulings direction. The singularities appear when the change of the zero-curvature direction reaches another region with high curvature values. This is why singular points usually appear at the junction between weakly curved regions and higher curved regions.

Let us first consider the case where $\mathbf{e}_{\text{lim}}$ is a ruling of a curved region $C$ (blue edge in Figure 6(d)). In order to derive a law for positioning the singular vertex, we repeated many times the same simple experiment shown in Figure 10 with real paper. We observed that the singular point does not exactly appear on $\mathbf{e}_{\text{lim}}$ but is shifted in the direction of $\mathbf{v}^{\perp}$ where the surface starts to bend. See Section 7.1 and Figure 17 for more details on this experiment. We therefore pursue the previous propagation algorithm until we reach a ruling with higher curvature in the orthogonal direction, which we quantify using the dihedral angle (i.e. the angle between the two adjacent triangles or quadrangles whose common edge is the ruling). Let $\mathbf{e}_{\text{sing}}$ be the first ruling encountered in the $\mathbf{v}^{\perp}$ direction with dihedral angle larger than $\theta_0$. The edge associated with this ruling will be considered as the most probable location for new singularity insertion.

We define $\overline{s}$, the actual position in the 2D pattern $\overline{S}$ of the singular point $s$ to be added, as being randomly close to the edge $\overline{e}_{\text{sing}}$ as illustrated in Figure 10 (left). To this end, we choose the probability $\chi(\overline{s})$ of creation of a singularity at a position $\overline{s}$ as being uniform along the edge $\overline{e}_{\text{sing}}$, and normally distributed according to the distance to this edge. This is done using the law $\chi(\overline{s}) = \exp(-d^2/\sigma^2)$, where $d$ is the distance between $\overline{s}$ and $\overline{e}_{\text{sing}}$, and $\sigma$ is a parameter taking into account the variability of the position of the singular vertex. In our experiments, we chose $\sigma = 0.1$ cm. Finally, we compute the 3D coordinates of $s$ from $\overline{s}$.

The second possible case we consider arises when the ruling $\mathbf{e}_{\text{lim}}$ is an edge shared by two triangles belonging to two different flat regions (blue edge in Figure 6(c)). Then, the large dihedral angle prevents the surface from bending. In this case, we insert the singularity directly on this edge at a random position and set $\mathbf{e}_{\text{sing}} = \mathbf{e}_{\text{lim}}$.

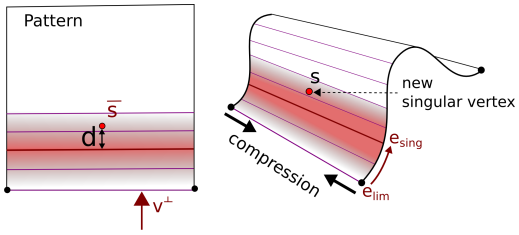


Fig. 10. A configuration where compression acts on top f an existing curved domain. The singular vertex to be inserted has a higher probability to appear in the red region. The edge $\mathbf{e}_{\text{sing}}$ at the center this region is highlighted in red.

**Remeshing around the singularity.**
When a new singularity $s$, i.e. the apex of a d-cone, is inserted, conical rulings are created in the adjacent flat region, as explained in Section 5.1. Let us now describe how we change the existing rulings in curved surface parts such as $C$ in Figure 6 (d), after a new singular point has been inserted. Figure 11 illustrates our method and shows corresponding experiments with real paper at the bottom.

As a singular point is the apex of a generalized cone, introducing one into a curved region implies a change of the nearby rulings so that they pass through the apex $s$. As changing the orientation of a ruling implies a modification of local curvatures, this modification can only be allowed where the surface is not too curved. We therefore perform a local remeshing around the singular vertex as long as the dihedral angle associated with the nearby rulings is smaller than the threshold $\theta_0$.

In practice, this is done by first removing the rulings of $C$ from $e_{\text{sing}}$ in the direction $\mathbf{v}^{\perp}$ until reaching a limit ruling $e_{\text{sup}}$, having a dihedral angle larger than $\theta_0$ or until reaching the border of $C$ (in which case $e_{\text{sup}}$ is the last ruling of $C$). We then create new curved region(s) defined by the same cone apex $s$,

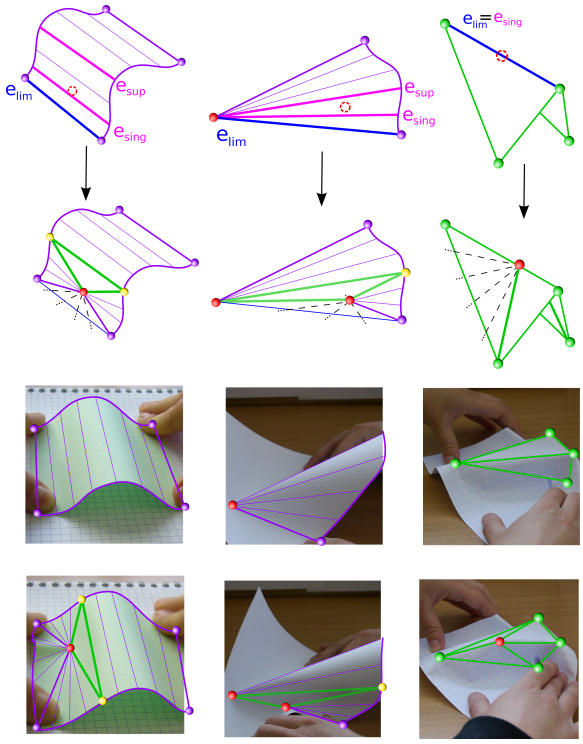


Fig. 11. Newly inserted singularity and associated remeshing.

between $e_{\text{lim}}$ and $e_{\text{sup}}$ (see Figure 11-left and middle). To enable geometric continuity between the newly remeshed region and the existing bent surface, we introduce a flat surface (green triangle in Figure 11-left and middle) as transition.

In the case of a singular point created between two triangles belonging to two different planar regions (Figure 11-right), the common edge is considered as a degenerate curved region with only one ruling.

## 6. DEVELOPABLE, ISOMETRIC TRACKING

The last step of our method consists in computing a fully developable approximation of the current geometric mesh $S_G$ and insuring its isometry with the 2D pattern $\overline{S}$. The resulting geometric mesh will be refined into $S_P$ by tessellating flat regions before display.

While segmenting an arbitrary mesh into approximately developable regions is a difficult problem requiring non-linear optimization [Leopoldseder and Pottmann 1998; Julius et al. 2005], we take benefits of the specific structure of $S_G$ in our case: we first update junction points at the limit between flat and curved regions (Section 6.1) and then segment the later into generalized cones, providing us rulings for edge alignment (Section 6.2). We then apply a new method, introduced in Section 6.3, for improving the length of mesh edges in order to keep accurate isometry with $S$. Note that ensuring developability first gives us a good starting point for this isometry optimization process.

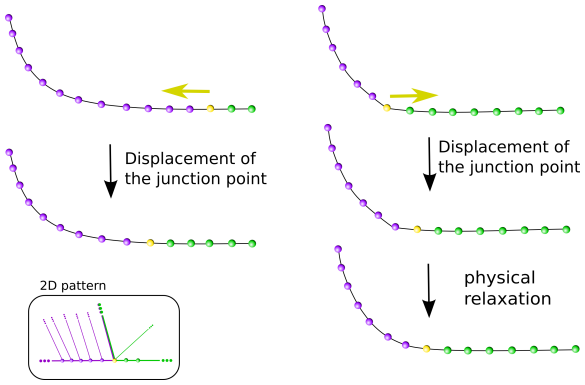


Fig. 12. 2D view of a border of the sheet of paper showing the displacement of a junction point joining a flat region and a curved region.

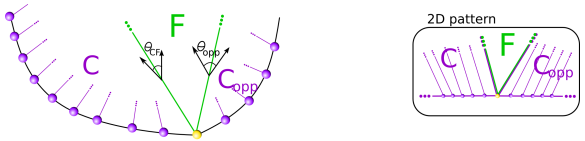


Fig. 13. A junction point joining a flat region and two curved regions.

### 6.1 Junction points between flat and curved regions

*Junction points* are points at the surface boundary $\partial S$ where at least one curved region and at least one flat region connect (yellow points in Figures 2, 8 and 9). These points do not model permanent plastic effect: therefore, contrary to singular vertices, they may move over the 2D pattern during the animation. For instance, if the constraints applied to a smoothly curved region progressively relax, junction points slide and may disappear leading to a large planar region. Our process for updating these points, described next, corresponds to the arrow (e) in Figure 16.

To take the possible expansion or shrinking of planar regions into account, we compute for each curved region $C$, the dihedral angle $\theta_{\text{CF}}$ at each border ruling between $C$ and an adjacent flat region $F$ (these border rulings are either defined by two junction points or by a singular point and a junction point). We compare this angle to two thresholds $\theta_{\text{low}}$ and $\theta_{\text{high}}$ as follows: if $\theta_{\text{CF}} < \theta_{\text{low}}$, we move the corresponding junction point(s) as to expand the planar region until we reach a ruling whose angle is larger than $\theta_{\text{low}}$ (see Figure 12-left). Inversely, if $\theta_{\text{CF}} > \theta_{\text{high}}$, we move the junction point(s) as to expand the curved region, enabling subsequent simulation steps to smooth out this angle (see Figure 12-right). If $F$ is also adjacent to another planar region $F_{\text{opp}}$ or to another curved region $C_{\text{opp}}$ over the same junction point, we allow $C$ to expand only if $\theta_{\text{CF}} > \theta_{\text{opp}}$ (see Figure 13).

We usually choose $\theta_{\text{high}} = \theta_0$ and $\theta_{\text{low}} = \theta_0/10$. Choosing $\theta_{\text{high}}$ sufficiently different from $\theta_{\text{low}}$ is important, so that hysteresis prevents junction points from oscillating too much. If the dihedral angle at all rulings of $C$ are $< \theta_{\text{low}}$, the curved region is completely replaced by a flat region, which captures the case of a curved region becoming flat.

## 6.2 Fitting generalized cones

We now seek to approximate each curved region $C \in \mathscr{C}$ without interior singular point using generalized conical surfaces as illustrated in Figure 14 with a paper ribbon. Note that in the case where $C$ contains an interior singular point $s$, $C$ is already represented by a single generalized cone whose rulings pass through the singular point $s$ (see Section 5.2).

The main idea is to segment the set of rulings $\mathscr{R}_c$ such that each subset can be approximated accurately enough by one generalized cone. Then we force every 2D ruling of the subset to pass through the apex of the computed cone to improve developability. This step corresponds to the arrow (f) in Figure 16.

**Finding the apex of a generalized cone.**
In order to find the apex of a generalized cone that approximates a list of rulings, we inspire from the surface reconstruction method from Peternell *et al.* [2004] and compute the best approximating generalized cones using a model based on the *Blaschke cylinder*[1], which is particularly well adapted to our setting. This model matches a generalized cone to a one-parameter family of tangent planes computed along the rulings of a curved region. We summarize this model and the way we use it below. See [Peternell 2004] and the references herein for a more detailed description in the context of Laguerre geometry.

Let $E$ be an oriented plane in Euclidean space $\mathbb{R}^3$. $E$ is uniquely defined by its unit normal vector $\mathbf{n} = (n_1, n_2, n_3)$ and the signed Euclidean distance $w$ between $E$ and the origin. Thus $E$ can be represented by the point $(n_1, n_2, n_3, w)$ of $\mathbb{R}^4$. Similarly, the whole set of oriented planes of $\mathbb{R}^3$ is represented by a set of points $u = (u_1, u_2, u_3, u_4)$ of $\mathbb{R}^4$ verifying:

$$B : u_1^2 + u_2^2 + u_3^2 = 1$$

This set is called the *Blaschke cylinder*.

All the tangent planes of a generalized cone pass though one point $\mathbf{a}$, called the apex of the cone. Thus, using the representation for planes we just described, they all are contained in the intersection between $B$ and the hyperplane:

$$H : \mathbf{n} \cdot \mathbf{a} + w = 0 \, .$$

We use this property to compute the apex $\mathbf{a}$ of the generalized cone that best approximates a surface, as follows.

Let us consider rulings $r^b$ associated with a smooth curved part of the mesh and their associated normals (constant along the ruling) $\mathbf{n}^b$. We call $\mathbf{n}_i^b$ (resp. $T_i^b$) the i-th consecutive normal (resp. tangent plane associated to this normal) from a set $\mathcal{R}$ of rulings. We thus find the best common apex $\mathbf{a}$ in a least square sense by minimizing the error $e$ given by:

$$e = \sum_{i \in \mathcal{R}} \|\mathbf{n}_i^b \cdot \mathbf{a} + w_i\|^2 \, . \qquad (2)$$

where $w_i$ is the signed Euclidean distance between the origin and the tangent plane $T_i^b$. Note that this error can be seen as the sum of squared distances between all the tangent planes and the apex $\mathbf{a}$.

[1] Wilhelm Blaschke (1885-1962) Mathematician, differential geometer.

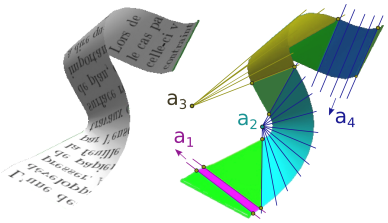


Fig. 14. Segmentation of the surface at the left into piecewise conical parts (right). The color parts correspond to pieces of developable generalized cones whose rulings are pointing to their common respective apex $\mathbf{a_i}$ satisfying Eq. (2). We add regions in green (portions of cones sometimes degenerated to planes - whose apex is the junction point between adjacent conical regions) to connect these regions while ensuring tangential continuity.

**Iterative segmentation algorithm.**
In practice, we use an iterative algorithm to find a segmentation of $C$ into generalized cones such that the error $e$ is smaller than a fixed threshold $\varepsilon_{\text{blaschke}}$ (in our implementation, $\varepsilon_{\text{blaschke}} = 0.5\% \times L_{\text{paper}}$, $L_{\text{paper}}$ being the maximal length size of a piece of paper).

We initialize $\mathcal{R}_{\text{cur}} = \mathscr{R}_C$ as the list of rulings defining a curved region in the current segmentation. We also initialize two empty lists $\mathcal{R}_{\text{front}}$ and $\mathcal{R}_{\text{back}}$. Using the Blaschke model presented above, we compute the apex $\mathbf{a}$ of a generalized cone that approximates the surface defined by $\mathcal{R}_{\text{cur}}$ and the corresponding error. Until the error is smaller than $\varepsilon_{\text{blaschke}}$, we iteratively remove either the first or the last ruling from $\mathcal{R}_{\text{cur}}$ and add it to either $\mathcal{R}_{\text{front}}$ or to $\mathcal{R}_{\text{back}}$. The selected ruling among these two is the one associated with the maximal error contribution $e_i = \|\mathbf{n}_i^b \cdot \mathbf{a} + w_i\|^2$, meaning that its tangent plane has the largest distance to $\mathbf{a}$. The result of this algorithm is a suitable subset of rulings that can be approximated by a generalized cone with apex $\mathbf{a}$ with an error smaller than $\varepsilon_{\text{blaschke}}$. We then restart the process on each of the two lists $\mathcal{R}_{\text{front}}$ and $\mathcal{R}_{\text{back}}$. In this way, we obtain a list of apices, each one corresponding to a subset of successive rulings.

The last step consists in moving the vertices of rulings in each subset in order to align their respective edges along the rulings of the generalized cone we just computed. We proceed as follows.

For each apex $\mathbf{a}$ we consider the ruling $r^j$, belonging to the corresponding subset, associated to the minimal error contribution $e_j = \|\mathbf{n}_j^b \cdot \mathbf{a} + w_j\|^2$ and compute the orthogonal projection of the apex $\mathbf{a}^{\text{proj}}$ onto this ruling. The image of $\mathbf{a}^{\text{proj}}$ in the 2D pattern space $\overline{\mathbf{a}}^{\text{proj}}$ is computed using its local coordinate within the ruling frame. Then, we resample the vertices on the boundary curve such that the extension of the 2D edge lines outside $\overline{S}$ are all passing through $\overline{\mathbf{a}}^{\text{proj}}$ as shown in Figure 15. The 3D coordinates of the newly sampled vertices are finally reported on the 3D boundary curve using cubic interpolation.

Two successive generalized cones $c_1$ and $c_2$ join on the boundary $\partial S$ at a singular point $s_b$ where their extreme rulings cross. We thus consider the gap between them (in green in Figure 14 and Figure 15) as a generalized junction cone $c_{\text{junction}}$ whose apex is $s_b$ and remesh it accordingly. Note that the first and the last rulings of $c_{\text{junction}}$ are respectively border rulings of $c_1$ and $c_2$. Each remeshed curved region is thus represented by a set of generalized

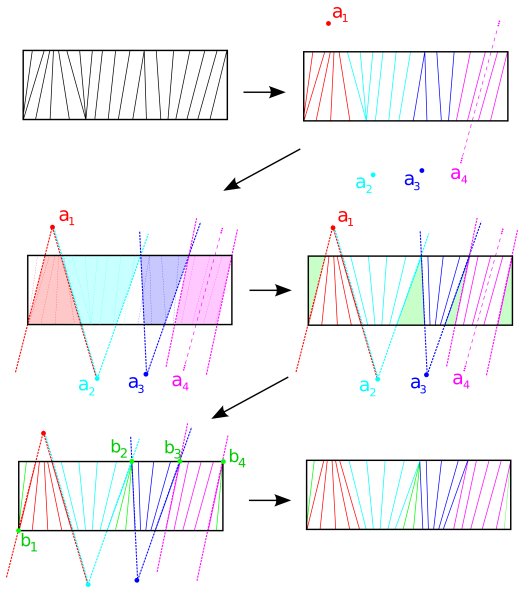


Fig. 15. First row : the current rulings are used to find a list of apices $a_1, a_2, a_3, a_4$ ($a_4$ being at the infinity) corresponding to the surface. Second row left: region of influence of each apex. Second row right: resampling of the boundaries in order to have in each region the rulings converging to the corresponding apex. Last row left: resampling of the junctions between the regions by considering them as cones whose apices are $b_1$, $b_2$, $b_3$, and $b_4$. Last row right: final mesh.

cones joined by common rulings and meshed accordingly, and is therefore exactly developable onto a plane.

### 6.3 Preserving length with respect to the pattern

To explicitly enforce the lengths of edges in $S$ to match those of the pattern $\bar{S}$, we use a new, efficient optimization process, which is decomposed into two phases. In the first phase, the optimal directional vectors of edges are computed using constrained optimization. In the second phase, the positions of vertices are updated using a least-square formulation. Unlike prior work on vertex-based shape optimization, this new formulation can explicitly enforce the preservation of edge lengths.

The optimization is applied on $S_P$ after updating it from $S_G$ (see arrow (h) in Figure 16). The optimized shape is then displayed.

**Phase I.**
For every edge $e_{ij} \in S_P$, of vertices $\mathbf{p}_i$ and $\mathbf{p}_j$, we define $\mathbf{u}_{ij}$ as a unit vector indicating the direction of $e_{ij}$ and denote $\bar{l}_{ij}$ the original length of $e_{ij}$ on $\bar{S}$. Given unit vectors $\mathbf{u}_{ij}$ for the directions of edges on $S_P$, the isometry can be enforced by the constraints, $C_{closed} \equiv 0$, using the edge lengths on $\bar{S}$ as

$$C_{closed} = \sum_{t_{ijk}} \|\bar{l}_{ij}\mathbf{u}_{ij} + \bar{l}_{jk}\mathbf{u}_{jk} + \bar{l}_{ki}\mathbf{u}_{ki}\|^2,$$

where $t_{ijk}$ is the triangle formed by edges $e_{ij}$, $e_{jk}$, and $e_{ki}$. We also enforce $\mathbf{u}_{ij}$ to be a unit vector, using $C_{unit} \equiv 0$, where:

$$C_{unit} = \sum_{e_{ij}} (\mathbf{u}_{ij}^T \mathbf{u}_{ij} - 1)^2.$$

To minimize the deviation from the current $S_P$ while satisfying the above two constraints, the optimal $\{\mathbf{u}_{ij}\}$ can be computed as:

$$\begin{aligned} &\min_{\{\mathbf{u}_{ij}\}} \frac{1}{2} \sum_{ij} \|\mathbf{u}_{ij} - \mathbf{u}_{ij}^o\|^2 \\ &s.t. \quad C_{unit} = 0,\ C_{closed} = 0 \end{aligned} \tag{3}$$

where $\mathbf{u}_{ij}^o$ is the current unit directional vector computed from $e_{ij}$. To solve this problem efficiently, we take inspiration from [Wang 2008a]. Lagrange multipliers (LMs) are exploited :

$$\begin{aligned} &\frac{1}{2} \sum_{ij} \|\mathbf{u}_{ij} - \mathbf{u}_{ij}^o\|^2 + \lambda_1 \sum_{ij} C_{unit}(\mathbf{u}_{ij}) + \\ &\lambda_2 \sum_{i,j,k} C_{closed}(\mathbf{u}_{ij}, \mathbf{u}_{jk}, \mathbf{u}_{ki}) \,. \end{aligned} \tag{4}$$

For simplicity, we represent this as $E + \lambda_1 C_1 + \lambda_2 C_2$. The use of two LMs could lead to simpler expression and better convergence performance than associating one LM with every single constraint. By neglecting the terms from second order derivatives of constraints in the Hessian matrix according to the sequential linearly constrained programming [Nocedal and Wright 2000], we obtain :

$$\begin{bmatrix} \mathbf{H} & \mathbf{L} \\ \mathbf{L}^T & 0 \end{bmatrix} \begin{bmatrix} \delta_{\mathbf{x}} \\ \delta_{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_{\mathbf{x}} \\ \mathbf{b}_{\lambda} \end{bmatrix}, \tag{5}$$

where $\mathbf{H} = \nabla^2 E$ is the Hessian matrix of $E$ and $\mathbf{L} = \{\nabla C_1, \nabla C_2\}$. The vectors on the right-hand side are

$$\mathbf{b}_{\mathbf{x}} = -\nabla E - \lambda_1 \nabla C_1 - \lambda_2 \nabla C_2 \text{ and } \mathbf{b}_{\lambda} = \{-C_i\}.$$

In our case, $\mathbf{H}$ is an identity matrix since we neglect the terms from second order derivatives of constraints. We get the following two formulas, enabling us to update the values iteratively:

$$\begin{aligned} \mathbf{L}^T\mathbf{L}\delta_{\lambda} &= \mathbf{L}^T\mathbf{b}_{\mathbf{x}} - \mathbf{b}_{\lambda} \\ \delta_{\mathbf{x}} &= \mathbf{b}_{\mathbf{x}} - \mathbf{L}\delta_{\lambda}. \end{aligned} \tag{6}$$

Therefore, only a 2 by 2 equation system needs to be solved in each iteration step to update the value of LMs, which greatly speeds up the efficiency of computation. Notice that our formulation does not guarantee theoretical convergence. As the system comes closer to satisfying the constraints, L approaches a zero matrix and the Lagrange multipliers tend to infinity leading to numerical difficulties near convergence. In practice we observed a good behavior of this approach when limiting the maximum number of iteration steps (200 in our implementation). Moreover, since we perform this isometry preservation step following each simulation step, we reduce the accumulation of simulation error as much as possible.

**Phase II.**
After finding the optimal unit vectors $\{\mathbf{u}_{ij}\}$ for edges, we compute the positions of vertices by minimizing the difference between the vector $(\mathbf{p}_j - \mathbf{p}_i)$ and the vector $\bar{l}_{ij}\mathbf{u}_{ij}$ on every edge $e_{ij}$.

$$\min_{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n} \frac{1}{2} \sum_{\{e_{ij}\}} \|(\mathbf{p}_j - \mathbf{p}_i) - \bar{l}_{ij}\mathbf{u}_{ij}\|^2 \tag{7}$$

This is a least-square minimization problem, which can be solved efficiently through a linear system. Meanwhile, the positions of handle vertices are fixed by moving the related terms to the right-hand side of the equations. Note that we can decouple the computation of the $x$, $y$ and $z$ components to reduce the dimension of the linear system.

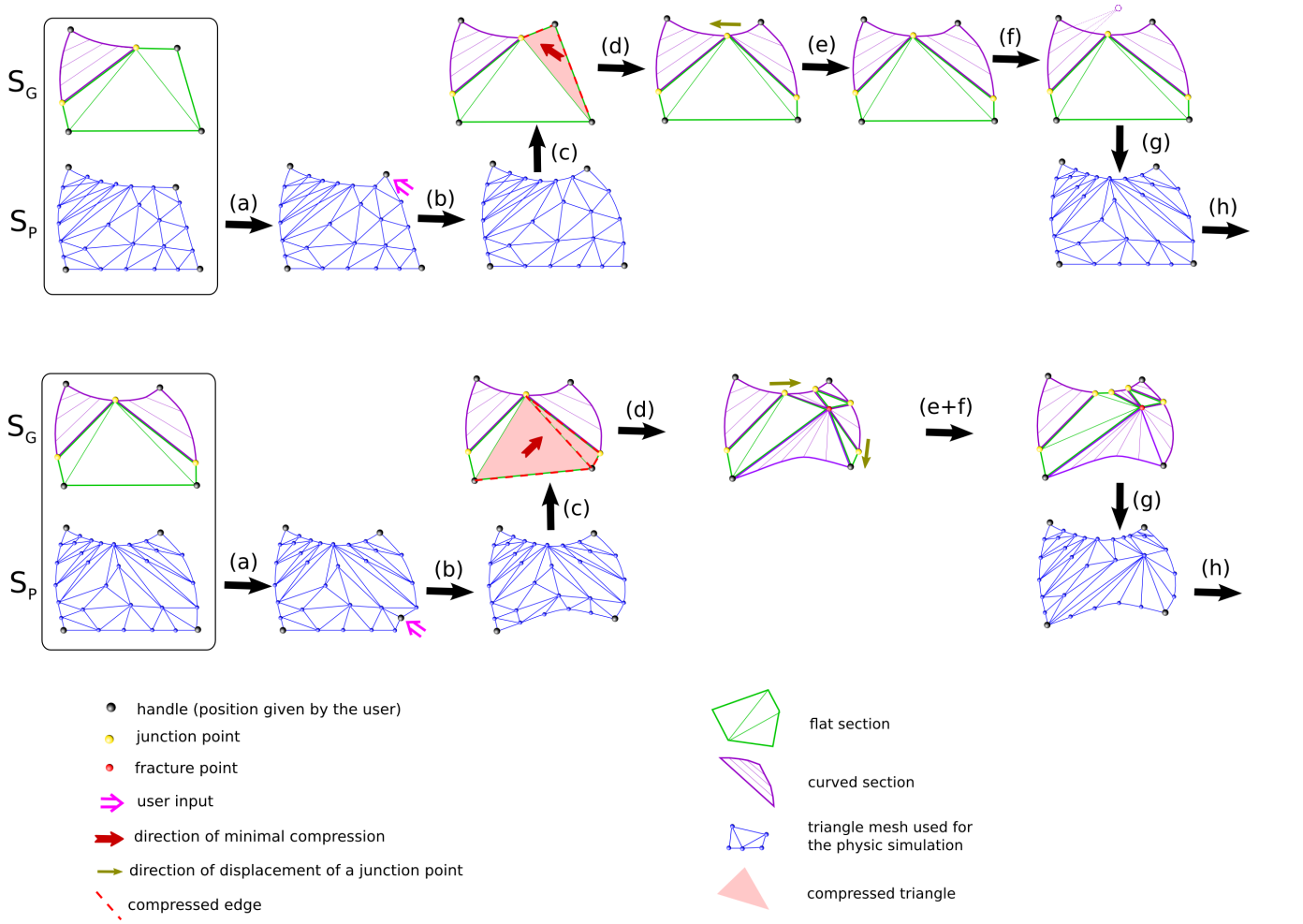


Fig. 16. A step-by-step example of two successive deformation steps:
(a): displacement of the handles according to the user input.
(b: physical relaxation of the triangle mesh (Section 4).
(c): updating of the position 3D of the point of the structure.
(d): creation of curved region from compressed flat regions (Section 5.1) and generation of fracture points if needed (Section 5.2).
(e): displacement of the junction points (Section 6.1).
(f): update the rulings of each curved region (Section 6.2).
(g): generation of a triangle mesh from the structure $S_G$ (Section 4).
(h): optimizing the isometry (Section 6.3) and displaying the triangle mesh on screen.

## 7. RESULTS & DISCUSSION

### 7.1 Validation

We used experiments involving real pieces of paper to validate our virtual model. Note that contrary to previous approaches, we specifically conduct comparisons in the critical situation when the first few singular vertices appear. Indeed, when and where these first sharp features appear is very noticeable and capturing these accurately is a key element towards realism.

For each validation test, we started from an initial flat mesh $S$ (which helped reproducing the same situation in reality), and set $L_{\text{paper}}$=10 centimeters and $\rho_{\text{bound}} = 1.2$ sample/cm and $\rho_{\text{int}} = 6$ sample/cm$^2$.

The first experiment, shown in Figure 17, was used to validate the way a new singular vertex appears: in reality, we moved the two opposite sides of an undamaged square sheet of paper closer to each other by a factor $10\%$. Then we applied an orthogonal constraint on one of the borders until a singular point appeared (see Figure 17-left). We then marked the paper with the position of this singular point. We repeated the experiment 20 times with other sheets of the same thickness. The same experiment was conducted the same number of times, using the random factors in the placement of singular vertices described in Section 5.2. The resulting distributions of the position of the first singular point are depicted in Figure 17 (center and right), for two different types of paper (a 70 $g/m^2$ paper and a 90 $g/m^2$ paper). Our results show a good accordance between real measurements and our model.

Visual comparison between real paper and our results are also provided in Figure 18. The top row illustrates the deformation of a smooth paper strip continuously twisted into a Moebius strip. The

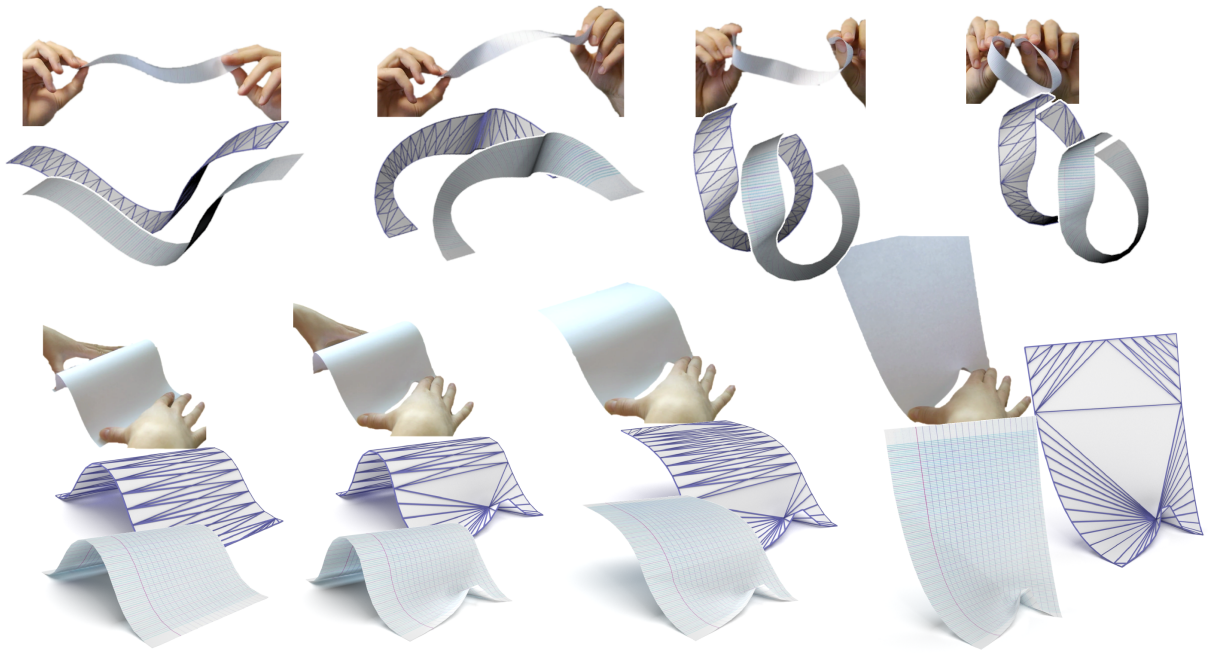

Fig. 18. Comparing a real paper deformation with our results for a smooth surface (top) and the creation of a singularity (bottom).

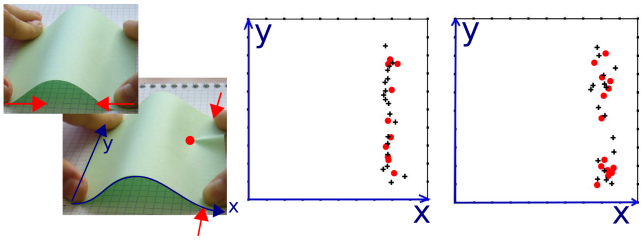


Fig. 17. Comparison of the positions of the first appearing singular vertex between real paper and our virtual model. Left: deformations applied to an initially flat sheet of real paper, repeated 20 times. Right: the measurements on real paper are shown as black crosses, while the positions generated by our method are represented as red dots. The graph at the center refers to a 70 $g/m^2$ paper, modeled using $\sigma = 0.01$ cm. The graph on the right refers to a 90 $g/m^2$ paper, modeled using $\sigma = 0.1$ cm.

second row compares the behavior of sheet of paper after a singular vertex is inserted: we get the same effect of the sheet straightening when the constrained points at the back are unloaded. Note that our mesh remains coarse throughout the animation, with its edges aligned with surface rulings. Moreover, the singular vertex is persistent in the model even if the surface is totally unloaded.

## 7.2 Comparison with other methods

**Comparison with FEM and adaptive meshing methods**
Figure 19 compares our method with the FEM model described at the beginning of Section 4 applied to a fine mesh of fixed connectivity (*standard FEM* experiment), and with simulation with adaptive remeshing from Narain *et al.* [2013].

Note that all three methods use the same physical FEM solver, and are able to obtain the comparable visual results (see Figure 19, upper row, in this case for the example used in the experiment of Sec. 7.1). The standard FEM approach with fixed connectivity requires very dense meshes to obtain visually sharp features. The triangulation obtained from the adaptive remeshing method in Narain *et al.* is roughly two times coarser than this dense mesh for similar visual result, but still requires approximately ten times more triangles than our method. As most computation time in all three methods is spent in the simulation step, which depends on the number of triangles, our approach reaches interactive frame rates when several seconds to a minute per frame are required using other approaches. See Table II. The lower row of Figure 19 show a comparison where we decreased the number of triangles of the standard FEM approach and of the approach from Narain *et al.* such that the corresponding methods reach roughly the same time rate than ours. We can note that the quality of the visual appearance drastically decreased and does not capture anymore the interesting sharp feature effect.

Table II. Average computation times for the example presented in Fig. 19.

| | Our method | standard FEM | Narain *et al.* | coarse FEM | coarse Narain |
|---|---|---|---|---|---|
| Remeshing time | 1 | - | 450 | - | 30 |
| Simulation time | 100 | 14 700 | 4 640 | 340 | 480 |
| Nb. of triangles | 89 | 1 668 | 788 | 212 | 221 |

*All timings are expressed in ms per frame.*

**Comparison with *"Fast Simulation of Mass-Spring Systems"* and *"Shape-Up"*.**
The technique we present in Section 6.3 treats the preservation of edge lengths as a geometric problem while Liu *et al.* [2013] simulate a mass-spring system in a physical perspective with their approach called *Fast Simulation of Mass-Spring Systems* (FSMS).

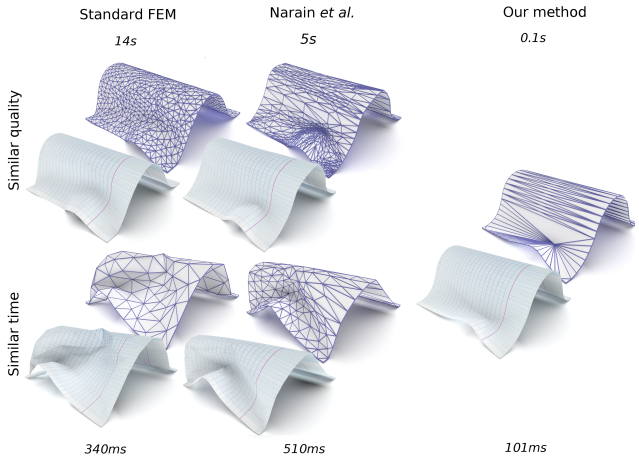


Fig. 19. Comparison with other methods. The upper row shows how much mesh refinement is required for Standard FEM (top left) and Narain *et al.* [2013] method (top middle), in order to get a result of similar visual quality as our method (right).
The lower row compares the mesh refinements for which the three methods converge with the same amount of time: Standard FEM (bottom left) and Narain *et al.* method (bottom middle), our method (right),

In the optimization method of FSMS, the edge length is set as a hard constraint. In our formulation, the optimization variables are the edge directions instead of vertex positions. Our method does not exploit any physical information so the objective function is simpler. As a result, it can be optimized by iteratively solving 2x2 linear systems.

The *Shape-Up* method [Bouaziz et al. 2012] provides a good solution for achieving isometry preservation. The original Shape-Up framework treats all constraints as soft constraints and requires minimizing an energy function that incorporates the projections of all constraints to the targets, which is solved in a least-square sense. Therefore, the resulting model will always be a balance between the input status and the constrained projections, which indicate that a shape distortion will occur. As we expect to preserve the isometry of edge lengths without significant shape distortion, we modify the Shape-Up framework to enforce all constraints as hard constraints. In the following discussion, we focus on the comparison between the Shape-Up with hard constraints and our scheme:

*About speed of computation:* although the Shape-Up method uses a single matrix pre-factorization step, our method is less expensive as it avoids solving large systems of equations. Comparison of computation time between the Shape-Up method and our isometry preservation method (Section 6.3) is given in Table III, and clearly shows that computational efficiency is higher with our method.

*About shape distortion and preservation of isometry:* both methods treat all requirements as hard constraints (with theoretically no conflict to each other) and find the nearest isometric status based on the current input. Experiments have shown, that our method leads generally to less deviation from the initial shape compared with the Shape-Up method, whereas the Shape-Up method achieves better isometry restoration. Quantitative comparisons are given in Table III. For our application it is important to keep the deviation as small as possible, otherwise it may introduce instabilities in the physical simulation. We further observed that the Shape-Up method may introduce perceptible visual artifacts such as discontinuities in a smooth region or flattening of weakly curved regions as depicted in Figure 20. This is particularly non-desirable for our application, where singularities, flat and curved regions are processed explicitly.

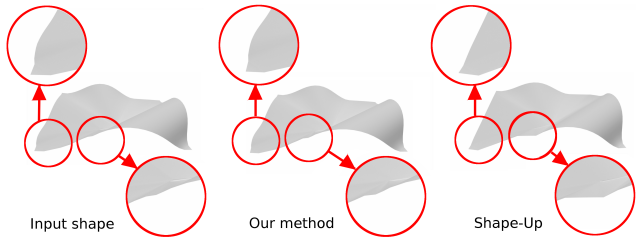


Fig. 20. Comparison of an input shape (left) with the results of our isometry optimization method (middle) and of Shape-Up isometry optimization (right). Contrary to our method, Shape-Up introduces unwanted discontinuities (right-most circle) or local shape flattening (left-most circle).

Table III. Comparison of our length preservation method (Sec. 6.3) and the Shape-Up method.

| | | Our | | | Shape-Up | | |
|---|---|---|---|---|---|---|---|
| Models | $c_{\text{init}}$ | $t_{\text{opt}}$ | $c_{\text{m}}$ | $d_{\text{m}}$ | $t_{\text{opt}}$ | $c_{\text{m}}$ | $d_{\text{m}}$ |
| Fig.20 | 0.75 | 196 | 0.38 | 2e-3 | 721 | 0.17 | 1e-2 |
| Fig.1 | 0.15 | 143 | 0.13 | 7e-4 | 760 | 0.03 | 3e-3 |
| Fig. 18 (top) | 0.11 | 112 | 0.06 | 9e-4 | 530 | 0.02 | 1e-3 |
| Fig. 18 (bottom) | 0.15 | 97 | 0.10 | 7e-4 | 480 | 0.03 | 1e-3 |
| Fig. 22 ($2^{nd}$ row) | 0.55 | 113 | 0.42 | 3e-3 | 580 | 0.15 | 8e-3 |
| Fig. 22 ($4^{th}$ row) | 0.22 | 122 | 0.20 | 2e-3 | 590 | 0.02 | 4e-3 |

*We apply our optimization (Section 6.3) and the Shape-Up optimization to a final mesh of some of ours examples.*
*$c_{init}$ is the mean percentage of compression computed over all the edges of the model before any optimization is applied. $t_{opt}$ is the time (in ms) taken by the optimization. $c_m$ is the mean percentage of compression over all the model edges. $d_m$ is the average displacement of the vertices from their initial position.*

## 7.3 Other results

Figure 1 shows a sheet of paper bent twice, in two perpendicular directions: after smoothly wrapping the sheet into a cylindrical shape, a second bending is applied orthogonally to the cylinder axis. The resulting shape exhibits sharp singularities while still remaining almost isometric to its pattern. In changing the parameter $\varepsilon_c$ defining the maximal admissible compression, we model different types of papers as shown in Figure 21, while keeping the same handle animation scenario. Larger values of $\varepsilon_c$ capture the behavior of stiff, thick paper, while smaller values capture the behavior or thinner paper, for which a larger number of singular points appear.

Figure 22 gathers other results of our method. The first row illustrates smooth deformation highlighting the dynamic adaptation of the triangulation. The second row shows a full crumpling animation where more than twenty singular vertices are progressively generated. The third row illustrates a newspaper type of surfaces

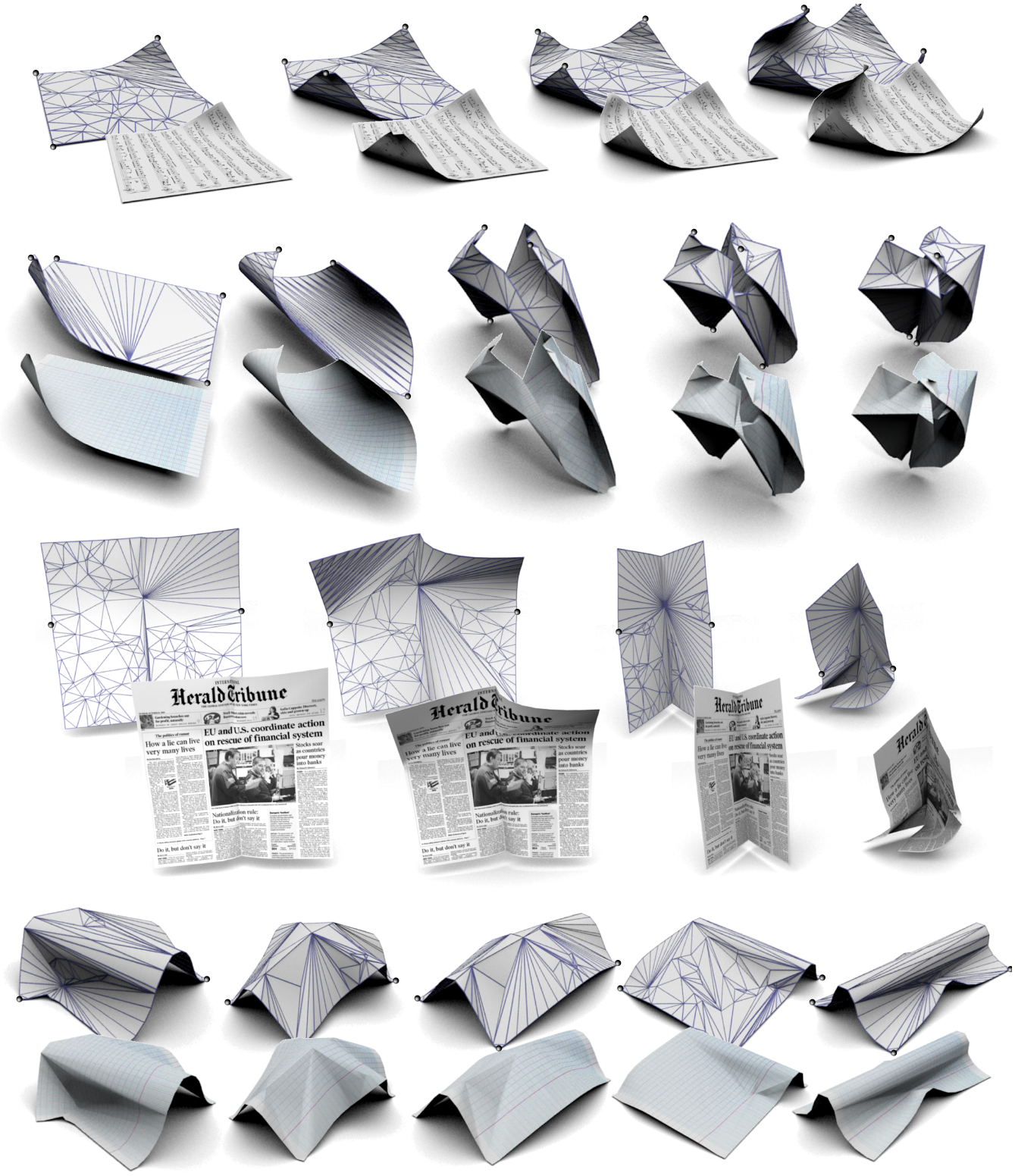


Fig. 22. Paper crumpling examples. The black points indicate the positions of the handles.

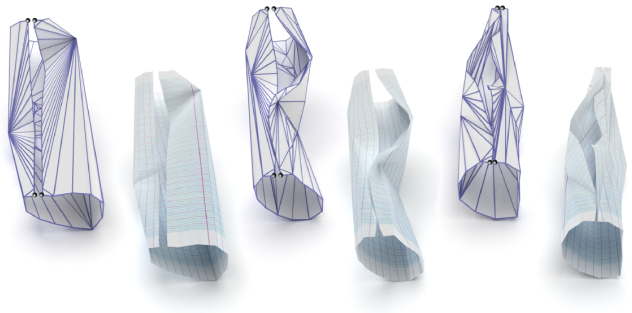

Fig. 21. The same experiment as in Figure 1 is performed with different values of the allowed compression factor $\varepsilon_c$: from left to right, $1\%$, $0.5\%$, and $0.2\%$.

Table IV. Measures of error in length for our examples.

| Models | $c_{\text{mean}}$ | $n_{\text{comp}}$ |
|---|---|---|
| Fig. 1 | 0.89 | 8.0 |
| Fig. 18 (top) | 0.17 | 0.1 |
| Fig. 18 (bottom) | 0.19 | 2.0 |
| Fig. 22 ($2^{nd}$ row) | 1.47 | 11 |
| Fig. 22 ($4^{th}$ row) | 0.20 | 2.0 |

*$c_{mean}$ is the mean percentage of compression over all the model edges and $n_{comp}$ is the percentage of edges whose compression is $> 1\%$.*

Table V. Computational time analysis for our examples. We separate the time spent in the simulation part and the time spent in the geometrical part.

| | Mesh size | | Simulation | | | Geometry | |
|---|---|---|---|---|---|---|---|
| Models | $N_t$ | $N_s$ | $N_{\text{it}}$ | $t_{\text{int}}$ | $t_{\text{col}}$ | $t_{\text{rem}}$ | $t_{\text{iso}}$ |
| Fig. 1 | 118 | 2 | 5 | 117 | 25 | 0.4 | 1.0 |
| Fig. 18 (top) | 117 | 0 | 11 | 232 | 62 | 0.2 | 3.1 |
| Fig. 18 (bottom) | 87 | 1 | 4 | 50 | 10 | 0.4 | 1.0 |
| Fig. 22 ($2^{nd}$ row) | 156 | 15 | 5 | 190 | 57 | 0.5 | 2.3 |
| Fig. 22 ($4^{th}$ row) | 120 | 21 | 8 | 190 | 48 | 0.8 | 0.9 |

*$N_t$ and $N_s$ are respectively the average number of triangles of $S_P$ through the animation and the maximum number of interior singular vertices. $N_{it}$ is the average number of iterations needed in the simulation steps. $t_{int}$ and $t_{col}$ are the respective average time spent in the integration steps of the physical simulation and to treat the collisions. The timings from the geometrical parts correspond to the time spent in remeshing $t_{rem}$, and in the isometry tracking $t_{iso}$. All times are given in ms.*

with a pre-set crease in the middle. The later highly influences the subsequent deformation of the surface. Finally, the last row shows various paper deformations obtained by interactively playing with the handles.

To get quantitative results, we measured the error of lengths compared to the pattern $E_{\text{length}} = \sum_{\text{edges } e}(\bar{l}_e - l_e)/\bar{l}_e$ , where $l_e$ and $\bar{l}_e$ are the length of the edge $e$ belonging respectively to $S_P$ and $\overline{S}$. Table IV provides the average value of $E_{\text{length}}$ over all the animation frames for the different examples, as well as the average number of compressed edges. The computational times for the different examples are given in Table V.

## 7.4 Discussion

While getting a variety of results that qualitatively look as real paper – even when animated (see the associated video) – indicates that our method has reached most of its goals, we also identified a number of limitations:

Firstly, although it is much faster than state-of-the-art methods, our model does not meet the real-time performances: our prototype currently runs at 1 to 10 fps. As shown in the timing table, most of the time is spent in the physically-based simulation step. Note that we re-used some existing code, including a collision detection module, without trying to optimize it specifically for our application. We believe that increasing efficiency should be possible, given the small size of our meshes. Moreover the detection and handling of collisions are only applied on $S_P$. We do not handle (internal or external) collisions that may possibly occur while modifying $S_G$.

Secondly, another limitation is the fact that we do not handle the case when a pressure is applied inside a curved area (such as the one caused by a finger pushing within a smoothly bended paper part) because it would requires to introduce further constraints to the rulings propagation algorithm. This prevents us from providing handles anywhere on the paper surface. We can also use boundary volumes to control the crumpling of the paper thanks to the collision detection used in the physical simulation. But due to this limitation, non-convex volumes would not act properly. Handling this case would require a couple of modifications of the approach such that the detection of compression inside a curved region and the generation of singular points caused by such a compression based on the same principles than in our algorithm.

Thirdly, as our meshes rely on very coarse triangulations, the dynamic adaptation of connectivity may introduce discontinuous behavior during the animation. This can be observed in some of our animations when the mesh suddenly changes. While such geometrical discontinuities may arise for real paper as well when singularities appear these effects are exaggerated by the coarse triangulation we use. Animation using temporal smoothing could be explored.

Fourthly, our model only generates singular points as sharp features, although previous studies indicate that crumpled paper geometry can exhibit creased curves as well. Kergosien *et al.* [1994] computes such curves using physical simulation on a geometric model even simpler than ours. Exploring the detection and generation of similar creased curves within our model is left for future research. Here, the creases modeled in [Kilian et al. 2008] could be a source of inspiration.

Lastly, although each individual generalized cone is developable, as well as flat parts, the overall surface may not be globally developable. It is however almost so, due to the approximative isometry with the flat pattern. Developability could be further improved by minimizing the angular defect around each singular vertex.

## 8. CONCLUSION & FUTURE WORK

In this work, we introduced a new hybrid geometric and physical model for paper, able to qualitatively capture the dynamic shapes of this complex material. The model is efficient enough to be

Table VI. Summary of user-defined parameters

| Symbol | unit | default value | |
|---|---|---|---|
| $\rho_{\text{bound}}$ | $\frac{\text{vertices}}{\text{cm}}$ | 1.2 | density of the sample on the boundary |
| $\rho_{\text{int}}$ | $\frac{\text{vertices}}{\text{cm}^2}$ | 6 | density of sample inside the surface |
| $\theta_0$ | rad | 0.01 | threshold angle for the limit planar region |
| $\theta_{\text{low}}$, $\theta_{\text{high}}$ | rad | $\theta_0/10$, $\theta_0$ | hysteresis threshold for the displacement of junction point |
| $\varepsilon_{\text{Blaschke}}$ | cm | $0.5\%$ of $L_{paper}$ | maximal acceptable error for an estimation with Blaschke cylinder |
| $L_{\text{paper}}$ | cm | 10 | maximal length on the surface of the paper |
| $\varepsilon_c$ | % | 0.1 | limit compression |
| $\sigma$ | cm | 0.1 | variance of the probability law of the position of the new singular point |

manipulated at interactive rates.

Quite interestingly, our model is based on some high-level understanding of the physical constraints that act on real sheets of paper, and on their geometric counterparts. This understanding enabled us to use an adaptive mesh carefully representing the main geometric features of paper in terms of singular points and rulings, throughout the animation. In addition to accelerating computation, this coarse mesh yielded fast, good quality rendering.

In addition to solving the limitations listed in the discussion above, we would also like to extend our work to more extreme situations, where paper is torn as well as crumpled: indeed, investigating the ability of our hybrid model to detect and progressively propagate tears is a promising direction for future research. Furthermore, our unique combination of interwoven geometric and physical processing could inspire the modeling of other complex materials, for which the same general methodology could be used.

**Acknowledgements:** This research was partially funded by the ERC advanced grant no. 291184 EXPRESSIVE.
Thanks to Yunbo Zhang for helping us to prepare the comparison with the Shape-Up method.

## REFERENCES

AMAR, M. B. AND POMEAU, Y. 1997. Crumpled paper. *Proceedings of the Royal Society. Mathematical, Physical and Engineering Sciences*.

BO, P. AND WANG, W. 2007. Geodesic-controlled developable surfaces for modeling paper bending. *Computer Graphics Forum (CGF) 26,* 3.

BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum (CGF) 31,* 5 (Aug.), 1657–1667.

BRIDSON, R., FEDKIW, R., AND ANDERSON, J. 2002. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graph. 21,* 3 (July).

BRIDSON, R., MARINO, R., AND FEDKIW, R. 2003. Simulation of clocloth with folds and wrinkles. *Symposium on Computer Animation (SCA)*.

BRONKHORST, C. 2003. Modelling paper as a two-dimensional elastic-plastic stochastic network. *International Journal of Solids and Structures 40*.

BURGOON, R., GRINSPUN, E., AND WOOD, Z. 2006. Discrete shells origami. *Proceedings of Computers And Their Applications*, 180–187.

CAMBOU, A. D. AND MENON, N. 2011. Three-dimensional structure of a sheet crumpled into a ball. *Proceedings of the National Academy of Sciences*.

CERDA, E. AND MAHADEVAN, L. 1998. Conical surfaces and crescent singularities in crumpled sheets. *Physical Review Letters 80,* 11.

CHU, C.-H. AND SÉQUIN, C. H. 2002. Developable Bézier patches: properties and design. *Computer-Aided Design 34,* 7, 511–527.

COFFIN, D. 2009. *Fundamental Research Symposium on Advances in Pulp and Paper Research.* Oxford.

DECAUDIN, P., JULIUS, D., WITHER, J., BOISSIEUX, L., SHEFFER, A., AND CANI, M.-P. 2006. Virtual garments: A fully geometric approach for cloth design. *Computer Graphics Forum (CGF) (Proc. of Eurographics) 25,* 3.

ENGLISH, E. AND BRIDSON, R. 2008. Animating developable surface using nonconforming elements. *ACM Trans. Graph. (Proc. of SIGGRAPH) 27,* 3.

FREY, W. H. 2004. Modeling buckled developable surfaces by triangulation. *Computer Aided Design 36*.

GINGOLD, Y., SECORD, A., HAN, J. Y., GRINSPUN, E., AND ZORIN, D. 2004. A discrete model for inelastic deformation of thin shells. *Technical Report*.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M., , AND SHRÖDER., P. 2003. Discrete shells. *Symposium on Computer Animation (SCA) 27*, 62–67.

HARMON, D., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. 2008. Robust treatment of simultaneous collisions. *ACM Trans. Graph. (Proc. of SIGGRAPH) 27,* 3, 1–4.

JULIUS, D., KRAEVOY, V., AND SHEFFER, A. 2005. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum (CGF) (Proc. of Eurographics)*.

KANG, Y.-M., ZHANG, H.-G., AND CHO, H.-G. 2009. Plausible virtual paper for real-time application. *Computer Animation and Social Agents (CASA), Short Paper*.

KERGOSIEN, Y., GOTODA, H., AND KUNII, T. 1994. Bending and creasing virtual paper. *IEEE Computer Graphics and Applications 14,* 1.

KILIAN, M., FLOERY, S., CHEN, Z., MITRA, N., SHEFFER, A., AND POTTMANN, H. 2008. Curved folding. *ACM Trans. Graph. (Proc. of SIGGRAPH) 27,* 3.

LEOPOLDSEDER, S. AND POTTMANN, H. 1998. Approximation of developable surfaces with cone spline surfaces. *Computer Aided Design 30*.

LIU, T., BARGTEIL, A., O'BRIEN, J., AND KAVAN, L. 2013. Fast simulation of mass-spring systems. *ACM Trans. Graph. (Proc. of SIGGRAPH Asia) 32,* 6.

LIU, Y., POTTMANN, H., WALLNER, J., YANG, Y.-L., AND WANG, W. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph. (Proc. of SIGGRAPH Asia)*.

MAHADEVAN, L. AND CERDA, E. 1998. Conical surfaces and crescent singularities in crumpled sheets. *Physical Review Letters 80,* 11.

MAKELA, P. AND OSTLUND, S. 2003. Cohesive crack modelling of thin sheet material exhibiting anisotropy, plasticity and large-scale damage evolution. *Intern. J. Solids Structures 40,* 21.

MITANI, J. AND IGARASHI, T. 2011. Interactive design of planar curved folding by reflection. *Pacific Graphics (short paper)*.

NARAIN, R., PFAFF, T., AND O'BRIEN, J. F. 2013. Folding and crumpling adaptive sheets. *ACM Trans. Graph. (Proc. of SIGGRAPH) 32,* 4.

NARAIN, R., SAMII, A., AND O'BRIEN, J. F. 2012. Adaptive anisotropic remeshing for cloth simulation. *ACM Trans. Graph. (Proc. of SIGGRAPH Asia) 31,* 6.

PETERNELL, M. 2004. Developable surface fitting to point clouds. *Comput. Aided Geom. Des. 21,* 8 (Oct.), 785–803.

POTTMANN, H., HUANG, Q., DENG, B., SCHIFTNER, A., KILIAN, M., GUIBAS, L., AND WALLNER, J. 2010. Geodesic patterns. *ACM Trans. Graph. 29,* 4, 43:1–43:10.

POTTMANN, H. AND WALLNER, J. 2001. *Computational Line Geometry*. Springer.

PROVOT, X. 1996. Deformation constraints in a mass-spring model to describe rigid cloth behavior. *Graphics Interface*, 147–154.

ROHMER, D., CANI, M.-P., HAHMANN, S., AND BORIS THIBERT. 2011. Folded paper geometry from 2D pattern and 3D contour. *Eurographics Short Paper*, 21–24.

ROHMER, D., POPA, T., CANI, M.-P., HAHMANN, S., AND SHEFFER, A. 2010. Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles. *ACM Trans. Graph. (Proc. of SIGGRAPH Asia) 29,* 5.

ROSE, K., SHEFFER, A., WITHER, J., CANI, M.-P., AND THIBERT, B. 2007. Developable surfaces from arbitrary sketched boundaries. *Symposium on Geometry Processing (SGP)*.

SCHROLL, R. D., KATIFORI, E., AND DAVIDOVITCH, B. 2011. Elastic building blocks for confined sheets. *Physical Review Letters 106*.

SIMNETT, T. J., LAYCOCK, S. D., AND DAY, A. M. 2009. An edge-based approach to adaptively refining a mesh for cloth deformation. In *TPCG*. 77–84.

SOLOMON, J., VOUGA, E., WARDETZKY, M., AND GRINSPUN, E. 2012. Flexible developable surfaces. *Computer Graphics Forum (CGF), Proc. of Symposium on Geometry Processing (SGP) 31,* 5.

STEENBERG, B. 1947. Svensk papperstidn. *Pappers 50,* 6, 127–140.

TACHI, T. 2010. Origamizing polyhedral surfaces. *IEEE Transactions on Visualization and Computer Graphics 16,* 2, 298–311.

TANG, M., MANOCHA, D., AND TONG, R. 2010. Fast continuous collision detection using deforming non-penetration filters. *Symposium on Interactive 3D Graphics and Games*.

THORPE, J.-L. 1981. Paper as an orthotropic thin plate. *TAPPI 64,* 3, 119–121.

WANG, C. 2008a. Computing length-preserved free boundary for quasi-developable mesh segmentation. *IEEE Transactions on Visualization and Computer Graphics 14,* 1, 25–36.

WANG, C. 2008b. Towards flattenable mesh surfaces. *Comput. Aided Des. 40,* 1, 109–122.

WANG, C. AND TANG, K. 2004. Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Visual Computer 20,* 8, 521–539.

WANG, H., O'BRIEN, J. F., AND RAMAMOORTHI, R. 2010. Multi-resolution isotropic strain limiting. *ACM Trans. Graph. (Proc. of SIGGRAPH Asia) 29,* 6, 160.

WITTEN, T. 2007. Stress focusing in elastic sheets. *Review of Modern Physics 79*.

ZHU, L., IGARASHI, T., AND MITANI, J. 2013. Soft folding. *Computer Graphics Forum (Proc. of Pacific Graphics) 32,* 7, 167–176.



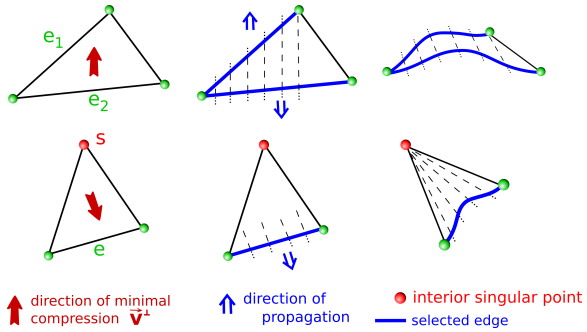


Fig. 23. Insertion of new edges (rulings) in direction of minimal compression makes a triangle likely to be bent as a conical surface during the next simulation step. Top: triangle without singular point. Bottom: triangle with singular point.

## APPENDIX

As a complement to Section 5.1, we describe in this appendix the technical details related to our algorithm for propagating imaginary rulings. The goal is to compute a triangle strip delimiting the influence of a compressed triangle within a flat region. This is done using a fast procedural method based on geometrical considerations (finding a physically accurate method for this step is left for future work). Our method, as already explained in Section 5.1, is divided into two steps:

First, we use an iterative algorithm in 2D pattern space to compute the prolongation of the imaginary rulings generated by the compression of $T_{\text{comp}}$. This allows to determine a strip of triangles of $\mathscr{T}_F$ affected by the bending of the $T_{\text{comp}}$ and the two borders which are the limit of the rulings of the new curved region. To get each border, we compute on each side whether the rulings can reach the boundary $\partial S$ on the paper, in which case the border is a boundary edge, or if they reach a already constraint edge, in which case the border is the interior singular point created to solve the conflict.

Second, we divide the region found into the parts which stay planar and the part corresponding to the new curved region and then remesh the region accordingly.

## A. FINDING THE TRIANGLE STRIP AND BORDERS BY PROPAGATION

This algorithm takes the compressed triangle $T_{\text{comp}}$ and the direction of minimal compression $\mathbf{v}^{\perp}$ as input. It computes and returns the triangle strip composed by all the triangles affected by the compression of $T_{\text{comp}}$ and the two borders.

To initialize the algorithm, we choose the two edges of $T_{\text{comp}}$ which are the most affected by the compression by selecting the edges which form the largest angle with $\mathbf{v}^{\perp}$, as shown in Figure 23-top. We then iteratively apply to each of the selected edges $e$ one of the following propagation steps with the direction of propagation being respectively $\mathbf{d}_{\text{prop}} = \mathbf{v}^{\perp}$ and then $-\mathbf{v}^{\perp}$ until finding the corresponding borders of the region.

A particular case arises when the edge $e$ of $T_{\text{comp}}$ with the

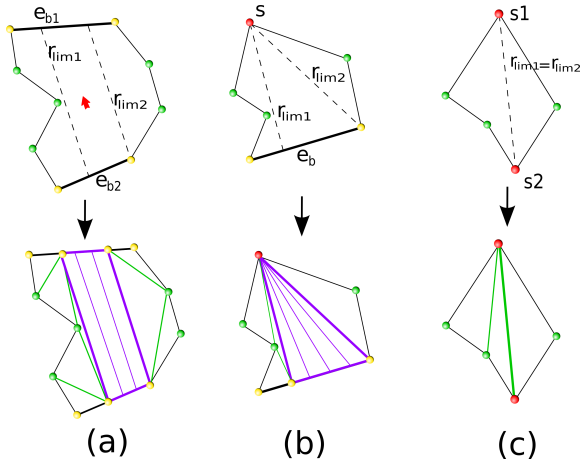


Fig. 24. The three cases of cutting the planar parts

largest angle to $\mathbf{v}^{\perp}$ is opposite to a singular point $s$ (see Figure 23-bottom). In this case, the singular point (i.e. apex of a d-cone) plays the role of a special border imposing that all rulings pass through it.

The propagation algorithm iterates for a given edge $e$ and a direction of propagation $\mathbf{d}_{\text{prop}}$ until a border is found as follows:

- if $e$ is common to another triangle $T \in \mathscr{T}_F$ (the dihedral angle of $e$ being smaller than $\theta_0$), then we collect $T$ and we selected the edge of $T$ (other than $e$) which forms the greatest angle with $\mathbf{d}_{\text{prop}}$, see Figure 6(a), and continue the propagation with the next iteration.
- if $e$ belongs to the boundary of the paper, it can bend freely. So we return $e$ as border of the triangle strip, see Figure 6(b).
- if $e$ is common with a triangle $T$ belonging to another flat region (the dihedral angle of $e$ being greater than $\theta_0$), the angle between the two triangles prevents the edge from bending freely. We need to add a singular point along the edge $e$ which will split it in two. The newly created singular point $s$ is returned as border of the triangle strip, see Figure 6(c).
- in the same way, if $e$ is a ruling of a curved region $C$, the rulings of the curved region to be created will be in conflict with the rulings of $C$. To deal with the two different directions of curvature, a singular point $s$ has to be added inside the curved region and is returned as border of the affected region, see Figure 6(d).

The technical details related to singular points generation are given in Section 5.2.

## B. CUTTING AND REMESHING THE PLANAR PARTS

The algorithm above gives us a triangle strip which will contain the new curved region. The deformation may however not necessarily affect the whole region. We define the actual limit of the curved region to be created by the extremal rulings $r_{\text{lim}_1}$ and $r_{\text{lim}_2}$ such that there is no singular point inside the region, see Figure 24. According to the borders previously computed, we have three cases:

- Both borders $e_{b1}$ and $e_{b2}$ are edges of the boundary of the paper $\partial S$. In this case the inserted rulings are parallel to the direction of propagation $\mathbf{d}_{\text{prop}}$ and are delimited by an endpoint on each border, see Figure 24(a). This configuration occurs also in the example shown in Figure 7.
- One border is an edge, $e_b$, of the boundary of the paper, the other is a singular point $s$ . In this case, $s$ being an apex of a generalized cone, the rulings are attracted by $s$ and are thus delimited by $s$ and an endpoint on $e_b$, see Figure 24(b). Another example is shown in Figure 8.
- Both borders are singular points $s_1$ and $s_2$. The rulings are attracted at the same time by $s_1$ and $s_2$, so the curved region degenerates into a single edge $(s_1, s_2)$, see Figure 24(c). Figure 9 shows such an example.

In the rare cases where we cannot find two rulings $r_{\text{lim}_1}$ and $r_{\text{lim}_2}$ without singular point between them, we do not generate a new curved region and let the physical simulation further deform this region until the geometrical step could find a proper change of connectivity.

If not degenerated the curved region is defined by a set of rulings between $r_{\text{lim}_1}$ and $r_{\text{lim}_2}$. We choose the number of rulings in order to respect the sampling density $\rho_{\text{bound}}$. Each ruling is defined by two endpoints (singular or belonging to the boundary $\partial S$ of the paper). The position of an endpoint belonging to $\partial S$ in the 2D pattern space $\overline{S}$ is computed as the intersection of the ruling and $\partial S$. The 3D coordinates of these points are computed using cubic interpolation along the boundary curve given by the mesh $S_P$ used in the physical simulation. Thus the newly created region is actually curved and its boundaries correspond to the curved boundary of $S_P$.

Finally, the rest of the region is triangulated, with the additional constraint in the degenerated case to enforce $(s_1, s_2)$ as an edge.

We can see in the Figure 7 (bottom left) that the deformation of a flat region induces bending of $S_P$ whereas the triangles of $S_G$ become compressed since the coarse triangulation of $S_G$ is not flexible enough. These two meshes are input to the "bending and crumpling" step of our animation loop as described in Section 5. The specific remeshing of $S_G$ (Figure 7 (bottom right)) with rulings inserted as explained in Section 5, can bend more easily and therefore better matches the shape of the input $S_P$.