# Interactive deformation of virtual paper

Camille Schreck

Communauté
UNIVERSITÉ Grenoble Alpes

# THÈSE

Pour obtenir le grade de

## DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques et Informatique**

Arrêté ministérial : 25 mai 2016


Présentée par

## Camille SCHRECK

Thèse dirigée par **Stefanie HAHMANN**
et codirigée par **Damien ROHMER**

préparée au sein du **Laboratoire Jean Kuntzmann**
et de l'**École Doctorale MSTII**
**Mathématiques, Sciences et Technologies de l'Information, Informatique**


# Déformation interactive de papier virtuel

Thèse soutenue publiquement le **24 Octobre 2016**,
devant le jury composé de :

**Dr. Bruno LEVY**
Directeur de recherche, Inria-Nancy Grand-Est, Président
**Dr. Maud MARCHAL**
Maître de conférences, IRISA-INSA, Rapporteur
**Dr. Bernhard THOMASZEWSKI**
Directeur de recherche, Disney Research, ETH Zurich, Rapporteur
**Dr. Jean-Francis BLOCH**
Maître de conférences, Grenoble INP - Pagora, Examinateur
**Pr. Stefanie HAHMANN**
Professeur, Inria Grenoble, Directeur de thèse
**Dr Damien ROHMER**
Maître de conférences, CPE Lyon / Inria Grenoble, Co-Directeur de thèse

Grenoble University

Doctoral Thesis

# Interactive Deformation of Virtual Paper

*Author:*
Camille Schreck

*Supervisor:*
Stefanie Hahmann
Damien Rohmer

*A thesis submitted in fulfilment of the requirements for the degree of Doctor of Computer Sciences*

*in the*

Imagine Team
Laboratoire Jean Kuntzman

October 2016

*"Something was creeping and creeping and waiting to be seen and felt and heard."*

H.P. Lovecraft, *The Colour Out of Space*, 1927.

# *Abstract*

## Interactive Deformation of Virtual Paper

Although paper is a very common material in our every-day life, it can hardly be found in 3D virtual environments. Indeed, due to its fibrous structure, paper material exhibits complex deformations and sound behavior which are hard to reproduce efficiently using standard methods. Most notably, the deforming surface of a sheet of paper is constantly isometric to its 2D pattern, and may be crumpled or torn leading to sharp and fine geometrical features. During deformation, paper material also has very characteristic sound, which highly depends on its complex shape.

In this thesis, we propose to combine usual physics-based simulation with new procedural, geometric methods in order to take advantage of prior knowledge to efficiently model the geometry and the sound of a deforming sheet of paper. Our goals are to reproduce a plausible behavior of paper rather than an entirely physically accurate one in order to enable a user to interactively deform and create animation of virtual paper.

We first focus on the case of paper being crumpled. We use the developability property of the paper to interleave the physics-based simulation with a geometric remeshing step that adapts the mesh with the folds and the sharp creases of crumpling paper. We obtain an interactive model able to reproduce the main features of crumpling paper.

We then take advantage of this model to develop a method for tearing paper in real-time. We use the geometric information provided by the remeshing step of the model to detect potential starting points of tears, and propose a new hybrid, geometric and physical, method to find their general direction of propagation while creating procedurally the details of the tearing path using a fibers' texture.

Finally, we propose to generate a plausible shape-dependant sound of the paper at run-time. We analyse the geometric and dynamic information provided by the model to detect sound-producing events and compute the regions in which the sound resonates. The resulting sound is synthesized from both pre-recorded sounds and procedural generation taking into account the geometry of the surface and its dynamics.

# *Résumé*

## Déformation interactive de papier virtuel

Le papier est un matériau très commun que l'on manipule quotidiennement. Pourtant on ne le trouve que rarement dans les environnements 3D. En effet, à cause de sa structure fibreuse, le papier, de même que le son qu'il produit, présente un comportement complexe qui se révèle difficile à reproduire avec les méthodes habituelles. En particulier, la surface du papier reste constamment isométrique à son patron 2D et peut se froisser ou se déchirer, créant ainsi de fins détails géométriques. Lorsqu'il se déforme, le papier produit également un son très caractéristique qui dépend fortement de la géométrie adoptée par la surface.

Dans cette thèse, nous proposons de combiner une simulation physique usuelle avec de nouvelles méthodes, procédurales ou géométriques, de façon à tirer parti de connaissances préalables afin de modéliser la surface et le son d'une feuille de papier manipulée virtuellement. Plutôt que d'obtenir des résultats précis au sens physique, nous cherchons à reproduire un comportement plausible du papier, permettant ainsi à un utilisateur de créer interactivement des animations de papier virtuel.

Nous nous concentrons dans un premier temps sur le cas du papier froissé. Pour cela, nous entrelaçons une étape de simulation physique avec une étape de remaillage géométrique qui adapte le maillage aux plis du papier froissé, exploitant pour cela la dévelopabilité du papier.

Nous tirons ensuite profit de ce modèle pour développer une méthode permettant de déchirer du papier virtuel en temps réel. Nous utilisons les informations sur la géométrie fournies par l'étape de remaillage pour trouver les points pouvant potentiellement être les points de départ d'une déchirure. Nous proposons aussi une nouvelle approche hybride, à la fois physique et géométrique, pour déterminer la direction générale de propagation tout en créant de façon procédurale les détails du tracé d'une déchirure en utilisant une texture représentant la répartition des fibres.

Enfin, nous proposons une génération de sons de papier, à la fois plausible, dépendant de la forme de la surface et qui s'opère en temps réel. Nous analysons les informations, géométriques et dynamiques, données par le modèle d'animation pour détecter les événements produisant du son et calculer les régions dans lesquelles le son résonne. Le son résultant est synthétisé à l'aide de sons pré-enregistrés et d'une génération procédurale, de façon à tenir compte de géométrie de la surface et sa dynamique.

# *Acknowledgements*

Cette thèse a été réalisée grâce à l'aide et au soutien de nombreuses personnes qui j'aimerais remercier ici.

Tout d'abord merci à mes directeurs de thèse, Stefanie et Damien, qui ont su me soutenir, m'encourager et me conseiller durant ces trois ans. Ils ont été là dans les bons moments comme dans les moins bons, aussi bien pour discuter de tout et de rien que pour finir une dead-line à 3 heures du matin.

Merci également à mes amis et collègues de l'équipe Imagine et une bonne partie de l'équipe Maverick pour leur bonne humeur. En particulier, je tiens à remercier Marie-Paule qui a pris le temps de s'intéresser à mes travaux. Mention spéciale également au infographistes de l'équipe, Estelle et Laura, qui ont mis leur talent d'artiste au service de mes vidéos de résultats. Et je n'oublie pas les différentes personnes qui ont partager mon bureau, en particulier Thomas, Quentin et Kevin, et leurs blagues (parfois douteuses:).

Merci aussi à mes amis. La bande de Xinra, Amélie, Benoît, Clemence, Floriane, Louis, Gilles et Raphaël, pour les échanges quuotidients et le support moral et parfois technique. Et mes amis de collège et lycée, Léa, Marie et Mathilde: malgré le temps et la distance, chaque fois que l'on se retrouve, c'est comme si l'on venait de quitter le lycée.

Enfin, un grand merci à ma famille, mes parents et ma soeur, pour avoir été là et m'avoir supportée depuis toujours.

# Contents

# Introduction

*"There are neither beginnings or endings to the turning of the Wheel of Time. But it was a beginning."*
Robert Jordan, *The Wheel of Time*, 1990.

Over the last few years, virtual computer-generated environments, and especially animated ones, have been taking an expanding place in our society. As 3D animated models are increasingly used for entertainment applications such as movies, video games, or advertising, the need for 3D virtual content has been growing likewise. Yet creating a 3D animated virtual model, is still cumbersome and time-consuming, even for expert digital artists. A long training process is also needed as mastering current 3D animation softwares requires both expert artistic and computer skills, but also some some understanding of advanced mathematical concepts to manipulate the degrees of freedom –usually limited to some low level of abstraction– proposed by these softwares. New 3D modeling and animation techniques are therefore required to make the work of professional 3D artists more efficient, but also to allow the general public to access 3D modeling and animation. Developing these new techniques is a challenging task as it should satisfy the following three constraints. Firstly, the resulting 3D model should be visually plausible in order to satisfy the public. Secondly, the method itself needs to be intuitive to use in the sense of being easy to understand and to learn by the artist. Thirdly, the method should be interactive, providing the artist with an immediate feedback while manipulating the 3D model and an accurate control of the 3D result. In addition, multimodal interactions –mainly audio, but also haptic or even olfactory cues– are of growing importance. Those feedbacks have been until now only scarcely researched and applied in 3D graphics, yet it should become an inherent part of a 3D model. In particular, an audio feedback is usually added to animation, as sound is often

necessary for realism. But, since existing softwares do not propose automatic ways to add it, sound is generally created "by-hand" by Foley artist.

Numerous researches focus on automatic, or semi-automatic animation, as, for example, animation for the clothes of a virtual character, the motion of fluids or the movement of a crowd. For the case of natural physical phenomena, the current automatic animation methods can be divided into two categories: the physically-based methods and procedural methods. First the physically-based methods aim at reproducing the physical phenomenon involved. Those methods usually give realistic animations, with natural movement but this also comes with a high computation time making it hardly usable for interactive applications. Moreover, the control of the animation mostly relies on, sometimes esoteric, physical parameters making it difficult to control the animation with accuracy. Second, the procedural methods use a set of rules to build the animation, for example geometric rules or stochastic rules that aim at reproducing some mathematical properties or phenomenological observations. They are often fast enough to obtain real or interactive times and offer more intuitive control to the users. But realism is harder to achieve particularly for deformations caused by natural physical phenomena. These methods also may be limited to specific cases or are not fully automatic and need more user-inputs. We may consider the hybrid methods as a third category; they are a mix between physically-based and procedural techniques. Hybrid methods aim ideally at obtaining the advantages of both procedural and physical methods without their drawbacks.

In the present thesis, we focus on virtual paper animation, for which the application of a hybrid method proves to be efficient. Paper is a very common material, yet it can hardly be found in virtual environments. The main reason for this absence is the lack of satisfying methods to model it automatically. Virtual paper is then rarely used and, when still modeled, often created by 3D Graphics artists. Paper indeed exhibits a complex behavior that usual techniques struggle to reproduce. The structure of microscopic fibers composing this material influences its macroscopic behavior. In constant experimental condition (humidity, temperature etc..), we can consider those fibers as inextensible. This leads to an extremely high in-plane stiffness (resistance to in-plane deformations, i.e., compression and extension), which, combined with the low bending stiffness, makes paper bend rather than undergo in-plane deformations. It can then support high curvature without damage but the bonds linking fibers between each other easily break under too much strain –making the paper crumple– or stress –creating tears. Due to its stiffness, paper also generates a sound that is strongly dependent on the shape, making it hard to reproduce. To our knowledge there is no prior research that focuses on this problem for paper-like material.

Neither procedural, nor physically-based methods are well-adapted to reproduce those characteristics. Indeed, procedural methods are ill-suited for modeling natural motion well enough to obtain satisfying results for a material as common as paper. On the other hand, physical simulations can lead to convincing results. But the dense mesh needed to represent sharp features and highly bent regions as well as the high stiffness required make the computational cost prohibitive for many applications, notably the interactive ones.

Our insight in the different works presented in this thesis is to use prior knowledge of the specific behavior of paper to make the physical simulation more efficient. More specifically, the methods proposed here aim at satisfy the following criteria:

**Realism.** By basing the procedural steps on phenomenological observations as well as geometrical properties of paper, we reproduce plausible behavior of paper rather than physically accurate one. We compare our results to real paper for simple, familiar cases in addition to more complex cases in which defects are less noticeable.

**Rapidity.** Real-time, or at least interactive-time is required in order to use the algorithms in interactive applications.

**Intuitive user interaction.** The user can control the deformation of the paper simply by manipulating virtual fingers on the paper.

Our contributions are divided into three methods:

**Crumpling paper.** We present in Chapter 2 the first method, to our knowledge, that automatically animates crumpling paper at interactive rates. We propose to interleave a conventional thin shell simulation step with a geometric step that adapts the mesh to meet the developablility constraints of real paper. To this end, we approximate folds by a set of generalized cones and sharp features where the fibrous structure is damaged by singular points. We therefore align the triangles of the mesh with the rulings of the cones and creases are explicitly represented as vertices of the mesh, such that both smooth folds and sharp features are obtained while maintaining a very coarse mesh. Moreover plastic behavior is taken into account as the positions of the creases are explicitly computed and recorded as singular points of the geometry.

**Tearing paper.** The method for crumpling paper summarized above handles the case of the fibrous structure undergoing strain. We focus instead in Chapter 3 on paper tearing, which happens when the structure breaks under stress. We first present an entirely procedural method that models paper being torn in real-time but only

handling a simple case. We then extend the basic idea to more complex situations by clustering the forces around the tip of the tear into two opposite forces. This reduces the problem to a situation for which a geometric criterion for the relevant direction of tear can be derived. We then couple it with our crumpling model taking advantage of its geometric features notably to find the potential starting points of a tear.

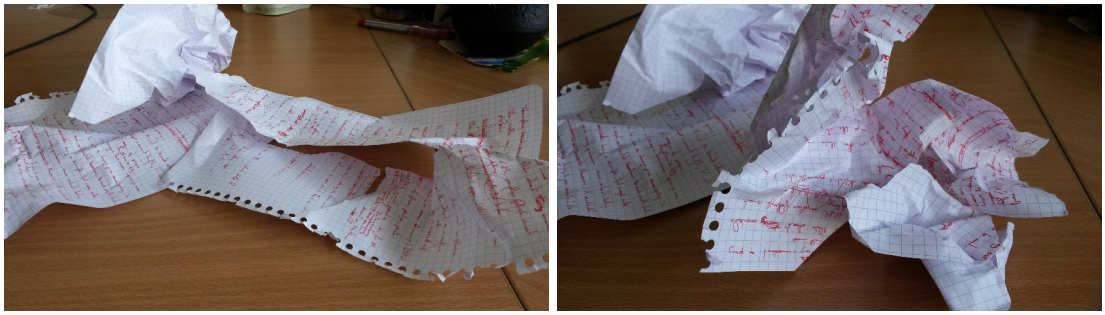


FIGURE 1: Crumpled and torn paper.

**Generating the sound of paper.** Since the geometry of paper influences its sound, we propose in Chapter 4 to take advantage of the geometric features computed by our animation model to generate plausible sounds for our virtual paper. We first propose a method to synthesize the sound of crumpling paper that introduces the notion of geometrically parameterized *resonators*. This enables us to generate a shape-dependent sound, based on a small database of sound units. As the paper is highly deformable, a new set of resonators needs to be computed at each frame; this is done by a procedural algorithm that is fast enough to achieve real-time rates. We then complete the sound model with a procedural method for generating the sound of paper being torn, parametrized only by the speed of the tearing.

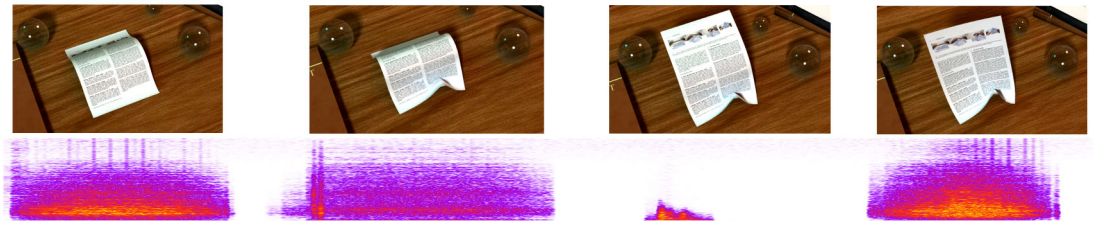


FIGURE 2: Animation of a sheet of virtual paper and the corresponding synthesized sound represented as a spectrogram.

## Notations

In this document, we use the following notations:

**Vectors** are denoted by bold lower case roman letters: $\mathbf{v} = (v_x, v_y)$. The same notation is used for spatial position. The scalar product between two vectors is denoted by . and the cross product is denoted by $\wedge$.

**Matrices** are named with bold upper case roman letters such as:

$$\mathbf{M} = \begin{pmatrix} m_{0,0} & m_{0,1} \\ m_{1,0} & m_{1,1} \end{pmatrix}.$$

The inverse of $\mathbf{M}$ is denoted $\mathbf{M}^{-1}$, and its transpose is $\mathbf{M}^T$. The trace is denoted $Tr(\mathbf{M})$.

**Coordinates in 2D and 3D.** The vertices used to described the surface of the paper are represented by a position in space $\mathbf{x}$. A sheet of paper is always associated to its unfolded 2D pattern. The corresponding 2D vertices of this pattern are denoted $\bar{\mathbf{x}}$. By extension any element $e$ (edge, triangle, vector...) of the 3D space is represented in the 2D pattern by $\bar{e}$.

**Complex exponential.** The complex exponential is represented as:

$$e^{i\alpha} = \cos(\alpha) + i\ \sin(\alpha).$$

# Related work

*"Books! The best weapons in the world!"*
The Doctor, *Tooth and Claw*, 2006.

Paper is a material with a very specific and complex behavior that has been studied by Physics and Material Science researchers but is not yet fully understood. This material has also gained more and more interest in the Computer Graphics domain as techniques to model and deform surfaces have become more efficient.

Paper has been the subject of numerous works. They give us many insights on paper behavior that are described in the first section. We will then mainly present the methods that have been used in Computer Graphics to model paper. They can be divided into two categories: methods derived from physical simulations of thin fabrics on one hand, procedural methods based on geometry or data on the other hand.

## 1.1 Physical and material studies of paper behavior

Paper presents interesting and complex physics. They are summarized, along with some associated studies, by Alava and Niskansen in [AN06].

Paper is composed of fibers, usually made of wood, bound together to form a random network (see Figure 1.3). As the length of a fiber (around 2mm) is large enough compared to the thickness of paper (around 0.1mm), paper may be accurately enough considered as a 2D surface (so the thickness is considered as negligible). The process of making paper leads to an inhomogeneous distribution of the fibers' density along the surface – as it can be observed by transparency while looking at a sheet of paper held in front of a light source. The position of the fibers is stochastic, yet in the industrial process, two

main directions ("machine direction" and "cross direction") appear in the orientation of the fibers. As a result, paper exhibits an anisotropic material behavior – for example the bending stiffness may be different according to the direction. In the works presented in this thesis, we neglect the phenomenon and consider paper as isotropic. However it could be an interesting aspect to take it into account in the future.

FIGURE 1.3: X-ray tomography of paper showing the fibers' structure, from [AN06].

The fibers are only slightly elastic – they can barely be compressed or stretched–, and the bonds in-between prevent them from sliding against one another. That is why paper is usually considered in our domain as rigid enough to preserve lengths upon deformation as long as the structure does not fail. Indeed when subject to too many constraints the bonds, occasionally the fibers, break which leads to irreversible damage. The latter appear at a macroscopic level either as sharp creases or tears in the paper.

**Crumpling of paper.** Numerous articles in Physics study the phenomenon of crumpling for thin sheets: the geometry and energy distribution around a single crease – either as a single point [AP97, MC98, CM98], or as a linear ridge [LW97, LW05, DDMS12], or as a more general crease [SKD11]– or for an entire sheet crumpled into a ball [TÅT09, CM11]. Some studies also focus on the ridge network formed by crumpling and unfolding a sheet [AHS07, BK05]. The review article of Witten *et al.* [Wit07] discusses many of these previous works.As explained in Chapter 2, our model of crumpling paper is based on insights from these Physics works.

**Tearing of paper.** Fracture Mechanics is a challenging research area. One of the main problems addressed is the prediction of the propagation of a crack in 2D or 3D material. For understanding the case of tearing of paper, we focus on the studies of quasi-static fracture in brittle thin sheet material. The early work of Griffith [Gri21] sets many bases for this problem and notably formulates what is referred to as the Griffith's criterion. It states that the crack propagates if the energy released by the propagation is equal to the fracture energy of the material (i.e. the energy required to create a new free surface in the material). The direction of the propagation is still a debated question, the two main theories being (1) the crack propagates in the direction

which maximizes the energy released (in the case of isotropic material) [BFM08], (2) the principle of local symmetry (PLS) [GS74, HS93], i.e. the crack propagates such that the in-plane shear stress vanishes around the crack tip. Roman [Rom13] reviews many of those works along with some studies of the geometry of the crack path in some specific conditions as [OKe94, ARR05]. Some works also focus on experimentally measuring the physical parameters of such a phenomenon. For example, Seth and Page look at the fracture toughness and the quasi-static fracture energy in [SP74].

**Acoustics of paper.** Power-law models of the acoustic emissions from crumpling elastic sheets [KL96] and paper [HS96] have been studied by the Physics community, as well as the sound of paper fracture [STA02]. Indeed acoustic emission is typically used to physically study irreversible damage in the internal structure of a material.

## 1.2 Simulating virtual paper using a physics-based model

Physics-based approaches are widely used in Computer Graphics to obtain physically realistic complex behavior. The principle is to derive physical laws to obtain partial differential equations describing the physical process on the virtual object. Numerical discretization and integration can then be applied to integrate the differential equation, and then retrieve the motion.

The methods used to model paper or other thin materials, such as fabric, represent the object as a 2D surface in a 3D environment. The most usual way to represent a surface is to use a mesh (see Figure 1.4): the surface is subdivided into small polygons –most often triangles as they are easy to manipulate– called faces, delimited by edges and vertices.

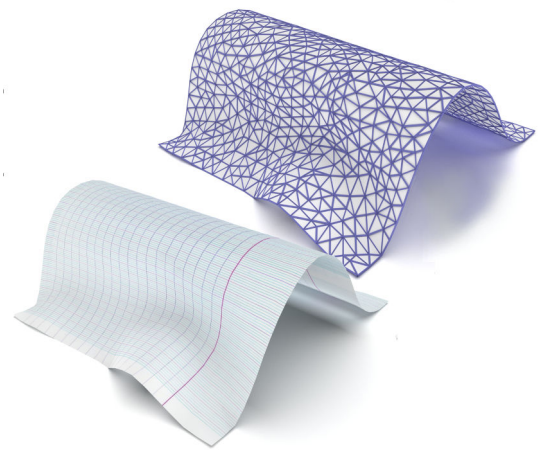


FIGURE 1.4: A triangle mesh (top-right) representing the surface of a virtual sheet of paper, and the same mesh after texturing and rendering (bottom-left)

### 1.2.1 Thin shell theory

The physics-based methods used to represent paper are inspired by thin shell models based on finite element methods (FEM). The standard framework for dealing with thin shells is the Kirchhoff Love Theory. Introductions to this theory can be found in [Cia97]. This approach was used initially in Computer Graphics by Terzopoulos in [TF88] is widely used for cloth and fabric simulation [VCMT95, BW98, CK02]. The idea is to reproduce linear elastic behavior by computing the forces caused by the deformation of the mesh applied on each vertex modeled by a small mass –the Mass-Spring system, for example, computes the forces applied by each edge by considering them as springs linking the vertices. In-plane stress can be computed using edges or faces deformation; this gives good results for material with negligible bending rigidity as fabric. The thin plate theory extends it to paper-like material; Grinspun *et al.*, later Burgoon *et al.* [GHD*03, BGW06], propose to add bending forces based on dihedral angles between triangles. In our work we use the code provided by Narain *et al.* [NSO12] based on Green's strain to compute the in-plane stretching forces and bending forces as described by Grinspun as follows.

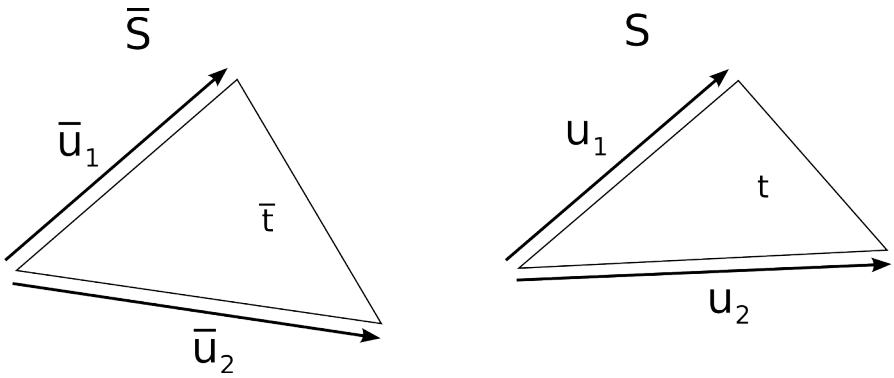



FIGURE 1.5: A triangle $t$ defined by the vectors $u_1$ and $u_2$ in its initial shape (left) and after some transformation (right)

A triangle $t$ can be represented by a matrix $\mathbf{U} = \{\mathbf{u_1}, \mathbf{u_2}\}$ where $\mathbf{u_1}$ and $\mathbf{u_2}$ are the vectors along the edges of the triangles as represented in Figure 1.5. The deformation between the current state $S$ of the mesh and the initial state $\bar{S}$ –in our case, $\bar{S}$ is the 2D pattern of the sheet– can be represented by the deformation gradient $F$ defined by :

$$\mathbf{F} = \mathbf{U}\bar{\mathbf{U}}^{-1}.$$

The Green's strain is then obtained by:

$$\mathbf{G} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}),$$

where $\mathbf{I}$ is the identity matrix.

The stress energy caused by this strain is dependent on the in-plane stiffness $k_s$ of the material and can be modeled by:

$$W_{stress} = k_s((1-\mathcal{V})\ Tr(\mathbf{G}^2) + \mathcal{V}\ \mathrm{Tr}(\mathbf{G})^2,$$

where $\mathcal{V}$ is the Poisson's ratio of the material.

The discrete bending energy of a triangle is based on the dihedral angle at each of its edges.

$$W_{bending} = k_b \sum_{i=1...3} (\sigma_{e_i} - \bar{\sigma}_{e_i})^2 \frac{||\bar{\mathbf{e}}_\mathbf{i}||}{\bar{h}_{e_i}}$$

where $k_b$ is the bending stiffness, $\{\mathbf{e_i}\}_{\mathbf{i=1...3}}$ are the edges of the triangle. $||\bar{\mathbf{e}}_\mathbf{i}||$, $\bar{h}_{e_i}$, $\sigma_{e_i}$ and $\bar{\sigma}_{e_i}$ are respectively the length (measured on the 2D pattern), height of the triangle, current dihedral angle and rest angle corresponding to the edge $e_i$. The elastic energy of a triangle is then:

$$W = W_{stress} + W_{bending}$$

The differential equation of motion is then obtain by:

$$\ddot{x} = \mathbf{M}^{-1}\nabla W(x)$$

It is also possible to add some external forces (like gravity for example) and/or damping to the equation of motion.

This approach may work well for slightly elastic material, but difficulties arise for inextensible or nearly inextensible materials. High stiffness causes instabilities in the simulation, and smaller time steps are thus required to get a stable animation implying the computation of a greater number of simulation steps for each frame. The direct application of the thin shell model alone is therefore ill-suited to obtain length-preserving simulation with interactive rates.

To get closer to inextensible material, a possible approach is to limit the strain at each simulation step as done by Provot [Pro96] or Wang *et al.* [WOR10]. English and Bridson [EB08] propose to apply physical simulation to nonconforming elements instead of the usual geometrically continuous mesh, allowing to compute exact isometric deformation. But the nonconforming elements also need to be coupled with a conforming mesh for collision detection and rendering.

Smooth deformations can efficiently be modeled by these approaches. Even sharp creases can be handled when manually defined by the user (see Figure 1.6). Burgoon *et al.* [BGW06] extend the approach for origami simulation where all the sharp folds are


FIGURE 1.6: Realist curved piece of virtual paper from [GHD*03]. By altering dihedral angles of the reference configuration, a sharper edge as been created.

predefined by the user. In this work, the triangulation is adapted such that the mesh's edges align with the sharp folds (see Figure 1.7). We note that origami has attracted specific attention from the research community (see Figure 1.8). For instance, Tachi [Tac11, Tac09] considers each face as a rigid planar object and uses the rigid object kinematics rather than thin shell theory. However, this approach is not able to reproduce the specific plastic behavior of paper material. Indeed as a piece of paper crumples, sharp creases spontaneously appear and change irreversibly the paper mechanical behavior. Sharp features can possibly appear in thin shell simulation if the mesh is dense enough, however the plastic behavior cannot be directly reproduced using this kind of method. In addition, the high number of triangles needed to have a mesh that fine makes the computational cost prohibitive.

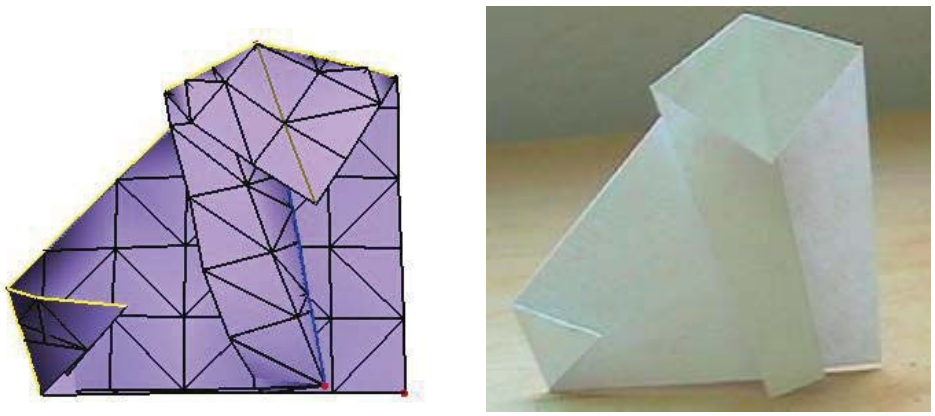


FIGURE 1.7: A comparison from [BGW06] showing an origami obtained with their approach (left) and a real one (right)

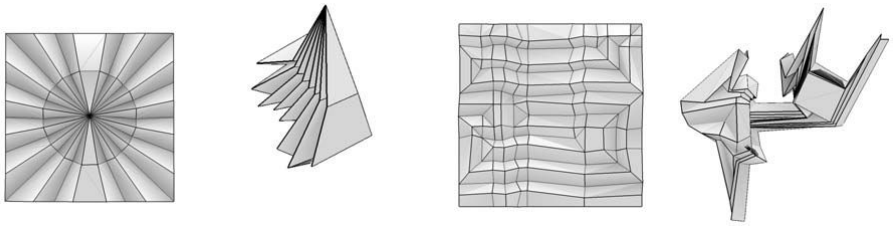


FIGURE 1.8: Folding pattern and the corresponding origami from [Tac09]

### 1.2.2 Dynamic meshes for crumpling and tearing

In order to avoid the prohibitive increase in mesh resolution when modeling crumpling paper, using automatic adaptive meshes instead of fixed ones has been proposed to handle the crumpling behavior. The principle is to remesh the surface to better adapt to the folds and constraints.

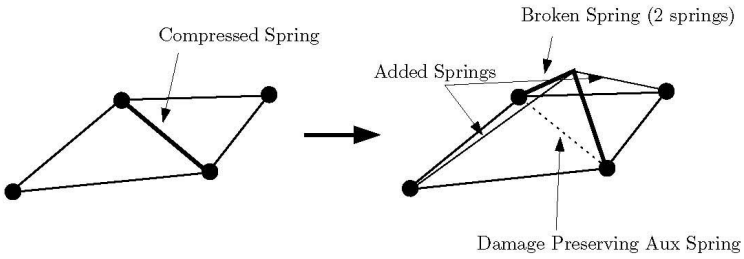



FIGURE 1.9: The remeshing scheme from [KZC09] based on breakable spring

Kang *et al.* in [KZC09] add plastic damage to classic mass-spring simulation by breaking springs which are highly compressed. More precisely, at time $t$, a spring between vertices $i$ an $j$ has a probability $P(i,j)$ to break where

$$P(i,j) = (1 - \frac{l^t_{i,j}}{l^0_{i,j}})^\phi,$$

with $l^t_{i,j}$ being the current length of the spring, $l^0_{i,j}$ its initial length and $\phi$ controlling the fragility of the paper. If the spring breaks, it is subdivided into two springs. As shown in Figure 1.9 two more springs are added, to keep the faces of the mesh triangular, and a damage preserving spring to keep the paper wrinkled even when the constraints are released. This is coupled with a fractal surface texture to create smaller details. The method has the advantages to be rather simple to implement and fast, and may produce natural looking wrinkles (see Figure 1.10). Yet the wrinkles appear along the already existing edges leading to some visual artifacts. Also the method does not handle well smooth shapes: the simulated paper will tend to crumple where it should bend smoothly.

This may hamper the realism as the smooth bending occurring at the beginning of the crumpling process influences notably the final crumpled shape.

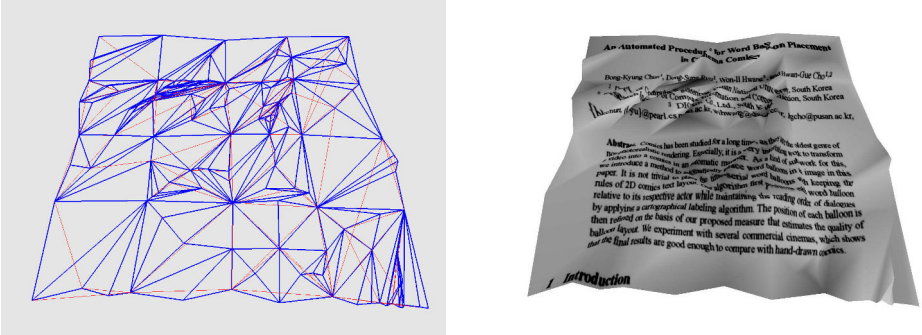


FIGURE 1.10: A crumpled paper and the corresponding mesh (the damage preserving springs are represented in red) from [KZC09]

A more recent work using adaptive meshes has been proposed by Narain *et al.* in [NSO12], using the FEM described above. The mesh is refined or coarsened according to a tensor field $M$ that represents the maximum permitted length of edges at each location and orientation. $M$ is defined by the curvature, compression, and velocities of the material. The mesh becomes denser in curved region, with anisotropic elements in the direction of the folds, and coarser in the flat regions. They obtain realistic cloth simulations with fine details much faster than with fixed mesh connectivity. Narain [NPO13] later expands this work to crumpling phenomena, notably by adding a damage parameter $\delta$ inherent to each face to obtain plastic behavior. $\delta$ increases when the bending strain $\mathbf{S}$ exceeds the material's yield curvature $\kappa$ in the following way:

$$\delta \leftarrow \delta + \frac{1}{\kappa}(||\mathbf{S}|| - \kappa).$$

The damage $\delta$ is used to attenuate the bending stiffness. The bending stiffness $k_e$ associated to an edge $e$ is:

$$k_e = \frac{1}{1 - \sigma\delta_e} k_{mat},$$

where $k_{mat}$ is the initial bending stiffness of the material, $\delta_e$ is the average of the damage of each face adjacent to $e$, and $\sigma$ is a user-defined parameter to control how much the damage weaken the bending stiffness of the paper. Note that they also propose a rib-stiffening method to account for the paper being less likely to buckle in the direction orthogonal of the already existing folds. Another interesting feature of this work is a projection applied after the remeshing to prevent the instability caused by dynamic changes. The results are quite compelling (see Figure 1.11); creased features appear naturally where the constraints are higher, making it physically plausible. As the mesh is locally densified only where needed, the method is much faster than classic FEM. Yet an large number of triangles is still needed to represent the creases, therefore the

computation time is still too high for any interactive application (see Section 2.6.2 for time and quality comparisons between FEM with fixed mesh, this method and our method described in Chapter 2). Another problem that may occur is the diffusion of the damage of a crease as the mesh gets flatter and so coarser. The exact position of the creases, where the fibers should have broken, is not retained.

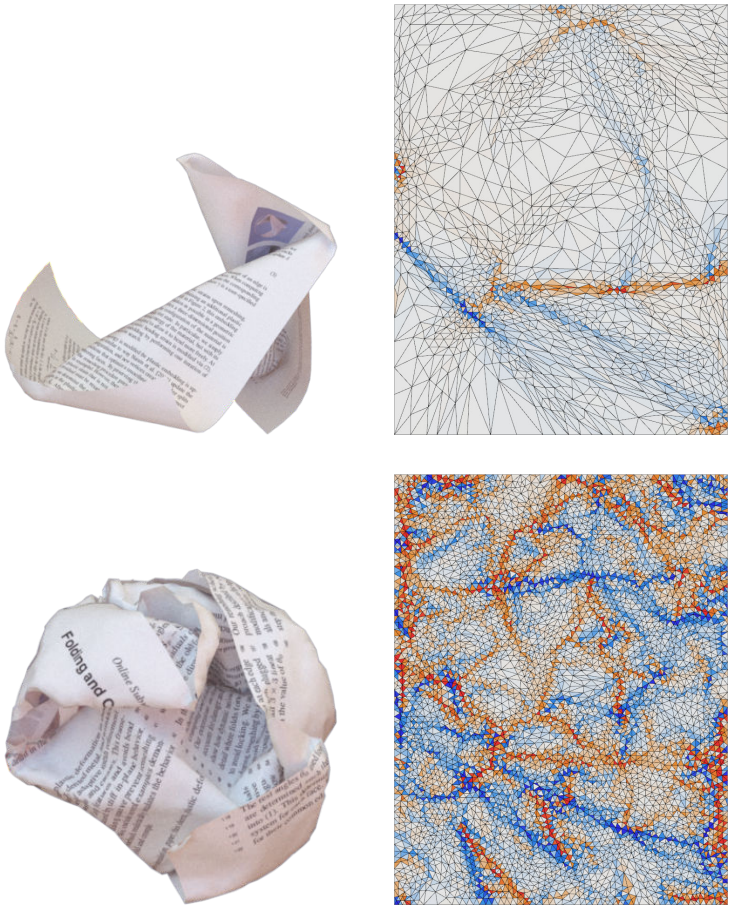


FIGURE 1.11: Two crumpled virtual piece of paper from [NSO12]. The right column represents the 2D pattern of the triangulation. The mesh is very dense in the highly curved regions and coarser in the near-flat regions

Many physics-based methods which deal with fracture phenomenon are also based on FEM. Although most of them treat the case of the fracture of volumetric solid objects (as for example [BHTF07, GMD12, OBH02, OH99]), some of them focus on thin shell tearing (note that they are not specific to paper). The basic idea, already used by Terzopoulos [TF88] is to create or propagate tears when the strain or stress becomes too high (see Figure 1.12). A simple approach, used for example by Souza [SvWC14] consists in splitting one vertex of the overstretched edges to create a tear that follows the existing edges as shown in Figure 1.13. The excess energy is then dissipated with a local physical relaxation around the tip of the tear.

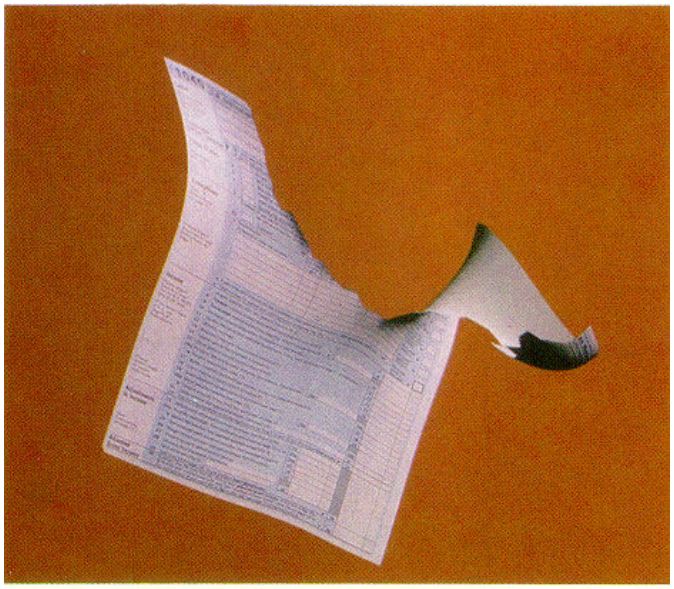


FIGURE 1.12: One of the first result of virtually torn paper from [TF88]

FIGURE 1.13: Virtual fabric being torn. Picture from [SvWC14].

Gingold *et al.* [GSH*04] based its fracture handling on the stain tensor **E** of each triangular face of the mesh: a face is split into two if the principal strain –the largest Eigen value of **E**– exceeds a material-specific threshold. The fracture splits the strained face along the direction orthogonal the principal strain direction –the associated Eigen vector. As a vertex needs to be inserted on one edge of the face, the adjacent face will split too as shown in Figure 1.14. The excess energy is absorbed by the plastic behavior that they also proposed for their model.

Pfaff *et al.* [PNdJO14], based on the prior work on adaptive meshes by Narain [NSO12], propose to refine the mesh where a crack is likely to propagate or be created and coarsen it elsewhere. The direction of a potential crack is computed by finding the potential splitting plane for which the most stress energy would be released. Local physical simulation around the tip is done to release the excess stress.

A work of Metaaphanon [MBCN09] aims at modeling frayed edges of woven fabrics being torn. A simple spring-mass system, the base cloth model, represents undamaged regions, and a yarn-level model where each mass particles are split in two loosely coupled

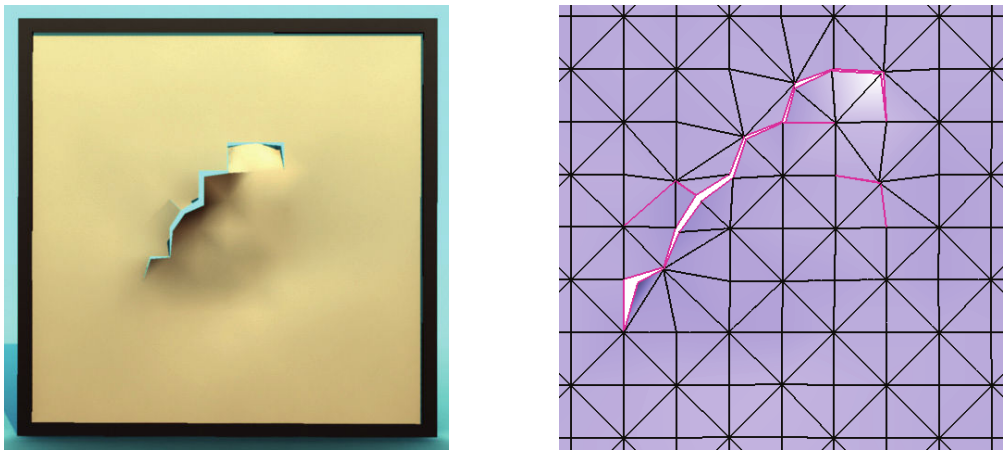


FIGURE 1.14: A fracture in a thin plate from [GSH*04] and the corresponding triangulation. Edges that underwent plastic damage without breaking are represented in red.

particles linked to different threads of springs models the two layers of interwoven yarn of the cloth. As shown in Figure 1.15 (a), springs from the base cloth model can be cut under stress. The vertices around the cut are split leading to the yarn-level model (b). Under more stress, coupled particles may be disconnected (c) causing the splitting of the neighboring particles of the base model (d). As the cloth is torn, the fraying propagates. They obtain detailed frayed edges as expected for woven fabric as shown in Figure 1.16. Still, although paper may be seen as a structure of interwoven fibers, the small size and stochastic position and orientation of the fibers make this kind of approach hardly efficient for tearing paper.

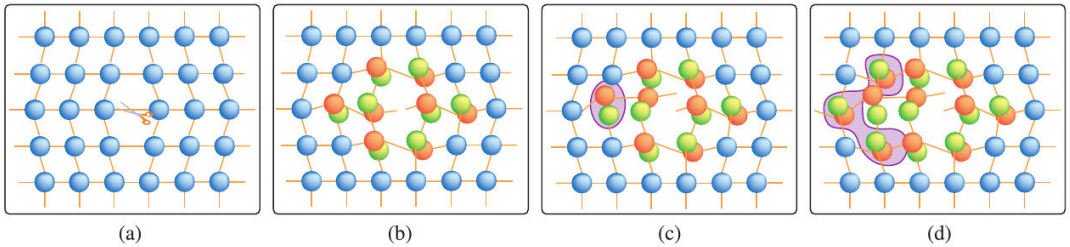



FIGURE 1.15: Tearing process from [MBCN09], the model switches from base cloth model to yarn-level model around the tear.

Some research to generate crack pattern, as the one proposed by Iben [IO06], also use physics-based methods, but they do not allow actual changes of geometry of the object as it is expected for paper.

### 1.2.3 Computing sounds with a thin shell model

The thin shell model can also be used to compute the small vibrations involved in sound generation. The pressure of the air above the surface can be inferred and be translated

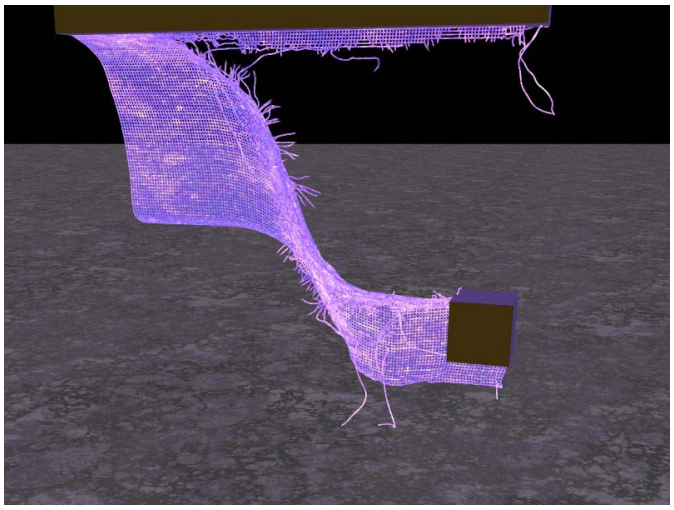


FIGURE 1.16: A torn piece of cloth from [MBCN09] with frayed edges.

into the sound produced by the object. This is done by O'Brien *et al.* in [OCE01]. The thin shell is then used to synthesize both the animation and the sound. A wide range of complex scenes (including sheet-like objects) can be simulated. But as the sound usually requires a sample rate of about 40kHz (20kHz for low quality sound), the computation time becomes quickly prohibitive.

To be more efficient, modal vibration models [PW89, JP02], introduced to digital sound generation by Adrien [Adr91], may reduce the computational cost. The main modes of vibration of the object are pre-computed so the vibrations are modeled only for a limited number of modes. The linear modal synthesis methods have become widely used for rigid-body sound [DP98, vdDKP01, Coo02, OSG02, JBP06, ZJ10]. Moreover it is possible to model a large number of sounding objects at interactive rates using acceleration techniques [RL06, BDT*08], and also memory compression [LAJJ14]. Ren *et al.* [RYL13] present a method to estimate modal parameters from recorded audio clips. The linear modal synthesis has also been considered for non-linear thin-shell sounds by Chadwick *et al.* [CAJ09]. However these approaches only support small deformations as modes depend on the shape. For highly deformable materials such as fabric or paper, the modes should be recomputed for each step, losing therefore most of the speed-up of the modal approach.

Finally, the physics-based methods presented in this section give realistic, even sometimes compelling, results for crumpling and tearing paper and also for generating sound. However they are not well suited for interactivity which is one of our purposes. Indeed the high rigidity imposes small time steps, and the sharp fine features, which influence the motion and the sound of paper, require a very dense mesh. The combination of both makes the computational cost far too high for interactive applications. To make the

physics-based simulations more efficient, we propose in this thesis to inspire from geometric methods summarized in the next Section 1.3 in order to represent sharp features and folds with coarser mesh and reduce the need of high rigidity.

## 1.3 Simulating virtual paper using procedural and geometric approaches

Contrary to the physics-based methods, geometric and procedural approaches are usually fast and provide more intuitive parameters for the user, giving them more control over the final result. Geometric methods for paper modeling use the well-known property of developability of paper. To our knowledge, no method so far is dedicated to the tearing of paper or the generation of its sound. But more general works on procedural fracturing or data-based sound generation are inspirational for creating methods specialized for paper.

### 1.3.1 Developable surfaces

Paper is a developable surface: it can be unfolded onto a plane without any in-plane deformation. Developable surfaces exhibit a number of interesting geometric properties [DCDC76]. Notably $C^2$-developable surfaces are a particular case of ruled surfaces. Ruled surfaces can be represented by a one-parameter family of lines, called rulings. Each point can then be defined by:

$$\mathbf{x}(u, v) = \mathbf{c}(u) + v\ \mathbf{r}(u)$$

with $\mathbf{c}$ being a curve of $\mathbb{R}^3$ and $\mathbf{r}$ a vector field. A ruled surface is developable if its tangent plane is constant along any ruling, meaning that $\mathbf{c}'(u)$, $\mathbf{r}(u)$ and $\mathbf{r}'(u)$ are coplanar.

$C^2$-developable surfaces are composed of four types of surfaces:

- tangent developables, which are defined by:

$$\mathbf{x}(u, v) = \mathbf{c}(u) + v\ \mathbf{c}'(u)$$

  The rulings of a tangent developable cross each other on a singular curve of the surface called the curve of regression.

- generalized cones for which the curve of regression is reduced to one point $\mathbf{s}$ called the apex.

$$\mathbf{x}(u, v) = \mathbf{s} + v\ \mathbf{r}(u)$$

- generalized cylinders for which the curve of regression is reduced to one point at the infinity. All the rulings are the parallel to a direction $\mathbf{r}$.

$$\mathbf{x}(u, v) = \mathbf{c}(u) + v\ \mathbf{r}$$

- planes.

A $C^2$-developable surface can also be defined as a surface whose Gaussian curvature (product of the main curvatures) is zero at any point. A fact that derives from the isometric nature of admissible deformations and is known as Gauss' Theorem Egregium.

As paper can be considered inextensible, in addition to be developable, it also stays isometric to its 2D initial flat shape, called 2D pattern: unless damaged, the geodesic distance between any two points of the paper surface equals the Cartesian distance between the two corresponding points on the 2D pattern.

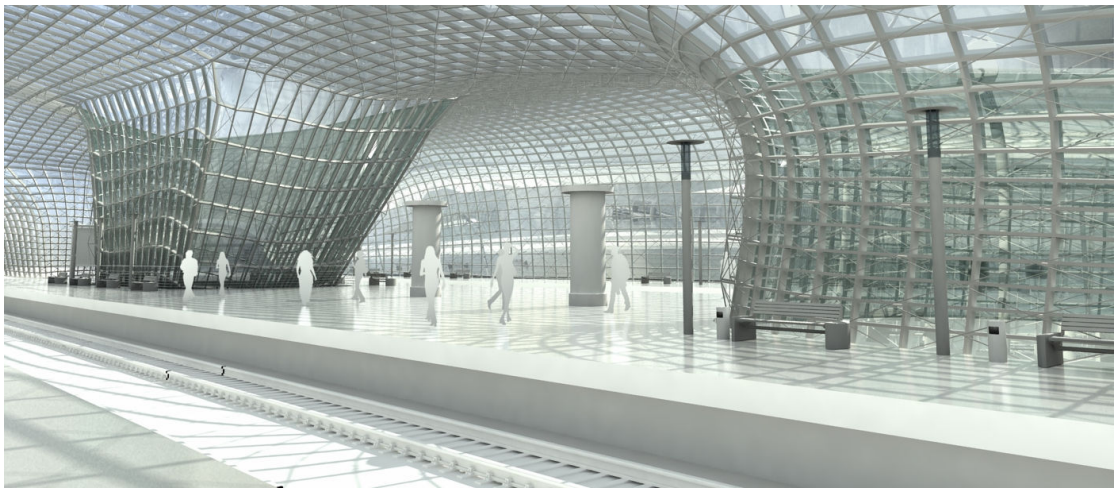


FIGURE 1.17: Developable surfaces can be used to design elegant shapes for architecture as this glass structure from [LPW*06]

Developable surfaces have been studied in geometric research. Solutions for generation [Aum03, PF95], approximation and interpolation of developable surfaces [PW01, Pet04] have been proposed.

Developable objects can be modeled from sketching as proposed by Rose *et al.* [RSW*07] who generate developable surfaces from 3D boundary curves, or from simple shapes –Chu and Séquin [CS02] propose a method to create quadratic and cubic Bezier developable patches. Liu *et al* [LPW*06] use developable surfaces to optimize a quadratic mesh

in order to help the design of free-form architecture (see Figure 1.17). Decaudin *et al.* [DJW*06] and later Jung *et al.* [JHR*15] create 3D models for clothes from users' sketches such that the models are optimized to be developable. They also compute the corresponding 2D pattern of the cloth so it is possible to build the model with real materials. Note that finding a 2D pattern from a 3D shape is another interesting challenge related to developable surfaces (see for example the work of Tachi [Tac10] on finding a folding diagram to create an origami of a given 3D shape). Notably, the problem of unfolding a polyhedron has been investigated since Albrecht Dürer in the early 16th century. The Shephard's Conjecture –every convex polyhedron can be cut along some of its edges and unfolded onto the plane without overlap– has still not be proved or disproved. These approaches only study smooth developable surfaces and none of them considers how to form and retain sharp features during the modeling process.

Some methods are dedicated to optimize an input surface in order to increase the developability. For example Wang *et al.* [WT04, Wan08] minimize an objective in terms of discrete Gaussian curvature. These approaches are computationally intensive. Moreover, although they create developable surfaces, they cannot explicitly formulate length preservation with respect to an original 2D pattern.

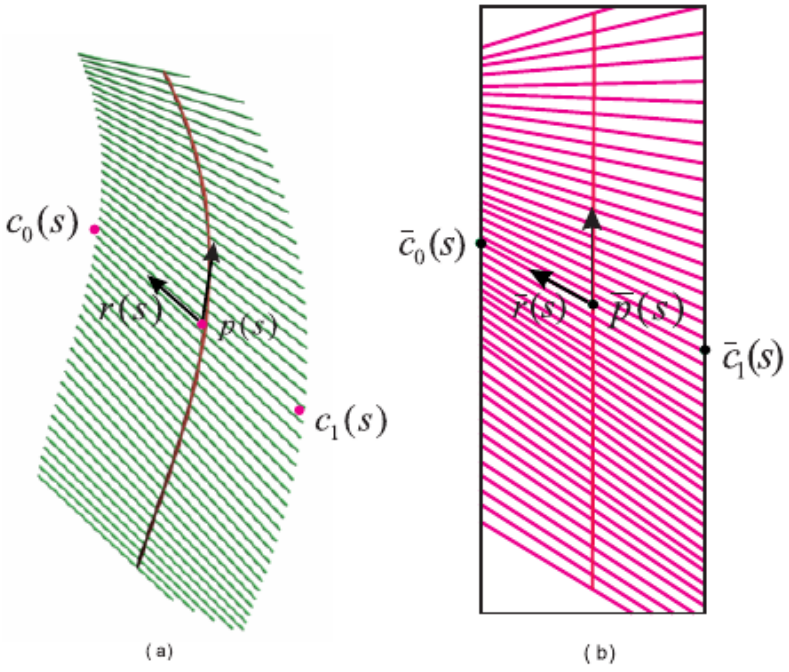



FIGURE 1.18: Construction of a smooth developable surface from [BW07]. The 3D boundary is computed by reporting the distance in the flat state (b) between each point $\bar{\mathbf{p}}(s)$ of the geodesic and the intersections of the corresponding ruling $\bar{\mathbf{r}}(s)$ and the boundary along the ruling $\mathbf{r}(s)$ on the 3D space (a).

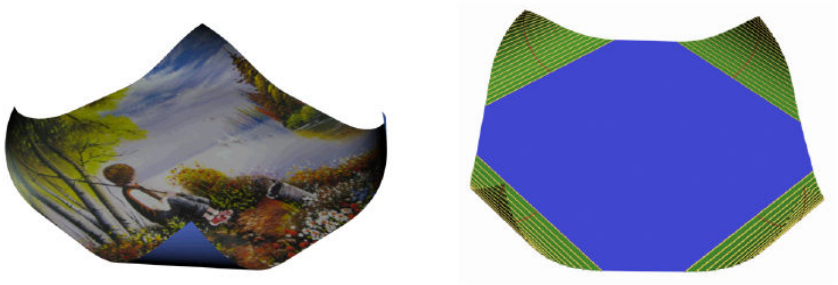


FIGURE 1.19: A smoothly developable surface from [BW07] modeling paper-like material.

### 1.3.2 Folding and crumpling paper using isometric deformations

Bo and Wang [BW07] model and animate paper-like material through a developable surface obtained by the isometric deformation of a given 2D pattern. The input is a 3D curve $\mathbf{p}$ with non-vanishing curvature representing a geodesic of the surface and its position (a line) on the 2D pattern. The normalized direction $\mathbf{r}(s)$ of a ruling going through the point $\mathbf{p}(s)$ is then given by:

$$\mathbf{r}(s) = \frac{\mathbf{p}''(s) \wedge \mathbf{p}'''(s)}{|\mathbf{p}''(s) \wedge \mathbf{p}'''(s)|}.$$

The intersection $\mathbf{c}(s)$ of a ruling and the paper surface boundary can be computed on the 2D pattern and its 3D coordinates are:

$$\mathbf{c}(s) = \mathbf{p}(s) + l(s)\,\mathbf{r}(s),$$

where $l(s)$ is the distance between the boundary and the curve $\mathbf{p}$ (see Figure 1.18). More complex smooth developable surfaces can be composed by joining several curved regions, computed as explained above, linked together by planar regions (the joining edges with a planar region need to be rulings of the curved regions) as shown in Figure 1.19. A paper-like surface can thus be modeled and manipulated by a user through the manipulation of the geodesic curve $\mathbf{p}$. The resulting surfaces are however smooth surfaces. The crumpling of paper with its typical sharp features is not proposed here.

Solomon *et al.* [SVWG12] and later Zhu *et al.* [ZIM13] create a developable surface by bending and folding an initially flat surface according to a folding pattern given by the user. Peraza Hernandez *et al.* [HHAL16] improve rigid origami simulation by joining two adjacent faces by a smooth flexible ruled surface instead of using an articulated edge. These methods enable to model smooth developable surface with eventually sharp linear folds explicitly prescribed by the user.

When crumpling paper, one can observe some creases that look like singular point. Huffman studies in [Huf76] how the surface of paper behaves near such singularities, in particular its convexity and trace on the Gaussian sphere. Kergosien [KGK94] proposes one of the first models to animate a developable surface defined by its boundary and a function associating each point on the boundary to another, each couple defining a ruling. He defines the need for creases, which he models either as a singular point or a singular curve, where some rulings cross each other. Frey [Fre04] proposes a buckled developable surface construction for 2D height fields using triangulation. Those curved creases can also be used to create elegant shapes for industrial design or origami as done by Killian *et al.* [KFC*08], who locally optimize to find the position of the faces adjacent to the crease, or Mitani *et al.* [MI11], who reflect the surface onto a plane to create a crease. For both of them the creases are defined on some initial simple smooth developable shape. For all these methods, the position of the creases are specified explicitly by the user. Another approach from Rohmer *et al.* automatically generates a developable creased surface from its 3D boundary curve [RCHbT11]. But this approach is not temporally consistent: a slight change in the boundary usually leads to a radical change in the shape. So it is not adapted for animation.

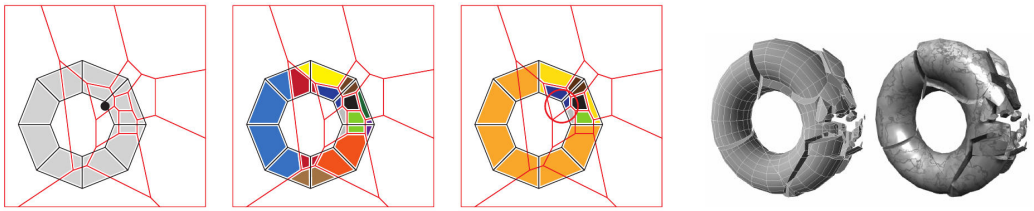


FIGURE 1.20: Pictures from [MCK13]. (left) Overview of the fracture algorithm, (right) fracture pattern applied to a 3D object.

### 1.3.3 Procedural methods for fracture

Currently there is no procedural or geometric method for tearing paper or thin shell known to us, but some previous works use procedural methods for fracturing volumetric objects.

For example, fracture patterns can be computed using fast procedural method, although the kinematics is still usually computed using a physical model. A popular method, particularly for explosion modeling, for cheap fracture computing is to use pre-fractured objects that will break along a pre-defined pattern. A more advanced idea proposed by Müller *et al.* [MCK13] is to align a fracture pattern with the impact location in order to subdivide at run-time an object composed of convex parts as shown in Figure 1.20. The fracture may also be spread from an initial location using a procedural algorithm as proposed by Neff *et al.* [NF99]. This kind of approaches enables fast fracture simulation

but cannot be applied to paper as the path of the tear explicitly depends on the motion applied to the paper.

FIGURE 1.21: Image from [CYFW14]. From left to right: the strength field of the material, the stress field centered on the fracture, the modified strength field.

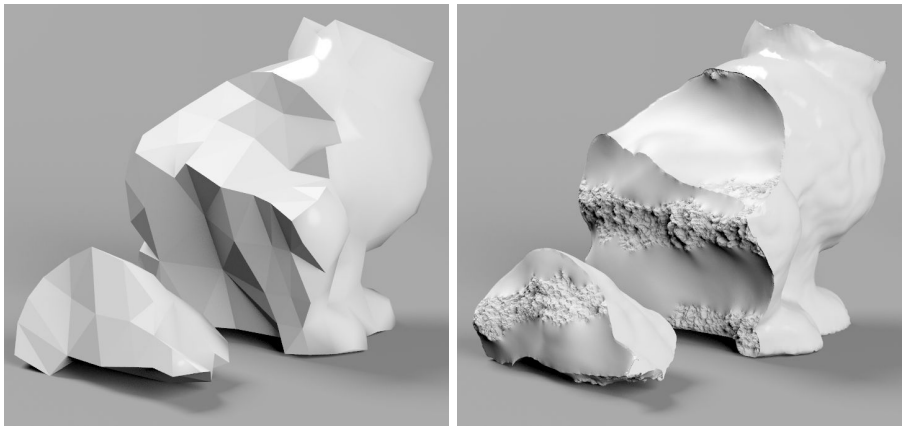


FIGURE 1.22: Example of results obtained by [CYFW14]. (left) The result of the coarse simulation; (right) the result after adding procedural details.

A mixed approach, proposed by Chen *et al.* [CYFW14], is to refine a 3D fracture animation, computed for example with a coarse physical simulation, by adding procedural details based on a 3D texture representing the strength field of the material (i.e. its ability to resist to fracture). The strength field is modified by subtracting the stress field (see Figure 1.21), centered on the fracture surface. The high resolution fracture surface is computed by minimizing the energy represented by the modified strength field. Highly detailed fracture of inhomogeneous material can thus be computed at low cost (see Figure 1.22). Still physical accuracy is limited by the coarse initial simulation, and the collision handling of the refined parts may prove to be problematic. Yet this concept seems to be well-adapted to represent small details of the torn fibrous structure.

Example-based fracture pattern can also be created by learning the statistics of patterns obtained from real data, as proposed by [GMD12].

### 1.3.4 Data-based methods for generating sounds

Real data are also used for sound generation in order to avoid the complex and costly process to simulate the physical phenomenon involved in sound generation. In the

traditional approach, still widely used in movies and video games, real sound effects are recorded and edited by Foley artists to match the visual appearance. The approach produces high quality results but is time consuming and cannot be used in interactive environments. The process can be automated for interactive applications by triggering the sounds by computer-generated events and rendering pre-recorded sound samples [TH92] (see Figure 1.23). Unfortunately, this approach lacks fine-grained variability and synchronization needed to interactively render the complex interactions occurring with paper manipulation.

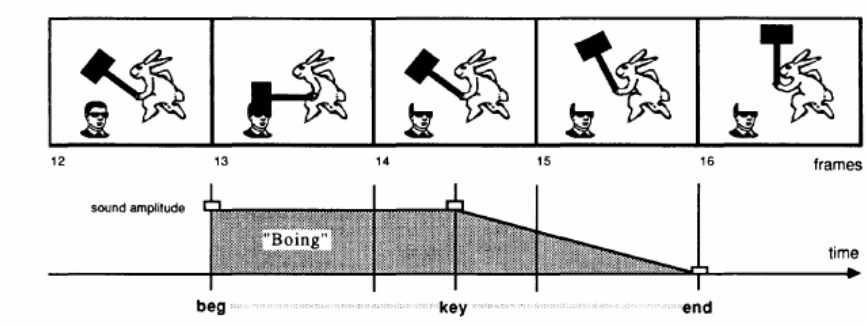



FIGURE 1.23: Image from [TH92]. The sounds are triggered by events like collisions.

Other data-driven methods are inspired by granular synthesis techniques [Roa04]. Dubnov *et al.* [DBJEY*02] and Bar-Joseph *et al.* [BJLW*99] propose to use wavelet trees to model sound texture and synthesize audio backgrounds. The "Sound-by-Numbers" method [CBBJR03] uses a granular synthesis model driven by a low-dimensional motion signal. Concatenative sound synthesis (CSS) [S*00] methods consist in selecting in a large database of sound units, the units that match the best the sound to be synthesized.

An *et al.* [AJM12] build on these methods to automatically synthesize the sound of a physically based cloth animation (see Figure 1.24). First they analyze the deformation to find crumpling and friction events that drive the synthesis of a low-quality target signal. A CSS process is then used to select the best units from a database of recorded cloth sounds. In Chapter 4 we take inspiration from the first part of this approach, and adapt it to paper material. In the case of paper, the sound is also produced by crumpling and frictional sliding mechanisms, but additionally the sound highly depends on the shape of the paper.

The sound of brittle elastic objects being fractured has been investigated by Zheng and James [ZJ10] (see Figure 1.25). They associate to each fragment an ellipsoidal proxy whose sound has been pre-computed. But this method assumes the fracture happens suddenly to divide a fragment into several pieces during a single simulation time step. There is therefore no transitional state where the fragment is only partially fractured,

FIGURE 1.24: Cloth animation and the spectrogram of the corresponding sound from [AJM12].

while in the case of paper, the tear should slowly propagate at the same time scale than the motion of the hands.

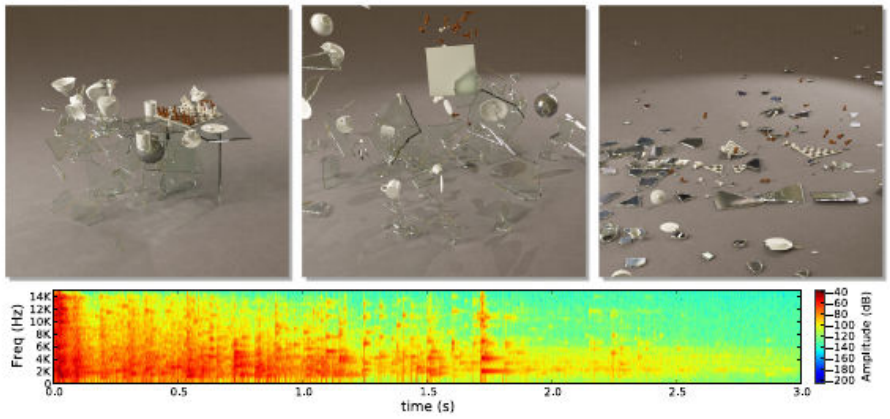



FIGURE 1.25: A glass smashed into over 300 pieces and the spectrogram of the corresponding sound from [ZJ10]

## 1.4 Conclusion

Finally, while physics-based simulations are usually too computationally expensive to allow interactivity, and geometric or procedural approaches still do not fully allow to represent the complex behavior of the paper, only few researches explore hybrid methods. By combining a usual physics-based simulation with new procedural and geometric methods taking advantage of prior knowledge, we aim at reproducing a plausible behavior of paper rather than an entirely physically accurate one in order to enable a user to interactively deform and create animation of virtual paper.

# Chapter 2

# Modeling crumpling paper

*"The lunatic is in the hall.*
*The lunatics are in my hall.*
*The paper holds their folded faces to the floor*
*And every day the paper boy brings more. "*
Pink Floyd, *Brain Damage*, 1973.

## 2.1 Introduction

We manipulate sheets of paper in our everyday life, when reading, writing or drawing, as well as for wrapping things up, and they often end up in a more or less advanced crumpled form. So far, most of the 3D virtual paper models are restricted to smooth sheets of paper, or sometimes even modeled as a rigid material. On the contrary, a more plausible model for paper animation should handle non-smooth developable geometry undergoing irreversible changes upon deformation in order to model its crumpling. While state of the art physically-based models are able to generate visually compelling animations of paper crumpling behaviors [NPO13], this is done at a high computational cost: no method is yet able to compute such deformations at interactive rates.

The goal of the work presented in this chapter is to achieve the animation of paper crumpling phenomenon at interactive rates. Our insight is to enforce the characteristic features of the paper –imposed by the microscopical fibrous structure– through a dedicated geometric model, enabling to relieve most of the time usually spent in stiff,

physically-based simulations. Let us list more accurately the main macroscopic characteristics which need to be captured in order to crumple a sheet of paper in a visually plausible way:

**Length preservation**. Paper has a very strong in-plane stiffness, i.e. it does not stretch nor compress under standard conditions of use. The surface model used for a sheet of paper should therefore be able to preserve lengths during deformation. In particular, although acceptable for animating cloth, the small elastic deformations between neighboring vertices typically allowed by deformable models should be prevented when modeling paper material. Therefore, a sheet of paper can always be isometrically developed onto the flat pattern representing its original state. This leads to the well known developablility property of paper.

**Sharp features.** Crumpled sheets of paper exhibit some sharp creases. While standard geometric representations assume that the surface to be captured is smooth, the representation we are looking for should be able to handle such discontinuities of the tangential plane.

**Plastic behavior and shape memory.** In real life, when the microscopic fibers of paper –or more accurately the bonds between them– break upon deformation, this causes irreversible damage to the structure. Therefore, all singularities and creases are persistent over time, even after trying to flatten the sheet. A dynamic model of paper must incorporate this persistence property, resulting in an increasing number of singularities throughout the animation.

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. The model is a piecewise developable surface made of generalized cones, it is designed to capture the sudden appearance of singular points during the deformation process. Its deformation is governed by both physical constraints and geometrical laws. The fact that only a small set of conical patches are used for representing geometry reduces computational cost, since it enables simulation to be performed on a sparse 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 handling the generation of tangent discontinuities. Its 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 sharp features are explicitly tracked throughout the animation process and are used to model physical plastic behavior such as non-planar rest pose.

**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 enables an accurate representation of isometric surfaces with sharp features using only a small number of triangles instead of using some dense mesh subdivision.

**Validation methodology through comparison to real paper.** We use experiments with real paper to validate our method: we quantitatively compare the location and time of appearance of surface singularities in some simple crumpling configurations. In addition to these statistical results, we provide visual comparisons with videos of real paper, enabling us to show that our method generates the same first few folds when a sheet of paper is crumpled.

*The work presented in this chapter has been published in ACM Transaction on Graphics (TOG) [SRH*15] and presented at SIGGRAPH 2016. It has also been presented at the "Journées du Groupe de Travail de Modélisation Géométrique" (Days of the work groups of geometric modeling) in April 2015 and at the conference WomEncourage in September 2015.*

## 2.2 A new hybrid model for paper material

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

### 2.2.1 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 typically leads to the use of dense meshes, which provide a large number of degrees of freedom

and reduce the visual artifacts due to badly oriented edges. However, 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 using 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. A coarse physical simulation, interwoven with this geometrical model, guides the 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 maintain an exact isometry to their 2D pattern, i.e. their configuration before deformation. Therefore, they are also exactly developable, which implies a zero Gaussian curvature. Gauss' Theorem Egregium states that the Gaussian curvature is an intrinsic invariant of a surface. A consequence is that developable surfaces that are curved in two directions must be strained. In addition, when a sheet of paper is bent leading to a non-zero principal curvature, the second curvature in the direction of the fold is necessarily zero. The surface thus becomes rigid in the direction orthogonal to the fold, forcing it to break rather than bend if an orthogonal force is applied. The surface then necessarily loses smoothness. Indeed, $C^2$-developable surfaces can be defined as ruled surfaces whose tangent plane is constant along a ruling. They were shown to be pieces of planes, cylinders, cones or tangential surfaces that join with tangent plane continuity along some common rulings [SVWG12]. These rulings being straight lines, a sheet of paper cannot be bent smoothly simultaneously in several directions. In consequence, forcing some general bending on a sheet of paper, as in Figure 2.1, leads to the appearance of at least one singularity 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 [CM11]. [Wit07] further explains that thin sheets have the specific behavior of concentrating elastic energy when being constrained. When the thickness of the thin sheet tends to 0, this energy concentrates into singular points, which are called d-cones. A quantitative investigation confirms that large deformations of thin elastic plates lead to the formation of singular structures which are either linear (ridges) [LW97] or point-like (d-cones) [AP97, MC98, CM98]. These generic structures are reflected in the branched network of vertices and sharp folds in a crumpled paper. Outside of the influence of the d-cones, the surface is smooth, exhibiting a low distribution of elastic energy [SKD11]:

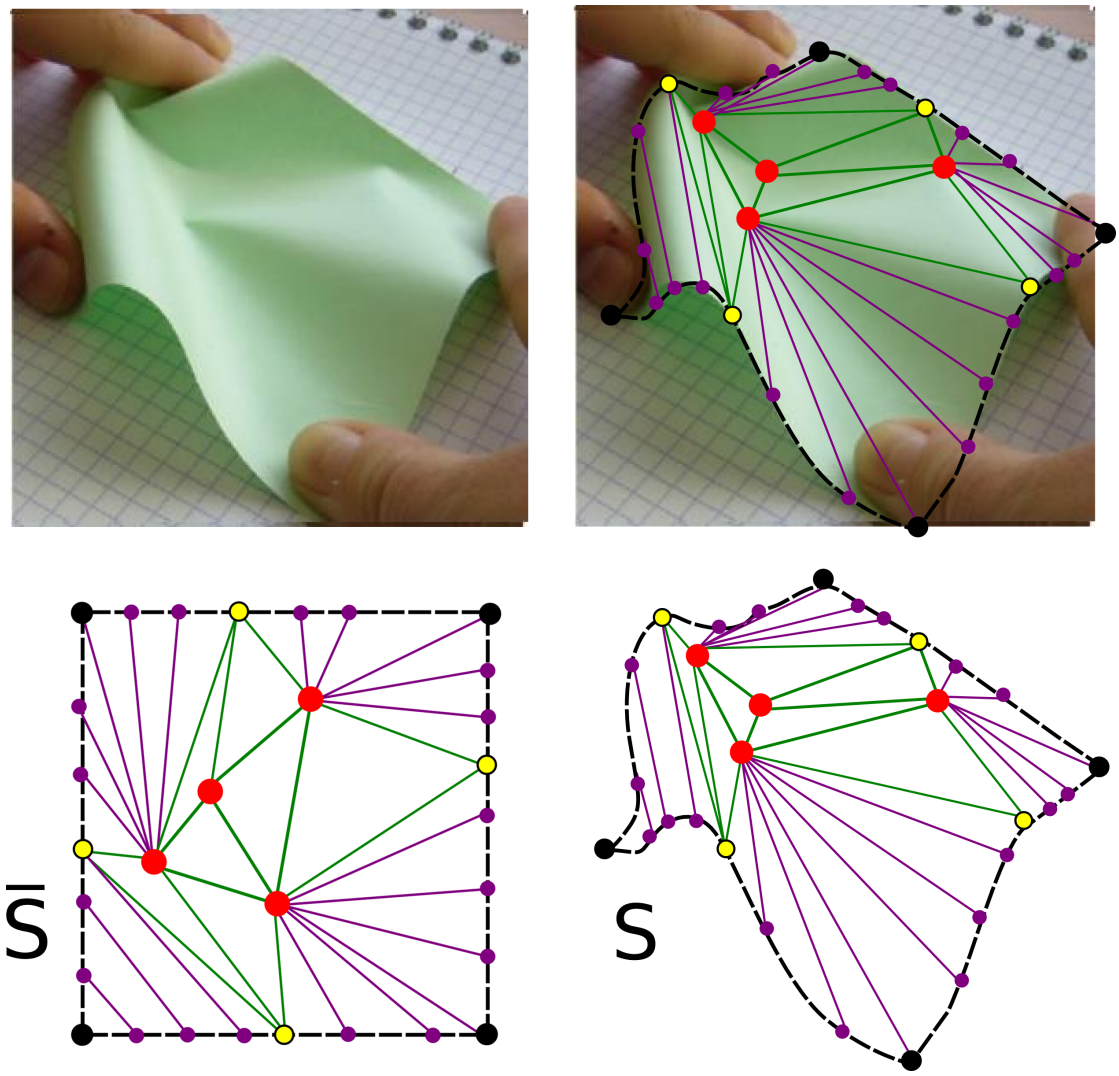



FIGURE 2.1: **Geometric structure.** Singular points appear when a sheet of paper is bent in several directions (photograph). Our surface representation (pattern on the bottom left, and 3D surface on the bottom right) includes flat areas (green triangles) and curved generalized cones (magenta triangles and quadrangles), whose apex is a singular vertex (red dots). Flat areas are either defined between singular vertices, or between singular vertices and junction vertices (yellow dots) separating flat and curved regions on the boundary. The user directly manipulates the model through handles (black dots) on the boundary.

it deforms accordingly to classical linear models of continuum mechanics laws. These *d-cones* denote developable generalized cones defined by an apex and a one-parameter family of straight lines (rulings) through a fixed point called the *apex*. Figure 2.1 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).

Based on these geometrical properties and physical observations, we propose the following paper geometry:

• The surface is defined as a parametric piecewise continuous developable surface composed of generalized cones (see the magenta rulings in Figure 2.1) and planar pieces (in green in Figure 2.1). Note that this structure is similar to the one use by Frey [Fre04].
• The apices of the d-cones are modeled as specific vertices that we call *singular vertices* (red dots in Figure 2.1). We track the position of these vertices on the 2D-pattern and on the 3D shape of the deformed sheet.
• The deformation of the surface is guided by continuous mechanics in between plasticity events, i.e. the creation of new singularities due to the breaking of fibers.

Animating the resulting 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 is especially well adapted to handle these constraints, since the singular vertices are tracked explicitly and govern the generation of the smooth surface parts in-between. The animation algorithm is described next.

### 2.2.2 Overview of the animation algorithm

**Geometric representation for paper:**

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 2.1) and a set of flat regions $\mathscr{F} = \{F_i\}$ (in green in Figure 2.1).
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 2.1). Second, one vertex lies on $\partial S$

and the other is an interior singular vertex (magenta triangles in Figure 2.1), 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 dihedral angle between any pair of adjacent triangles of a same flat region is lower than a threshold, $\theta_0$. All thresholds and user-defined parameters are summarized in Table 2.6.

The interior vertices of $S_G$ (red in Figure 2.1) are singular points of the surface $S$. They are the apices of developable cones. The yellow vertices of $S_G$ in Figure 2.1 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 2.1) 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 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 2.3. 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 2.1.

| 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 $F$ |

TABLE 2.1: **Summary of symbols used.**

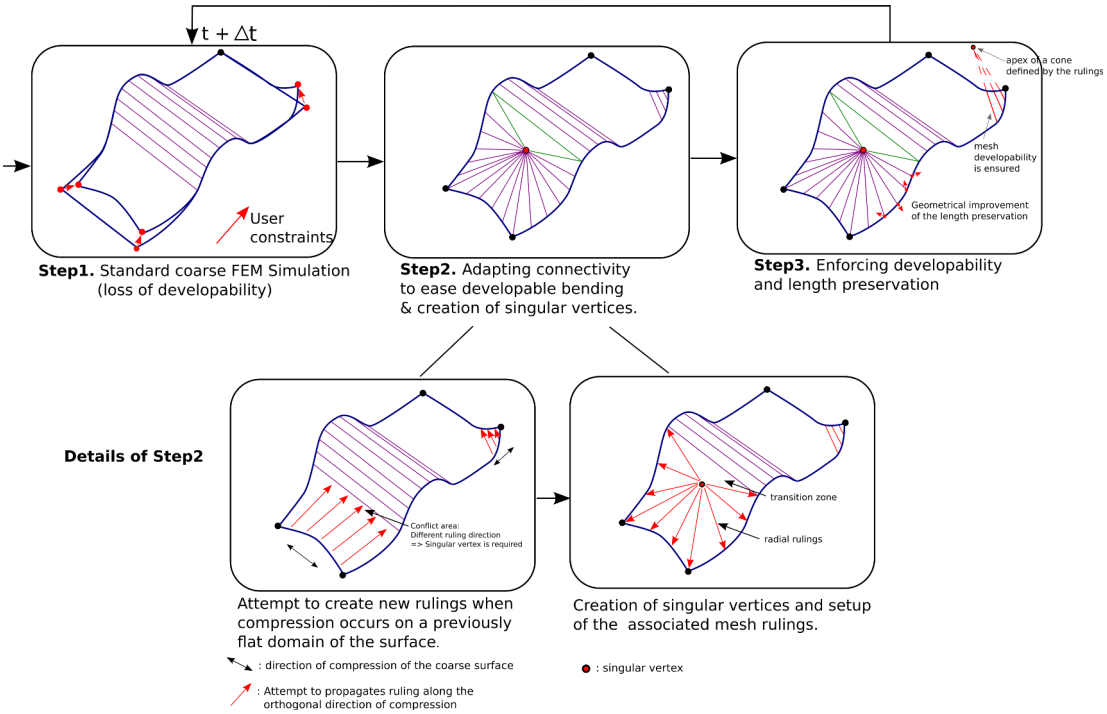



FIGURE 2.2: **Overview of the algorithm.** Each animation step combines a standard FEM simulation (step 1) with a remeshing step (step 2) that approximate the surfaces using generalized cones and mesh it accordingly, notably by finding the position of singular points and enforcing them as apex of cones. Developability is improved in step 3.

Our algorithm develops as follows:

**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 dealt with. Additionally, some *handles*, i.e. a set of points that the user can manipulate, are defined. They act as hard constraints and are used to govern the deformation.

**Deformation 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 the surface $S$ with two geometry-based steps, namely a geometrical analysis implying the eventual creation of singularities and the update of the piecewise developable paper geometry (see Figure 2.2). These interweaving steps consist of:

**Step 1. Elastic deformation.** (Section 2.3)

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 get lost. This step outputs new positions of the vertices of $S_G$.

**Step 2: Modeling bending and crumpling** (Section 2.4)

- *Remeshing of the compressed planar parts.* Some of the flat regions are remeshed according to locally measured compression 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 2.2-middle.

**Step 3: Developable and isometric tracking** (Section 2.5)

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 2.2-right.

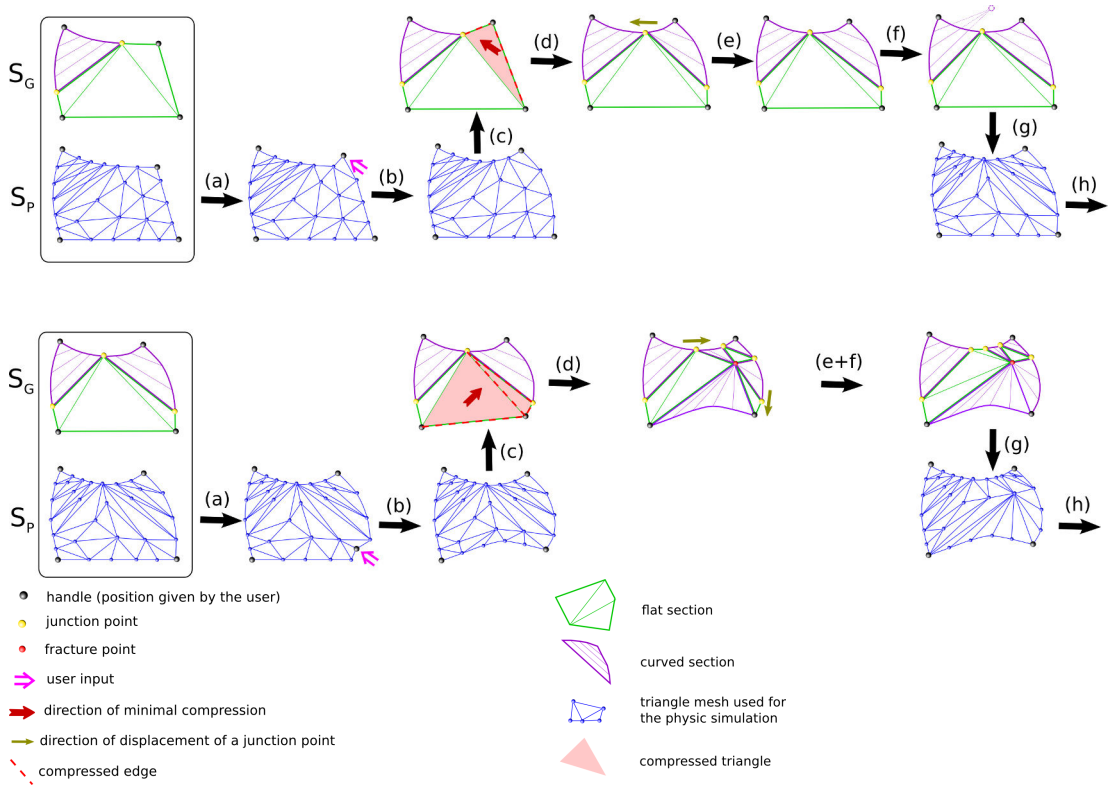
Throughout the description of our method, we will refer to Figure 2.3, which shows another general example case spanning two successive steps of the algorithm loop.

## 2.3 Physically-based deformation

### 2.3.1 Physical simulation

The first step in the animation loop consists in computing the deformation of the sheet of paper using a standard physical-based simulation method, based on elastic energy. More precisely, our implementation makes use of the simulation code provided by Narain et al. [NSO12]. It relies on Green strain computation for stretching forces, while bending forces are computed using *discrete flexural energy* [BMF03, GHD*03]. The 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 [TMT10].

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 neighboring vertices – between two and four fixed vertices




(a) : displacement of the handles according to the user input.
(b) : physic relaxation of the triangle-mesh (Section 2.3.1).
(c) : updating of the position 3D of the point of the structure.
(d) : flipping the compressed edges (if possible) for each flat area (Section 2.4.1).
(e) : creation of curved area from compressed flat areas (Section 2.4.2) and generation of fracture points if needed (Section 2.4.3).
(f) : displacement of the junction points (Section 2.5.1).
(g) : update the rules of each curved area (Section 2.5.2).
(h) : generation of a triangle mesh from the structure (Section 2.3.2).
(i) : displaying the triangle mesh on screen.

FIGURE 2.3: **An example of two successive animation loops.**

around the handle defined the tangent plane according to whether the handle is in a corner, a border of the paper or inside the surface. 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 2.3 illustrate the user-defined deformation followed by a physics-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 leads to specific constraints when setting up the simulation, described next.

### 2.3.2 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 2.4-left) enables us to obtain 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 us 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 inside a same flat region are connected using Delaunay triangulation.

Note that an edge of $S_G$ between two flat regions has a higher dihedral angle and can then be considered as the one ruling of a degenerate cone. Thus such an edge is not sampled and is then also an edge of $S_P$ (see for example Figure 2.5). In this way, edges between two flat regions will be sharper than edges inside a flat region.

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 2.4. Note that these long triangles aligned with folds increase the stiffness in this direction as a natural side effect, which correlates well with real paper behavior.

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 step (g) in Figure 2.3. $S_P$ is then used for simulation at the next animation step (steps (a) and (b)). Finally, the new position of its vertices are used to update the position of the $S_G$ vertices (step (c)).

### 2.3.3 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

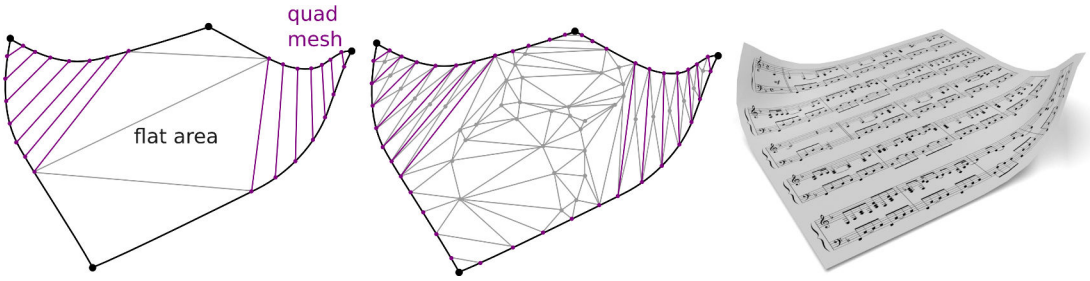



FIGURE 2.4: **Creation of the physical triangle mesh structure** $S_P$ (middle) from the geometrical mesh structure $S_G$ (left). The added vertices and edges in $S_P$ are shown in gray.

crumpled and contains singular vertices, the surface should not come back to such a flat configuration by itself, as the paper fibrous structure has been damaged. We model this plastic behavior using a non-zero rest angle for the bending force and by weakening the resistance to bending around the interior 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:

$$\theta_i = \begin{cases} \theta_{rest}/(N_i^{\text{sing}}) & \text{if the vertex } i \text{ is a singular vertex} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

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.

In the same way, we weaken the bending stiffness around a singular point by associating a damage parameter $d$ to each edge. For each vertex $i$ we define:

$$d_i = \begin{cases} D/(N_i^{\text{sing}}) & \text{if the vertex } i \text{ is a singular vertex} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

and set the damage parameter of edge $(i, j)$ to

$$d = d_i + d_j.$$

The bending stiffness for each edge is then the general bending stiffness set for the simulation multiplied by the corresponding factor $\frac{1}{1+d}$ (which is conveniently 1 when the damage is null).

We experimentally set the parameters $\Theta_{rest} = \pi/8$ and $D = 10$ while trying to match the behavior of standard sheets of paper (70 $g/m^2$). Those values could be tuned to model thinner or thicker paper material. Note that those formula aim at procedurally modeling the effect of the damage of the structure of the paper with sharp singular vertices (and sharp edges between two singular vertices) that stay marked even after releasing all the constraints. Still, note that such formula may not accurately reflect the involved physics.

## 2.4 Modeling bending and crumpling

**Algorithm 1:** Geometric step

```
Data:
list of curved_regions : curved_list ;          // list of current curved regions
list of flat_ region: flat_list ;               // list of current flat regions
for cr ∈ curved_list do
    update_junction_points (cr)                          ▷ See Section 2.5.1
    if is_null (cr) then
        rm cr from curved_list
    end
    update_ruling (cr)                                   ▷ See Section 2.5.2
end
for fr ∈ flat_list do
    flip_edges (fr)                                      ▷ See Section 2.4.1
    treat_compression (fr, out list_of_new_curv_r, out list_of_new_flat_r)
                                          ▷ See Section 2.4.2 and Section 2.4.3
    push list_of_new_curv_r into curved_list
    push list_of_new_flat_r into flat_list
    if is_null (fr) then
        rm fr from flat_list
    end
end
```

We restrict our surface $S_G$ to deform with bending and crumpling instead of stretching. This is achieved by adapting the connectivity of $S_G$ while still keeping a very coarse triangulation, thanks to our hybrid model taking into account the specific developability properties of paper.

In the following, we consider two adjacent triangles to be 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 $\leq \theta_0$).

### 2.4.1 Flipping edges in the flat regions

As explained before, a flat region $F$ has two different representations: 1) it is represented by a set of triangles $\mathscr{T}_F$ whose vertices are either finger points, i.e. fixed points representing the positions of fingers, or (interior or boundary) singular points (in black on Figure 2.5) for $S_G$, 2) a denser triangulation (in gray) is done by adding isotropically sampled vertices for the physical representation $S_P$. The denser triangulation of $S_p$ is used for the simulation step so the sparse triangulation of a flat region in $S_G$ does not matter for the simulation step. This is why we rather choose this triangulation in order to ease the detection of the separation between two flat regions. Indeed if the dihedral angle between two triangles of $\mathscr{T}_F$ becomes $\geq \theta_0$, $F$ must be divided into two new regions.

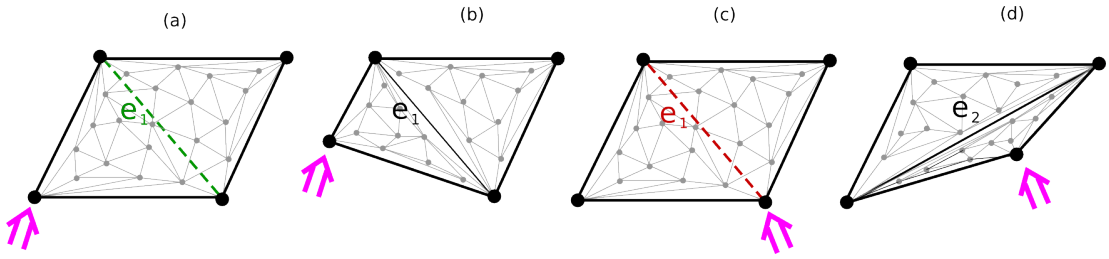



FIGURE 2.5: **Swapping of an edge.** In order to divide a flat region on the right edge to handle a fold, the edges representing a flat region in $S_G$ may need to be flipped.

This is explained by the example shown in Figure 2.5. A sheet of paper is held horizontally flat by four fingers on the four corners. The surface is only composed of one flat region $F$ (see Figure 2.5 (a)). The left bottom corner is then lifted upward. We detect that the green edge $e_1$ has a dihedral angle $\theta > \theta_0$ so $F$ is divided into two new flat regions separate by $e_1$ (Figure 2.5 (b)). Going back to the flat configuration of the sheet, if we rather lift the bottom right corner, $F$ needs again to be divided in two, but $e_1$ does represent this separation well, it will be compressed but its dihedral angle will stay low (Figure 2.5) (c)). This is why we need to detect such a situation and flip edges if needed (Figure 2.5) (d)) in order to be able to model the division of a flat region in any direction. This is done in the following way.

To measure the compression of an edge $e$, we consider the deformation of its length

$$c_e = (\bar{l}_e - l_e)/\bar{l}_e,$$

where $l_e$ and $\bar{l}_e$ are the length of $e$ belonging respectively to $S$ and $\overline{S}$.
We consider $e$ as compressed if $c_e$ is greater than a user defined threshold $\varepsilon_c$ (between $10^{-3}$ to $10^{-2}$ in our examples). We flip en edge $e_1$ between two triangles of the same flat region when $e_1$ is compressed and the alternative edge $e_2$ is not.

Note that it is important to find the edges which separate two flat regions because, contrary to edges inside a flat region, they are also edges of $S_P$ and have a notable influence on the physical simulation and also on the visual representation.

### 2.4.2 Analyzing mesh compression to generate bent surfaces

Most of the time, flipping edges does not enable to fully prevent triangles in a flat region from getting compressed. In such case, a planar part of the paper must be able to bend in the direction implied by the compression. Note that such direction may be arbitrary within the triangle plane. The refined triangulation (see Section 2.3) of the planar regions enables the physical simulation to bend them slightly when compressed. But in order to enable them to bend further in a smooth way, we need to adapt the geometry of the compressed region.

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 curvature) are inserted into $S_G$ along the direction of minimal compression (see Figure 2.6). 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, with possibly the insertion of new singular vertices (see Section 2.4.3).

#### 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.* [RPC*10] which computes the principal directions of strain:
Given a triangle $t$ (respectively $\bar{t}$) from the 3D mesh $S$ (respectively $\overline{S}$), let $(\mathbf{e}_0, \mathbf{e}_1)$ and $(\overline{\mathbf{e}_0}, \overline{\mathbf{e}_1})$ be the 2D edge vectors of the shape in the local frames of $t$ and $\bar{t}$. Let $\mathrm{F} = [\mathbf{e}_0\ \mathbf{e}_1][\overline{\mathbf{e}_0}\ \overline{\mathbf{e}_1}]^{-1}$, be the 2x2 transformation matrix and $\mathrm{U} = (\mathrm{F}^T\,\mathrm{F})^{1/2}$ the stretch

tensor. If the triangle $t$ is compressed with respect to $\bar{t}$ in at least one direction, then the largest Eigen vector $\lambda$ of $U$ with $0 < \lambda < 1$ indicates compression and the corresponding Eigen vector $\mathbf{v}$ the direction of maximal compression. ($\lambda = 1$ when $t$ and $\bar{t}$ are congruent, $\lambda > 1$ indicates extension: note that we do not treat such cases in this chapter; Tearing being handled in Chapter 3.) The maximal compression ratio of $t$ can thus be measured by $1 - \lambda$.

We call $T_{\text{comp}}$ the triangle of $\mathscr{T}_F$ (set of triangles of $F$) of the largest compression ratio (red triangle in Figures 2.10,2.11,2.12(a)). This triangle should actually not belong to a flat region anymore but should bend during the next simulation step. To encourage this bending behavior, our algorithm remeshes the surface locally such that the new edges align as well as possible with $\mathbf{v}^\perp$, i.e. the direction of minimal compression, orthogonal to $\mathbf{v}$, see Figure 2.6. This remeshing step (corresponding to Section 2.4.2 and Section 2.4.3) is then repeated until there is no flat region with a highly compressed triangle anymore.

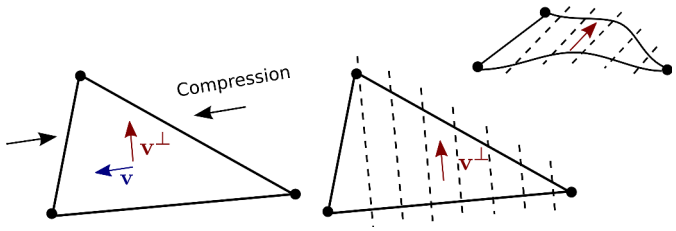



FIGURE 2.6: **Bending of a triangle.** We remesh a compressed triangle from $S_G$ using rulings in the direction of minimal compression $\mathbf{v}^\perp$, to favor uni-directional bending in subsequent simulation steps.

We now detail the actual remeshing process relying on a propagation algorithm computed in the 2D pattern space. 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). The idea is to compute the region affected by the bending of $T_{comp}$ by propagating endpoints of imaginary rulings aligned along the direction of minimal compression $\mathbf{v}^\perp$ in both directions until reaching the boundary of $F$.

To achieve this, we use a method divided in two steps: First, as explained in the paragraph (Step 1), we use an iterative algorithm in the 2D pattern space to compute the prolongation of the imaginary rulings generated by the compression of $T_{comp}$. This allows us to determine a strip of triangles of $\mathscr{T}_F$ affected by the bending of the $T_{comp}$ and the two borders which are the limit of the rulings of the new curved region. To obtain 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 an already constrained edge, in which case the border is the interior singular point created to solve the conflict.

Second (see the paragraph (Step 2) for more details), 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.

**(Step 1) Computing a triangle strip aligned with $\mathbf{v}^\perp$:** this algorithm takes the compressed triangle $T_{comp}$ and the direction of minimal compression $\mathbf{v}^\perp$ as input. It computes and returns a triangle strip composed by triangles affected by the compression of $T_{comp}$ and the two borders of the strip. We call *border* the edges of triangles in $\mathscr{T}_F$ where the triangle strip ends. This *border* is either a part of $\partial S$ (Figure 2.8(b)), or an internal ruling (Figure 2.8(c,d)).

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 2.7-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.

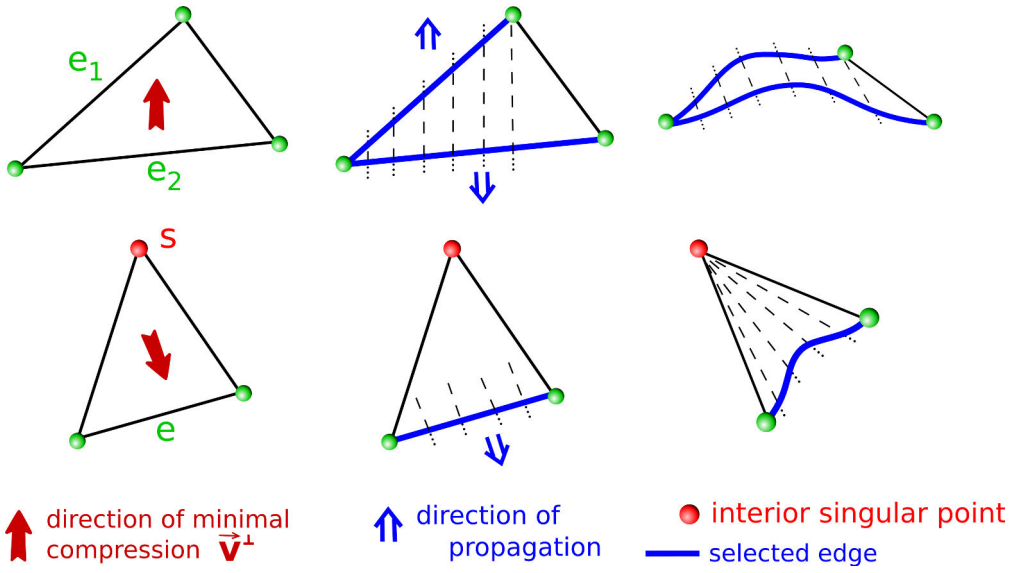



FIGURE 2.7: **Initialization of the propagation method.** New edges (rulings) inserted in the direction of minimal compression make a triangle likely to be bent as a conical surface during the next simulation step. Top: triangle without singular point. Bottom: triangle with singular point.

A particular case arises when the edge $e$ of $T_{\text{comp}}$ with the largest angle to $\mathbf{v}^\perp$ is opposite to a singular point $s$ (see Figure 2.7-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 largest angle with $\mathbf{d}_{\text{prop}}$, see Figure 2.8(a), and continue the propagation with the next iteration. Note that of the other edge of $T$ may also be affected by the compression, in this case, a following step of remeshing will affect it.

- 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 2.8(b).

- if $e$ is common with a triangle $T$ belonging to another flat region (the dihedral angle of $e$ being larger 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 2.8(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 2.8(d).

The last two cases (Figure 2.8) (c) and (d)) lead to the main contribution in this section, namely the insertion of new singular points. Indeed 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. The technical details related to singular points generation are given in Section 2.4.3.

**(Step 2) Remeshing $F$ locally:**

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 2.9. 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 $\mathbf{v}^{\perp}$ and are delimited by an endpoint on each border, see Figure 2.9(a). This configuration also occurs in the example shown in Figure 2.10.

- 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 $e_b$. So the rulings actually

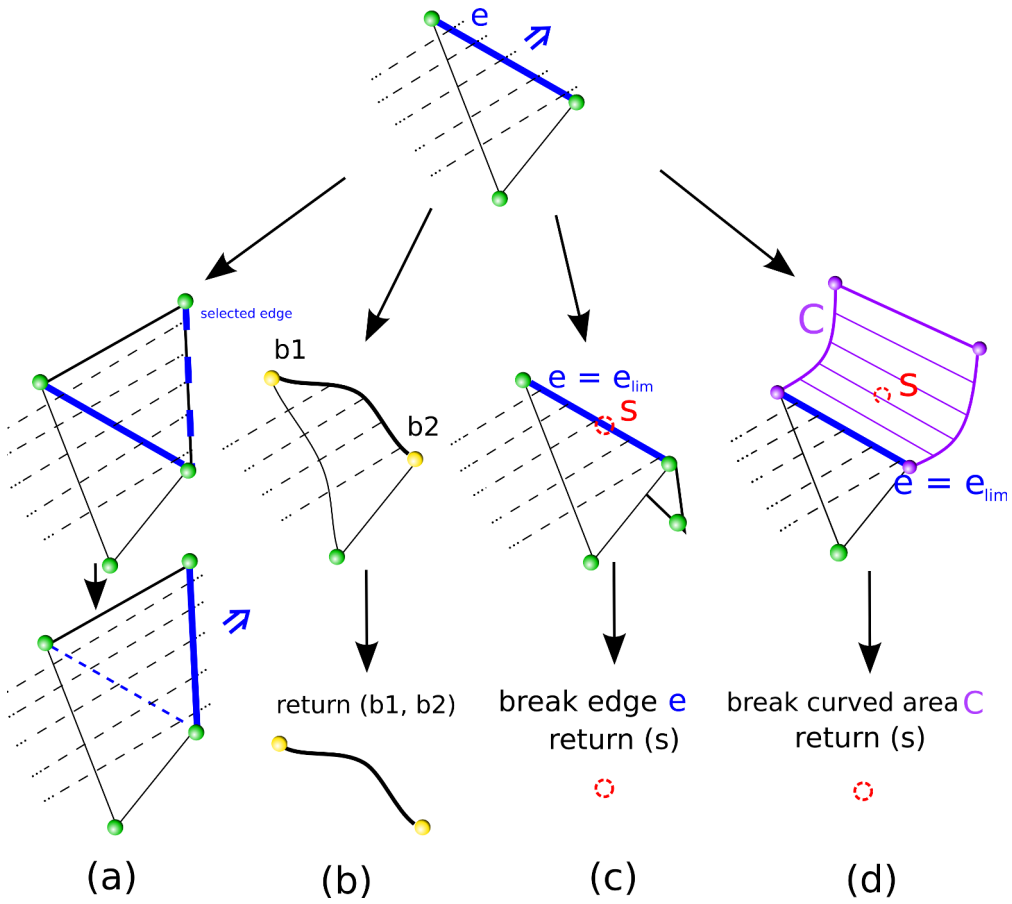



FIGURE 2.8: **One step of the propagation algorithm.** The four possible cases of the imaginary rulings propagation algorithm. The edges met during propagation may respectively belong to: another triangle of the same flat region (a), $\partial S$ (b), another flat region (c), a curved region (d).

inserted are conical instead of parallel to $\mathbf{v}^{\perp}$, see Figure 2.9(b). Another example is shown in Figure 2.11.

- 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 2.9(c). Figure 2.12 shows such an example.

In the rare cases where we cannot find two rulings $r_{\text{lim}_1}$ and $r_{\text{lim}_2}$ without any 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

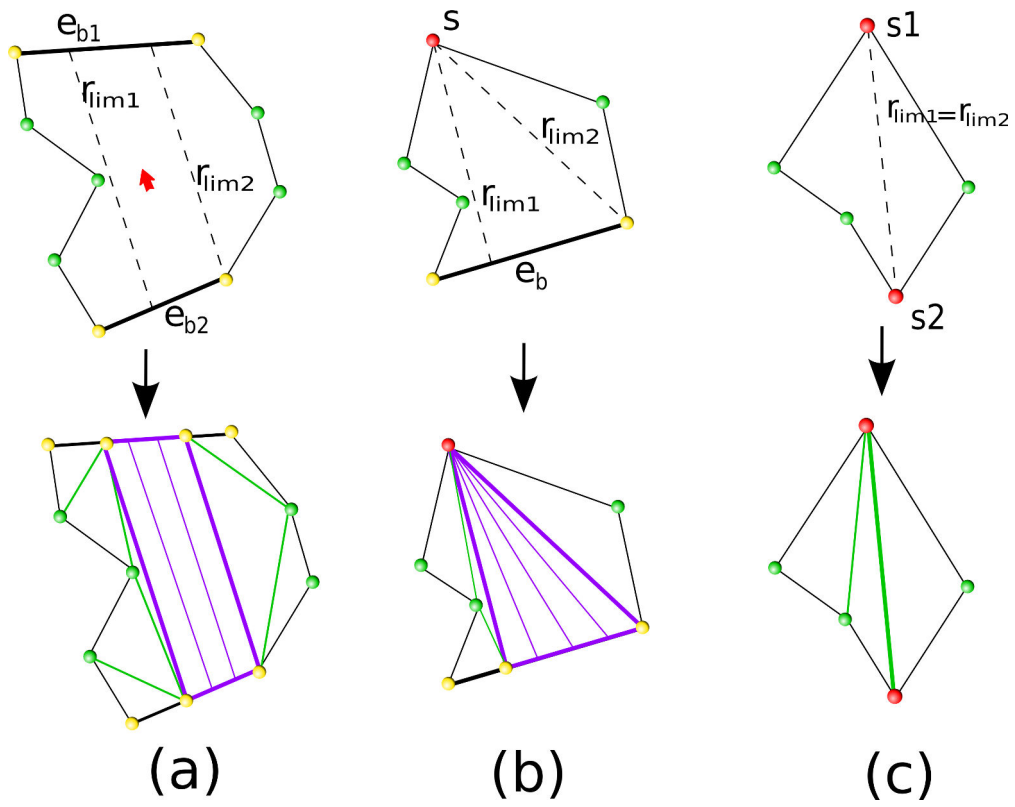



FIGURE 2.9: **Remeshing locally.** The propagation algorithm returns two *borders* – either a point or an edge – of the region that needs to be remeshed. This gives us three cases: (a) both borders are edges, (b) one edge and one singular point, and (c) two singular points.

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 2.10 (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 2.10 (bottom right)) with rulings inserted as explained in Section 5, can bend more easily and therefore better matches the shape of the input $S_P$.

The whole process is also represented in Figure 2.3(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.

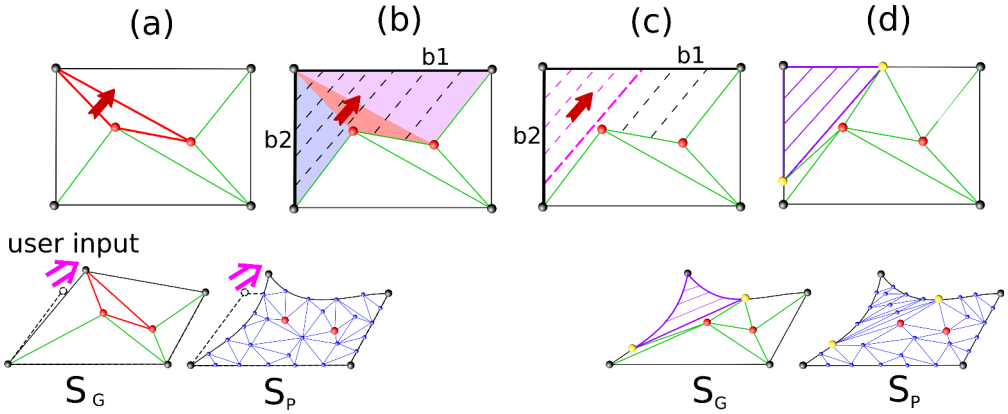



FIGURE 2.10: **An example of remeshing without creating any singular point.** Generation of rulings to favor further bending when the propagation reaches the boundary $\partial S$ at both ends. (a) The triangle of maximal compression is marked in red. (b) Propagation of triangle strip with imaginary 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.

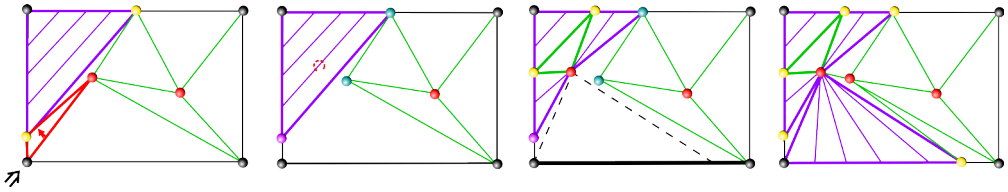


FIGURE 2.11: **An example of remeshing with the creation of one singular point.** (a) compressed triangle in flat region. (b) The imaginary rulings of the triangle strip encounter a non-constrained border (fat black) at one end and a constrained existing ruling (fat purple) at the other end. (c) A singular vertex is created and (d) conical rulings are inserted in $S_G$.

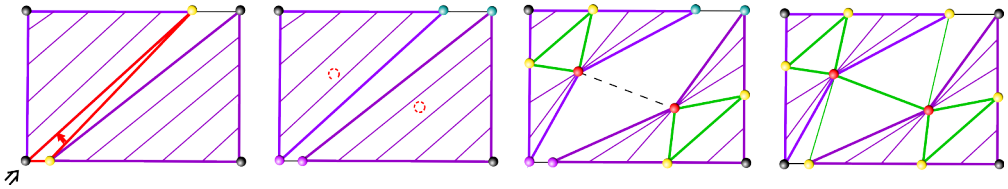


FIGURE 2.12: **An example of remeshing with the creation of two singular points.** (a) compressed triangle in flat region. (b) Imaginary rulings and 2 constrained borders (fat purple lines). (c) 2 singular points and a ruling between them are then inserted. (d) Final meshing of $S_G$.

### 2.4.3 Singularity generation

As stated previously, a piece of surface that starts to bend generates rulings that propagate throughout the whole surface. When these rulings intersect another preexisting ruling, see Figure 2.8 (c) and (d), defined in our mesh by an edge called $\mathbf{e}_{\text{lim}}$, we introduce a new singular vertex to accommodate for the two non-compatible constraints.

**Finding the singularity position.**
The formation 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 sheets of paper of the same thickness and under similar constraints.

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 2.8(d)). In order to derive a law for placing the singular vertex, we repeated many times the same simple experiment shown in Figure 2.13 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 2.6.1 and Figure 2.20 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 at a random position close to the edge $\overline{e}_{\text{sing}}$ as illustrated in Figure 2.13 (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 2.8(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 $\mathbf{e}_{\text{sing}} = \mathbf{e}_{\text{lim}}$.

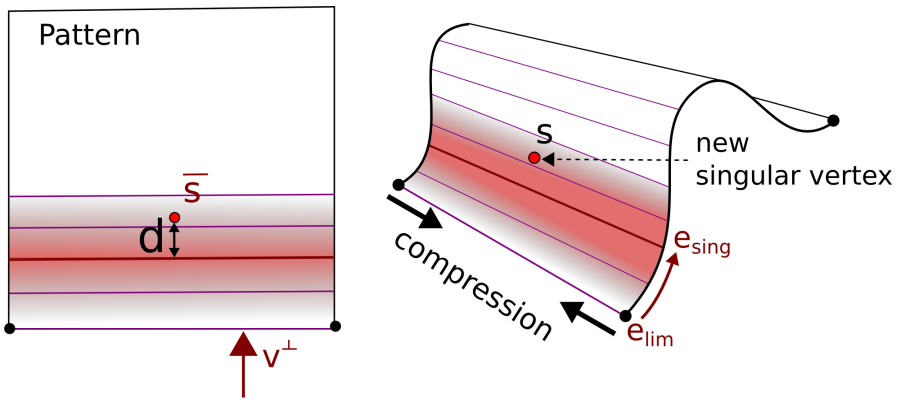



FIGURE 2.13: **Computing the position of a singular point.** A configuration where compression acts on top of an existing curved domain. The singular vertex to be inserted has a higher probability to appear in the red region. The edge $\mathbf{e}_{\text{sing}}$ at the center of this region is highlighted in red.

**Remeshing the pattern and the surface around the singularity.**
The singularity $\overline{p}_s$, i.e. the apex of a d-cone, is inserted when the propagating rulings of the flat surface region meet the ruling of an existing surface part. We have already described in Section 2.4.2 how we set actually the newly introduced rulings of the flat surface. In this section, we describe how we change the existing rulings of the already bent surface.

The introduction of the singular point into a curved area also makes the closest rulings of this area converge toward the singularity. Changing arbitrarily the direction of the rulings is only possible in the flat regions, i.e. as long as the dihedral angle of the rulings is smaller than $\theta_0$. In practice, when adding a singular point $s$ into a curved area $C$, we suppress the rulings of $C$ in the direction $\mathbf{v}^{\perp}$ from $e_{\text{sing}}$ until reaching the next rulings to have an angle superior to $\theta_0$ –or the opposite border of $C$– that we call $e_{\text{sup}}$, and we create new curved areas, which are represented by a single generalized cone whose apex is $s$, between the old position of the border ruling and the new one as show in Figure 2.14 left and middle. To enable geometric continuity between the newly remeshed region and the existing bent surface, we introduce a transition surface as a flat surface (green triangle in Figure 2.14 left and middle).

In the case of a singular point created between two triangles (Figure 2.14 right), the common edge will be considered as a degenerate curved area with only one ruling.

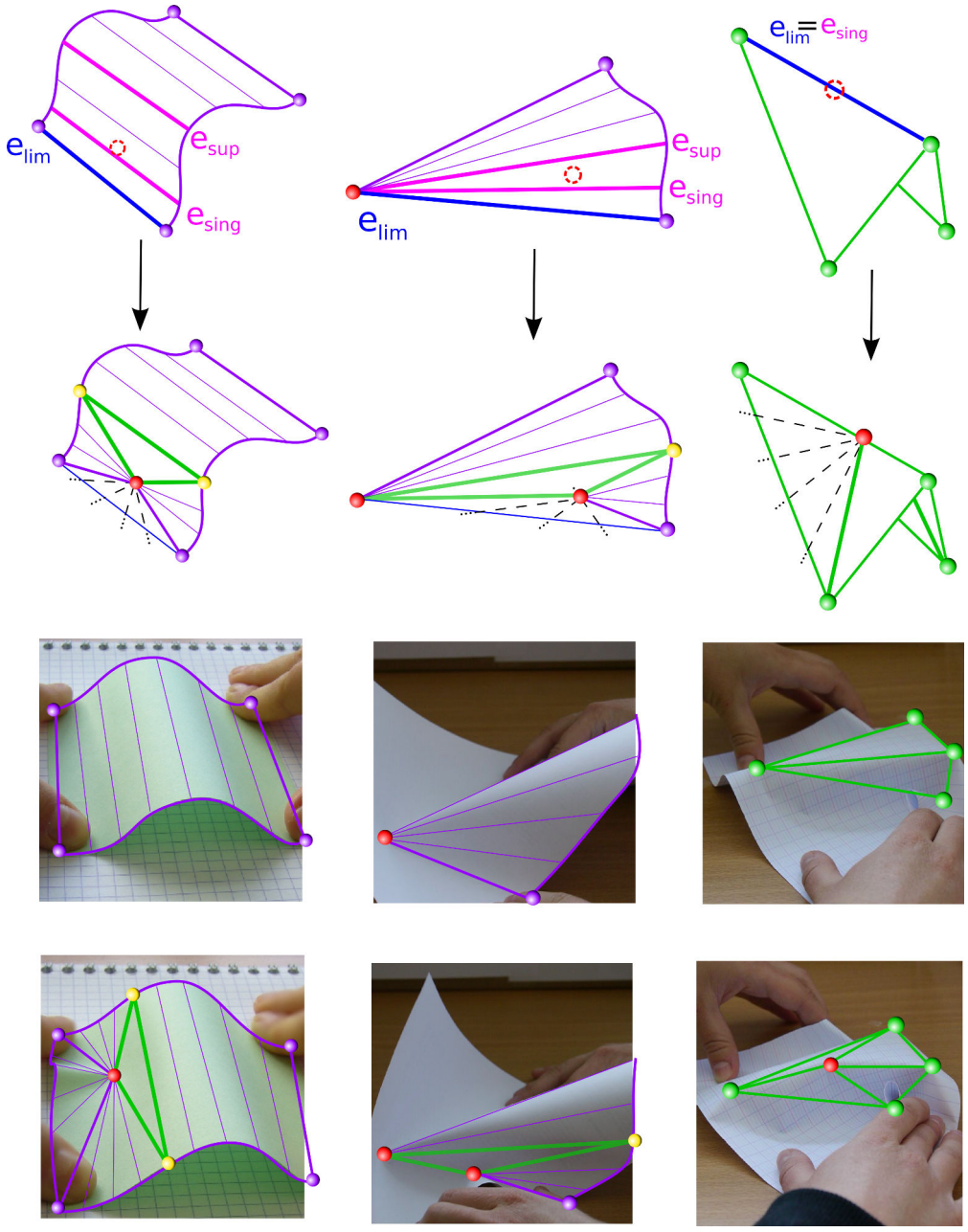



FIGURE 2.14: **Newly inserted singularity and associated remeshing.**

## 2.5 Developable, isometric tracking

The last step of our method consists in computing a fully developable approximation of the current geometric mesh $S_G$ and ensuring 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 [LP98, JKS05], 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 2.5.1) and then segment the latter into generalized cones, providing us with rulings for edge alignment (Section 2.5.2). We then apply a new method, introduced in Section 2.5.3, 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.

### 2.5.1 Finding the limit between flat areas and curved areas

*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 Figure 2.1, Figure 2.11 and Figure 2.12). They can be considered singular as they are points where rulings (from different regions) cross. Yet these points do not model a permanent plastic effect: therefore, contrary to interior singular vertices, which represent actual damage in the fibrous structure of the paper, 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 2.3.

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 2.15-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 2.15-right). If $F$ is also adjacent to another planar region $F_{\text{opp}}$ or to another curved region $C_{\text{opp}}$ with the same junction point, we allow $C$ to expand only if $\theta_{\text{CF}} > \theta_{\text{opp}}$ (see Figure 2.16).

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, such that the 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.

In order to maintain the consistency of the geometric model, we also prevent a junction point from moving if it would cause an interior singular point to get inside a curved region.

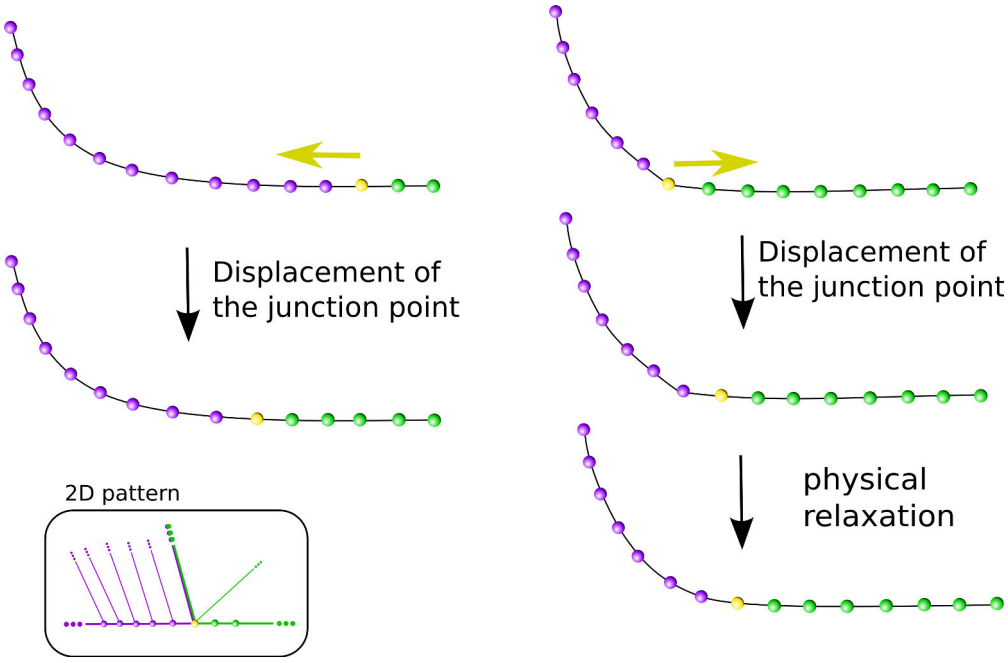



FIGURE 2.15: **Displacement of a junction point.** 2D view of a border of the sheet of paper showing the displacement of a junction point between a curved region and a flat region.

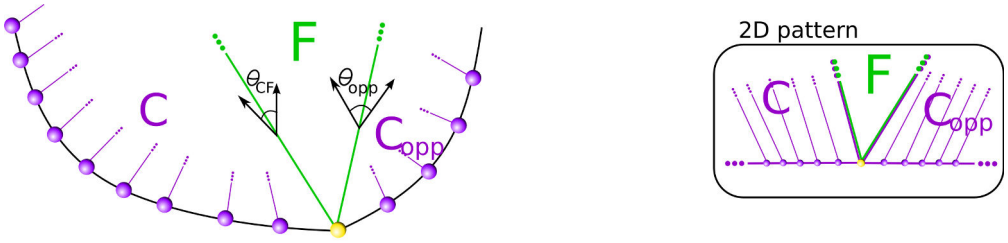



FIGURE 2.16: **Displacement of a junction point** joining a flat region and two curved regions.

### 2.5.2 Fitting generalized cones

We now seek to approximate each curved region by generalized conical surfaces as illustrated in Figure 2.17. A curved region which contains an interior singular point $s$ is necessarily represented only by one generalized cone whose apex is $s$. In this case, we just ensure that the corresponding border of the surface is correctly sampled (i.e that the number of points on this border corresponds to the density $\rho_{\text{bound}}$). The following explanations therefore concern only the other case, when the curved region does not contain any interior singular point.

We approximate the surface of a curved region $C$ in a set of generalized cones based on the current rulings. Note that, as we saw earlier, when a curved region is created

the rulings are initialized all in the same direction parallel to the direction of minimum compression. The rulings are then updated to fit the set of approximating generalized cones.

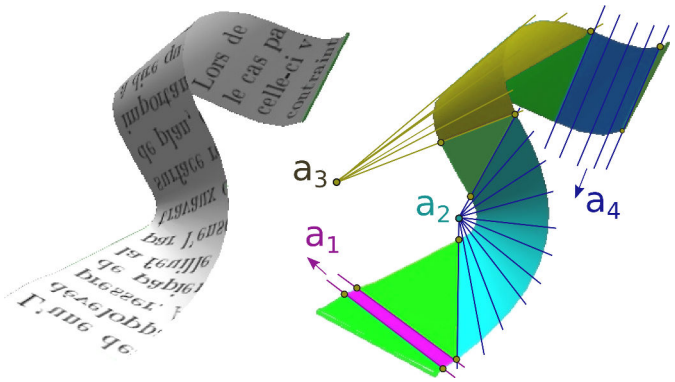



FIGURE 2.17: **Segmentation of the surface into piecewise conical pieces in the top.** The colored parts correspond to pieces of developable generalized cones whose rulings are converging toward their respective apex $a$ satisfying equation 2.3. Each region may be separated by flat triangular regions in green ensuring tangential continuity. The junction vertices are shown as yellow dots.

**Finding the apex of a generalized cone.**

In order to find the apex of a generalized cone that approximates a list of rulings, we take inspiration from the surface reconstruction method in [Pet04] and compute the best approximating generalized cones using the *Blaschke model*[1], which is particularly well adapted to our setting. This model enables us to fit a one-parameter family of tangent planes from the estimated tangent planes along the rulings of our surface strips. The resulting surface is developable. We summarize this model and the way we use it below. See [Pet04] 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 of $\mathbb{R}^3$. 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 $\mathbf{u} = (u_1, u_2, u_3, u_4)$ of $\mathbb{R}^4$ obeying:

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

This set is called the *Blaschke cylinder*. You can visualize it as the Gaussian sphere extruded along the fourth coordinate.

Each tangent plane, represented by a point $(n_1, n_2, n_3, w)$ in $\mathbb{R}^4$, of a generalized cone passes though the same point $\mathbf{a}$ which is the apex of the cone. Thus, they all are

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

contained in the hyperplane:

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

with $\|\mathbf{n}\|^2 = 1$. We use this property to compute the apex $\mathbf{a}$ of the generalized cone that best approximates a surface as follows:
Let us consider $N$ rulings $r$ associated with a smooth part of the mesh and their associated normals (constant along the ruling) $\mathbf{n}$. We call $\mathbf{n}_i$ (resp. $T_i$) the i-th consecutive normal (resp. tangent plane associated to this normal). Finding the point $\mathbf{a}$ in a least squares sense requires to minimize the error $e$ given by:

$$e = \sum_{i=1...N} \|\mathbf{n}_i \cdot \mathbf{a} + w_i\|^2, \tag{2.3}$$

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

**Algorithm 2:** Segmentation

```
Data: rulings_list ;                          // current rulings of the curved region
Result: apices_list ;      // apices of the cones segmenting the curved region
    indices_list ;         // indices of the rulings corresponding to each cones
n ← size(rulings_list)
apices_list ← {}
indices_list ← {0, n }
segmentation_rec (rulings_list, apices_list, indices_list)
```

**An iterative segmentation algorithm.**

In practice, we use an iterative algorithm to find a segmentation into generalized cones of our curved surface strips such that the error $e$ is smaller than a fixed threshold $\varepsilon_{\text{blaschke}}$ that we set to $0.5\% \times L_{\text{paper}}$. We start our algorithm by setting $\mathcal{I}_{cur} = \{r_i\}_{i=1...N}$ as the list of ordered rulings defining a curved area in the current segmentation. We also initialize two empty lists $\mathcal{I}_{begin}$ and $\mathcal{I}_{end}$. Using the Blaschke model presented above, we compute the apex $\mathbf{a}$ of the approximating generalized cone defined by $\mathcal{I}_{cur}$ and the corresponding error. Until the error is smaller than $\varepsilon_{\text{blaschke}}$, we iteratively remove either the first or the last ruling from $\mathcal{I}_{cur}$ and either add it to the list $\mathcal{I}_{begin}$ or to $\mathcal{I}_{end}$. The selected ruling is the one associated to the largest, between the two possible rulings, 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 $\mathcal{I}_{begin}$ and then $\mathcal{I}_{end}$. In this way, we obtain a list of apices with, for each one, a list of corresponding rulings. The recursive algorithm is

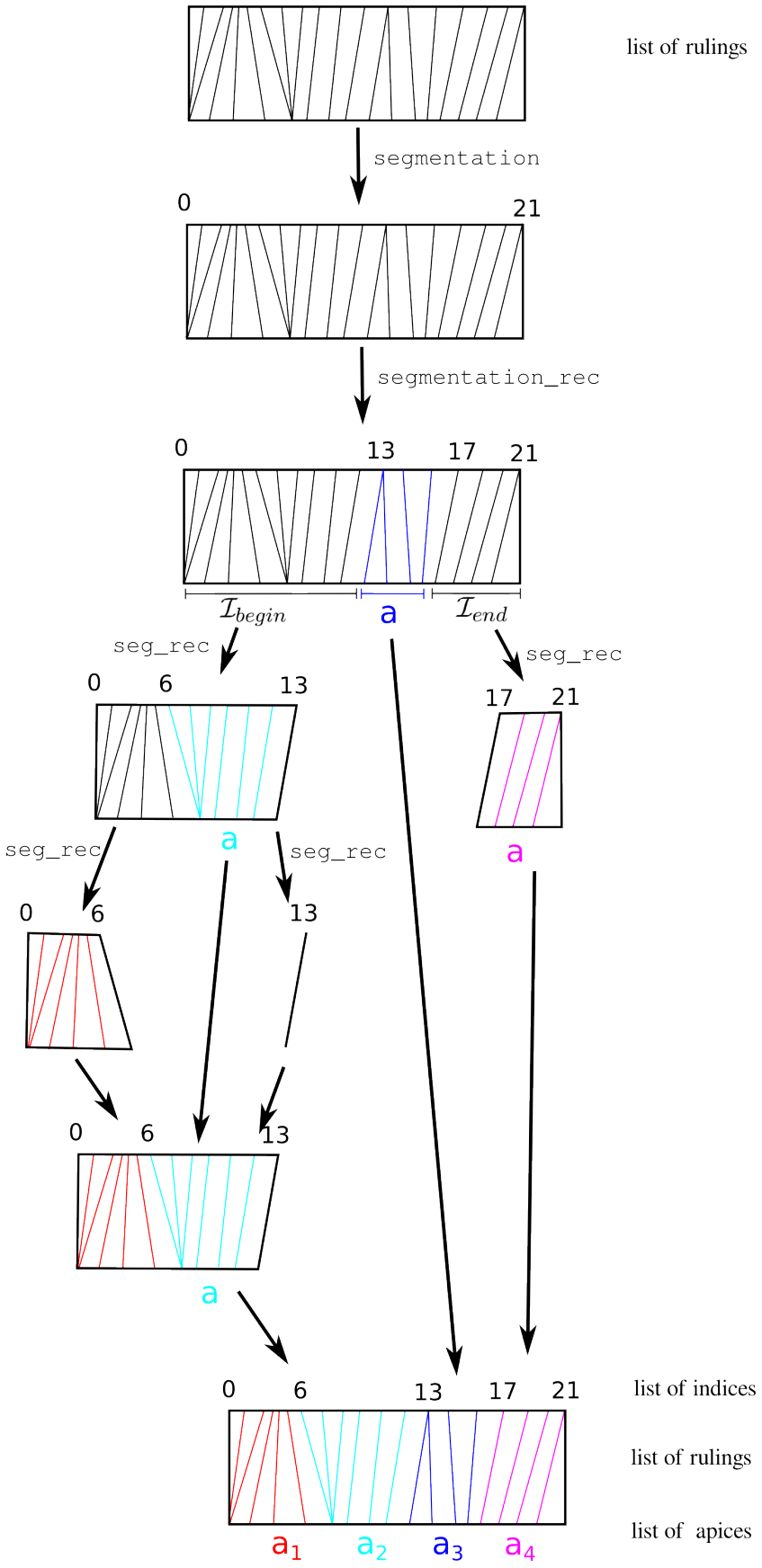



FIGURE 2.18: **Segmentation algorithm:** an example.

**Algorithm 3: Segmentation:** takes in entry a list of rulings and two indices and return the list of apices approximating the rulings between those two indices and the corresponding list of indices.

**Data:** rulings_list, apices_list, indices_list
**Result:** apices_list, indices_list
$i_f \leftarrow$ indices_list.front() ;
$i_b \leftarrow$ indices_list.back() ;
apex $\leftarrow$ `compute_apex_approx` (rulings_list, $i_f$, $i_b$) ;
▷ compute the apex of the cone the best approximate rulings between the indices using Blaschke Cylinder
error = `compute_error` (apex, rulings_list, $i_f$, $i_b$) ;
▷ compute the error of this apex
**while** error $> \varepsilon_{blaschke}$ **do**
  **if** `compute_error` *(*apex, rulings_list, $i_f$*) >* *error(*apex, rulings_list, $i_l$*)* **then**
    $i_f++$ ;
  **else**
    $i_b--$ ;
  **end**
  apex $\leftarrow$ `compute_apex_approx` (rulings_list, $i_f$, $i_b$) ;
  error $\leftarrow$ `compute_error` (apex, rulings_list, $i_f$, $i_b$) ;
**end**
**if** indices_list.*front()* $\neq i_f$ **then**
  `segmentation_rec` (rulings_list, front_apices_list $= \{\}$,
   front_indices_list $= \{$indices_list.front(), $i_f\}$) ;
**else**
  front_indices_list $\leftarrow \{i_f\}$ ;
  front_apices_list $\leftarrow \{\}$ ;
**end**
**if** indices_list.*back()* $\neq i_b$ **then**
  `segmentation_rec` (rulings_list, back_apices_list $= \{\}$,
   back_indices_list $= \{i_b,$ indices_list.back()$\}$) ;
**else**
  back_indices_list $\leftarrow \{i_b\}$ ;
  back_apices_list $\leftarrow \{\}$ ;
**end**
apices_list $\leftarrow$ front_apices_list :: apex :: back_apices_list ;
indices_list $\leftarrow$ front_indices_list :: back_indices_list ;

described in Algorithm 2 and Algorithm 3. An example of this algorithm is depicted in Figure 2.18.

The last step consists in aligning the existing rulings (magenta lines in Figure 2.1) with the rulings of the corresponding generalized cones, as follows. We consider the ruling $r^j$ 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 2D edge lines are all

converging to $\overline{\mathbf{a}}^{\text{proj}}$ as shown in Figure 2.19. The 3D coordinates of the newly sampled vertices are finally mapped to the 3D boundary curve using cubic interpolation.

Two successive generalized cones $c_1$ and $c_2$ join on the boundary $\partial S$ either on a common ruling (as the two most left cones on Figure 2.19) or at the point $b$ where their extremal rulings cross. In the second case, the point $s_b$ is singular. Thus we consider the gap between the two cones (in green in Figure 2.17 and Figure 2.19) as a generalized cone $c_{gap}$ whose apex is $b$ and remesh it accordingly. Note that the first and the last rulings of $c_{gap}$ are respectively the extremal ruling of $c_1$ and $c_2$. In this way, the remeshed curved area is represented by a set of generalized cones joined by common rules and thus is developable.

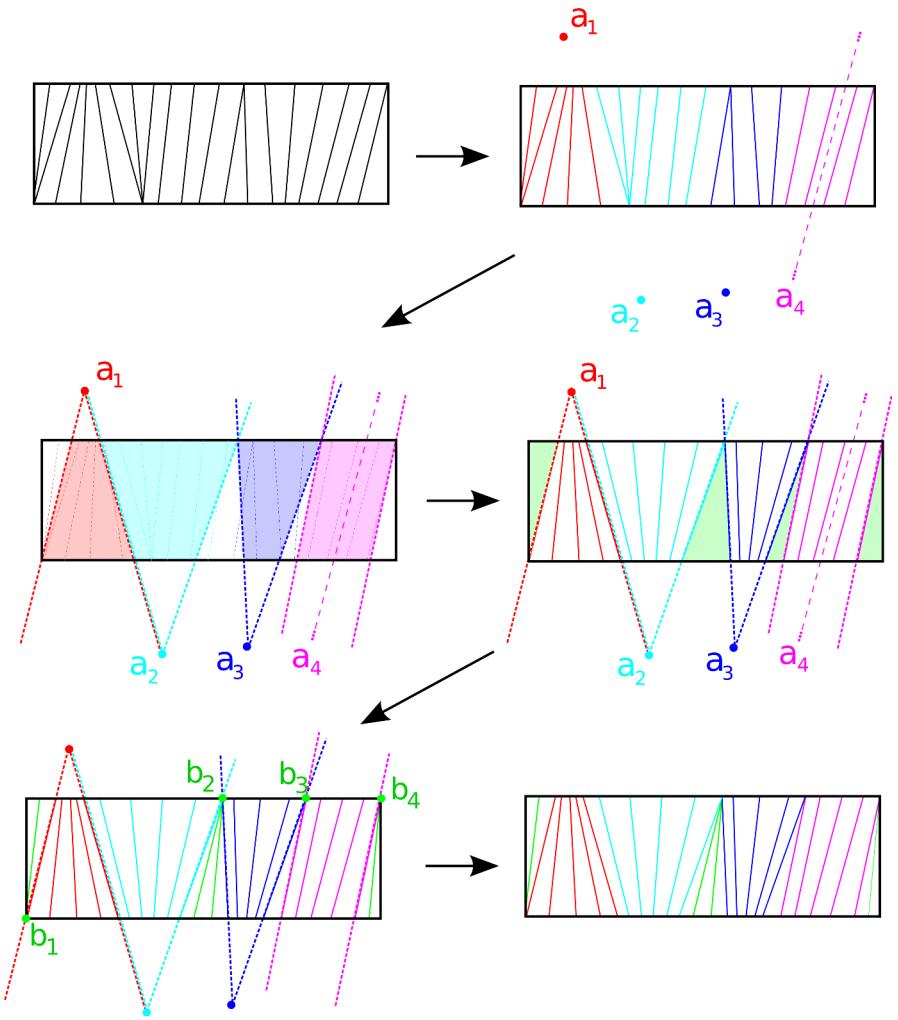



FIGURE 2.19: **Remeshing inside a curved region.** First row: the current rulings are used to find a list of apices $a_1, a_2, a_3, a_4$ ($a_4$ being at the infinity) corresponding to the surface. Second row left : area of influence of each apex. Second row right: resampling of the boundaries in order to have in each area the rulings converging to the corresponding area. Last row left: resampling of the gaps between the area by considering them as cones whose apices are $b_1$, $b_2$, $b_3$ and $b_4$. Last row right: final mesh

During the resampling process we have just explained, we may need to add or remove rulings in order to keep a sample density $\geq \rho_{\text{bound}}$ along the border of the paper for each cone.

The approximating surface is a succession of patches of generalized cones joined along a common ruling, so it is developable.

### 2.5.3 Preserving length with respect to the pattern

To explicitly enforce the lengths of edges of the mesh in $S$ to match those of the pattern $\bar{S}$, we use a new, efficient optimization process implemented by Jerry Shuo Jin and Charlie Wang. The algorithm 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 vertex-based prior work of shape optimization, this new formulation can explicitly enforce the preservation of edge lengths.

More details can be found in [SRH*15].

## 2.6 Results

In addition to the results provided in this section, we encourage the reader to look at the video available at the following link `https://hal.inria.fr/hal-01202571/file/video_with_sound.mp4`. The video shows notably the animations presented in Figure 2.21 and Figure 2.26.

### 2.6.1 Validation

In this section, we use experiments involving real pieces of paper to validate our animated model. Note that contrary to previous approaches, we specifically conduct comparisons in the critical state when the paper begins to deform and just a few singular vertices appear. Indeed, these cases are easily reproducible with real paper and their specific behavior is rather familiar. Contrary to very crumpled paper in which one can hardly predict where the next sharp features will appear, the shape of little crumpled paper and also when and where the first sharp features appear is very visually noticeable. Capturing these accurately is thus a key element towards realism.

For each validation test, we started from an flat initial mesh $S$ (which helped to reproduce the same situation in real life), of size $L_{\text{paper}} = 10\text{cm}$. The boundary $\partial S$ of the virtual paper is initially sampled with 12 vertices.

**First sharp feature.** The first experiment, shown in Figure 2.20 (left), was used to validate the way new singular vertices appear: we move the two opposite sides of an undamaged sheet of paper closer to each other by a factor of 10% and then apply an orthogonal constraint on one of the edges until a singular point appears. We then mark the real paper with the position of this singular point, and repeat the experiment 20 times with a new sheet each time. The same experiment is conducted the same number of times with our virtual paper model, using the random factors in the placement of singular vertices described in Section 2.4.3. The resulting distributions of the position of the first singular point are depicted in Figure 2.20 (center and right), for printer paper and writing paper. In order to adapt our model to the type of paper, we choose the variance of the probability law to be $\sigma = 0.001$ for the printer paper (whose density is 70 $g/m^2$) and $\sigma = 0.01$ for the writing paper (whose density is 90 $g/m^2$). Our results show a good correspondence between real measurements and our model.

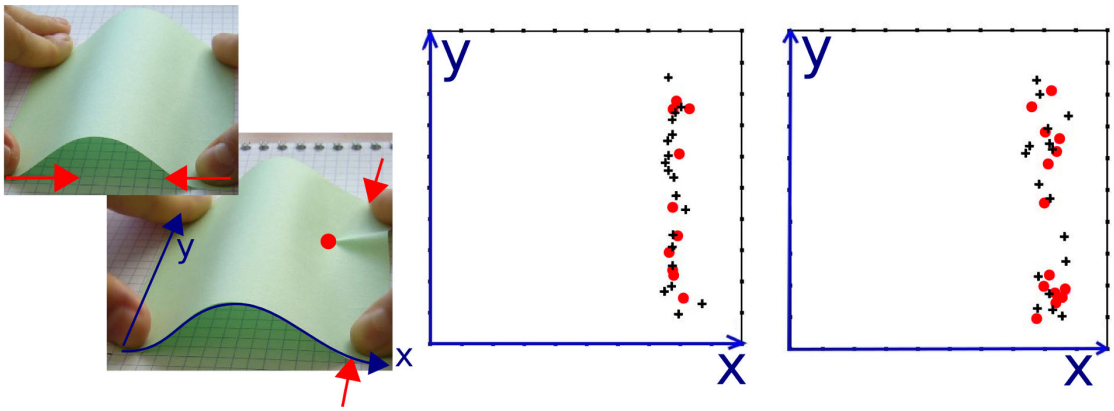



FIGURE 2.20: **Comparison between the positions of the first singular vertex, in real and in virtual.** The real measurements are shown as black crosses, while the positions generated by our method are represented as red dots. The graph in the center refers to a 70 $g/m^2$ paper, modeled using $\sigma = 0.001$. The graph on the right refers to a 90 $g/m^2$ paper, modeled using $\sigma = 0.01$.

**Visual comparisons** between real paper and our results are also provided in Figure 2.21 with two experiments. The top row illustrates the deformation of a smooth strip of paper continuously twisted towards a Moebius strip. This experiment validates in particular our method to segment a smooth curved region into a set of generalized cones (Section 2.5.2). Our method enables to handle even complicated twisted shapes, with different directions of curvature and highly bent parts. Note that a similar idea has been recently proposed in [SHCB15] to simulate inextensible ribbons.

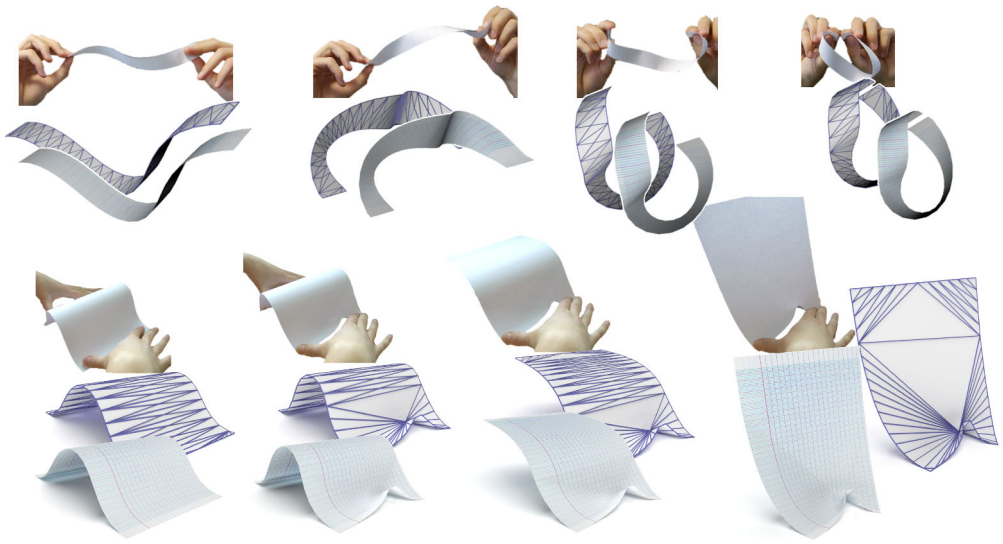


FIGURE 2.21: **Comparison with real paper.** We compare our results with the deformation obtained with real paper for a smooth surface (top) and the creation of a singularity (bottom).

The second row compares the behavior of a 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. The rigidification effect of the curved parts mentioned earlier is visible here. Note also that our mesh remains coarse throughout the animation, with its edges aligned along the surface rulings. Moreover, the singular vertex is persistent in the model even if the surface is totally unloaded, which is one of the key features of the method.

Figure 2.22 shows a step-by-step comparison between our virtual paper and real one. The paper is deformed from a flat undamaged state to a mildly crumpled state with a few singular points.

### 2.6.2 Comparison with other methods

**Comparison with FEM and adaptive meshing methods.**
Figure 2.23 compares our method with a standard FEM applied to a mesh of fixed connectivity and with simulation with adaptive remeshing from Narain *et al.* [NPO13]. Each of the three methods relies on a physical simulation so we use the same FEM solver (*arcsim* from Narain) for all three. The differences in the results come then mainly from the remeshing strategy (or its absence). We use the set-up of the experiment described in Section 2.6.1. We compare first the computational times needed to obtain results of similar quality. We then observe the quality of the results obtained for similar times –by adapting the density of the meshes to adjust the computational times. The results are shown in Table 2.2.

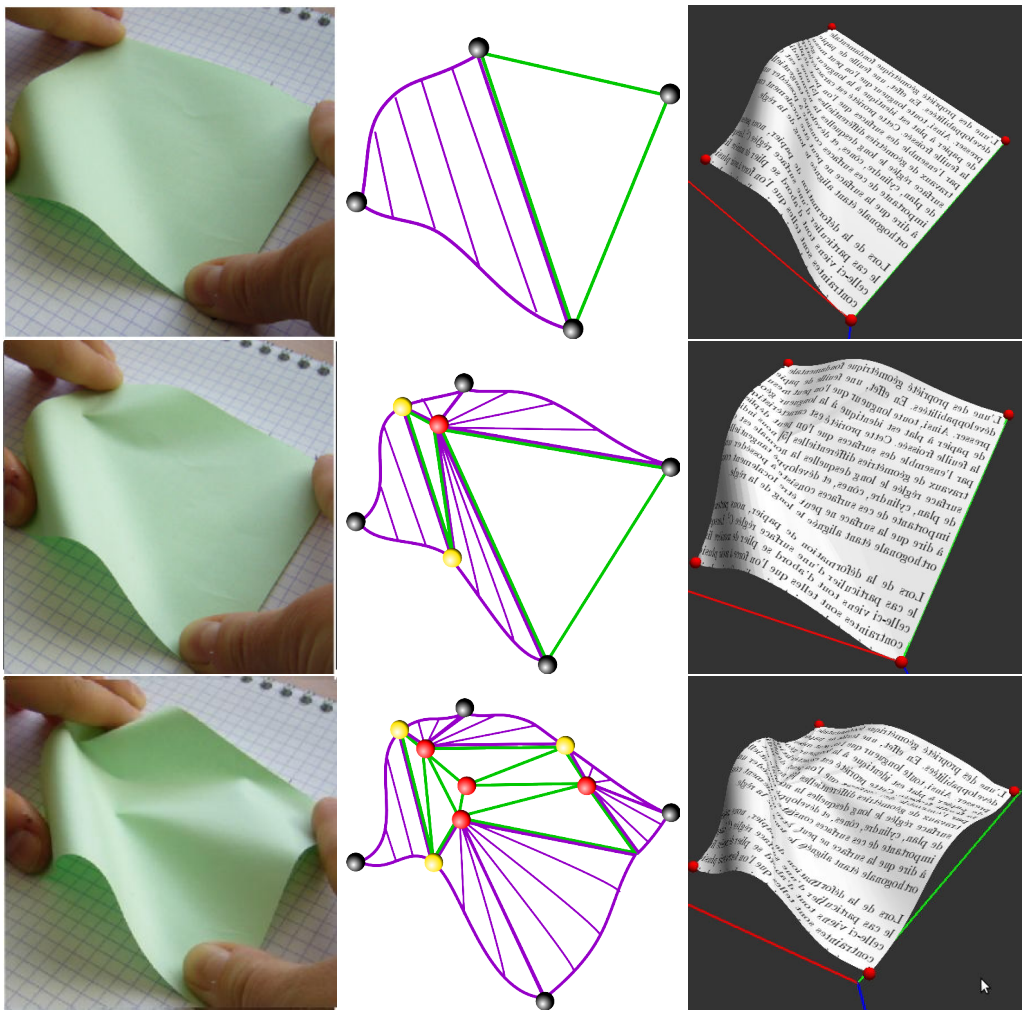


FIGURE 2.22: **Step-by-step comparison with real paper.** (left) real paper, (middle) geometric structure, (right) virtual paper.

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 results, 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 2.2. The lower row of Figure 2.23 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.

**Comparison with *"Fast Simulation of Mass-Spring Systems"*.**

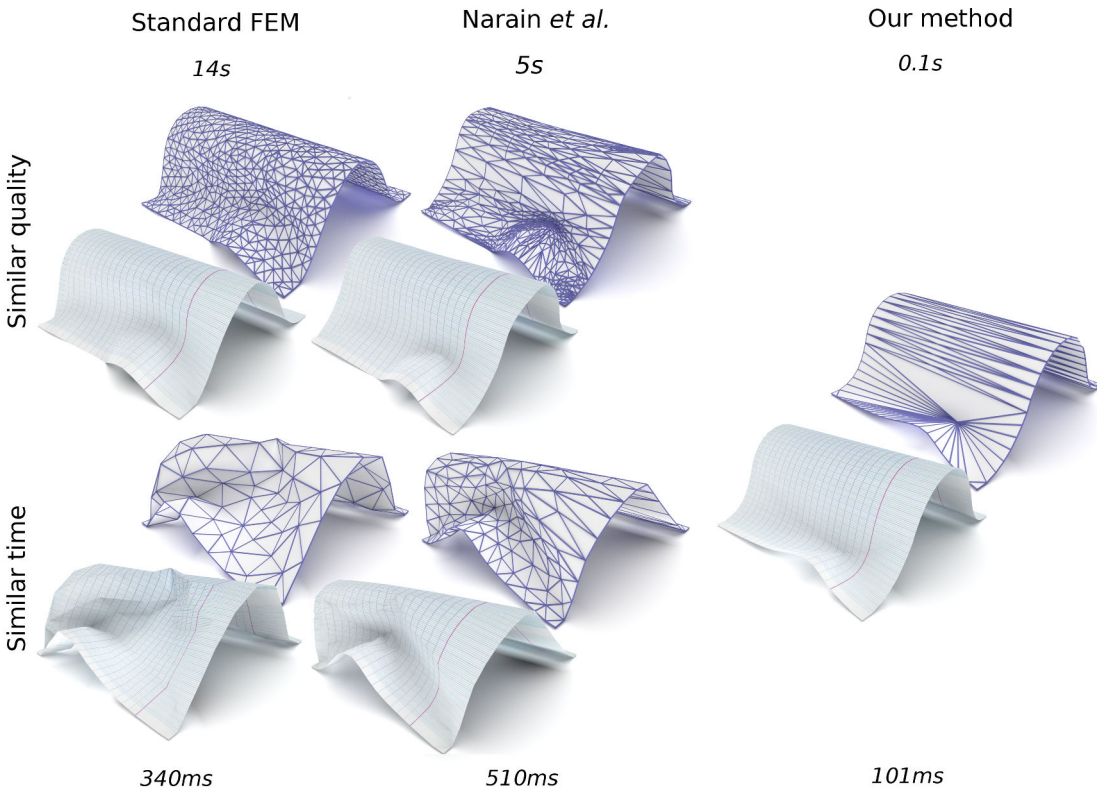



FIGURE 2.23: **Comparison with other methods.** The upper row shows how much mesh refinement is required for Standard FEM (top left) and Narain *et al.* [NPO13] method (top middle), in order to get a result of similar visual quality as our method (right).
The lower row compares the mesh refinements for 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),

| | 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 |

TABLE 2.2: **Average computation times** for the example presented in Figure 2.23. All timings are expressed in ms per frame.

The technique we present in Section 2.5.3 treats the preservation of edge lengths as a geometric problem while Liu *et al.* [LBOK13] simulate a mass-spring system in a physical perspective with their approach called *Fast Simulation of Mass-Spring Systems* (FSMS). 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.

**Comparison with *"Shape-Up"*.**

The *Shape-Up* method [BDS*12] provides a good solution for achieving isometry preservation. The original Shape-Up framework treats all constraints as soft constraints and requires minimizing, in a least-square sense, an energy function that incorporates the projections of all constraints to the targets. 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 keep the surface isometric to its 2D pattern, i.e. to preserve the length of the edges 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 prefactorization 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 2.5.3) is given in Table 2.3, 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 2.3. 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 2.24. This is particularly non-desirable for our application, where singularities, flat and curved regions are processed explicitly.

### 2.6.3 Other results

**Variation of compression factor.** The first row of Figure 2.26 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 2.25, while keeping the same handle animation scenario. Larger values of $\varepsilon_c$ capture the behavior of stiff, thick paper,

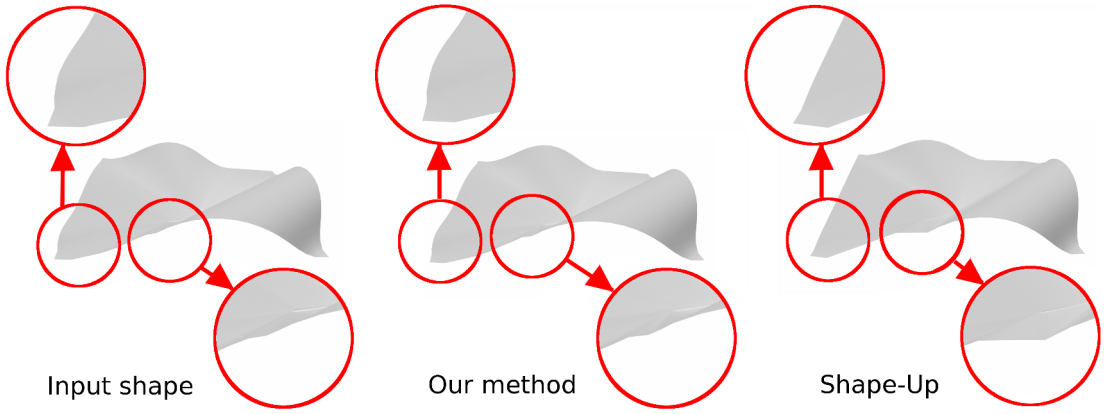



FIGURE 2.24: **Distortion caused by the isometry optimization.** Comparison of an input shape (left) with the results of our isometry optimization method (middle) and of Shape-Up isometry optimization (right). Contrary to our method, Shape-Up introduces unwanted discontinuities (right-most circle) or local shape flattening (left-most circle).

| | | Our | | | Shape-Up | | |
|---|---|---|---|---|---|---|---|
| Models | $c_{\text{init}}$ | $t_{\text{opt}}$ | $c_{\text{m}}$ | $d_{\text{m}}$ | $t_{\text{opt}}$ | $c_{\text{m}}$ | $d_{\text{m}}$ |
| Figure 2.24 | 0.75 | 196 | 0.38 | 2e-3 | 721 | 0.17 | 1e-2 |
| Figure 2.21 (top) | 0.11 | 112 | 0.06 | 9e-4 | 530 | 0.02 | 1e-3 |
| Figure 2.21 (bottom) | 0.15 | 97 | 0.10 | 7e-4 | 480 | 0.03 | 1e-3 |
| Figure 2.26 ($1^{st}$ row) | 0.15 | 143 | 0.13 | 7e-4 | 760 | 0.03 | 3e-3 |
| Figure 2.26 ($3^{rd}$ row) | 0.55 | 113 | 0.42 | 3e-3 | 580 | 0.15 | 8e-3 |
| Figure 2.26 ($5^{th}$ row) | 0.22 | 122 | 0.20 | 2e-3 | 590 | 0.02 | 4e-3 |

TABLE 2.3: **Comparison of our length preservation method (Section 2.5.3) and the Shape-Up method**. We apply our optimization (Section 2.5.3) and the Shape-Up optimization to a final mesh of some of ours examples.
$c_{\text{init}}$ is the mean percentage of compression computed over all the edges of the model before any optimization is applied. $t_{\text{opt}}$ is the time (in ms) taken by the optimization. $c_{\text{m}}$ is the mean percentage of compression over all the model edges. $d_{\text{m}}$ is the average displacement of the vertices from their initial position.

while smaller values capture the behavior or thinner paper, for which a larger number of singular points appear.

**Some other examples.** Figure 2.26 gathers other results of our method. The second row illustrates smooth deformations highlighting the dynamic adaptation of the triangulation. The third row shows a full crumpling animation where more than twenty singular vertices are progressively generated. The fourth row illustrates a newspaper type of surfaces 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.

**Isometry.** To get quantitative results, we measured the error of lengths compared to the

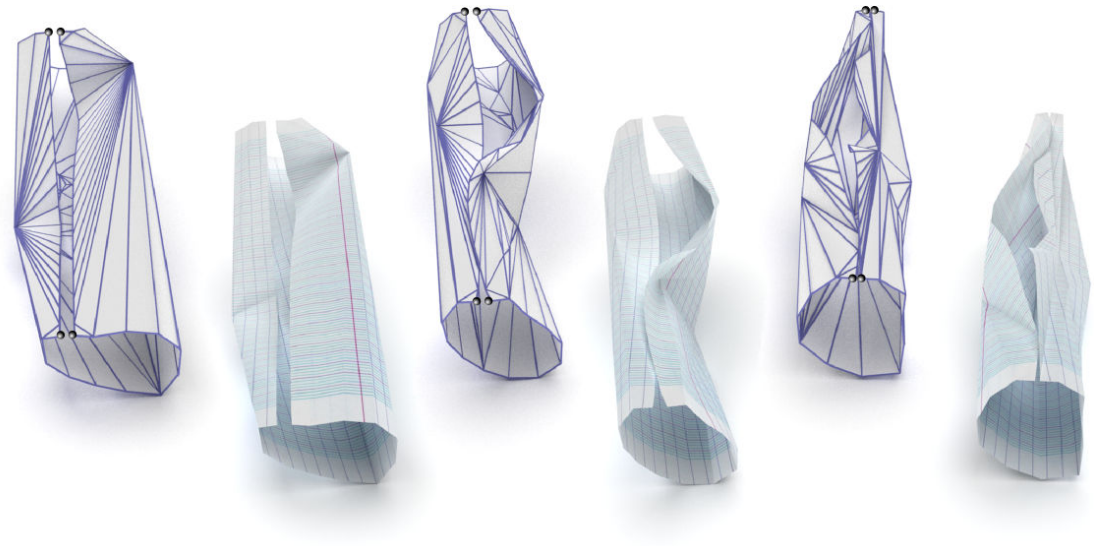


FIGURE 2.25: **Variation of the compression factor.** The same experiment as in Figure 2.26 (first row) is performed with different values of the allowed compression factor $\varepsilon_c$: from left to right, 1%, 0.5%, and 0.2%.

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 2.4 provides, for different examples, the mean value of $E_{\text{length}}$ for all the edges of the mesh, averaged over all the animation frames. This value, $c_{\text{mean}}$, is given in %. We can see that this mean compression value stays below 1% except when the paper begins to get really crumpled as in the $3^{rd}$ row of Figure 2.26. In a similar way, we observe that the average number $n_{\text{comp}}$ of compressed edges (aka with $E_{\text{length}} \geq 1\%$) stays under 10% when the paper is not too much crumpled. As the paper gets more crumpled, many singular points appear and the simulation cannot anymore take advantage of the developable properties as area of the regions modeled by smooth developable surfaces becomes negligible. Also the constraints are stronger in the case of very crumpled paper, so the model struggles more to solve all of them.

**Computation times.** The computational times for the different examples are given in Table 2.5. We can see that the time used by the geometric remeshing step is negligible compared to the time of the physical simulation. The simulation times is much reduced, due mainly to the very small number of triangles, compared to other methods using FEM (see Section 2.6.2). Indeed we reach interactive times but the simulation remains the bottleneck preventing us from reaching real times.

### 2.6.4 Discussion and future work

We have obtained a variety of results that qualitatively look like real paper, even when animated (see the associated video). This indicates that our method has reached most of its goals. However, we have also identified a number of limitations, listed next:

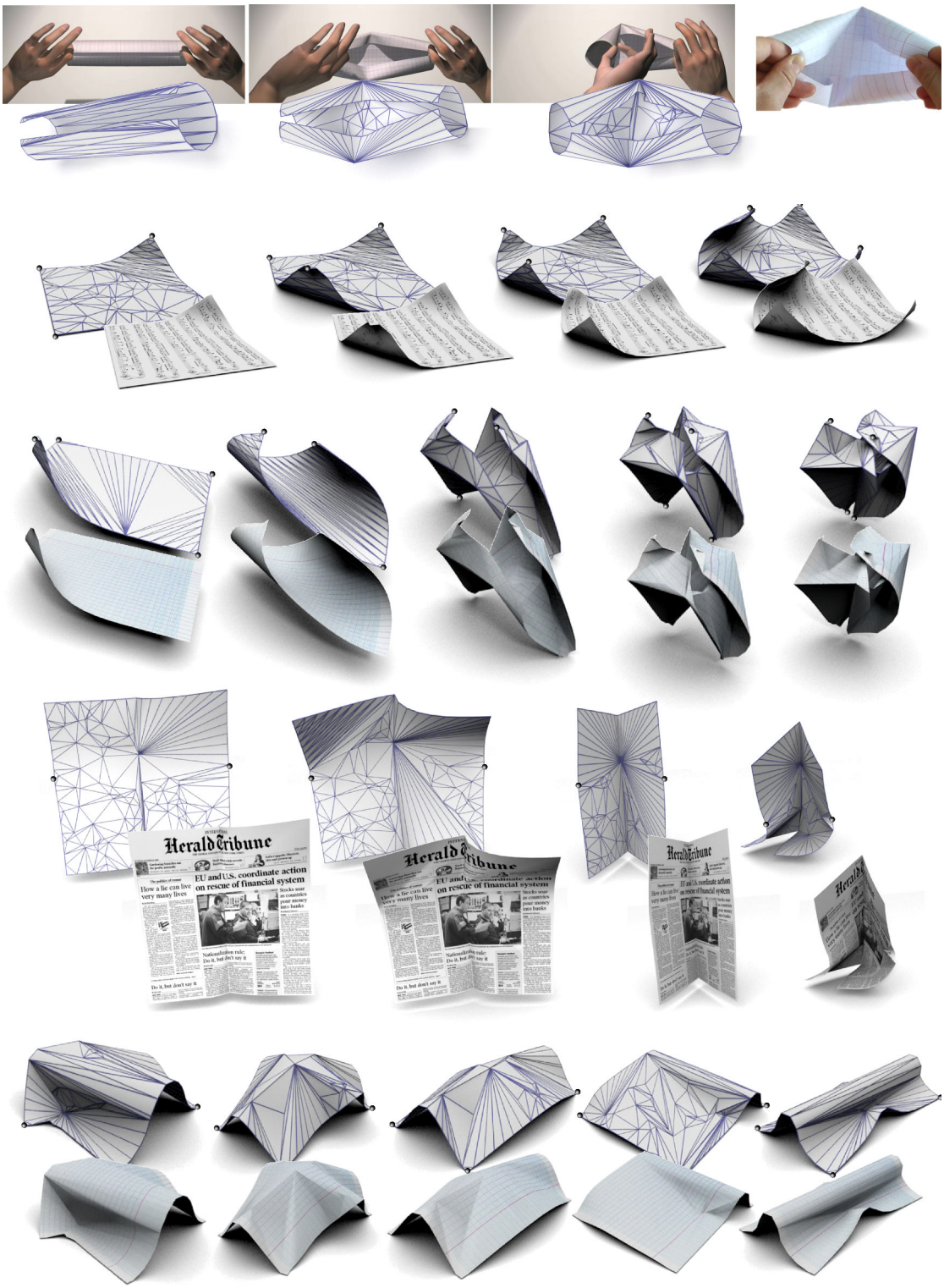



FIGURE 2.26: **Paper crumpling examples.** The black points indicate the positions of the handles.

Firstly, although much faster than state of the art methods, our model does not meet the real-time performances we were expecting: our prototype currently runs at 1 to 10 fps. As shown by the timing table, most of the time is spent in the physically-based

| Models | $c_{\text{mean}}$ | $n_{\text{comp}}$ |
|---|---|---|
| Figure 2.21 (top) | 0.17 | 0.1 |
| Figure 2.21 (bottom) | 0.19 | 2.0 |
| Figure 2.26 ($1^{st}$ row) | 0.89 | 8.0 |
| Figure 2.26 ($3^{rd}$ row) | 1.47 | 11 |
| Figure 2.26 ($5^{th}$ row) | 0.20 | 2.0 |

TABLE 2.4: **Measures of error in length for our examples.** $c_{\text{mean}}$ is the mean percentage of compression over all the model edges and $n_{\text{comp}}$ is the percentage of edges whose compression is $> 1\%$.

| | Mesh size | | Simulation | | | Geometry | |
|---|---|---|---|---|---|---|---|
| Models | $N_t$ | $N_s$ | $N_{\text{it}}$ | $t_{\text{int}}$ | $t_{\text{col}}$ | $t_{\text{rem}}$ | $t_{\text{iso}}$ |
| Figure 2.21 (top) | 117 | 0 | 11 | 232 | 62 | 0.2 | 3.1 |
| Figure 2.21 (bottom) | 87 | 1 | 4 | 50 | 10 | 0.4 | 1.0 |
| Figure 2.26 ($1^{st}$ row) | 118 | 2 | 5 | 117 | 25 | 0.4 | 1.0 |
| Figure 2.26 ($3^{rd}$ row) | 156 | 15 | 5 | 190 | 57 | 0.5 | 2.3 |
| Figure 2.26 ($5^{th}$ row) | 120 | 21 | 8 | 190 | 48 | 0.8 | 0.9 |

TABLE 2.5: **Computational time analysis for our examples**. We separate the time spent in the simulation part and the time spent in the geometrical part. $N_t$ 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. $t_{\text{int}}$ and $t_{\text{col}}$ 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 $t_{\text{rem}}$, and in the isometry tracking $t_{\text{iso}}$. All times are given in ms.

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. Particularly, possibly due to the very specific structure of the mesh, the collision module is neither really efficient nor accurate. A specific implementation for our model may prove to be a noticeable improvement.

Secondly, another limitation hampering the interactive manipulation is the fact that we only provide point-wise handles to the user. Offering more complex handles, such as the ability to crumple paper within some bounding volume, is not only a user-interface concern but would deeply modify the existing approach. To get this effect, one could imagine to constrain all points located on the convex hull, at each time step. These two limitations explain why for now, we are not able to create tightly crumpled balls of paper.

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 animation. While such geometrical discontinuities may arise for real paper as well when fibers break and singularities appear, these effects are

| 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 |

TABLE 2.6: **Summary of user-defined parameters**

exaggerated by the coarse triangulation we use. Smoother temporal animation could be explored using temporal smoothing. We observe here that when happening for real paper, those sudden changes are accompanied by a characteristic "clac" sound. We will use this observation in Chapter 4.

Moreover, our model only generates singular points as sharp features, although previous studies indicate that crumpled paper geometry can exhibit creased curves as well. [KGK94] computes such curves using physical simulation on a geometric model even simpler than ours. Exploring the detection and generation of similar creased curves into our model is left for future research. Here, the creases modeled in [KFC*08] could be a source of inspiration. Also it may be interesting to explore the use of tangent developable instead of generalized cones, as they have singular curves, and not only singular points.

Lastly, although each individual generalized cone, as well as each flat part, is developable, the overall surface may not be globally developable. It is however almost the case, due to the quasi-isometry with the 2D pattern. Developability could be further improved by minimizing the angular defect around each singular vertex.

## 2.7 Conclusion

In this chapter, we introduced a new hybrid geometric and physical model for paper, able to qualitatively capture the dynamic shapes of this complex material, and efficient enough to be manipulated at interactive times. 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 the model in terms of singular points and rulings, throughout the simulation. In addition to accelerating animation, this coarse mesh yields fast, good quality renderings.

Our unique combination of interwoven geometric and physical processing may inspire the modeling of other complex materials, for which the same general methodology could be used.

Finally, our model outputs not only the animation of the mesh representing the surface, but also information about the geometry of the surface that can be used for different purposes, notably tearing the paper or modeling its sound, as we will see in the following chapters.

# Chapter 3

# Modeling tearing paper

*"Help me, you tear down my reason"*
Nine Inch Nails, *Closer*, 1994.

## 3.1 Introduction

As explained in Chapter 2, the length-preserving nature of the paper surface leads to crumpling and bending behavior when the surface undergoes compression forces. This chapter, instead, will focus on the opposite case, i.e. applying extension forces onto the paper. In this case, tearing will occur, still preventing any in-plane deformation, but instead modifying the boundaries of the sheet of paper. Similarly to the crumpling case, the fibrous structure is also related to this tearing behavior: Under extension forces, the bonds between fibers break which leads to tearing.

We identify four features needed to reproduce a plausible tearing behavior:

**Local length preservation.** As already explained, the fibrous structure of paper prevents in-plane deformations. Note that in the tearing case, the boundary of the 2D pattern changes to integrate the tears.

**Generation and propagation of the tear.** A tear is initiated or grows only if enough stress is applied to the tear's tip. At a large scale, the tear follows a predictable path that depends mainly of the motion imposed on the sheet: the same deformation applied on two different sheets will produce similar paths. In some specific

cases, it is possible to mathematically predict the curve formed by the path as described in Roman's work [Rom13].

**Tear appearance.** As the distribution of the fibers is inhomogeneous within the paper material, the shape of the tear exhibits small details where fibers have been pulled apart leading to highly detailed tearing shape at small scale. The details may have a size of the same order that the size of the fibers, having a width about 30 $\mu$m (see Figure 3.1).

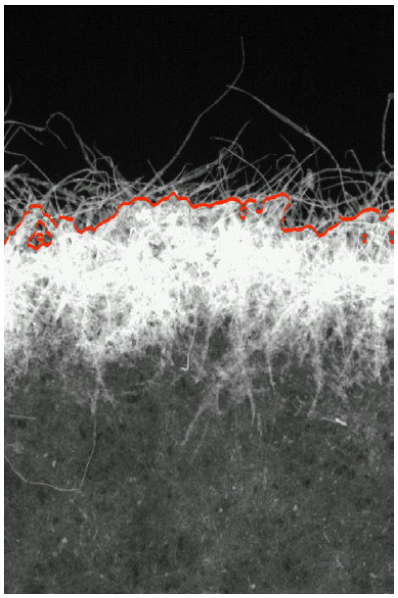


FIGURE 3.1: Fracture process zone in paper made visible by impregnation with a silicone that has been cured before the fracture test. Image from [AN06].

As explained previously, teared paper exhibits highly detailed geometry along the tearing path. Computing tearing at the scale of the smallest detail would therefore require a very dense mesh and would not be appropriate for interactive applications. At the opposite of most previous works handling geometry and details of the tear simultaneously, we use a layered model to separate two levels of detail. At the larger scale, a first layer models the macroscopic geometry of the sheet and handles the computation of the global tearing path using simulation or geometrical assumptions. At the smaller scale, a second layer models the microscopic details around the fracture path. As the fibrous structure within the paper material exhibits stochastic organization, this layer can be used locally without interfering with the macroscopic scale. Separating these two layers enables very efficient computation at both scales using only a coarse triangle mesh at the macroscopic level, and local texture for details at the microscopic scale.

This chapter introduces two new methods for efficiently handling paper tearing using this layered model. Firstly, we present a procedural approach for modeling the tear in a specific scenario. Although this first model is restricted to a limited set of interaction

possibilities, the fully procedural description enables real-time computation. Secondly, we describe a more general approach to model the tearing with wider interaction possibilities using physics-based criteria and underlying simulation adapting the stress field around the tip of the tear. This second model is coupled with our geometrical paper model described in Chapter 2 and enables both tearing and crumpling at interactive rates.

*The procedural method presented in the first section has been published as a short paper in Motion In Games (MIG) [LFD*15].*

## 3.2 A particular case using a procedural method

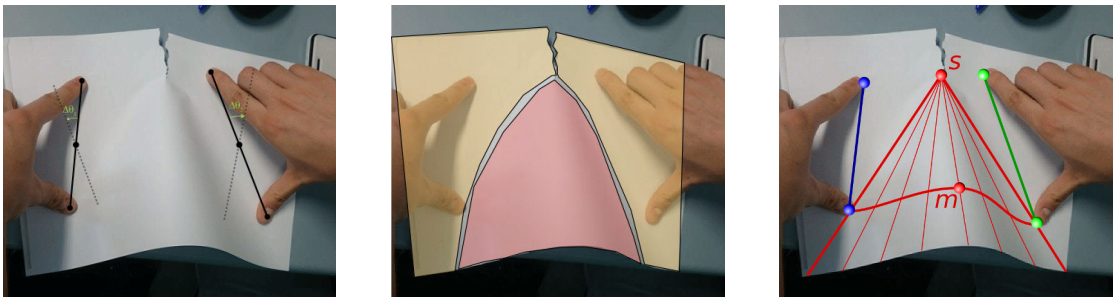
FIGURE 3.2: **Possible application of our technique** using interactive paper tearing for torn paper collage creation on a multi-touch tablet.

In this section we present the first technique in computer graphics for real-time paper tearing using a phenomenological model. Note that the work presented is this section has been realized in the context of a student project I co-advised.

We propose here an interactive method that allows easy manipulation and tearing of virtual paper for games or artistic applications, such as the virtual crafts application proposed in Figure 3.2. In addition to the realism of the results, we must ensure real-time computation that we achieved based on a phenomenological approach that procedurally reproduces the behavior of paper being torn.

We focus on a simplified interactive case illustrated in Figure 3.2 where positional constraints are modeled by the displacement of the index and thumb of the two hands. These constrained positions are constantly lying within the 2D plane. Starting from an initially flat sheet of paper, the relative rotation of the hands implies both the tearing of the piece of paper and a 3D deformation due to out of plane buckling.

Our contribution is a real-time solution for the challenges of modeling and animating detailed tear shapes and paper bucking.




(A) (B) (C)

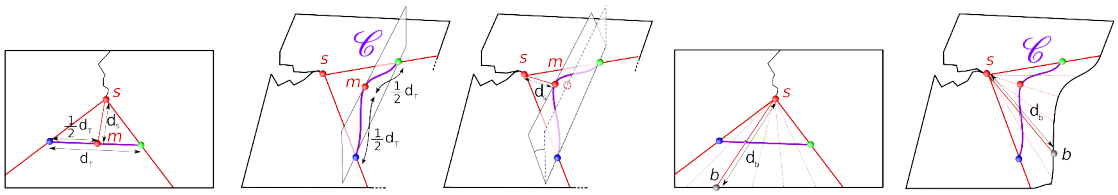
FIGURE 3.3: (a) The model for the movement of the hands. (b) The two planar regions in yellow and the generalized cone shown in red. (c) An example of the cone shape formed between the tip of the tear $s$ and the thumbs. The position and orientation of the two flat regions of the paper on either side of the cone are driven by the motion of the hands. The cone is defined by its apex $s$ and two Hermit curves joining, respectively, each thumb and the vertex $m$ at the middle of the geodesic passing though both thumbs.

### 3.2.1 Paper deformation model

In order to capture the key features of paper deformation, we performed several informal experiments and observed the behavior of tearing paper. The setting of our experiments (and our model) is a paper sheet placed on a planar surface and manipulated with two hands as described above. An example observation can be seen in Figure 3.3a.

Even when external constraints are purely 2D, torn paper is necessarily associated with 3D deformation in order to keep constant local length. In our case, where the fingers are kept on the 2D plane, bending occurs as soon as the paper is torn. We consider the assumption introduced in the previous chapter that paper can be modeled as a piecewise $C^2$-developable surface. The resulting surface consists therefore of patches of planes, cylinders, cones and tangent developables. In order to model the scenario illustrated in Figure 3.3b, we consider three geometrical surfaces: two planar surfaces respectively on the left and right of the tear, and a curved region linking them. As the tip of the tear can be considered as a singular point, we choose to model this curved region by a generalized cone whose apex is the tip (see Figure 3.3c).

In our model, each hand is modeled by two points that represent the index finger and the thumb. They can be placed anywhere on the paper sheet. At each time step, the hands perform a rotation, which we denote $\Delta\theta_1$ for the left hand and $\Delta\theta_2$ for the right. This hand motion induces tearing (as described in Section 3.2.2) and a deformation of the surface between the thumbs. The planar regions follow a rigid motion directly imposed by the hands, rotating around the tip of the tear at each step by a small




(a) 2D configuration (b) 3D bend (c) 3D tilt for isometry (d) 2D and 3D triangulation

FIGURE 3.4: **Construction of the generalized cone** begins (a) with the 2D position of $m$ set at the midpoint between the thumbs, and (b) with the 3D position of $m$ initially set vertically above the plane such that the Hermite curves preserve the distance between the thumbs. The 3D position of $m$ is then rotated about an axis going through the thumbs (c) to ensure the distance between $m$ and the apex $s$ matches the corresponding distance in the initial flat state. Finally, the cone is discretized (d) in the initial flat state, and the 3D positions of the points on the border of the paper are computed such that they lie on the corresponding ruling and that the distance between border points and the cone apex is preserved.

angle corresponding to the hand rotation. The conical region articulates the two planar regions, enabling to keep the isometry as the planar parts rotate.

The generalized cone is defined by the tear's tip $s$ (its apex) and the geodesic curve $\mathscr{C}$ that joins the two thumbs. The generalized cone is thus delimited by the two lines respectively going through $s$ and each thumb. This conical region is composed of the set of rulings going through the apex $s$ and each vertices of $\mathscr{C}$. We now detail the geometrical process to generate this generalized cone, illustrated in Figure 3.4.

The first step is to build the curve $\mathscr{C}$. $\mathscr{C}$ has to be $C^2$ to get a smooth surface. We choose to approximate it by two cubic Hermite splines which meet at vertex $m$ which is the point at equal geodesic distance of each thumb. To get a $C^2$ transition between the two curves and to simulate the pressure applied by the thumbs on the paper we constrain the tangents at the ends of the Hermite's curves to be parallel to the line between the thumbs. Note that we select in advance an appropriate magnitude for the tangents according to the size and type of paper, as discussed in Section 4.3.5.1.

The length of $\mathscr{C}$ has to be equal to the distance $d_T$ between the thumbs on the 2D pattern as shown in Figure 3.4 (a). So we adjust the vertical position of the middle point $m$ using an iterative algorithm until the length of each curve between $m$ and a thumb matches the distance $\frac{d_T}{2}$ (see Figure 3.4 (b)). In order to approximately preserve the distance between the tear's tip $s$ and the points of the curve, the next step consists in rotating the curve such that the distance $d_s$ between $m$ and $s$ on the 3D shape corresponds to the one on the 2D pattern (see Figure 3.4 (c)).

The last step is to compute the 3D position of the vertices on the curved part of the paper's boundary, i.e. the part of the boundary between the two lines going through $s$ and the thumbs. Each ruling of the cone is defined by $s$ and one point of $\mathscr{C}$. So to find

each vertex $b$ on the curved border of the paper, we compute the point at the distance $d_s$, distance between $\bar{s}$ and $\bar{b}$ in the 2D pattern, of $s$ along the corresponding ruling (see Figure 3.4 (d)).

### 3.2.2 Tearing model

Paper tearing in real-time is not a trivial problem. An extremely fine discretization is necessary to produce realistic small scale details of torn paper. This makes stress-based crack direction computations, as used in physics-based simulation, inappropriate for real-time paper tearing.

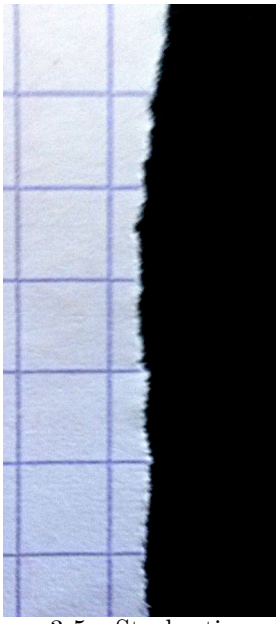


FIGURE 3.5: Stochastic appearance of the details of a real torn piece of paper

Instead, we develop a procedural tearing algorithm that exhibits the characteristic stochastic details of torn paper that we observe in empirical experiments (see Figure 3.5). Our algorithm is based on two observations. First, the main crack direction is guided by the motion of the hands. Second, the details along the crack are due to the distribution of fibers and a resistance to stress that are not uniform across the paper's surface. Our algorithm is therefore made of two steps. We first compute a main tearing direction based on hand motion, and then add tearing details based on an anisotropic fiber distribution model.

#### Tearing direction

We observe that tearing initiates at a random position on the edge between the two index fingers. In our model, we therefore choose to initiate tearing at a random point along the edge of the sheet, specifically, at a point inbetween the positions where the rays from thumb to index positions intersect the edge. Then, at each time step $t$, we compute the position of the next crack tip $s_{t+1}$ according to the motion of the hands. For this, we define a vector $\mathbf{t}$ that represents the advance of the tear between $s_{t+1}$ and the previous crack tip position $s_t$, as shown in Figure 3.6.

In our experiments, we observe that rotating a single hand produces a tear in the direction of the thumb of that hand. Therefore we compute $\mathbf{t}$ as a linear combination of the unit vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ representing the direction defined by the crack tip $s_t$ and the thumb of each hand. We weight each vector by the corresponding rotation angle $\Delta\theta$ to

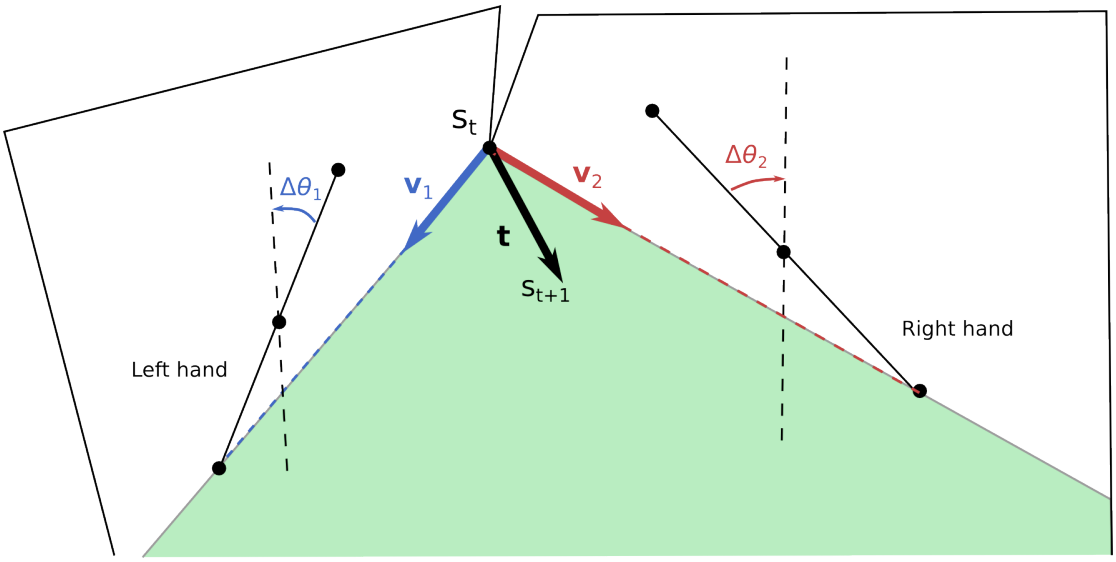



FIGURE 3.6: Our model for determining the direction of the tear at each time step depends on the incremental rotation of each hand.

compute the current tearing direction

$$\mathbf{t} = C\left(\Delta\theta_1 \mathbf{v}_1 + \Delta\theta_2 \mathbf{v}_2\right), \tag{3.1}$$

where $C$ is a constant that controls the tearing speed $||\mathbf{t}||$.

### Tearing details

Advancing the tear using only the tear direction computed above would not produce the rough edges we see in real torn paper. Therefore we procedurally add fine details along the path between two successive crack tip positions. Our algorithm is inspired by the work of Chen *et al.* [CYFW14], and makes use of two 2D functions as shown in Figure 3.7. A 2D texture $T_{fibers}$ lets us represent the stress resistance of fibers at each point of the paper. We used here a Perlin noise to create the fiber texture. A Gaussian function models the stress field $G_t$ produced by hand motion between the steps $t$ and $t+1$. Specifically, the Gaussian is centered on the edge between $s_t$ and $s_{t+1}$. We combine these two functions to obtain a 2D texture

$$\nu_t(u,v) = (1-p)\, T_{fibers}(u,v) + p\, G_t(u,v), \tag{3.2}$$

where $T_{fibers}(x,y) \in [0,1]$ is the value of the fibers texture at the position $(u,v)$ and $G_t(u,v) \in [0,1]$ the value of the stress field at $(u,v)$. So each pixel $\nu_t(u,v) \in [0,1]$ can be considered as a breaking coefficient that intuitively represents the probability that the tear will go through this pixel. The parameter $p \in [0,1]$ lets us adjust the blend between the two functions to achieve a desired rough appearance of the tear.

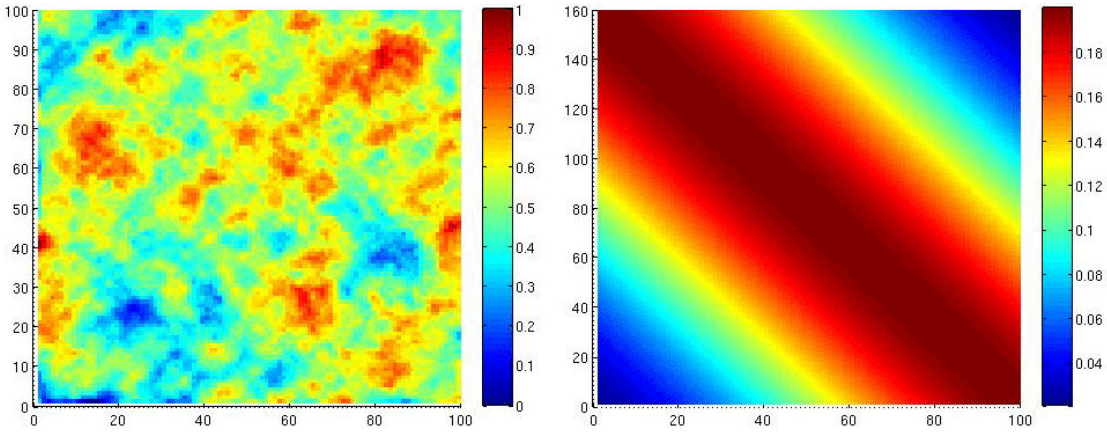


FIGURE 3.7: Variation in the tearing direction is produced with the combination of 2D Perlin noise (left) with a 2D Gaussian function (right) aligned with the current tear-trajectory direction.

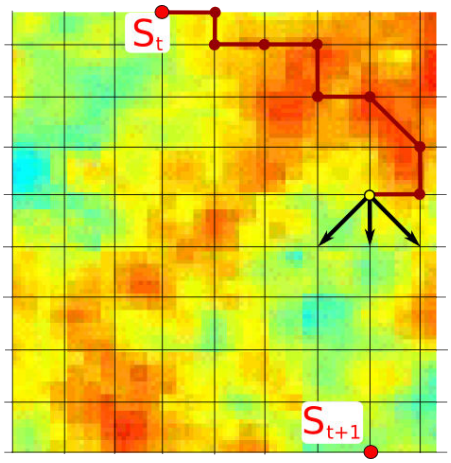



FIGURE 3.8: **The path between $s_t$ and $s_{t+1}$** is computed by iteratively choosing among the neighboring pixels that are closer to $s_{t+1}$ the one with the highest breaking coefficient.

From this texture, the tear details between $s_t$ and $s_{t+1}$ can be chosen as the shortest path, but we prefer a more intuitive and simple approach. Going from the pixel representing $s_t$, we iteratively choose the neighboring pixel with the greatest breaking coefficient until we reach the destination $s_{t+1}$ (see Figure 3.8). To ensure that we actually reach $s_{t+1}$ without going backward, we only consider the pixels in directions that form an angle smaller than $\frac{\pi}{2}$ with the vector between the current pixel and the destination $s_{t+1}$.

We can control the resolution of the details by adapting the resolution of the texture.

Note that we represent here the fibers by a Perlin noise, but it is possible to replace it by any texture as, for example, an anisotropic pattern obtained from real data.

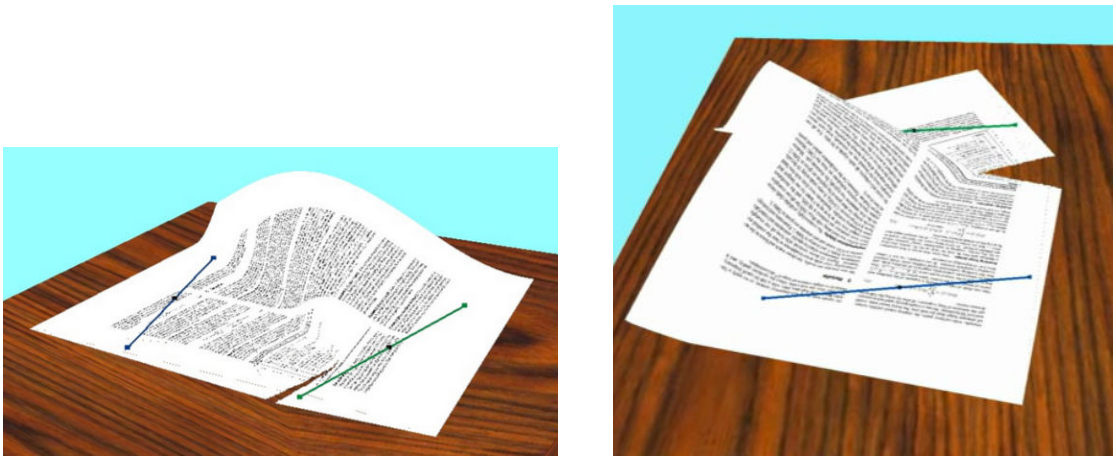


FIGURE 3.9: **Results of cone.** Two different views of the generalized cone.

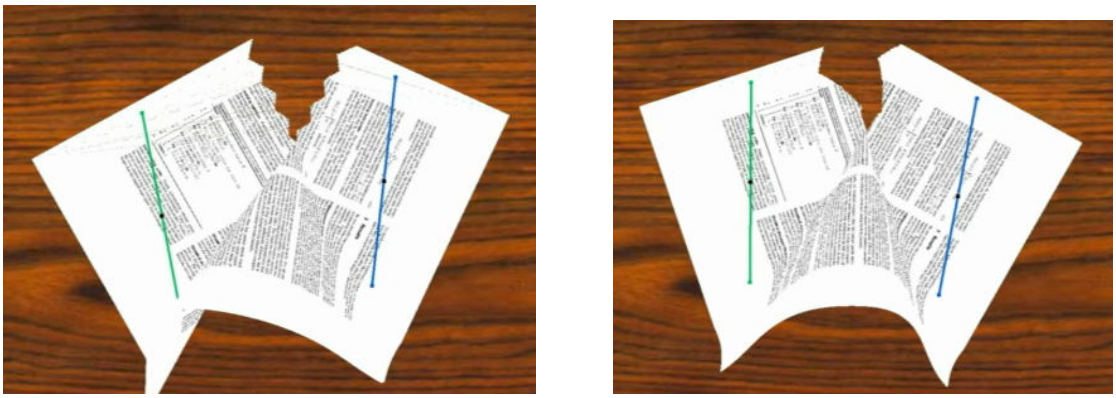


FIGURE 3.10: **Results on the path.** (left) Path obtained by moving successively each hand, (right) path obtained for three different motions of the hands.

FIGURE 3.11: **Results on details.** A closer view of the geometry of the tear.

### 3.2.3 Results and discussion for the procedural model

We use scripted motions to produce different real-time simulations of tearing paper. Figure 3.9 shows two examples with different views of the generalized cone. It is easily possible to control the direction of the tear, see Figure 3.10 for two examples of paths

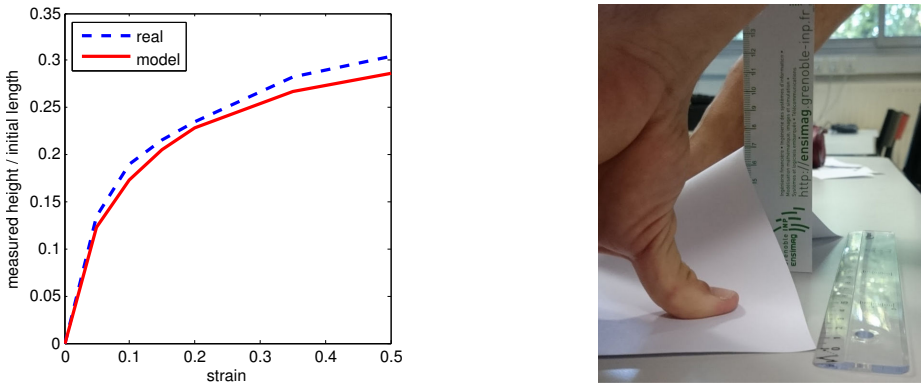



FIGURE 3.12: **Experimental evaluation** shows that our cone deformation model produces results that correspond to real paper.

that we can obtain just by controlling the angle of rotation of the hands. The details of the tear are shown in Figure 3.30. We have fine geometry while keeping the animation fast enough for real-time. A video providing additional examples that also include synthesized tearing sounds can be found at the following link: https://hal.inria.fr/hal-01206764/file/Lejemble_MIG2015_IPSPTS.mp4.

We performed a simple experimental evaluation of our deformation model. Figure 3.12 shows a comparison of the cone height produced by our model with that of a real sheet of paper. Specifically, we compare the real and simulated cone height normalized by original distance between the thumbs at different amounts of strain (i.e. compression normalized by the original distance between the thumbs). The results suggest that the choice of the magnitude of Hermite curve tangents in our model works well for the scale of paper and our particular interactions (e.g. thumb placement as shown in the figure). We note that our model could be improved by making the tangent magnitude a function of the type of paper and the original distance between the thumbs.

Note that our examples in the video are accompanied by the sound of the tearing paper, and explain how this sound is generated in Chapter 4.

We have presented in this section a first geometric model for tearing paper in real time. Although we obtain plausible visual and even audio results, these are limited to the described scenario. Note that the method was originally designed to be used on multi-touch tablets or tables with limited user positional inputs, therefore in this context this scenario may be fully sufficient. For more general applications however, the position and number of fingers may not longer be restricted to the described case. Then tearing and crumpling may occur at the same time, and the tearing computation may not be longer fully described as a procedural function of the finger positions. The next section describes a new model compatible with crumpling including physics based computation

of the tearing shape in order to model more general scenarios while still being very efficiently computed.

## 3.3 General case of paper tearing

The fracture of any material, either thin or solid, is a difficult problem. But modeling paper being torn is a particularly challenging one. For most material, the fracture pattern can hardly be intuitively predicted or controlled. For example, thin materials less rigid than paper, as fabrics, often undergo large elastic and plastic deformations before being torn, making it hard for someone to predict the path followed by the tear. In a different way, it is also difficult to predict the fracture pattern of a solid object being broken, as the fracture usually happens too fast for the eyes to follow. On the contrary, tearing paper is a rather slow process and the small plastic deformation at the tip of a tear is hardly noticeable. Most people can intuitively control the direction of propagation of a tear in paper. And thus a user should expect the same kind of intuitive control over the tearing of virtual paper.

In the previous section, we proposed a model that procedurally creates a path that a user might expect, based on observations in a particular case. The method presented in the rest of this chapter can be seen both as a generalization of this procedural method and as an additional improvement of our crumpling model. Indeed, we couple the crumpling model with a generalized version of the procedural method of tearing. If the basic idea of separating the general direction of the tear and the details of the path stays the same, the propagation based on the motion of the hands is not valid anymore in the general case. We base our new solution on previous studies of tears in paper. As for the crumpling, our method is specifically dedicated to paper or paper-like materials since the solutions adapted for a wider range of material struggle to reproduce the very specific and complex behavior of paper.

Our contributions are the following:

- **potential tearing points:** we use the information provided by the geometric structure of the crumpling method to identify a small number of points where a tear can potentially be created or propagated. We thus limit the number of points that need to be tested.
- **propagation:** we propose a new criterion for deciding when and how a tear propagates by reducing the situation to a known case for which we know a 2D geometric solution.

- **texture-based details:** we propose an efficient representation of the details of the path, based on the texture rather than on the triangulation in order to keep the triangle mesh coarse.

### 3.3.1 Hybrid model for tearing

**Physics of the tearing of paper.** Most of the features of our model rely on studies from Physics or Material Sciences researches, rather than on observations as the procedural model proposed Section 3.2. Roman reviews in [Rom13] numerous studies on tearing brittle thin sheets and shows that crack paths in material that can be approximated as inextensible and infinitely flexible are highly reproducible and seem to follow geometric rules. This leads us to base our computation of the propagation direction of a tear on few simple geometric rules. In the context of linear elastic fracture mechanics (LEFM), the propagation behavior of a tear is mainly influenced by the stress field around the tip [Fre98, LG03]. We use this assumption and simplify our tearing model to only take into account the forces applied on the tip of the tear, and not the one applied on the other part of the surface. As for the procedural model proposed in the previous section, the small details are handled by a stochastic procedural algorithm.

The Griffith criterion [Gri21] is used in many works to predict if a crack should propagate. The condition for a crack to propagate is to have:

$$E_r >= E_f, \tag{3.3}$$

where $E_r$ is the energy release rate, i.e. the energy released by the propagation of the tear, and $E_f$ is the fracture energy, i.e. the energy required to create a new free surface in the material. We use this criterion in our method and explain how we compute $E_f$ in Section 3.3.2 and $E_r$ in Section 3.3.3.

Roman [Rom13] looks into several cases of crack paths following geometric rules. For instance, he studied the case of pulling a flap or pushing a cylindrical object through a sheet of paper. In particular, he examines the propagation of a tear when pulling away with two points, $A$ and $B$, of the piece of paper (see Figure 3.13). In this case, he showed that the energy release rate can be computed as:

$$E_r = F \cdot 2\, cos(\frac{\theta}{2}), \tag{3.4}$$

where $F$ is the amplitude of the force applied on the pulling points and $\theta \in [0, 2\pi]$ is the angle between $\vec{AS}$ and $\vec{BS}$ (see Figure 3.13) where $S$ is the tip of the tear. In addition, the direction of propagation is given by the bisector of the vectors $\vec{AS}$ and $\vec{BS}$. So when

tearing is created by two pulling points which are fixed with respect to the pattern of the paper, the tear trajectory follows a hyperbola with focal points $A$ and $B$. Note that this case is only valid for isotropic paper, and the complete study for anisotropic paper can be found in [OKe94]. In the present thesis, we take the assumption that a general tearing scenario can be equivalently modeled by two opposite forces acting locally on the tip. Under this assumption the propagation of the tear can be computed on the 2D pattern, reducing therefore the complexity of the problem (see Section 3.3.3).

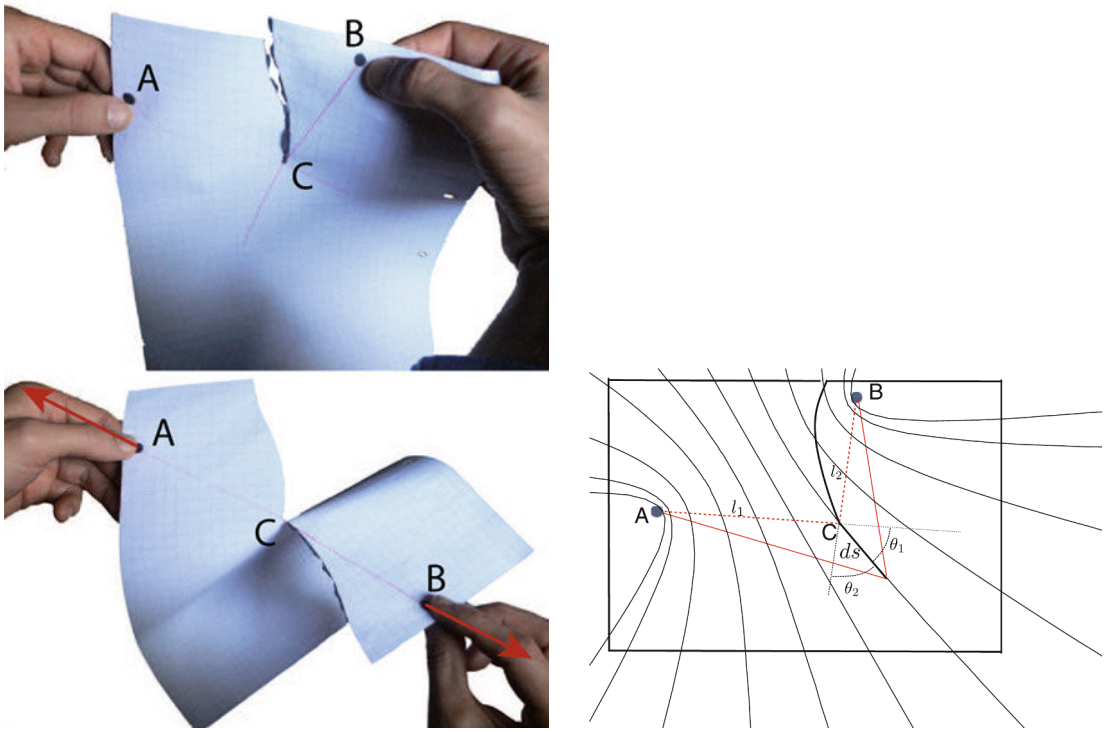



FIGURE 3.13: **Tearing hyperbolae:** (right) Two points on a flat sheet are pulled away from each other. The crack in $C$ propagates when the sheet is pulled by points $A$ and $B$. (left) The optimal direction bisects the angles $ACB$ ($\theta_1 = \theta_2$), so that the possible trajectories are hyperbolae, with focal points in A and B. Pictures from [Rom13].

Most of the existing works studying the propagation of tears use a predefined notch to initiate the tear. But we aim in this work to develop an interactive model where the start of the tear cannot be pre-defined. Phenomenological observations lead us to make the hypothesis that most often a tear starts on a so-called singular point of the border of the paper. The stress tends to focus on d-cones [AP97, Wit07] making these points more likely to be under large stress and therefore to be the starting point of a tear. Although it is possible to create a tear by pulling a piece of paper in opposite directions while keeping it flat, one can observe than the force required to initiate the tear must in this case be quite large as the stress spreads on a large area. In our work, we therefore consider only singular points as potential starting points for a tear.

**Overview.** Our tearing process relies on the crumpling model explained in Chapter 2 for the deformation of the surface. The tearing-related computations are done for each frame on the surface computed by the crumpling model. The tearing method is summarized as follows, see also Figure 3.14.

For each frame $t$ of the animation:

- **Step 1: Potential tearing points** (see Section 3.3.2)
  Get a list of potential, or candidate, tearing points and their associate resistance energy to tearing (i.e: the fracture energy at this point).
- **Step 2: Propagation of tears** (see Section 3.3.3)
  For each potential tearing point $s_t$:
  - approximate forces around $s_t$ into two opposite forces.
  - determine if a tear must be generated or propagated at this point, and if so, compute the position of the next tip of the tear $s_{t+1}$.
  - if needed, remesh to add the tear to the boundary of the paper.
- **Step 3: Adding details** (see Section 3.3.4)
  if the tear is propagated:
  - compute the detailed path between $s_t$ and $s_{t+1}$ using a fiber texture.
  - create the path on the texture of the paper.
  - eventually update the fibers texture to model damages.
- **Physical simulation step** to update the mesh surface geometry with respect to the internal and external forces.

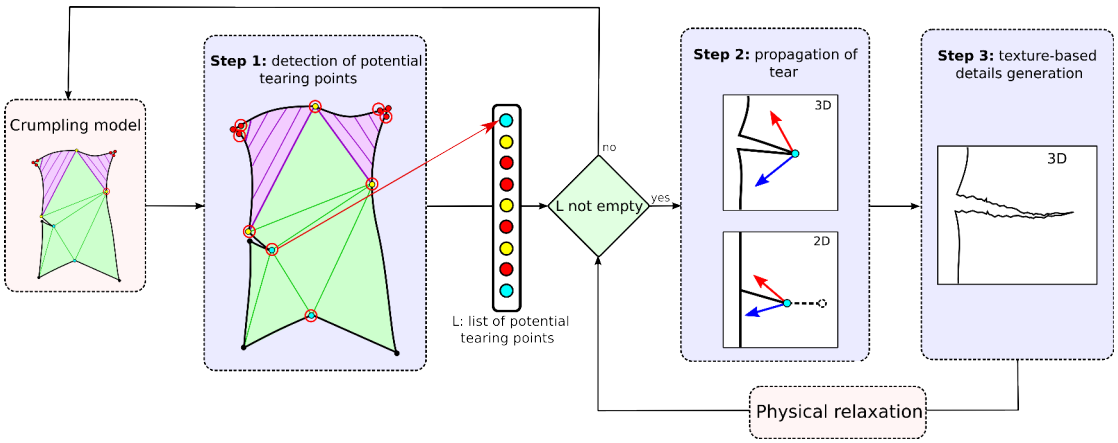



FIGURE 3.14: Overview of our method obtained in adding three extra steps to the existing model of crumpled paper introduced in the previous chapter.

### 3.3.2 Singular points as potential tearing points

**Potential tearing points** Finding the location of the starting points of a tear is an interesting problem that has not been explicitly studied in Computer Graphics so far. Most of the works dealing with the tearing of thin shells test all the points of the mesh to determine which ones should be torn. In addition to be expensive, this may lead to inaccurately positioned starting point or unwanted branching. This kind of artifacts is particularly noticeable in the case of paper since the tearing is generally a rather slow and well controlled process.

We saw in Chapter 2 that the stress tends to concentrate into singular regions, that we approximated by singular points, leading damages to the fibrous structure of paper. We can observe by tearing paper that the usual location for a tear to be initiated (see Figure 3.15) corresponds to those singular regions of the border of the surface of paper.

In our model for crumpling paper, we not only detect the interior singular points where the fibers' links are broken, but also the singular points (in yellow in Figure 3.15) on the border of the paper at the junction between some flat and curved regions. We consider those points to be our potential starting points of a tear.

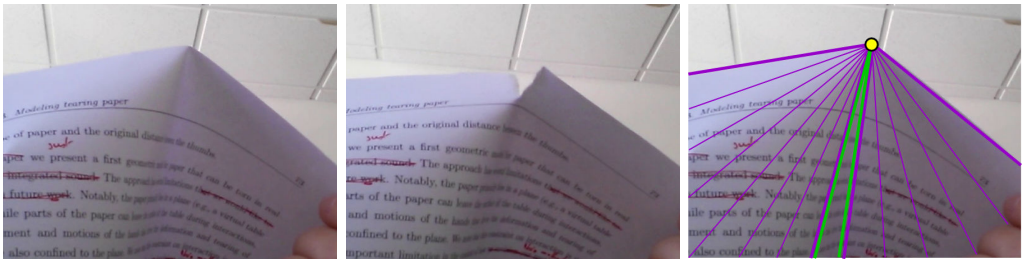


FIGURE 3.15: The starting points of the tears seem to be in the singular regions on the border of the paper. These correspond to the junction points in our crumpling model.

At each frame, we need to check only the following small number of points as potential tearing points (see Section 3.3.3):

- the junction points (yellow in Figure 3.14), points on the boundary of the paper belonging to at least on flat region and one curved region.
- the points delimiting the position of the fingers (red in Figure 3.14), as junction between the flat part of the surface imposed by the pressure of a finger and the rest of the surface.
- the points where the boundary of the 2D pattern of the piece of paper makes an inward angle (sky blue in Figure 3.14). They usually become singular when applying a tearing motion near them but to ease the work of the simulation we

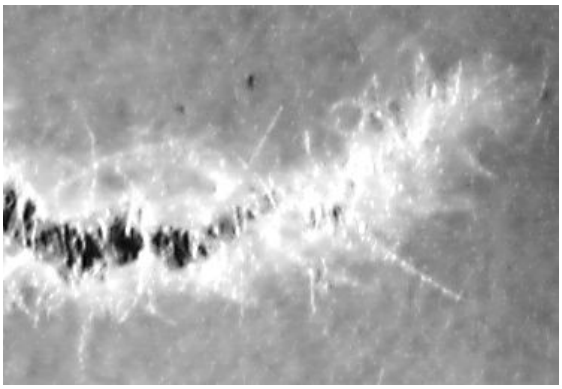


FIGURE 3.16: The fracture process zone is rather large and diffuse and extends well ahead of the region where noticeable microcracks or voids can be seen. Image from [AN06]

consider them as singular by default. In this way, all the tear tips are always considered as singular.

**Fracture energy of the potential tearing points.** We define the fracture energy $E_f$, representing the threshold above which the energy release rate $E_r$ is sufficient for generating or propagating a tear, for each potential tearing points. We model the resistance of the fibers by a 2D texture $T_{fibers}$ over the surface. Similarly to the procedural method described in Section 3.2, we use a Perlin noise as fibers' texture. The parameters of noise can be easily modified to adapt the appearance of the details of the tear. Note that it would be possible to replace the Perlin noise by a more physically accurate representation of the distribution of the fibers as a future work.

In order to take into account the plastic damage occurring at the tip of a tear (see Figure 3.16), we associate a damage parameter $d_p$ to each potential tearing point $p = (u, v)$ ($u$ and $v$ being the coordinates of the point in the 2D pattern). If $p$ is a tip of a tear, $d_p = D$, where $D \in [0, 1]$ is a constant value chosen by the user to define how easily of tear can be propagated. For any other point, we have $d_p = 1$. The fracture energy of $p$ is then defined as

$$E_f(p) = C\ T_{fiber}(u, v)\ d_p,$$

where $T_{fibers}(u, v) \in [0, 1]$ is the value of the fibers' texture at the position $(u, v)$ and $C$ is a constant modeling the general resistance of the paper. We use $C = 0.005$ in our examples.

### 3.3.3 Generation and propagation of tears

For each potential tearing point we compute at each time step whether it has to be torn or not. Note that our computation is the same for a tip currently torn than for any other potential tearing points. Moreover, we proceed the same way whether a tear is created or propagated. The first problem to tackle is find a criterion to decide whether a tear needs to be propagated –or created– or not. The second problem, when the tear is propagated, is to determine the position of the tip of the tear at the next time step.

Let us call the potential tearing point to be tested $s$. We solve both problems by approximating the 3D in-plane forces applied on $s$ by the physical simulation by two opposite forces. We can then relate to the case of two pulling points and determine the propagation using the 2D solution described in Section 3.3.1. We neglect the bending forces as they are several orders of magnitude weaker than the strain forces.

**Computing two opposite forces.** We want to separate the strain forces applied on $s$ into two clusters, each one gathering forces directed in a similar direction. To this end, we inspire from classical clustering methods and use the following iterative algorithm. This computation takes place in the 3D space. We obtain from the physical simulation the forces applied on $s_t$ by the deformation of each adjacent triangles –these forces are based on Green strain described in Chapter 5. We use the direction of the tear as first estimate for the direction of separating the two clusters, $\mathbf{t}$ is computed as the bisector of the tear lips (as they are often along different directions during tearing) or as the bisector of the directions of the boundary at each side of $s$ (see Figure 3.17). We have then $\mathbf{b}^0 = \mathbf{t}$.

At the iteration $i$, we call $\mathbf{F}^i_{left}$ and $\mathbf{F}^i_{right}$ the sum of the forces of each cluster and $\mathbf{b}^i$ the direction bisecting $\mathbf{F}^i_{left}$ and $\mathbf{F}^i_{right}$ (see Figure 3.17). For each iteration $i$, we compute $\mathbf{F}^{i+1}_{right}$ and $\mathbf{F}^{i+1}_{left}$ (see Figure 3.18) as being respectively the sum of the forces of each side of the plane $\mathcal{P}^i$ defined by $\mathbf{n}$, the normal of the surface at $s$, and $\mathbf{b}^i$. Concretely, we compute the direction $\mathbf{d}^i$ orthogonal to $\mathcal{P}^i$:

$$\mathbf{d}^i = \mathbf{b}^i \wedge \mathbf{n}.$$

Then

$$\mathbf{F}^{i+1}_{left} = \sum_{\mathbf{f}.\mathbf{d}^i > 0} f \text{ and } \mathbf{F}^{i+1}_{right} = \sum_{\mathbf{f}.\mathbf{d}^i < 0} f.$$

We stop the iterative algorithm when $\mathbf{b}^i = \mathbf{b}^{i+1}$ which usually takes less than 10 iterations, since there are only a small number of forces, and set $\mathbf{F}_{left} = \mathbf{F}^{i+1}_{left}$ and

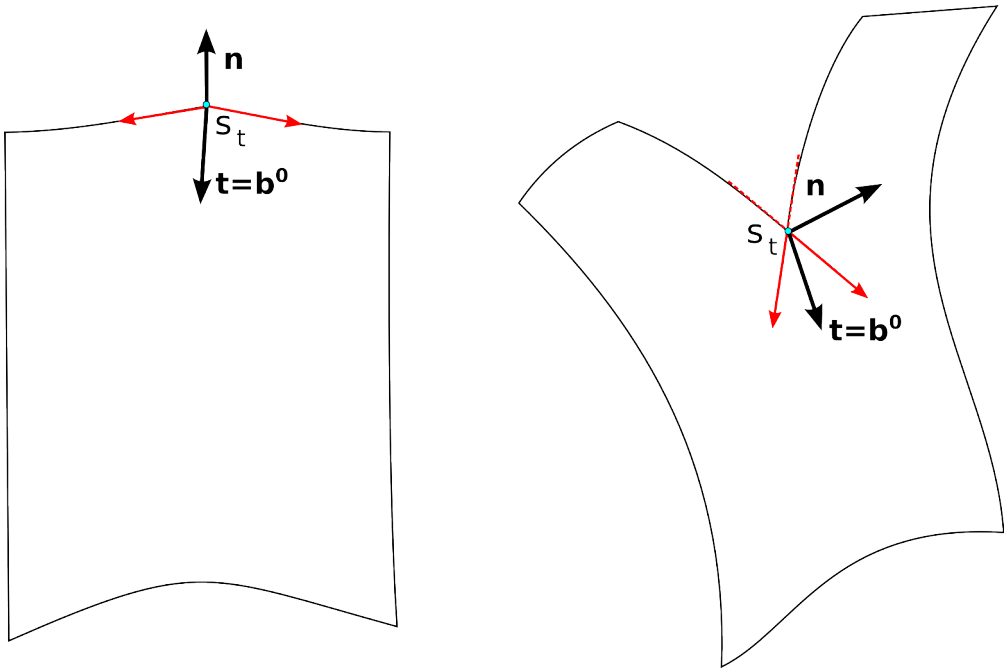



FIGURE 3.17: The direction **t** of the tear is computed as the bisector of the directions of the boundary at each side of $s$.

$\mathbf{F}_{right} = \mathbf{F}^{i+1}_{right}$. Otherwise we limit the number of iterations to 10 to be sure to avoid any infinite loop.

**Computing the next tear tip.** Once we have obtained the two opposites forces $\mathbf{F}_{left}$ and $\mathbf{F}_{right}$, we can consider them as forces applied by two pulling points, and compute the energy release rate and the direction of propagation as described before in Section 3.3.1.

The forces composing $\mathbf{F}_{left}$ and $\mathbf{F}_{right}$ are in-plane forces, so we can map them in the 2D pattern to get $\bar{\mathbf{F}}_{left}$ and $\bar{\mathbf{F}}_{right}$.The direction $\bar{\mathbf{d}}_{prop}$ (shown in Figure 3.19) in which the tear should propagate to release the maximum of energy is then the bisector of $\bar{\mathbf{F}}_{left}$ and $\bar{\mathbf{F}}_{right}$ and the corresponding energy release rate (derived from 3.4) is:

$$E_r = (F_{left} + F_{right})\ cos(\frac{\theta}{2}),$$

where $\theta$ is the angle $\in [0, 2\pi]$ between $\bar{\mathbf{F}}_{left}$ and $\bar{\mathbf{F}}_{right}$.

If the energy release rate $E_r$ is superior to the fracture energy $E_f$ at the point $\mathbf{s}$ then the tear propagate and a new tip of the tear is created in the direction $\bar{\mathbf{d}}_{prop}$. We adjust the speed of propagation according to the energy released by the propagation:

$$\bar{\mathbf{s}}_{t+1} = \bar{\mathbf{s}}_t + V\ E_r\ \bar{\mathbf{d}}_{prop},$$

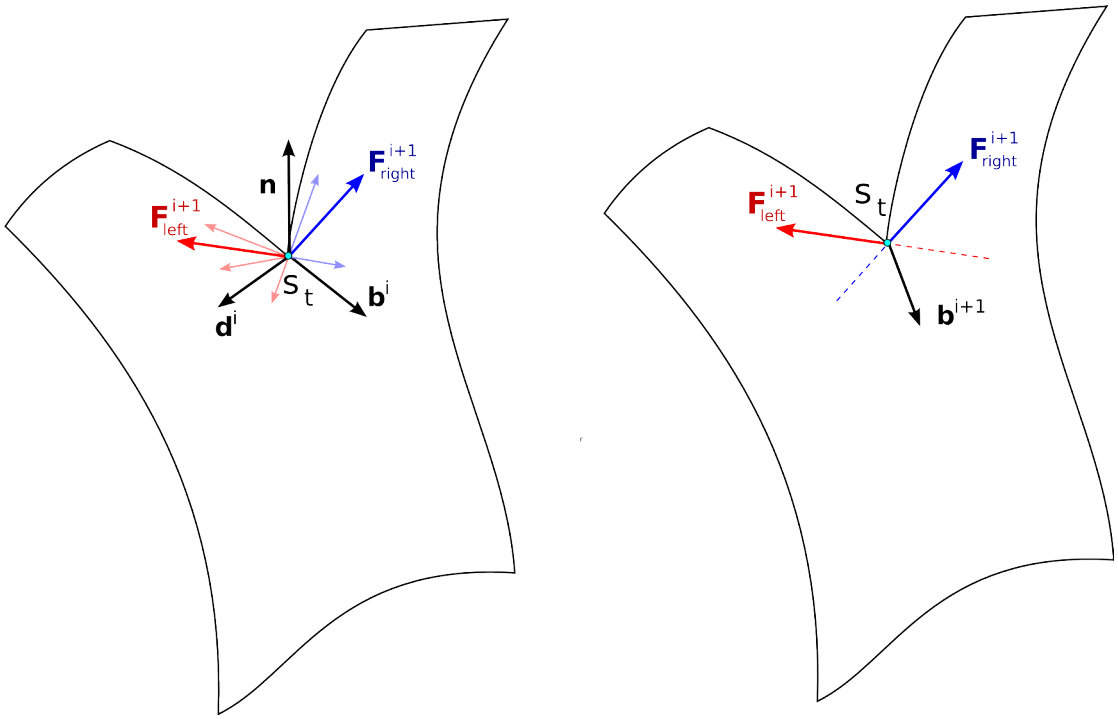



FIGURE 3.18: **An iteration of the clustering algorithm.** (left) $\mathbf{F}^{i+1}_{left}$ is computed as the sum of the forces (in red) of the left side of the plane defined by $\mathbf{b}^i$ and the normal to the surface and $\mathbf{F}^{i+1}_{right}$ as the sum of the forces on the right side (in blue). (right) $\mathbf{b}^{i+1}$ is then computed as the bisector of $\mathbf{F}^{i+1}_{left}$ and $\mathbf{F}^{i+1}_{right}$.

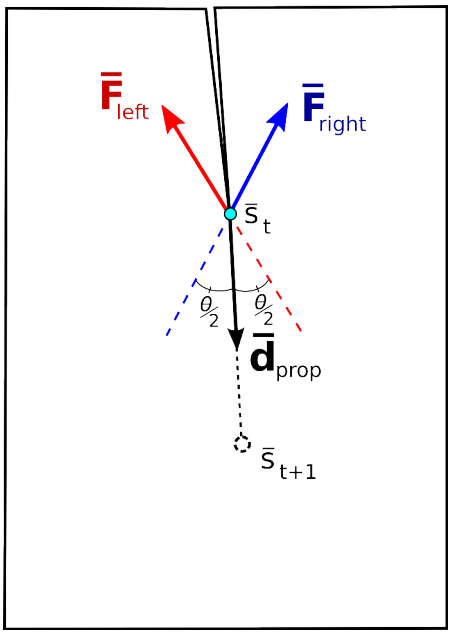



FIGURE 3.19: The propagation of a tear depends on the 2D angle $\theta$ between $\bar{\mathbf{F}}_{left}$ and $\bar{\mathbf{F}}_{right}$. If propagated, the next tip is found in the direction $\mathbf{d}_{prop}$, the bisector of $\bar{\mathbf{F}}_{left}$ and $\bar{\mathbf{F}}_{right}$.

where $V$ is a constant parameter enabling the user to tune the speed of propagation of the tear.

The 3D position of $\mathbf{s}_{t+1}$ is finally computed by projecting $\bar{\mathbf{s}}_{t+1}$ onto the 3D surface.

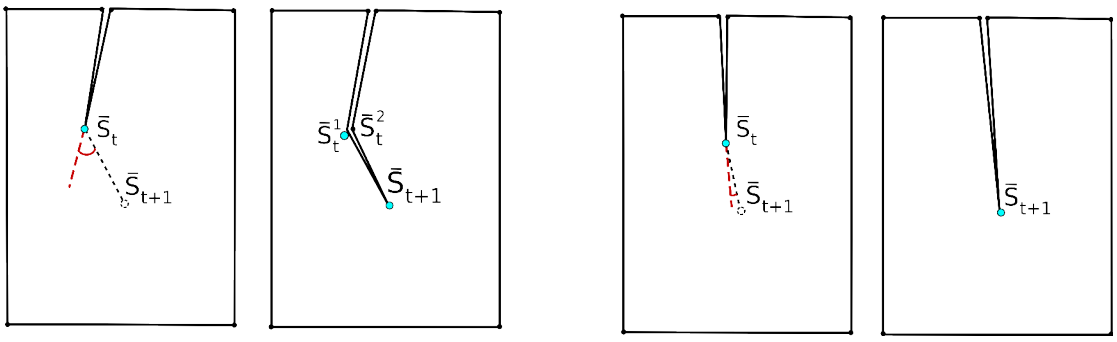



FIGURE 3.20: Depending on the angle made by the tear at $s_t$, we either (right) split or (left) remove it.

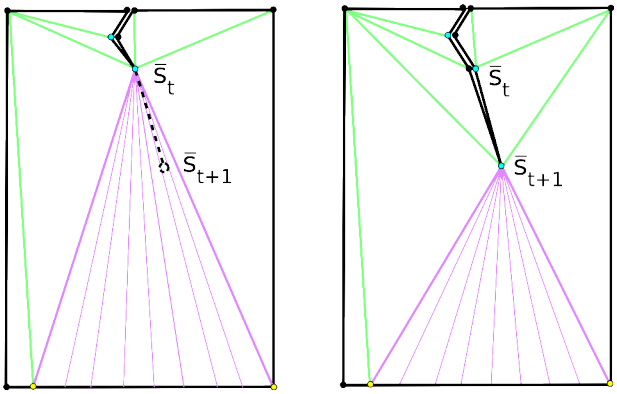



FIGURE 3.21: Case 1 of the remeshing scheme after the propagation of a tear: $s_t$ is the apex of a cone and $s_{t+1}$ is inside this cone. After remeshing, $s_{t+1}$ is the apex.

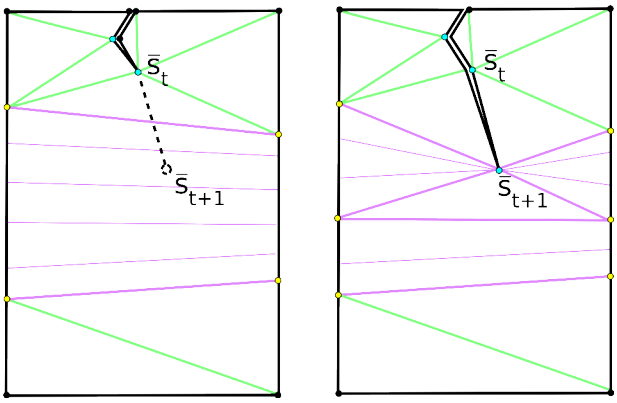



FIGURE 3.22: Case 2 of the remeshing scheme: $s_{t+1}$ is inside a curved region.

**Remeshing.** We aim at keeping the mesh consistent and as coarse as possible while propagating a tear. As $s_t$ is a point defining the boundary of the paper, it is a point of the physical simulation, and at the same time a point belonging to the geometric structure used for the crumpling model. We either split $s_t$ into two points, $s_t^1$ and $s_t^2$, if the angle, in the 2D pattern, made by the general path of the tear at $s_t$ is $> \varepsilon_a$ or just remove it otherwise (see Figure 3.20). We choose to have $\varepsilon_a = 0.1$ gradient in our

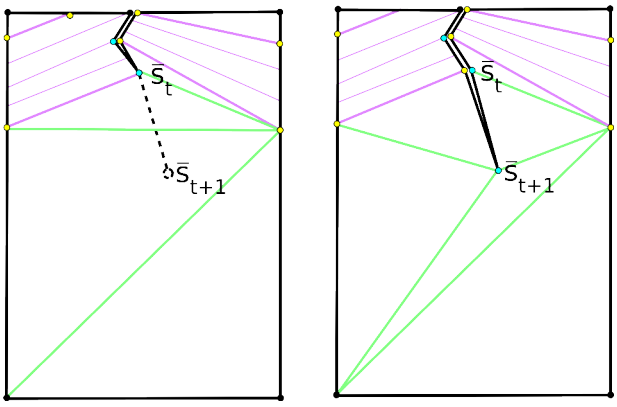



FIGURE 3.23: Case 3 of the remeshing scheme: $s_{t+1}$ is inside a flat region.

examples. In this way, if the tear goes straight, $s_t$ is no longer a point defining the boundary and does not belong to the geometric structure anymore. Thus we can keep the representation of the flat regions in the geometric structure very coarse. Note that for the mesh used by the physical simulation, the newly created edges are sampled as any other edge of the boundary of the paper. The suppression of $s_t$ does not hamper the degrees of freedom of the physical simulation. In the other case, when the tear forms an angle, one of the two points, $s_t^1$ or $s_t^2$, has an inward angle of the boundary. This means that it is a potential tearing point and so that the tear could possibly branch at this point.

We also need to keep the geometric structure used by the crumpling model consistent. Three main cases occur:

**Case 1** $s_{t+1}$ is inside a generalized cone (i.e. $\bar{\mathbf{s}}_{t+1}$ is inside the 2D pattern of the cone) whose apex is $s_t$, in which case we modify the cone such that its apex becomes $s_{t+1}$ (see Figure 3.21).

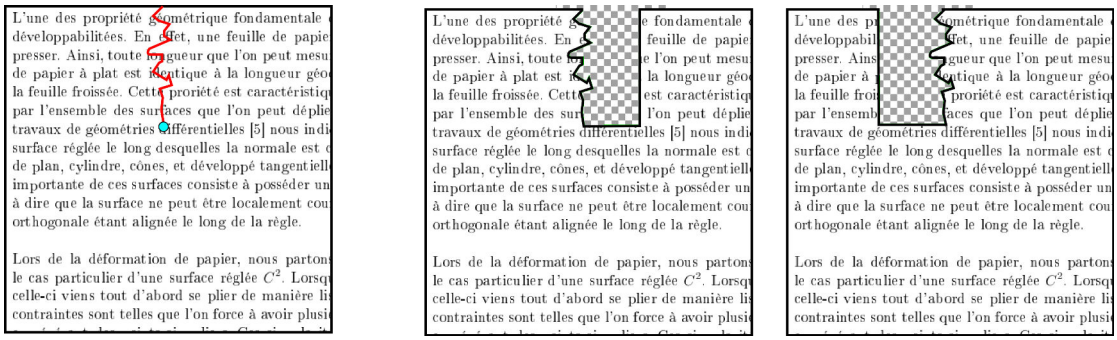
**Case 2** $s_{t+1}$ is inside a curved region and $s_t$ is not the apex of a generalized cone. We apply a remeshing scheme similar to the one used when creating a singular point inside a curved region (see Figure 3.22).

**Case 3** $s_{t+1}$ is inside a flat region. We just update the coarse triangulation representing the flat region in the geometric structure (see Figure 3.23).

### 3.3.4 Texture-based details

We compute the detailed path between two successive tear tips by using the algorithm explained in Section 3.2.2 that consists in finding a low energy path in the combination (described by 3.2) of a fibers texture $T_{fibers}$ and a fracture-centered stress field $G_t$.

**Representation of the detailed path as textures.** Let us call the initial texture used to render the piece of paper $T_{paper}$. To effectively represent the detailed path we modify this texture. When a new tear is created, we compute two textures from $T_{paper}$, $T_{right}$ and $T_{left}$, that are respectively transparent on each side of the detailed path of tear (see Figure 3.24). The triangles that are adjacent to the tear are textured by either $T_{right}$ or $T_{left}$ according to which side of the tear they belong to.



$T_{paper}$ $T_{left}$ $T_{right}$

FIGURE 3.24: The two textures (right) corresponding to the tear (in red) and the initial texture on the left.

When a tear is propagated, the textures $T_{right}$ and $T_{left}$ are updated in order to include the new path. When a tear is branching, two new textures are created for the new branch of the tear from the texture corresponding the parent tear.

The detailed path between two successive tear tips is shared by two triangles. To have all the details of the path represented for both of them, we extend those triangles by a small textured rectangle as shown in Figure 3.25.

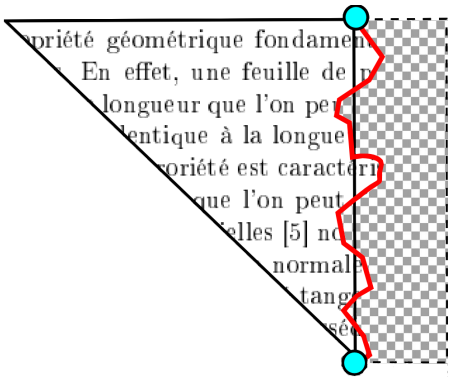


FIGURE 3.25: A support rectangle is used to represent the details outside a triangle bordering a tear.

### 3.3.5 Preliminary results

We present in this section our results for the general scenario including both crumpling and tearing. Note that all the presented results are still at preliminary state and correspond to unpublished work which may be improved in the near future.

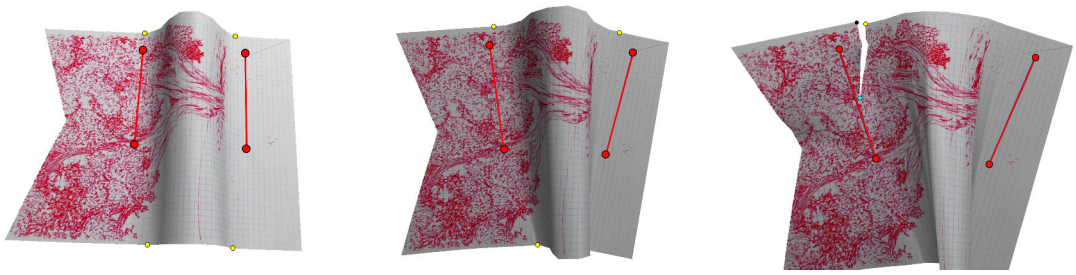


FIGURE 3.26: A tear is created on virtual paper by flattening on border of the bent surface. The deformation is generated by two hands represented each by two red points (modeling two fingers) connected by a red line. The tear appears at a junction point between the curved region and one of the flat ones.

FIGURE 3.27: A tear created in real paper by reproducing the situation of Figure 3.26.

**Validation of the starting points** Figure 3.26 shows a piece of paper bent into a cylinder by two hands modeled by two points (each represented by two red points linked by a red line). Then the two hands rotate to flatten one border of the cylinder until creating a tear. The same experiment is realized with a real piece of paper (see Figure 3.27). We can see that with both virtual and real paper the tear appears at one finger between the curved region and one of the parts maintained flat by a hand.

We can also notice that when a piece of paper is torn near a concave border, the tear usually starts from an inward angle of the border. We reproduce this case in Figure 3.28 and Figure 3.29.

**Validation of the tear trajectory** As explained in Section 4.3.1, and in more details in [Rom13] and [OKe94], when the propagation of the tear is only due to two fixed points being pulled apart, the curve followed by the tear on the 2D pattern is a hyperbola with

focal points given by the two pulling points. It means that at each point $c$ of the curve, the tangent to the curve is the bisector of the lines between each pulling points at the point $c$. In the next experiment, we validate our model with respect to this behavior.

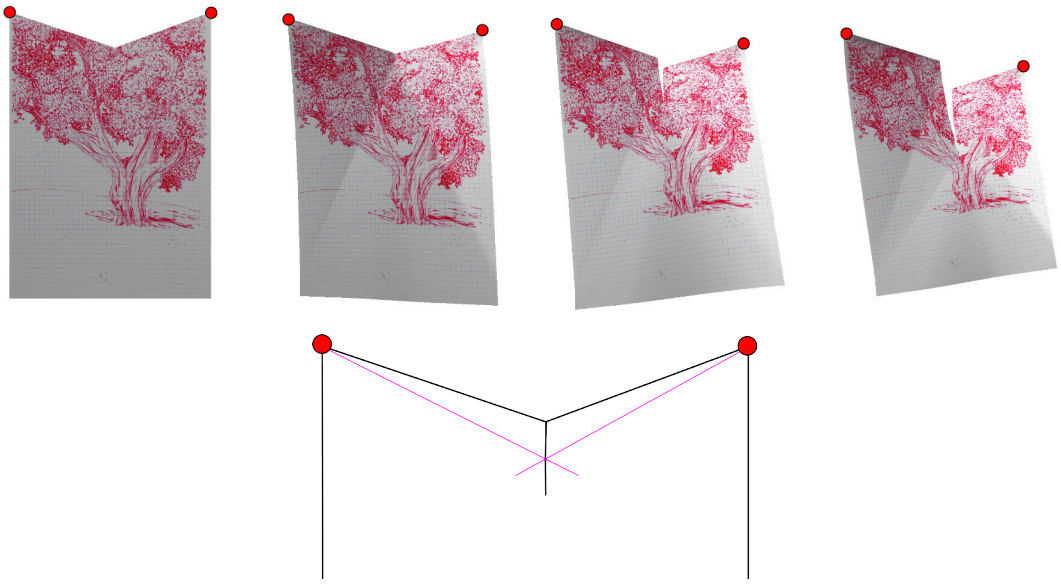


FIGURE 3.28: Two pulling points a equal distance of the starting point of the tear create a straight tear. (top) Evolution of the 3D surface as the tear propagate. (bottom) 2D pattern, the tear path is the mediator of the two pulling points.

Figure 3.28 shows the paper being torn by pulling two points located at equal distances of the starting point of the tear. As seen on the picture, the tear follows the mediators of the line between the two pulling points describing therefore the expected tear trajectory.

Figure 3.29 shows the tear obtained by pulling two points located at different distances from the starting points. The tear follows indeed a curve whose tangent at each point corresponds closely to the expected bisector.

**Details** Figure 3.30 shows the tear with the added stochastic details. We obtain small details without needing a denser mesh.

When the successive tips of the tear are separated by a very small and regular space step, the tear may seems to regular (see Figure 3.30 (top-left)) as the path is defined by successive tips and so goes through each of them. One possible improvement would thus be to allows the path to deviate a little from its defining point in order to take better advantage of the stochastic path method.

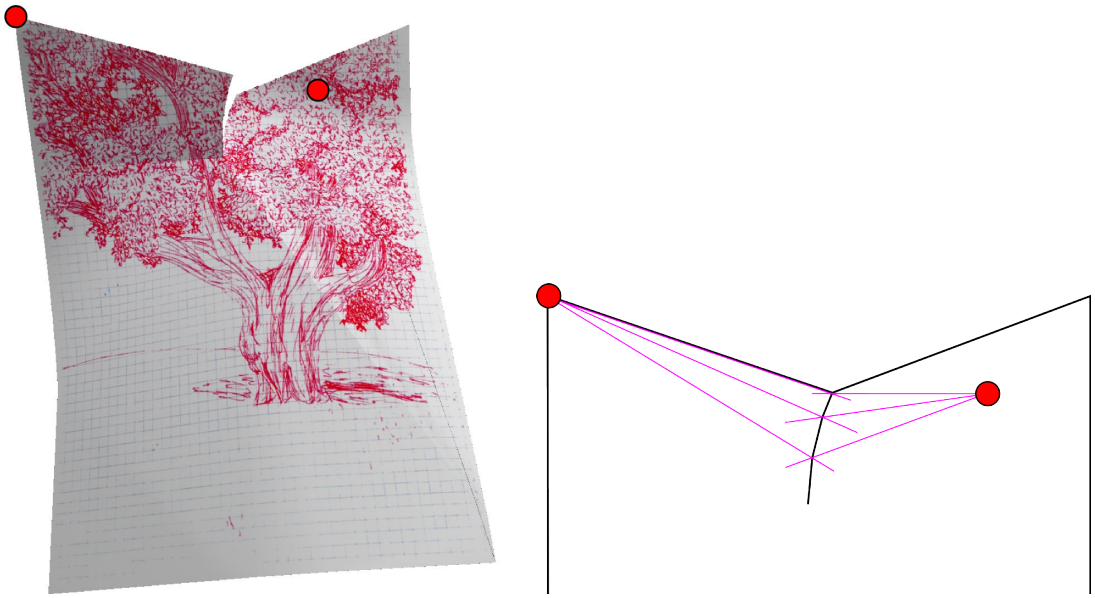


FIGURE 3.29: Two pulling points at different distances of the starting point of the tear create a curved tear. (right) The 3D surface of the torn paper. (left) 2D pattern, at each point, the tangent to the path is approximately cut the angle between the directions toward each pulling points into two equal part.

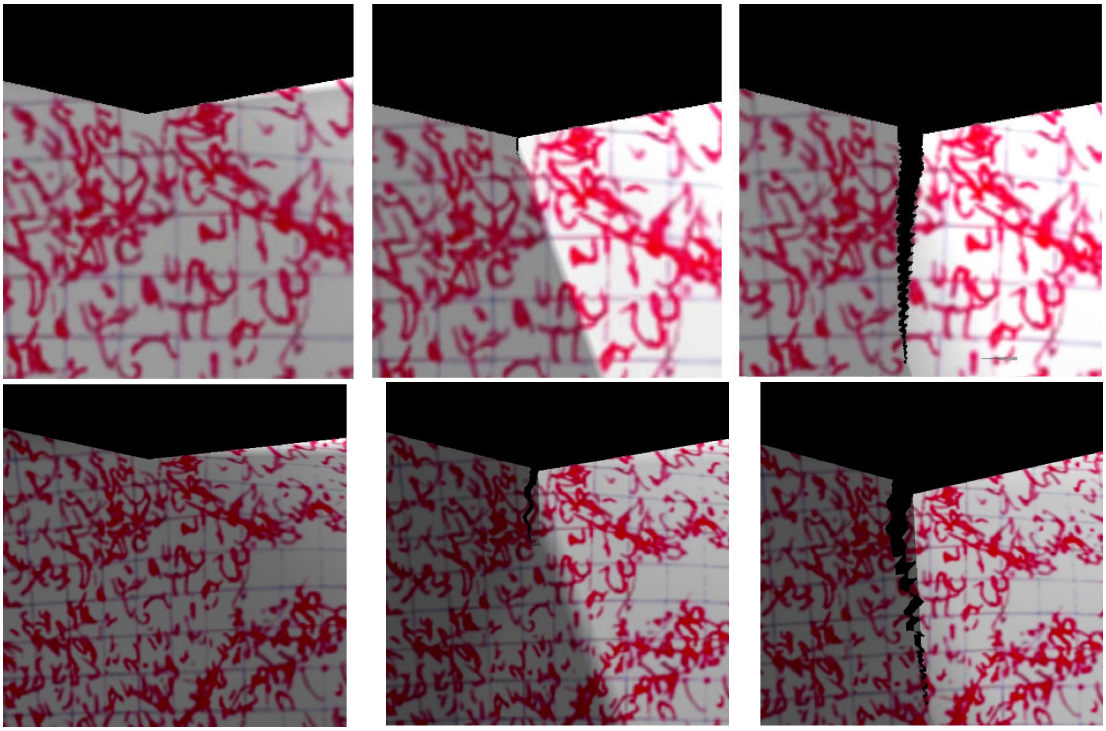


FIGURE 3.30: **Evolution of a tear with stochastic details.** (right) Just before the start of the tear, (middle) first step of the tear, (left) the whole tear.
(top) The speed of the tear is set at $V = 2$. (bottom) The speed of the tear is $V = 10$.

## 3.4 Discussion and conclusion

We presented in this chapter a procedural method for tearing paper in a specific case and a generalization based on our crumpling model and on Material Sciences studies. Although more general scenarios could be explored, the method already provides promising results. Notably we can reproduce some characteristic of paper, geometrically-ruled tearing path which are validated with respect to behaviors described in the physical literature.

In the near future, we plan to improve the appearance of the details by generating damages in the fiber texture around singular regions representing broken fibers and where the fibrous structure should thus be weaker. Another idea would be to use several fiber textures to represent multiple layers of fibers. In this way, the same tear would correspond to several detailed path and then represented by layered semi-transparent textures.

# Chapter 4

# Sound synthesis for paper

*"The empty vessel makes the loudest sound."*
William Shakespeare, *Henry V*, 1599.

## 4.1 Introduction

In the two previous chapters, we have introduced a model to simulate paper that creates visually compelling animations. Yet to aim at a fully immersive experience, a synthetic model needs to stimulate also the other senses, particularly the hearing. Realistic auditory feedback plays an important part in the creation of compelling virtual experiences in providing to the listener feedback information about the environment and material involved.

The automatic generation of sound for virtual animation is still a recent research area. Generating sound for video games and interactive applications is a particularly difficult problem as the sound has to be synthesized interactively while only using limited computing resources –2 to 50 MB of memory and around 10% of the total CPU.

The case of sound synthesis for paper material is particularly challenging as paper is not only extremely deformable – preventing from using rigid body simulation – but also, contrary to the sound of cloth garments, the sound of paper depends heavily on its shape –which may change dramatically during crumpling. In addition, paper sound is a complex combination of different sound styles. It is a mix between continuous

noisy sounds produced by frictional sliding, and discrete events produced by geometric bending and crumpling processes. These discrete sounds may also vary between long "*flap*" sounds when the sheet is still smooth, and more short "*clac*" sounds in more crumpled cases. Also tearing sound is again another kind of sound composed of small bursts of energy caused by the breaking of the bonds between fibers. Those characteristic sounds of paper-like material are very familiar to humans – as for example the universally recognized crumpling of money bills.

The work presented in this chapter tackles these challenges. We aim at obtaining a realistic sound model able to synthesize the sound at run time as the user interacts with the virtual paper while taking into account the shape of the paper and its environment. We first introduce a shape and environment dependent sound synthesis for crumpling, bending, and friction. Secondly, we present a method to procedurally synthesize the sound of torn paper.

*The method for synthesizing the sound of crumpling paper described in Section 4.3 has been published and presented at Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2016) [SRJ*16]. This work has also been presented earlier at the "Journées de l'AFIG" in November 2015.*

## 4.2 Digital sound

This section explains some basic notions about digital sound required to understand the rest of the chapter.

**Sampling rate.** The average human hearing range –i.e. the average range of frequencies audible for human ears– goes from 20Hz to 20kHz. To capture a band of frequencies without aliasing, according to the Nyquist criterion, the sampling rate $r_s$ must be such that $r_s > 2\,B$ where $B$ is the highest frequency of the sound. This means that to accurately represent a sound (intended for human hearing), we need a sample rate of a least 40k samples per second –i.e. 1 second of sound is represented in the time domain by 40k samples of the amplitude of the sound, so 40k numbers between $[-1; 1]$. In this chapter, we use a sample rate of 44100Hz which is commonly used in audio, notably for audio CDs.

**Sound spectrum.** A sound can be represented in the frequency domain by its spectrum. For digital sound, the FFT (Fast Fourier Transform) algorithm is usually used to go from the temporal domain to the frequency domain. This algorithm takes a window of N sound samples –so a window of $\frac{N}{r_s}$ seconds– and returns a spectrum $\mathbf{s} = [s_i]$ represented by N complex numbers sampling the frequency band $[0, \frac{r_s}{2}]$. The $i$th value $s_i$

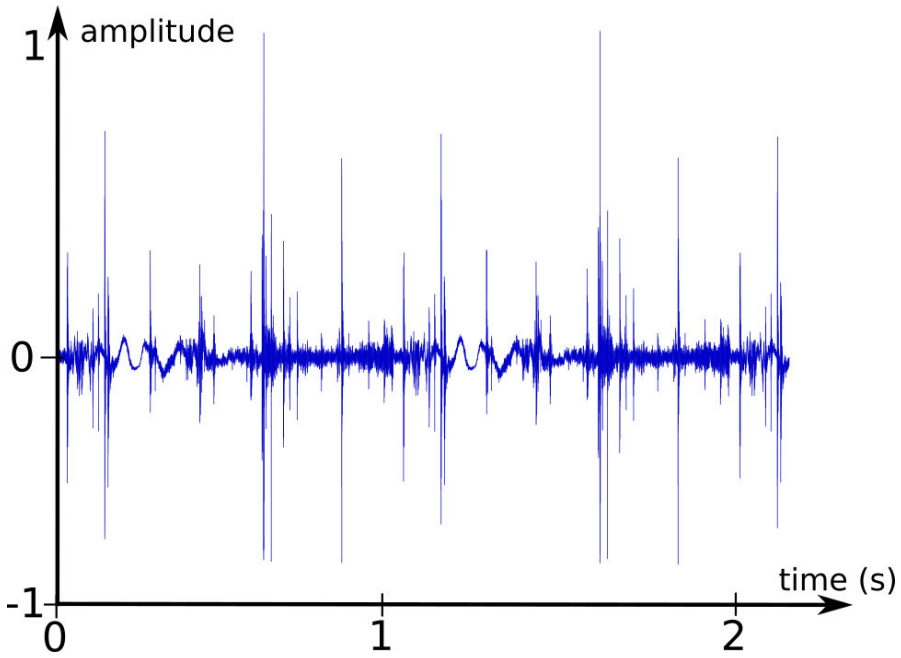



FIGURE 4.1: Sound of crumpling paper.

corresponds to the frequency $f_i = \frac{r_s}{2N} i$. The modulus $|s_i|$ is the amplitude of the sound at the frequency $f_i$ and the argument of $s_i$ is its phase. The power spectrum represents only the amplitude of the spectrum. It is common to compute the average power spectrum of a sound by averaging the spectra obtained for a sliding window spanning the whole sound.

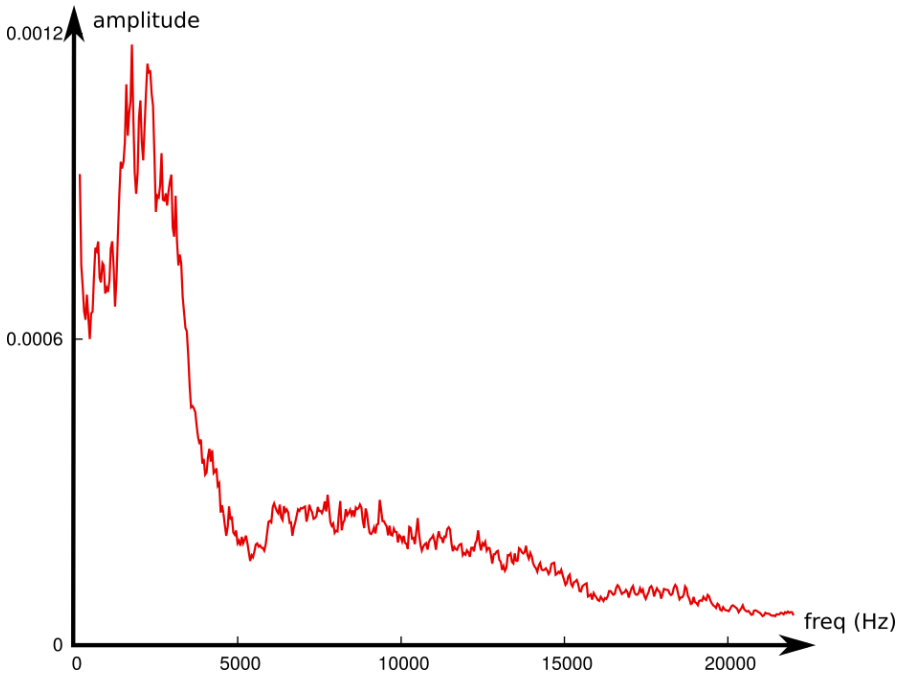



FIGURE 4.2: A spectrum of a sound.

**Spectrogram of a sound.**

A spectrogram represents a time-frequency analysis of a sound: it shows the variation of the power spectrum of a sound over time. It is obtained by computing the power spectrum for a window spanning the sound over time. As shown in Figure 4.3, the horizontal axis of the spectrogram represents the time while the vertical axis are frequencies. The pixel color in position $(t, f)$ encodes the amplitude of the frequency $f$ at time $t$. Note that frequency may be represented on a logarithmic scale in order to model octaves, i.e. a multiplication of frequency by a factor of 2. It is well known that temporal precision is inversely proportional to frequency precision. Therefore the width of the sliding windows is a trade-off between fast time change expected to be captured, and frequency precision.

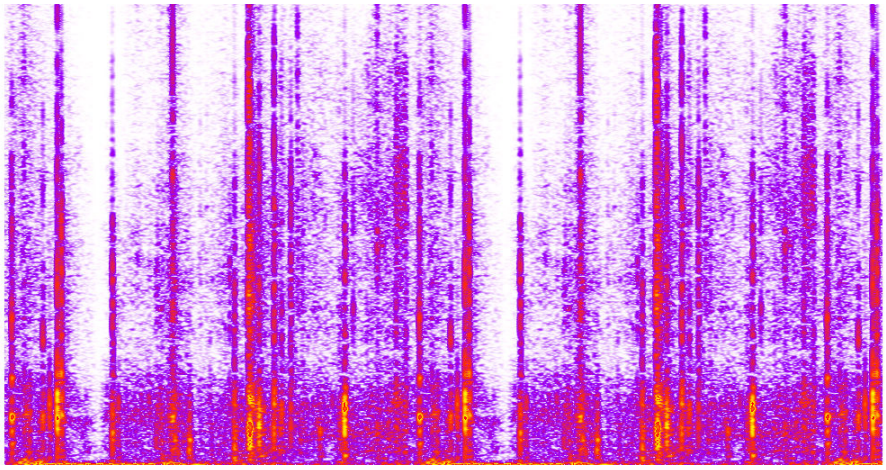


FIGURE 4.3: Spectrogram of a sound of crumpling paper.

## 4.3 Shape-dependent sound synthesis for crumpling paper

For generating the sound of paper being crumpled, we choose to adapt motion-driven concatenative synthesis of sound to the case of paper by introducing the concept of

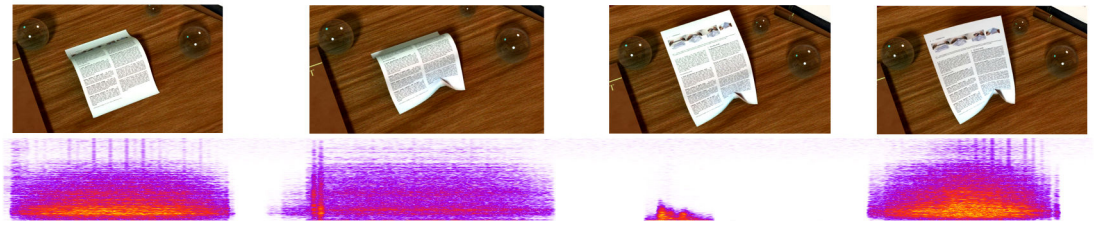


FIGURE 4.4: **Real-time paper sounds:** Our method can automatically synthesize a plausible, synchronized soundtrack (shown as a spectrogram) for interactive simulations of a rectangular sheet of paper. In this 3.3s animation, the back edge of the sheet is held while the front edge slides toward it, thereby curving the paper; then, when the back edge is released and the front edge is pinched, the back edge stands up.

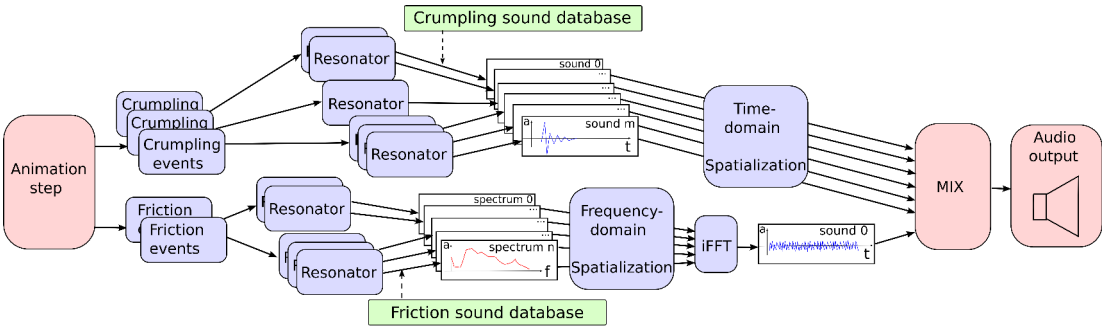



FIGURE 4.5: **Overview:** After computing the sound-producing events, each of them is associated with a set of *resonators*. Each of those *resonators* leads to a sound unit after querying it on a pre-computed sound data-base. Each sound unit is spatialized to adapt to our scene and the final resulting sound is obtained by mixing all the contributing sounds.

geometry-dependent resonators. We also propose a spatialization algorithm in order to handle the environment's influence.

Our approach consists of four steps performed at run-time during the visual simulation of the sheet of paper. First, sound-producing events are detected by analyzing the paper surface animation. Second, each event is characterized using the geometry and the motion of the surface to parameterize our sound models. Third, the sound is synthesized using the estimated model parameters using computed characteristics and two pre-recorded data-bases of primitive sounds: one for the discrete crumpling sounds, and another for the continuous friction sounds. Fourth, the resulting sound is spatialized in 3D to account for the influence of nearby planar surfaces, such as a wall or table.

Our three main contributions are as follows. First, we develop a new shape-dependent model for friction and crumpling sounds. Our model is based on local surface regions we call *resonators*, which characterize the regions where significant sound-related vibrations occur. Second, we propose a two-sided detection and characterization method for both friction and crumpling sounds, that incorporates their dependence on nearby free (unconstrained) boundaries of the sheet, leading to four different sound types: constrained and unconstrained crumpling sounds depending on the presence of a free border in the producing region; and similarly, constrained and unconstrained friction sounds. Third, we propose a new efficient approximation method for sound spatialization, adapted to model the sound of thin-sheet surfaces near flat surfaces, such as tables or walls.

### 4.3.1 Overview

Two kinds of sounds are usually identified for thin-sheet material:

1. **Friction sounds** caused by regions of the surface sliding along either another object of the scene or another part of the same surface. The sound is produced by the sliding part bumping randomly against small irregularities of the other surface and can be approximated by a colored noise (a stochastic signal with a slowly time-varying power spectrum, see Figure 4.6).

2. **Crumpling sounds** generated in regions where the bending direction changes suddenly, such as when the sign of surface curvature changes. The transition between two equilibrium states can produce a sudden burst of energy which is quickly dampened (see Figure 4.7). Although it may happen in some forced configurations, we discard the cases when the bending direction change suddenly enough to cause sound without the curvature changing sign. We only consider crumpling sound occurring because of an inversion of curvature.

Paper material is extremely rigid compared to other thin materials, such as fabric. Consequently, resonant vibrations can have a strong impact on the sound. The size and the shape of theses resonating regions influence the resulting sound. A good experiment to convince oneself of the importance of this phenomenon is to listen to the difference between the sounds produced while sliding a flat piece of paper over a table, compared to sliding a curved piece of paper enabling the sound to resonate (Figure 4.6).

We call free borders natural edges of the paper that are curved enough to be able to vibrate (see Figure 4.8 (left)). As they have more degrees of freedom than interior edges or borders constrained to be flat, they have a notable influence on the sound. When the resonating region has no free border, we observe that the friction sound is close to a white noise (Figure 4.6 left) and is less loud than friction sounds produced by a bent surface containing therefore a free border (Figure 4.6 right). Similarly for the crumpling sounds, we register sharp "clac" sounds (Figure 4.7 left) – which can be heard while crumpling a paper into a ball – that are produced by constrained regions dampening the sound in a few milliseconds. On the opposite, when the vibrations can reach a free border, a longer "flap" sound lasting a few hundreds of milliseconds (Figure 4.7 right) – which can be heard while wiggling a sheet of paper held on one border – is produced. Based on these observations, we therefore distinguish between sounds produced by regions containing one or more free border(s), that we call unconstrained regions, and those produced by regions that contain none, i.e. constrained regions (see Figure 4.8 (right)).

Our key idea is to reproduce these resonance phenomena using a procedural detection of representative regions in which the vibrations caused by a crumpling or friction event occur. This enables us to produce, for each of these regions we call *resonators*, an

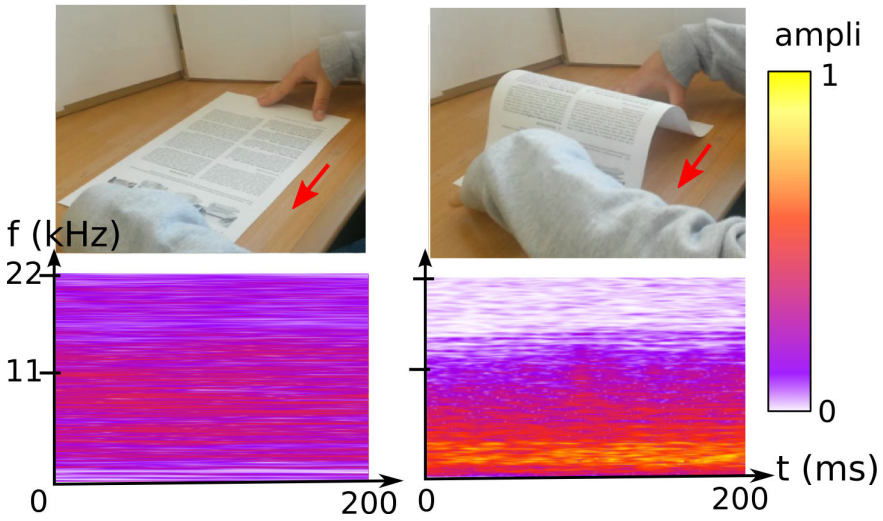



FIGURE 4.6: **Spectrogram of recorded friction sound** reveals slowly varying spectra in time, which is well approximated by a colored noise model. (left) Constrained friction sound obtained for a flat sheet of paper. (right) Unconstrained friction sound produced by a curved sheet.

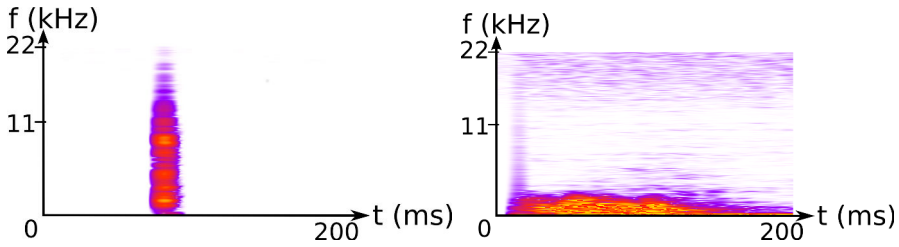



FIGURE 4.7: **Spectrogram of recorded crumpling sound:** (left) "clac" sound produced by a constrained resonating region. (right) "flap" sound produced by an unconstrained region.

appropriate shape-dependent sound. Resonators are then parameterized by their shape and size (see Section 4.3.3) and subdivided into two classes: *unconstrained* resonators or *constrained* resonators, depending on whether or not they have a free border that is able to oscillate. This leads to the four categories of sounds described above: *clac* and *flap* sounds for respectively constrained and unconstrained resonators in case of crumpling events, and different friction sounds for constrained and unconstrained resonators in case of friction events.

In order to output sounds as rich as those of real paper in real time, we use pre-recorded sound units stored in Crumpling and Friction sound databases. Due to their different natures, we need the whole sound to replay a crumpling sound, while we just use the time-averaged power spectrum to reproduce a friction sound (see Section 4.3.3). Therefore our Friction sound database only stores spectra.

Our processing pipeline is summarized in Figure 4.5. Given an animated mesh modeling the deformation of the paper surface over time (Figure 4.5 (a)), we extract two types

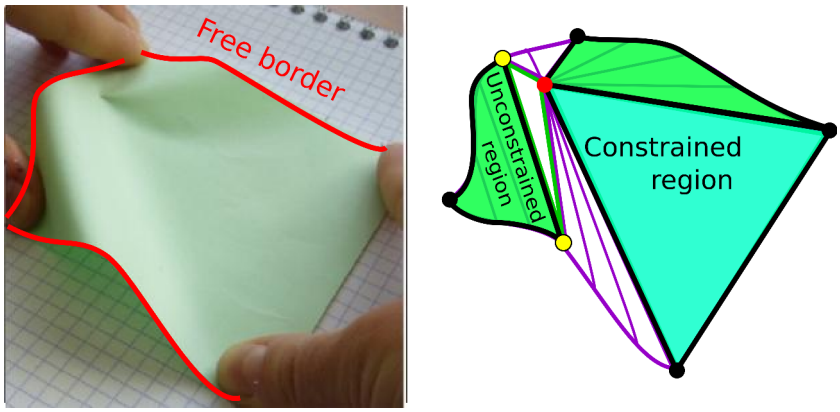



FIGURE 4.8: **Free borders** are smoothly curved edges of the paper allowing enough degrees of freedom to oscillate (in red). A region of the surface is considered as constrained if it has no free border, and unconstrained if it contains at least one.

of *sound source events* at each time step (Figure 4.5 (b)): (1) we compute the parts of the surface in contact with other objects, since each of these parts defines a *friction event*; and (2) we detect surface regions where the curvature changes sign, and use these features to identify *crumpling events*. For each sound source event, we then compute a set of *resonators* (Figure 4.5 (c)), i.e., the regions in which vibrations caused by this event occur (see Section 4.3.2). For each resonator, a sound is extracted from the appropriate database depending on the type of resonator and the type of sound event (Figure 4.5 (d)). A specific sound is extracted based on the geometric parameters of the resonator, and then reduced or amplified depending on the magnitude of the sound-source event. Finally, the assembled sound is spatially embedded within a 3D environment (Figure 4.5 (e)), by taking into account the relative positioning of the paper with respect to surrounding obstacles such as tables or walls, and the position of the listener (see Section 4.3.4).

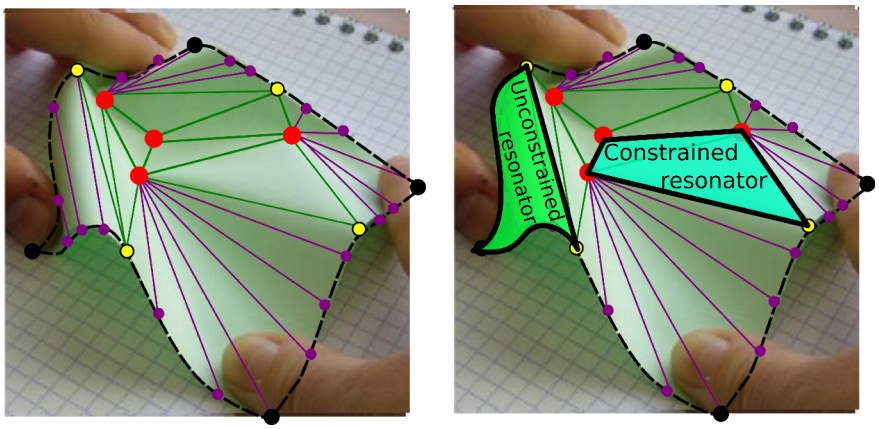



FIGURE 4.9: **Geometric structure of surface animation model:** The surface is segmented by a set of curved regions (purple) and flat regions (green).

These steps are more precisely described by the algorithm shown in Algorithm 4.

**Algorithm 4:** Animation loop with sound

```
animation ()
crumpling_events_detection (out crumpling_events_list)
for ce ∈ crumpling_events_list do
    compute_resonators (ce, out crumpling_resonators_list)        ▷ See Section 4.3.2
    for cr ∈ crumpling_resonators_list do
        sound_buffer ← gsfdb(cr, crumpling_database)
        multiply sound_buffer by amplitude of ce
        send sound_buffer to the audio output
    end
end
friction_events_detection(out friction_events_list)
for fe ∈ friction_events_list do
    compute_resonators (fe, out friction_resonators_list)         ▷ See Section 4.3.2
    for fr ∈ friction_resonators_list do
        spectrum ← spectrum + get_spectrum_from_db (fr, friction_database) ×
          amplitude of fe
    end
end
sound_buffer ← IFFT (spectrum)
send sound_buffer to the audio output
```

### 4.3.2 Detecting resonators

**Geometric Model:** At each animation step, the first stage of our method consists in detecting the sound-source events on a 3D virtual paper surface provided by an external

animation or simulation engine. As we aim at providing sound synthesis in real-time, choosing an interactive animation system for paper crumpling is mandatory. We use the model described in Chapter 2, which is, to our knowledge, the only existing model capable of animating paper crumpling at interactive rates. It has the advantage of being built on high level geometric descriptors, which we can reuse. In particular, we get from the geometric structure the partition $R_{paper}$ of the surface, composed of:

1. **Curved regions** (in purple in Figure 4.9) that segment sections of generalized cones, and are represented by the rulings of these cones (purple lines in Figure 4.9).
2. **Flat regions** (in green in Figure 4.9) that are represented by a set of coarse triangles whose vertices are either singular points or anchored points. The dihedral angle between every pair of adjacent triangles of a same flat region is inferior to a threshold, $\varepsilon_a$.

We use this surface partition to help represent surface resonators by considering each region of $R_{paper}$ as a possible resonator. As curved regions have necessarily a curved border they are considered in the following as *unconstrained regions*, whereas flat regions are considered as *constrained regions* (see Figure 4.9 (left)).

**Event Detection:** We detect the sound-producing events using a heuristic similar to the one proposed in An *et al* [AJM12]. Friction events are associated with vertices in contact with an obstacle (based on proximity computations), that are spatially connected. Each connected component represents the locus of a friction event. The mean speed of its vertices represents the amplitude of the event.

The location and time of crumpling events are estimated by analyzing the mean curvature over the surface. As the mesh given as input is constantly recomputed, we chose to compute the curvature using a regular grid projected onto the surface (see fig. 4.10). The curvature $H_v$ can thus be easily computed at each vertex $v$ of the grid using its direct neighbors, and its previous curvature can be stored. Crumpling events are represented by connected components of vertices whose mean curvature has undergone the same change of sign between the current and the previous animation step. To avoid having too many sounds caused by small instabilities in the simulation, we only select vertices where the change of curvature is significant; a vertex $v$ belongs to a crumpling region if its mean curvature change of sign from time $t-1$ to time $t$ and if the difference between the curvature $\Delta_v^t = |H_v^t - H_v^{t-1}|$ is greater than a threshold $\varepsilon_c$. The associated amplitude is computed as the sum of the curvature changes $\Delta_v^t$ for all the vertices $v$ contributing to the event.

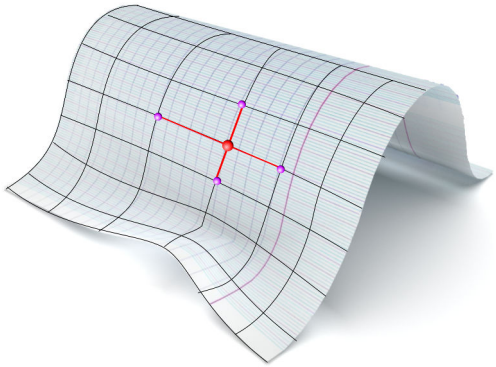


FIGURE 4.10: By projecting a grid on the surface, the curvature can be computed in any vertex of the grid using its neighbors.

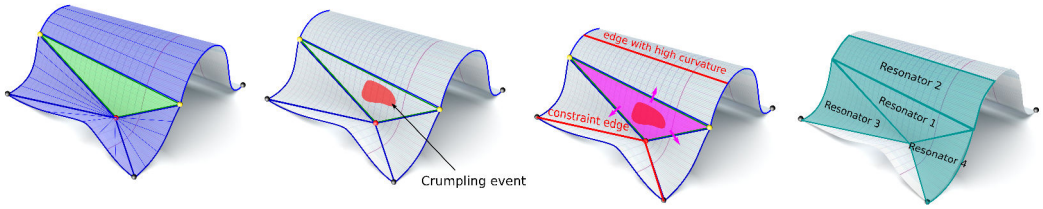



FIGURE 4.11: **Resonator estimation** (for a crumpling event): (left) curved and flat regions, (middle-left) detecting a sound event, (middle-right) finding the region of the event and recursively finding the regions in which the vibrations can propagate, (right) resonators.

As already stressed, contrary to cloth material where crumpling and vibrations only occur locally within the material, paper is very stiff (because of strong isometry constraints), and can support spatially coherent vibrations and buckling events over large regions. The sound produced by friction and crumpling events is thus much more dependent on the current geometrical state of the surface than it is for softer materials, such as cloth. Therefore, instead of directly associating a sound to each event as in [AJM12], we first associate a set of geometrically defined resonators to each event, and then synthesize the sound accordingly. While this is a approximation of the complex underlying phenomena, it enables the sound to have geometric dependence which can improve realism.

**Resonator Estimation:** In order to compute the set of resonators for an event, defined as a collection of connected surface pieces, we need to identify the boundary of the surface region that the event causes to vibrate. We choose to define this boundary in an intuitive way as follows. The borders of the sheet naturally belong to this boundary. Moreover, paper material tends to get very stiff in the direction of highly curved folds,

or "ridges" (which are aligned to some of the rulings; see Figure 4.9), but the sheet is more flexible in the main curvature direction. Consequently, transverse vibrations tend to be trapped in these interior resonator regions by ridges [GWV02]. Therefore, we assume that vibrations do not propagate across ridges, which we define as a ruling whose dihedral angle is above a threshold, $a_{\max}$; we use $a_{\max} = 0.5$ rad in our implementation. We then consider as boundaries of the propagation region all the edges with large dihedral angle, in addition to free borders and to edges with an anchored point (position of virtual fingers manipulating the sheet) or in contact with another object, since this also prevents further propagation.

Defining these limits enable us to estimate the set of resonators where a sound event propagates by recursively collecting the neighboring regions until reaching an edge labeled as propagation border. In more details, to compute the set of resonators $S_{res}$ associated to an event $e$, we use the following algorithm.

We initialize a list of regions $L_{regions}$ with the region(s) of $R_{paper}$ where $e$ took place (see fig. 4.11 (b) and (c)). While $L_{regions}$ is not empty, we remove its first item $r_0$ and add it to $S_{res}$. Then for each neighboring region $r$ of $r_0$:

- if a common edge of $r$ and $r_0$ has a dihedral angle inferior to $a_{max}$:
  - If $r$ is a flat region, the angle between two adjacent triangles is smaller than $\varepsilon_a$. We choose $a_{max} \leq \varepsilon_a$ such that the vibrations can propagate through the whole flat region. So we just add it to $L_{regions}$.
  - If $r$ is a curved region, the vibration can be stopped by a ruling, within the region, whose dihedral angle is greater than $a_{max}$ . In this case, we only add the relevant sub-part of $r$ to the set of resonators $S_{res}$. Else, all the rulings have angle small enough, so we add $r$ to $L_{regions}$.

Each of the regions of $R_{paper}$ collected in $S_{res}$ is considered as a resonator and will produce a sound as explained in the next section.

### 4.3.3 Geometry-based sound synthesis

**Resonator Parameters:** As already stated, our method to produce rich and natural sounds in real time is to re-play pre-recorded sound units, selected from a database. We therefore need to parameterize resonators in order to match them to a specific sound unit, easy to query in a database, and we need to register sounds in the database accordingly. The number of parameters needs to be kept small in order to keep the database to a reasonable size. They should also distinguish well enough the different possible sounds.

In this work, we propose to parameterize each resonator by the following two variables: $l_c$ the sum of lengths of all the free curved borders (red in Figure 4.12), and $l_r$ the mean length of the rulings (purple in Figure 4.12). As already discussed, free borders of the surface are of major importance to characterize the sound. In selecting their length, we aim at extracting the parameters of the dominant sound. By associating it with the mean length of the rulings, we sample uncorrelated variables related to the area of the resonator. We therefore claim that the couple $(l_r, l_c)$ is a good choice of parameters to correlate significantly a geometrical measurement of a region to the sound it produces. Note that we further validated this choice by performing real measurements, as described in Section 4.3.5.1.

The constrained resonators can then be identified as the resonators whose parameter $l_c$ is null.

**Sound Databases:** Let us detail how we pre-record the databases of sound units to be played at run time. Firstly, such databases are specific to a given type of paper material and to a given type of obstacle for the friction sound. In our experiments we used printer paper ($80\text{g}/m^2$) and a wooden table. We also recorded databases for tracing paper and paper of a bank note.

We actually created two different databases, one for *friction sounds* and the other for *crumpling sounds* (see Figure 4.13). As friction sounds can easily be continuously synthesized as noise, without taking into account their temporal phase, we only store the power spectrum in the frequency domain of the sounds recorded in the sound database. In contrast, *crumpling sounds* have a specific time duration with a specific beginning and end, and are therefore stored in the temporal domain.

For these two databases, we sampled the $(l_r, l_c)$ space, and for each value, we cut a rectangular sheet of paper with $(l_r, l_c/2)$ edge length, bent its edge of length $l_c/2$, recorded the sound produced by its friction, and by a change of sign of curvature (for the crumpling database). The *unconstrained friction* sound and the *flap* sounds are related to non-zero free border lengths and are the most dependent on the shape of the surface. The two others constrained types of sound, namely *constrained friction* and *clac* (corresponding to $l_c = 0$) do not depend much on the shape of the surface and were therefore considered as special cases. To ease the recording process, we stored these two specific sounds in the $(l_r = 0, l_c = 0)$ entry of their respective database. Several sound samples were stored per entry to increase variety.

**Sound Synthesis:** Given the databases and the resonator, sounds are synthesized at run-time in the following way: For each resonator, we select the closest recorded $(l_r, l_c)$

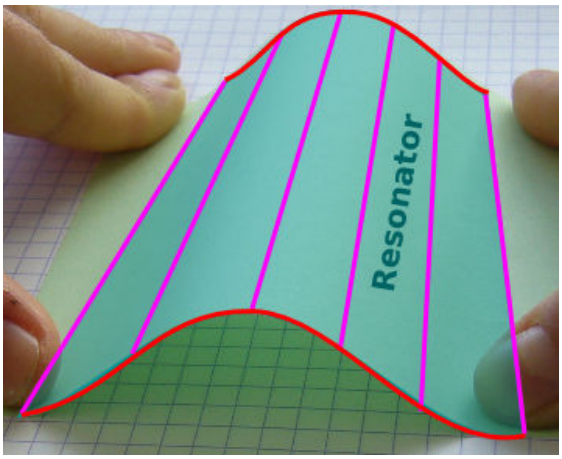



FIGURE 4.12: **Resonator parameterization** using the mean length of its (purple) rulings ($l_r$) and the sum of the lengths of the (red) free borders ($l_c$).

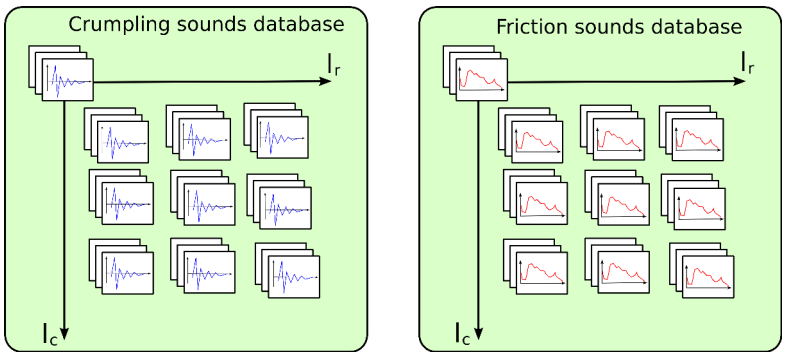



FIGURE 4.13: **Sound Databases:** Ordered in a matrix form according to the dimensions $l_r$ and $l_c$. The crumpling database contains short sound clips, whereas the friction database contains sound spectra.

parameter in the relevant database, considering also the special case $(0, 0)$ when the resonator does not have any free border, and randomly select one of the corresponding sound samples. Finally, all sounds are assembled together.

As explained before, each crumpling sound is entirely stored in the temporal domain in the database. In order to take into account the magnitude of the event, the sound magnitude is linearly scaled with respect to the event amplitude. Note that we let the computer system handle the mixing of the sounds. Each sound is individually handled in a separated thread such that geometrical animation can be updated interactively in parallel with the sound synthesis.

Contrary to crumpling sounds, the duration of a friction sound is not known *a priori*. Thus we need to generate the sound for as long as the friction lasts. As repetition is very noticeable for human, looping over the same recorded sound several times should be avoided, instead we choose to approximate a friction sound by a colored noise with equivalent spectral properties. To synthesize such sound, we use the inverse Fast Fourier

Transform method [RDRD92, MAKMV10] in order to generate a stochastic signal with a specific power spectrum: we call $\mathbf{s} = [s_k]_{0\dots M-1}$ the vector of size $M$ representing the power spectrum ($M = 2048$ in our case). We obtain a vector of size $M$ describing the sound by applying an iFFt to the complex vector $\mathbf{v} = [v_k]_{0\dots M-1}$ such that:

$$v_k = s_k \, e^{i\,\omega_k} \quad \text{for } k = 0 \dots M-1,$$

where $\omega_k$ is a random phase uniformly chosen in the range $[-\pi, \pi]$. The sound thus obtained has $s$ for power spectrum and a stochastic phase.

In order to avoid to much computation, instead of generating separately each friction sound, we first sum all the friction spectra selected for the current frame, each one weighted with the amplitude of its corresponding event. We thus get a general friction spectrum and apply the iFFT method to it. Buffers of general friction sound are produced and send to the audio output on a different thread as long as friction is detected. There are no guaranties that the transition between two successive buffers would be continuous, so in order to smooth it, two consecutive buffers are overlapped of $\frac{M}{2}$.

It may be interesting to note that the synthesized sound is always guaranteed to be synchronized with the frame of the animation, independently of the possibly variable time between two frames. The parameters of the sounds (i.e: the nature and amplitude of the events and the size of their corresponding resonators) are computed at each frame, fast enough not to hamper the animation time. Then the sound can be produced in real time, the most costly operation being the iFFT (see Section 4.3.5.1 of experimental times). This method can produce the sound at run time during the interactive animation of the paper, as well as being created for an already recorder animation with static fps if the parameters of the sound have been recorded for each frame.

### 4.3.4 Spatialization

In this section, we describe our approach to embed the synthesized sound in the 3D space by taking into account both the listener's position and sound reflection on the surrounding table or walls. Such planar surfaces interact with the sound vibrations of paper material which can be modeled as a dipole.

The dipole-like radiation behavior of paper surfaces leads to significant frequency-related interference effects above planar surfaces. For example when sliding a sheet of paper on a table to bend it more, as in Figure 4.4 (left), one may notice a pitch-shift. This is partly caused by the fact that the friction sound is reflecting on the table and the reflected source interferes with the original source. A pitch-shift is also noticeable when

one moves a sheet of paper closer to a wall while wiggling it. As the paper gets closer to the wall, the "flap" sounds change. Note that in this work, we do not consider the interference caused by self-reflections on the surface by cavities formed by the paper.

We develop below a simplified model for these phenomena for both time-domain (crumpling) and frequency-domain (friction) sounds. Transverse vibrations of the paper sheet give rise to positive and negative pressure fluctuations on either side of the sheet. If we approximate each resonator to a small vibrating disk sound sources, then we can reason about each disk's sound emission and any planar reflections.

Let us consider the sound field, perceived by a listener located at $\mathbf{x}$, due to a vibrating rigid disk of radius $a$ and zero thickness whose center is positioned at $\mathbf{y}$ and whose unit normal vector is $\mathbf{n}$ (see Figure 4.14). If we assume that the normal velocity is $U(t)$, and that the disk is acoustically compact, i.e., the important wavelengths associated with the frequencies in $U(t)$ are much larger than $a$, then one can derive the acoustic pressure field $p$ generated by the motion (see [How02] page 81, problem 9) as:

$$p(\mathbf{R},t) \approx \frac{2\rho}{3\pi c} a^3 \frac{\cos\theta}{R} \frac{\partial^2 U}{\partial t^2}(t - \frac{R}{c}), \tag{4.1}$$

where $c$ is the speed of sound in the air and the relative location is given by:

$$\mathbf{R} = \mathbf{x} - \mathbf{y}, \tag{4.2}$$

$$R = ||\mathbf{R}||, \tag{4.3}$$

$$\hat{\mathbf{R}} = \mathbf{R}/R, \tag{4.4}$$

$$cos\theta = \hat{\mathbf{R}} \cdot \mathbf{n}, \tag{4.5}$$

as shown on Figure 4.14. This vibrating disk produces a *dipole* pressure field, with characteristic $1/R$ distance scaling, and a cosine lobe about direction $\mathbf{n}$. Without loss of generality, we can discard the scalar factor $\frac{2\rho}{3\pi c}$ (which is common to all our sound sources) and use:

$$p(\mathbf{R},t) \approx a^3 \frac{\cos\theta}{R} A(t - \frac{R}{c}), \tag{4.6}$$

where $A = \frac{\partial^2 U}{\partial t^2}$ is the acceleration of the patch along $\mathbf{n}$.

Consider the dipole source from (4.6) at height $d$ above an infinite rigid plane, as shown in Figure 4.14. Since the rigid plane imposes a zero boundary condition $\frac{\partial p}{\partial \mathbf{n}} = 0$ on the normal acoustic particle velocity on the surface, a fictitious image dipole source $\widetilde{p}$ is introduced in the sound field. The global pressure field $P$ is then the sum of the pressure

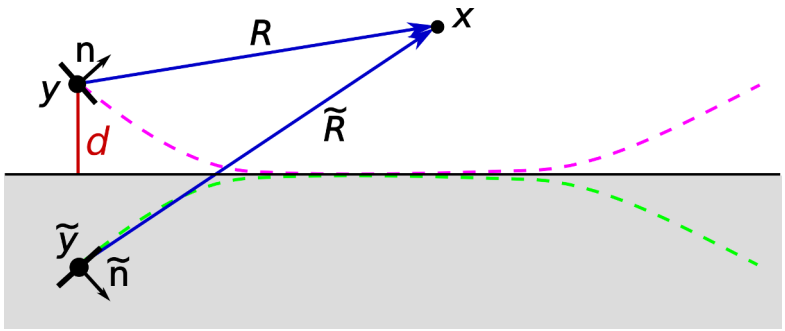



FIGURE 4.14: **Scattering:** Geometric configuration of a vibrating paper patch above a planar surface, along with its image source.

fields of the dipole source and its image source:

$$P(\mathbf{x},t) = p(\mathbf{x},t) + \widetilde{p}(\mathbf{x},t) \tag{4.7}$$

$$= a^3 \frac{\cos\theta}{R} A(t - \frac{R}{c}) + a^3 \frac{\cos\widetilde{\theta}}{\widetilde{R}} A(t - \frac{\widetilde{R}}{c}) \tag{4.8}$$

The image source has reflected position $\widetilde{\mathbf{y}}$ and normal $\widetilde{\mathbf{n}}$, and corresponding relative coordinates $\widetilde{R}$ and $cos\widetilde{\theta} = \widetilde{\mathbf{R}} \cdot \widetilde{n}$. The frequency domain form of (4.11) can be obtained by replacing $A(t)$ by $A(\omega)e^{-i\omega t}$ to obtain

$$P(\mathbf{x},\omega) = p(\mathbf{x},\omega) + \widetilde{p}(\mathbf{x},\omega) \tag{4.9}$$

$$= a^3 A(\omega) \left[ \frac{\cos\theta}{R} e^{i\omega\frac{R}{c}} + \frac{\cos\widetilde{\theta}}{\widetilde{R}} e^{i\omega\frac{\widetilde{R}}{c}} \right], \tag{4.10}$$

where the second term in brackets is the reflection.

Using this simplified model, we approximate the sound reflections in crumpling and friction sounds using the time- and frequency-domain reflection models, respectively, as follows.

**Time-domain crumpling sounds:** we scale each current time-domain crumpling sound pressure event $A(t)$ selected from the database to obtain its dipole-spatialized version and generate the following delayed sound source for the image dipole:

$$P(\mathbf{x},t) = \left( a^3 \frac{\cos\theta}{R} \right) A(t) + \left( a^3 \frac{\cos\widetilde{\theta}}{\widetilde{R}} \right) A(t - \frac{\widetilde{R} - R}{c}). \tag{4.11}$$

This change is done to each crumpling sound before synthesis.

**Frequency-domain friction sounds:** we obtain the dipole-spatialized version of each friction sound by placing Equation (4.11) in the frequency-domain (effectively replacing

$A(t)$ by $A(\omega)e^{-i\omega t}$):

$$P(\mathbf{x},\omega) = \left( a^3 \frac{\cos\theta}{R} e^{i\omega\frac{R}{c}} + a^3 \frac{\cos\widetilde{\theta}}{\widetilde{R}} e^{i\omega\frac{\widetilde{R}}{c}} \right) A(\omega). \tag{4.12}$$

This multiplication factor is applied to the friction spectra $A(\omega)$ prior to region-based summation and synthesis of the final sound.

### 4.3.5 Results and discussion

Most of the sounds presented in this section are represented graphically by their spectrum or spectrogram, but we strongly encourage the reader to watch and listen to the video provided at the following link, preferably using good headphones in a quiet environment: https://hal.inria.fr/hal-01333238v3/file/sound_of_paper.mp4

**Implementation Details:** In our results, we use sounds recorded at a sampling rate of 44100 Hz and play the generated sounds at the same rate. The spectra of the friction sounds contains 2048 frequencies which means that a buffer contains 2048 samples ($\approx$ 50 milliseconds) and we use an overlap of 1024 samples between two buffers. Our implementation makes use of the C++ library STK [CS99, SC05] to read and play sounds.

**Recording of the database:** Our two databases contain samples of sizes $l_c$ and $l_r$, using the values {2.5, 5, 7.5, 10, 12.5, 15, 20, 25, 30} cm. We recorded three crumpling sounds and one friction sound for each sheet size. The sounds were recorded manually under the same conditions. The microphone was placed one meter away from the sound source.

We used for the real recording several rectangular pieces of paper that we bent in order to give them the shape of a generalized cylinder . Each rectangular piece has therefore a curved and a straight border. We call $L$ the length of the curved border, and $l$ the length of the straight one. We consider that the recorded sound corresponds to the one produced by a resonator of size $l_c = \frac{L}{2}$ and $l_r = l$. We quantify the amount of bending of the cylinder by a so called *compression factor* referring to the equivalent difference of length between the border extremity in Cartesian distance. For example, if the curved border measures 10cm and is *compressed* by 30%, this means that the distance between the two endpoints of the curved border is 7cm, as they have moved closer of each other by 3cm (30% of 10cm). This measure will be used later to quantify the influence of the curvature on the sound.

For the friction sounds, the speed was measured by the time needed to go over the friction distance (2s for 20cm). To avoid artifacts caused by the beginning or the end of the slide, we compute the average power spectrum from a 200ms window at the middle of the friction . The shape was retained by keeping the same amount of compression of the curved edges (30%). We recorded the unconstrained ("flap") sounds by holding both ends of a sheet (slightly curved by compression curved edges of 10 %) of the required size and twisting them to change the sign of the curvature. The constraint ("clac") sounds were isolated in a recording of crumpling paper.

#### 4.3.5.1 Validation of the resonators' parameters

We used two parameters $(l_c, l_r)$ to characterize the sound of a given resonator. To validate this choice in the case of the friction sound, we show that modifying these two parameters has a larger influence on the sound produced by friction than modifying other geometrical shape features with constant $(l_c, l_r)$.

We perform several experiments in which we compare the friction sounds obtained with resonators of different shapes. To compare the different recordings, we computed the histogram corresponding to the spectrum of each sound and then compared them using the quadratic-form distance [PW10]. In a first experiment, we use a square paper sheet of size $10cm \times 10cm$ and modified its shape by compressing the two extremities of the sheet as illustrated in Figure 4.15 (a) (we measure as parameter the compression of the curved borders, expressed in % In the following experiment represented in Figure 4.15 (b), we record the sound of different rectangular sheets such that they have the same compression (= 30%) and the same value for the parameter $l_c$ (= $20cm$) but different values for the parameter $l_r$. And then we do the same (as shown in Figure 4.15 (c)) for the parameter $l_c$ ($l_r = 10cm$, $comp = 30\%$).

Table 4.1 gathers our results. The first four columns show that modifying the $l_c$ or $l_r$ parameters respectively by a factor of 50 to 75% leads to a modification of their spectrum, quantified between 0.07 and 0.16. The next two columns show that keeping constant $l_c$ and $l_r$ parameters but changing the curvature of the sheet leads to a change of spectrum quantified between 0.03 to 0.05, which always stays below the influence of modifying $l_c$ or $l_r$.

To further study the influence of shape, we also compared the sound produced by three different sheets of shape $S_1$, $S_2$ and $S_3$, shown in Figure 4.15 (d). Each piece of paper is compressed by a factor of 30% and the results on their respective histogram distances computed using the same approach as above are gathered in the last two columns of

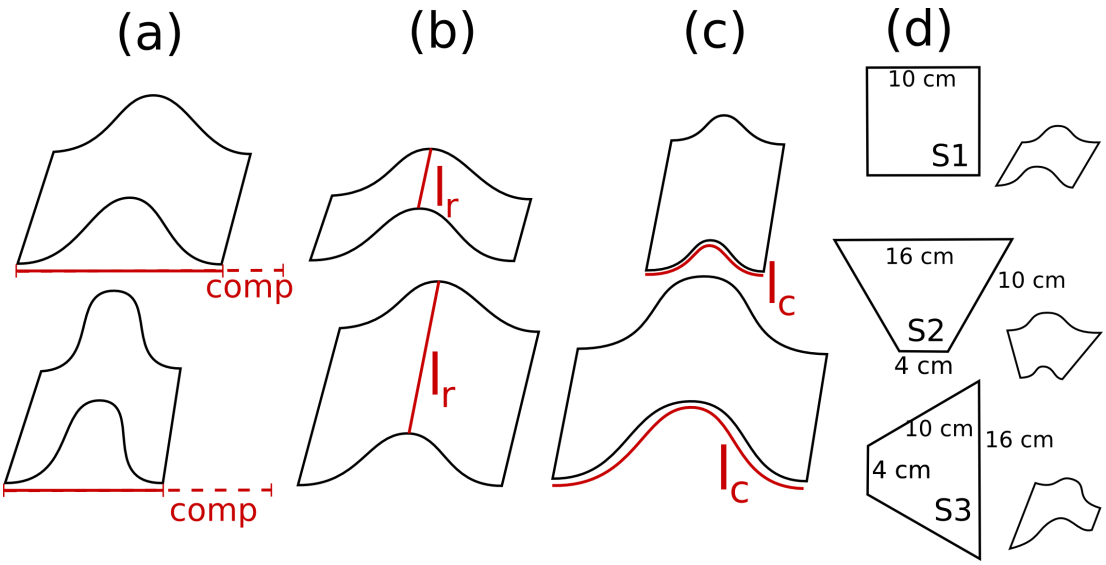



FIGURE 4.15: **Validation experiment:** We compare sounds obtained while varying three parameters: amount of compression (a), $l_r$ (b) and $l_c$ (c). We also measured the sounds obtained for 3 sheet shapes with the same compression, $l_r$ and $l_c$: S1 is a square 10 cm x 10cm (d top); S2 is a trapezoid whose bases, of respective sizes 4cm and 16cm, are the curved borders (d middle); S3 is a trapezoid whose bases of sizes 4cm and 16cm, are the flat borders (d bottom).

Table 4.1. We obtained distances comparable to those obtained for change of curvature, and once again lower than those obtained for changes of $l_r$ and $l_c$.

| $l_r$ | $d_r$ |
|---|---|
| 5 vs. 10 | 0.10 |
| 10 vs. 20 | 0.07 |
| 5 vs. 20 | 0.16 |

| $l_c/2$ | $d_c$ |
|---|---|
| 5 vs. 10 | 0.04 |
| 10 vs. 20 | 0.12 |
| 5 vs. 20 | 0.13 |

| comp | $d_{\text{comp}}$ |
|---|---|
| 20 vs. 40 | 0.03 |
| 40 vs. 80 | 0.05 |
| 20 vs. 80 | 0.03 |

| s | $d_s$ |
|---|---|
| s1 vs. s2 | 0.04 |
| s1 vs. s3 | 0.01 |
| s2 vs. s3 | 0.03 |

TABLE 4.1: **Distances between friction spectra** of rectangular sheets of paper while varying the parameters as shown in Figure 4.15. $l_r$ and $l_c/2$ are the considered edge length parameters in *cm*, $d_r$ and $d_c$ are the respective histograms distances. *comp* is the relative compression ratio, $d_{\text{comp}}$ the associated histogram distance. $d_s$ are the distances between friction spectra of the shapes $s$ described in Figure 4.15(d) compressed of 30%. $s$ refers to the shape pattern show in Figure 4.15 (right), and $d_s$ corresponds to the histograms distances. For example, the first line of the first table indicates that the histogram distance is 0.10 between the friction spectrum produced by a resonator with $l_r = 5\,\text{cm}$ and the one produced by a resonator with $l_r = 10\,\text{cm}$.

Those experiments show that $l_r$ and $l_c$ are plausible parameters to characterize a resonator, although other factors that we did not took into account may indeed influence the sound. Still limiting the sound parametrization to these two values gives plausible results (see the rest of this section) with a limited number of sound samples.

To achieve more accurate results, one could aim at using into account more geometrical parameters, and ideally, to procedurally adapt the sound generation to the shape

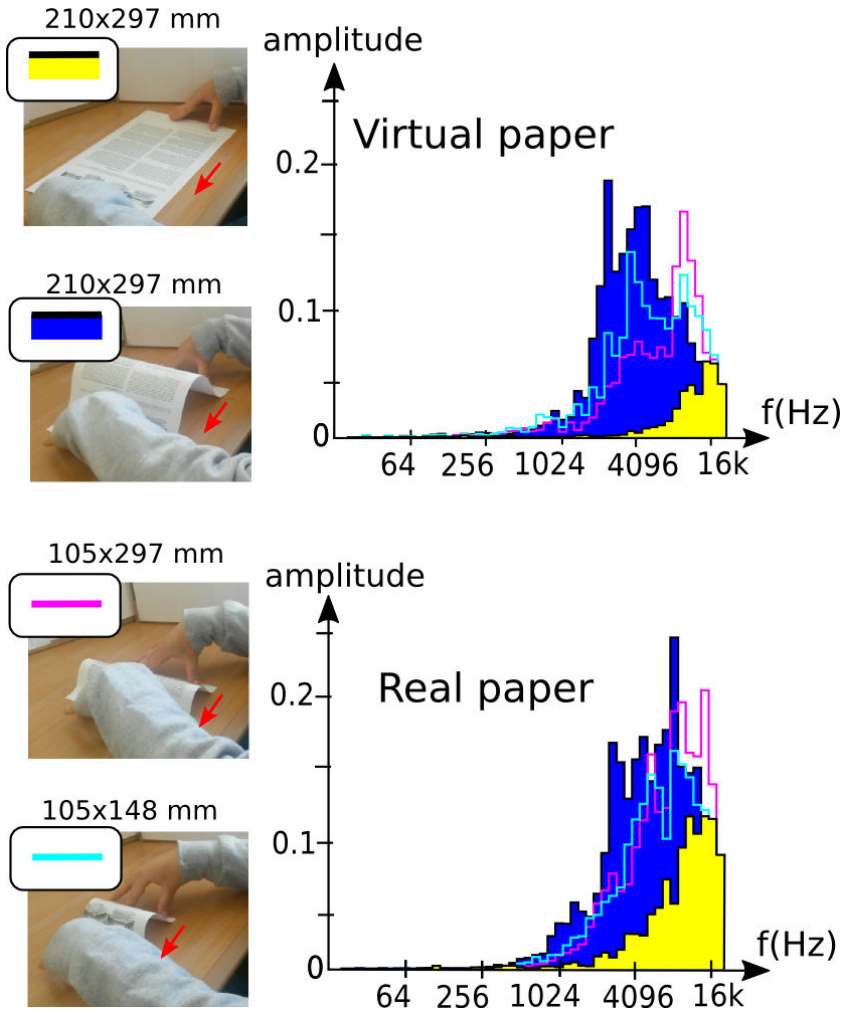



FIGURE 4.16: **Real-virtual comparison of friction sound spectra** for different sheet sizes.

described with a larger number of parameters. Another idea would be to use the developable property to simplify the physical computation of the vibrations in a resonator. All these suggestions could be developed as future works.

#### 4.3.5.2 Comparison with real paper material

To validate our resonator model, we compared the results obtained using our method with sounds produced by real sheets of paper of different sizes. First, we focused on friction sounds produced by sheets of paper slid over a table. We used rectangular sheets of different sizes –210x297 millimeters (A4 format), 210x148 (half A4),105x297 (A5), 105x148 (A6)– and folded them as shown in Figure 4.16 (left). We also recorded the sound of an A4-sized flat sheet. We compared the resulting five spectra of virtual and real sounds in log scaled space (respectively top-right and bottom-right of Figure 4.16). The results show that our model follows a similar evolution of the spectrum with respect

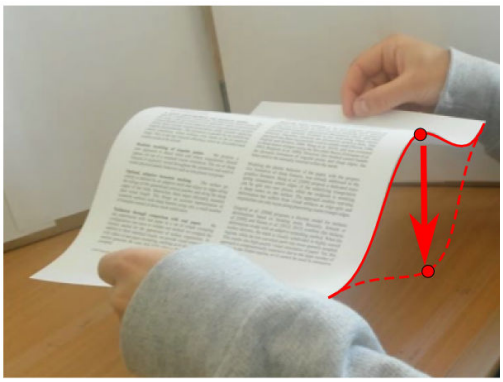


FIGURE 4.17: **Real-virtual comparison of "flap" sounds** for different sheet sizes, obtained by changing the curvature of bent paper.

to the sheet size, i.e. a general translation of the spectrum towards higher frequencies when the size decreases. This evolution –and particularly the clear difference between flat paper and folded paper being slid along the same table– can also be heard in the video.

Next, we compared real and virtual crumpling *flap* sounds for different sizes of sheets. Considering the same sizes than in the previous experiment, we folded each sheet of paper as shown in Figure 4.17 and revert the sheet while recording its sound. The video plays the resulting sounds. In both (real and virtual) cases, we get different sounds according to the size of the sheet.

#### 4.3.5.3 Spatialization

Here, we conducted two experiments. Figure 4.18 (left and middle) and the associated video presents the result of our frequency-domain spatialization method for the friction sound. The friction sound of our virtual paper is recorded for an A4 sheet compressed by a factor of 30% first and 10% then, leading to a change of height of the sound source. One can hear in the video that the friction sound depends on the height of the sound source, as it is the case for real paper sound. We can also hear the slight pitch shift as the sheet is bent.

We also show in the video the modification of sound depending on the position of the listener in front and on the side of the sheet of paper (Figure 4.18). Figure 4.19 corresponds to a sheet of paper being wiggled while changing its distance with respect to the wall. The time-domain spatialization of the crumpling sounds causes the sound shift we can hear in the video.

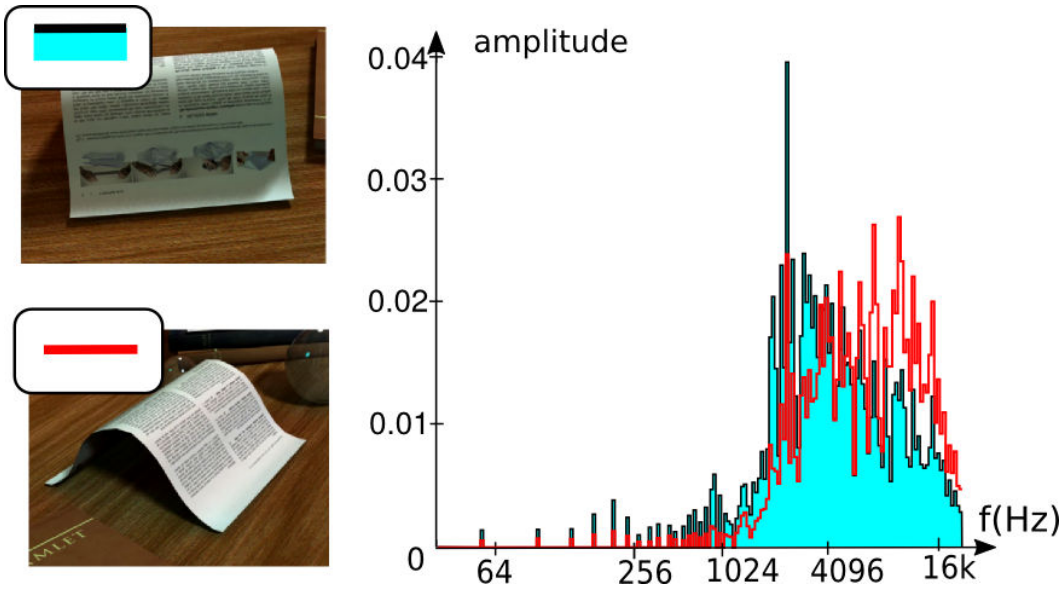



FIGURE 4.18: **Spatialization:** The accompanying video shows the result of spatialization for friction sound from two different points of view.

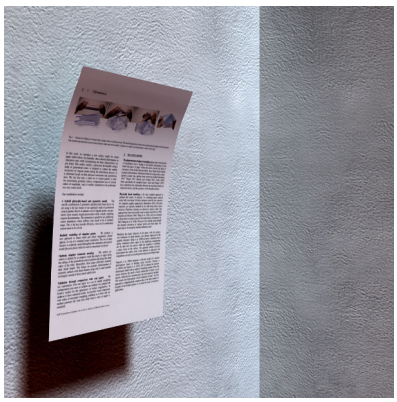


FIGURE 4.19: **Spatialization:** The sound may be modified by the proximity of a wall. It is noticeable for example when one wriggles a sheet of paper near a wall. A change a pitch can be heard while moving the sheet away from the wall. The accompanying video shows similar effect obtains thanks to the spatialization step.

#### 4.3.5.4 Computational Performance

**Memory requirements:** The crumpling database contains 249 sound units whose lengths are going from 200 to 800 milliseconds. The friction database contains 63 spectra sampling 2048 frequencies –between 0 and 22kHz. The memory footprint of the database is around 10MB.

**Computation times:** The detection of the events and the computation of their resonators are done once per frame. Table 4.2 compares the mean time per frame for these audio operations to the computation time of the animation. We computed the sound of the animation represented Figure 4.22 (right) with three different resolutions. The

computation time of the curvature depends on the resolution, so real-time performances get harder to reach when the resolution increases. Still, the computational time required for the audio-related operations always remains negligible compared to the one required for the animation.

Playing sounds is done using different threads than computing the animation To achieve real time sound synthesis, it is necessary to generate at least 44100 audio samples per second. A thread is created for each crumpling sound. Note that at run time, only scaling operations need to be applied to the recorded samples before sending them to the audio output. A single thread is used for friction sounds. A new buffer is computed every 1024 samples using the weighted sum of the spectra of all the friction sounds. Then the actual sound is computed by applying an Inverse Fast Fourier Transform. This computation has to be done 44 times per second and took 4ms in average in our examples, which makes ∼200ms for generating 1 second of sound.

| examples (Figure number) | animation (ms per frame) | audio (ms per frame) |
|---|---:|---:|
| Figure 4.4 | 168 | 5 |
| Figure 4.21 | 182 | 6 |
| Figure 4.22 (left) (183 triangles) | 252 | 6 |
| Figure 4.22 (left) (446 triangles) | 1322 | 17 |
| Figure 4.22 (left) (1333 triangles) | 11860 | 75 |

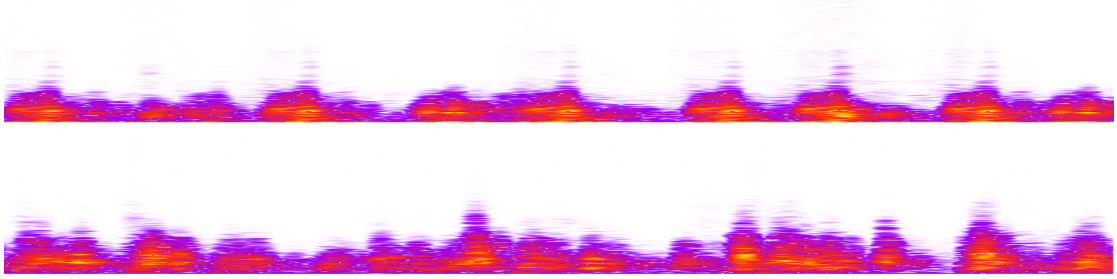
TABLE 4.2: **Mean computation time** for the animation and mean computation time for the audio operations, done once per simulation step

#### 4.3.5.5 Others results

In this section, we present some more complex examples, which are also shown in the video. Figure 4.4 and Figure 4.20 illustrate, respectively in the virtual case and in the real case, the spectrogram of the sound generated while sliding one edge of a sheet along the table thereby curving the paper, then pinching the front border causing the back to stand up. This example mixes all the different categories of sounds we considered. We can note that the synthetic friction sounds are much more regular than the real ones. This is mainly due to the fact that it is very difficult to reproduce a smooth movement "by hand." Nonetheless, we obtain the same type of event at the same moment, notably the "flap" sound when the sheet stands up and different kinds of friction sound for the different shapes that the sheet takes. Figure 4.21 shows a sheet (real and virtual) held by one border and wiggled and the corresponding spectrograms. Figure 4.22 (left) shows the example of a sheet being crumpled. We also simulated the sound of different categories of paper material by recording different databases, such as for tracing paper

FIGURE 4.20: **Comparison:** Spectrogram obtained while reproducing the experiment shown in Figure 4.4 with real paper.

(A) *(Left) Virtual paper. (Right) Real paper.*



(B) *Spectrogram obtained for (Top) virtual paper, and (Bottom) real paper. Both examples show 1.5s of the sound*

FIGURE 4.21: **An A4 sheet of paper** held by two corners is lightly shaken.


FIGURE 4.22: **Different paper materials** being crumpled can be synthesized in recording different databases: (Left) printer paper, (Middle) tracing paper, (Right) a bank note.

and paper of a bank note (Figure 4.22 (middle) and (right)).

### 4.3.6 Conclusion for the sound of crumpling paper

We presented the first method that automatically generates plausible sounds fitting an interactive animation of paper material. We achieved real-time sound synthesis through a trade-off between speed and accuracy, leading to several noticeable limitations. First,

the range of possible sounds is limited when using pre-recorded sound databases. Moreover, each database is associated with a specific type of paper and obstacle materials (for friction sounds), and changing them requires recording a new database.

The visual simulation of paper is also a bottleneck, and currently there are no methods that are both interactive and able to model very crumpled paper, especially with self contact. We used an interactive method to demonstrate our fast sound synthesis, and also because it enabled efficient estimation of resonators. Challenging but familiar scenarios, such as crumpling paper into a ball, remain outside the scope of this work, since they would require detailed simulation of self contact, as well as more sophisticated modeling of sound inter-reflections and scattering. Nevertheless, our results still depict a large range of scenarios.

By requiring only animated mesh sequences, our method can handle a large variety of animation inputs. However, since it does not have access to physical contact force information, it cannot, for instance, include dependence of friction sound volume on contact force. Moreover, our resonator model approximates paper vibrations in a resolution-dependent manner, and does not actually model vibrations directly.

Another limitation is due to the small set of physical parameters taken into account during the synthesis process. Studying more the effect of different parameters, in addition to the two we already use, on the sound or taking into account the specific properties of developable surfaces to allow for more accurate vibration modeling without sacrificing computational efficiency, these are some ideas for obtaining a more efficient sound model. Ideally, this may permit to know how to adapt a sound –either a recording, a reconstructed noise or a procedural sound– to any shape.

In the next section, we complete the crumpling and friction sound by a tearing sound. The same principle than the approach proposed in this section could apply, but as it is much more difficult to record only the tearing sound while controlling reliably the size a the resonator and the speed of tearing, we propose to generate it using a fast procedural method and shape independent method, avoiding thus the need of recording a database and its memory cost.

## 4.4 The sound of tearing paper

To generate a sound with minimal memory and computational costs, a sensible solution is to use a procedural method. So in this section, we try a phenomenological approach and enhance our interactive tearing animation with a simple and fast algorithm to create

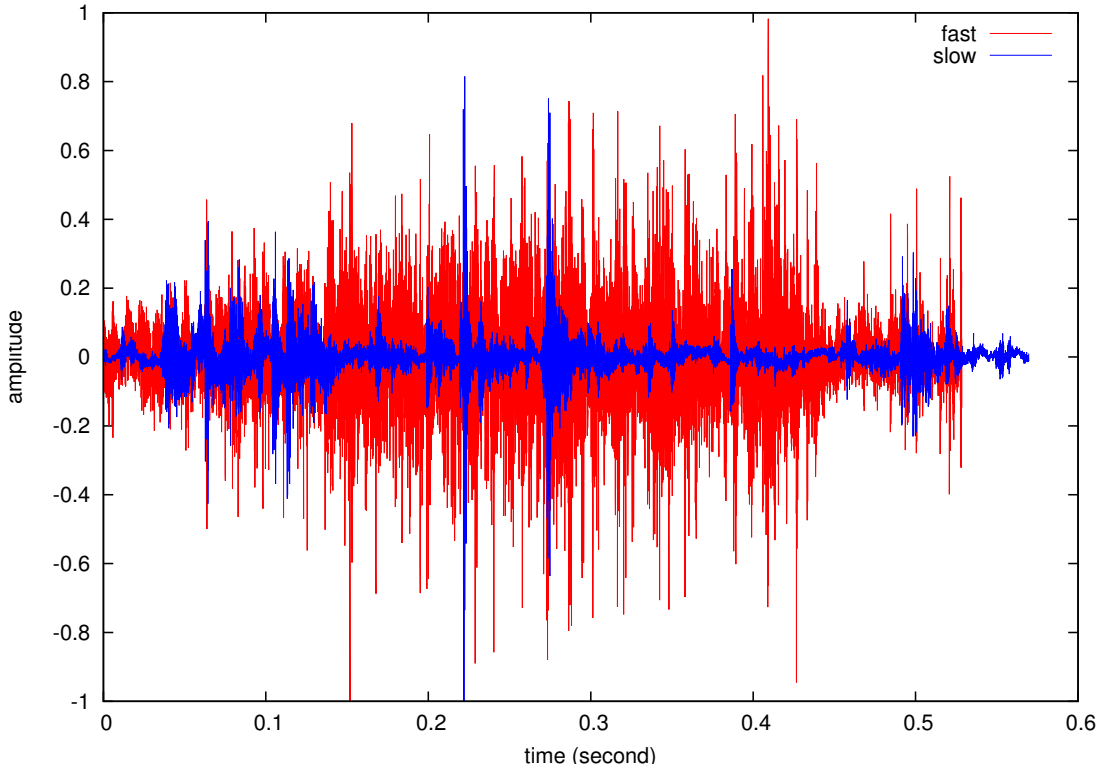
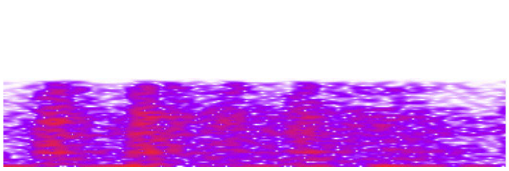


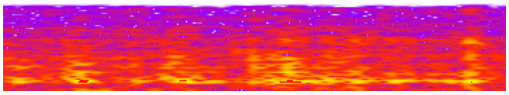

FIGURE 4.23: Comparison between the audio signal of a fast tear (red) and a slow tear (blue). The amplitude of the fast tearing is larger than the slow one, while the large peaks are generally closer for the fast tear.

sound. Our method is based on observations made from audio recordings of real paper, and hypotheses concerning the breaking of fibers and the speed of the tear.

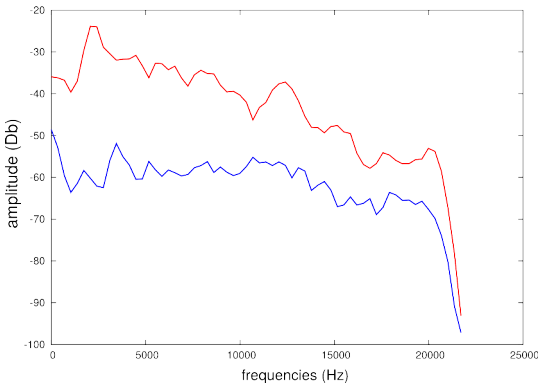
### 4.4.1 A fast procedural model



(A) Slow tearing



(B) Fast tearing




(C) Spectra of the fast tearing (red) and the slow tearing (blue).

FIGURE 4.24: Spectrograms of 0.05 s clips from the middle of the recorded sounds shown in Figure 4.23, with a plot of the spectral power densities. All frequencies are present at a relatively constant speed-dependent power, with a small frequency dependent power drop for fast tearing.

The sound of paper being torn is generated when fibers, or more often links between them, are breaking releasing energy. As fibers are randomly distributed within the paper material, this process can be considered as stochastic, and therefore synthesized using a noise model.

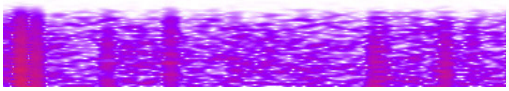
We recorded slow and fast paper tearing and reported the respective spectrogram and spectrum of the record in Figure 4.24. Spectrum exhibits a rather constant value on the hearing bandwidth and indicates that the sound is close to a white noise. The spectrograms show temporal variations in the amplitude of the sound. This can be explained by the physical organization of paper material. While being torn apart, fibers produce an energy burst leading to a sound. Moreover, as noted previously, the fiber density is not homogeneous within the paper material. These variations of density encountered during the tearing process lead to sudden changes of energy making the sound resemble a crackling noise. These variations are mostly visible in the slow tearing example shown in blue in Figure 4.23.

We reproduced this phenomenon using the following model. We first model a white noise. Then each sample of the noise is multiply by $r^{\alpha}$ , $r$ being a random number between 0 and 1 generated for each sample. $\alpha$ is a parameter used to tune the sound between a white noise, for $\alpha = 1$, and a crackling noise for large value of $\alpha$. We typically used in our experiment $\alpha = 100$.

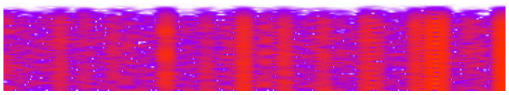
The crack propagation speed also affects the sound. First, the sound energy changes according to the speed, as can be seen with the comparison between a slow tearing and a fast tearing in Figure 4.23. This can be explained by the fact that a faster tear involves a larger number of broken fibers in the same amount of time. Second, the amplitude changes that are caused by varying density of fibers across the sheet will also occur faster. Thus, we linearly tune the amplitude and pitch of the modified white noise according to the speed.

### 4.4.2 Results

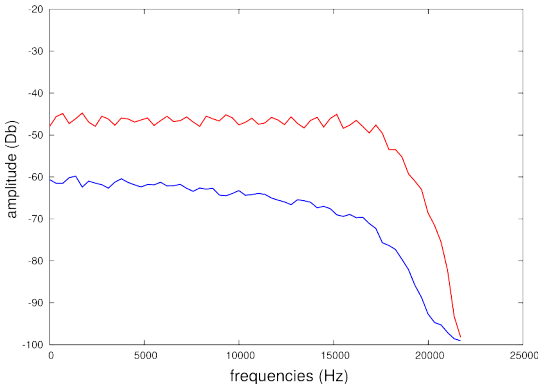
In Figure 4.25, we show spectrograms and spectral power density plots of synthesized slow and fast tearing sounds. We are able to capture the noise and crackling features of tearing, but in comparison to recorded sounds, our synthesized sounds focus only on the tear and are therefore missing several important qualities related to the shape. Indeed, we can see in Figure 4.24c that the sound of real tearing has a specific tone while the spectrum of the synthesized one (see Figure 4.25c) is nearly flat (it makes sense since it is based on a white noise).



(A) Slow tearing



(B) Fast tearing




(C) Spectra of the fast (red) and slow (blue) tearings.

FIGURE 4.25: Spectrograms of 0.05 s clips of synthesized sounds, with a plot of the spectral power densities.

## 4.5 Conclusion

We have proposed here two different approaches to model the sound of different behaviors of paper corresponding to the behaviors of our animation model. Our approach for generating the sound of tearing paper uses very few time and memory resources making it particularly well-adapted to games, that dedicated only little computation time to the sound generation. But it fails to take into account the shape of paper. The pitch shift that one can hear as the undamaged region get smaller is not reproduced. On the contrary, our approach for modeling crumpling and friction sound is shape-dependent but is more expensive in time and memory although it can still reach real-time.

A deeper study of the behavior of the sound of paper, how it changes according to its shape, or how the vibrations propagate in developable surfaces would be some interesting future works as it may put within reach an approach combining the advantages of both methods proposed in this section.

# Conclusion:
# The future of the virtual paper

*"Welcome back my friends, to the show that never ends."*
Emerson, Lake and Palmer, *Brain Salad Surgery*, 1973.

This thesis presented a complete model for interactively creating animations of virtual paper, including bending, crumpling, tearing and automatic on-the-fly generation of the corresponding sounds. By using a hybrid model mixing physical and geometric approaches we are able to reach most of the objectives we aimed at:

**Realism.** Our model draws upon a physical simulation, which has the advantage of enabling to obtain natural motion of the surface for general scenarios. The procedural and geometric layers, on their side, ensured to reproduce either observed or described phenomena in studies dedicated to paper material, thus enabling a plausible behavior. Although our model would still need to incorporate collision handling to allow fully crumpled modeling, we can already simulate many interesting scenarios.

**Efficiency.** The geometric step of our crumpling model computes a remeshing of the surface that gives the necessary degrees of freedom to the surface while keeping the mesh as coarse as possible. The methods for tearing paper and generating sound rely on the information given by this step to efficiently reduce computation time. As a result our model is fast enough for interactive applications.

**Intuitive control.** The physical simulation enables a smooth deformation and the specific features of the paper such as folds, tears or sound are automatically computed based on prior knowledge. The user does not need to provide complex inputs, like the position of sharp edges or folding patterns for example.

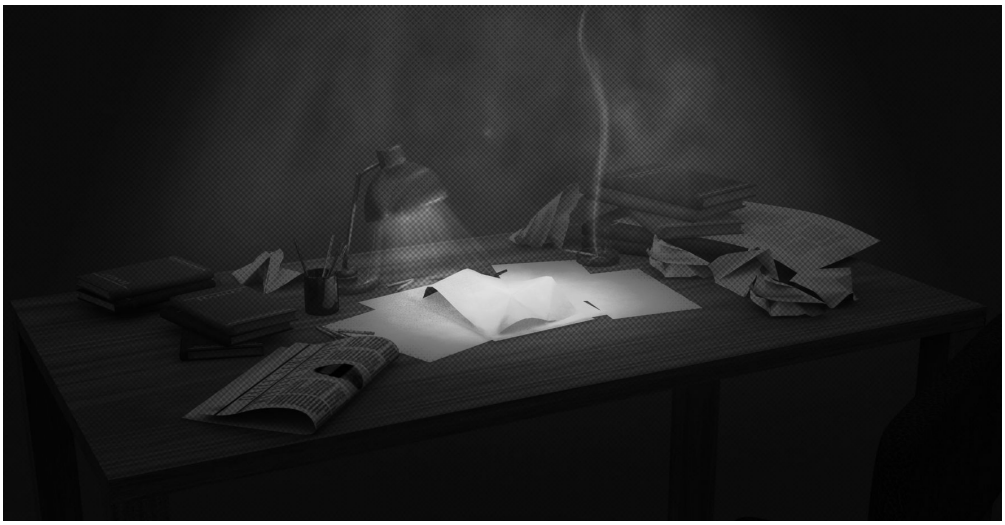
In the different works proposed in this thesis, we made a point of testing and comparing our virtual paper and real paper in situations when the sheet of paper starts to crumple or tear. These cases –when paper transitions from an undamaged flat state to a more distorted state– are often skipped in standard computer graphics literature. We believe instead that they are essential for plausible interactive deformations. The deformations at the beginning of the process do not only influence notably the final shape, but they are also very common to our daily life, and contrary to very distorted states where small artifacts may be unnoticed, these situations require therefore special care with respect to visual and audio feedback plausibility. We thus claim that they are a key point toward interactive realism.

Our model still has some limitations and there are many possibilities for improvements for future works.

- For the moment, self-collisions are not properly handled. This prevents us from obtaining entirely crumpled balls of paper. The sound of friction of the surface against itself is also a major component of the sound of crumpling paper. One interesting future work would be to integrate an efficient self-collision handling that takes advantage of the specific structure of our model and may provide the geometrical deformation and re-meshing adapted to the collision state.
- Many parts of our model rely on phenomenological observations. Although it still lets us obtain plausible results, better understanding of the underlying physics of paper may help to improve the method and its accuracy.
- Some properties of the paper that we neglected or simplified could be integrated into the model. Notably we consider the paper as an isotropic material. Actually most of papers are anisotropic, leading to some interesting phenomena. For example the bending stiffness would be different according to the direction of curvature. Also the path taken by the tear would change according to the direction of propagation.
- Our user-interface is not very advanced and only offers control over some chosen handles on the surface and the corresponding tangents as if moving fingers handling the paper. A better user-interface combined with a good handling of the collisions could enable a wide range of interactions and make the manipulation of the virtual paper really similar to the manipulation of real paper.
- Another future work, to improve the user-interface, would be to include haptic or pseudo-haptic feedbacks in order to better guide the motion given by the user. Indeed the stiffness of the paper, in particular in highly bent regions, influences the motions of someone manipulating real paper.

- In this thesis, we kept the assumptions that paper fibers can be considered as inextensible. However when paper gets burned or wet, the length of some fibers may change and the paper would not be developable anymore. Investigating this kind of phenomena may also be the object of some interesting future works.

Finally, combining the advantages of geometric, procedural and physics-based method leads us to present the first interactive model, to our knowledge, for creating animations of paper with immediate visual and audio feedback. While this model is dedicated to paper or material with similar properties, the idea to use prior knowledge to optimize a physical simulation may be applied to other materials that would be difficult to model with more general methods.



# Appendix A

# Résumé

## A.1 Introduction

Le papier est un matériau très commun que l'on utilise dans la vie de tous les jours. Pourtant on ne le trouve que très rarement dans les environnements virtuels. En effet le papier présente un comportement très spécifique et particulièrement complexe à reproduire par les moyens habituels. De ce fait, il n'existe pas pour le moment de méthode efficace, à la fois interactive et reproduisant le comportement caractéristique du papier.

Les principales caractéristiques nécessaires pour obtenir un comportement plausible du papier sont les suivantes:

- La conservation des longueurs: le papier est constitué de fibres que l'on peut considérer comme inextensibles. La surface du papier reste isométrique à son patron 2D (à savoir l'état initial de la feuille plane et intacte). Ainsi, lorsqu'il est soumis à des contraintes, le papier se plie plutôt que de se compresser, ou se déchire plutôt que de s'étirer.
- Les plis francs et les détails fins: lorsqu'il est déchiré et froissé, le papier présente des plis marqués et des détails le long des déchirures. Ces caractéristiques nécessitent habituellement un maillage très dense afin d'être représentées par les méthodes usuelles.
- Plasticité: la déchirure et le froissement du papier causent des dommages irréversibles dans la structure fibreuse du papier, ce qui modifie le comportement mécanique du papier.
- Le son dépendant de la géométrie: le son du papier est également très spécifique à ce matériau. En particulier, il dépend fortement de la forme prise par la surface.

Dans cette thèse, nous proposons un modèle capable de reproduire de façon plausible toutes ces caractéristiques tout en étant à la fois suffisamment rapide à calculer afin de pouvoir être manipulé interactivement.

Les méthodes existantes permettant de modéliser du papier ou de générer le son du papier peuvent être classifiées suivant deux catégories. D'un côté les méthodes dites *physiques* qui s'appuient sur des modèles à éléments finis appliquées à un maillage surfacique (employées initialement par Terzopoulos [TF88] en informatique graphique). Ces méthodes permettent d'obtenir des mouvements et déformations qui paraissent naturels et réalistes. Cependant modéliser du papier avec ces méthodes est particulièrement coûteux sur le plan calculatoire. Non seulement simuler l'inextensiblité de la surface impose une rigidité très importante et donc des pas de temps très courts, mais aussi représenter les plis francs nécessite un maillage extrêmement dense. Il est possible d'améliorer ces méthodes en utilisant un maillage adaptatif, comme proposé par Narain *et. al.* [NSO12, NPO13], qui devient plus dense dans les régions très courbées et plus grossier dans les régions qui le sont moins. Mais le maillage doit toujours être très dense au niveau de plis francs et des régions très courbées, ce qui empêche d'atteindre des temps interactifs. D'un autre côté, les méthodes procédurales s'appuient sur les propriétés géométriques du papier, des connaissances préalables ou des données. Ces méthodes sont en général beaucoup plus rapides que les simulations physiques mais elles nécessitent habituellement des entrées complexes de la part de l'utilisateur comme le bord 3D ([RCHbT11]) ou la position des plis sur le patron 2D ([SVWG12]). De plus il est difficile de reproduire un mouvement naturel en utilisant ces méthodes.

L'idée commune aux travaux présentés dans cette thèse est de combiner une simulation physique avec de nouvelles méthodes procédurales de façon à cumuler les avantages des deux types d'approches.

Nous appliquons notre approche hybride à trois cas d'applications différents:

- le papier froissé : notre méthode entrelace une simulation physique avec une étape géométrique qui remaille la surface de façon à l'adapter aux plis du papier.
- la déchirure du papier : nous proposons de séparer le calcul de la direction générale de la déchirure et la génération de détails fins le long de la déchirure. Nous proposons d'abord une méthode qui détermine la direction de déchirure de façon entièrement procédurale dans un cas particulier. Puis nous généralisons cette méthode en la couplant avec notre modèle de papier froissé et en appuyant le calcul de la direction de déchirure sur différents critères issus de la littérature physique.

- le son du papier : nous proposons une méthode pour générer le son du papier froissé qui s'appuie sur une base de données d'unités de son et le concept de *résonateur* pour produire un son qui dépend de la géométrie de la surface. Nous décrivons également une méthode procédurale pour créer le son correspondant à la déchirure du papier.

## A.2 Modéliser le papier froissé

En froissant du papier, on peut observer différents phénomènes. Notamment une feuille de papier plane peut être courbée dans n'importe quelle direction mais devient alors rigide dans la direction orthogonale de courbure nulle. Dans ce cas si l'on essaie de lui donner une seconde direction de courbure, des plis francs vont alors se créer et rompre le caractère lisse de la surface modélisant la feuille de papier. Dans notre approche, nous approximons ces régions où apparaissent les plis francs sous forme de points singuliers. Le papier est une surface développable, c'est-à-dire qu'il peut être déplié sur un plan sans être ni compressé ni étiré. Sa surface peut notamment être décomposée en morceaux de surface développable $C^2$ (surfaces réglées dont le plan tangent est constant le long d'une règle). Dans ce cas, et en tenant compte que l'on considère les plis francs comme des points, nous approximons la surface par un ensemble de morceaux de cônes généralisés -formés par un ensemble de règles qui se croisent en un point unique-, et de plans. Les sommets des cônes généralisés se situant à l'intérieur de la surface modélisent les points singuliers, et donc les régions plissées du papier.

La boucle d'animation de notre algorithme entrelace une simulation physique à éléments finis avec une étape de remaillage géométrique. Cette dernière est fondée sur les hypothèses précédemment décrites et approxime la surface par un ensemble régions courbes, modélisées par des cônes généralisés, et de régions planes afin de synthétiser un maillage spécifiquement adapté aux plis du papier.

L'étape géométrique est divisée en trois parties:

- la création de nouvelles régions courbes : la compression des régions planes du modèle physique élastique est analysée. Des régions courbes sont créées dans les parties compressées, et des règles sont insérées dans la direction de minimum de courbure. Lors de la création d'une région courbe, il peut arriver que les règles nouvellement créées croisent des règles déjà existantes. Dans ce cas, le conflit est résolu par l'insertion d'un point singulier.
- insertion d'un point singulier : dans la situation décrite ci-dessus, un point singulier est créé dans la zone de conflit. Sa position est choisie selon une loi de probabilité

basée sur la courbure de la zone de conflit. Un point singulier représentant le sommet d'un cône généralisé, un remaillage local est réalisé dans la région avoisinant ce point.

- mise à jour des régions courbes : afin de pouvoir représenter des surfaces lisses complexes, nous segmentons chacune des régions courbes en un ou plusieurs cônes généralisés.

Le maillage utilisé par la simulation physique est créé à partir de l'approximation en régions courbes et planes de sorte que :

- les points singuliers soient explicitement des points du maillage.
- les règles des régions courbes correspondent à des arrêtes du maillage.
- les régions planes sont triangulées de manière isotrope afin de permettre une courbure suivant une direction quelconque.

Cette approche est associée aux deux avantages suivants. Premièrement le maillage contient très peu de triangles et convient donc pour des applications interactives. Deuxièmement, les plis francs sont représentés de manière explicite et permettent de modéliser ainsi aisément le comportement plastique du papier en conservant ceux-ci dans le reste de l'animation.

Nous comparons nos résultats avec du papier réel dans des scénarios simples mais familiers dans lesquels le papier commence à se froisser. Nous comparons notamment la position du premier point singulier. Notre méthode est comparée également avec une méthode élément finis sur un maillage fixe, puis avec le maillage adaptatif proposé par Narain [NPO13]. Notre méthode parvient à des résultats de qualité similaire en temps interactifs alors que les deux autres méthodes prennent plusieurs secondes par image. D'autres déformations plus générales telles que la pliure d'un cylindre en papier ou encore d'une feuille de journal sont également proposées.

L'une des limitations actuelles de la méthode concerne la non-prise en compte des collisions internes qui ne permettent pas de modéliser, par exemple, une balle de papier entièrement froissée. Notons qu'une méthode de gestion de collision utilisant la structure particulière de notre modèle pourrait faire l'objet de travaux futurs. Une autre voie intéressante à développer serait de modéliser plus précisément les régions froissées sous la forme de courbes à la place de simplifier ceux-ci sous forme de points.

À notre connaissance, nous avons présenté le premier modèle permettant d'animer du papier froissé de façon interactive. Ce modèle traitant explicitement les zones lisses et

froissées du papier procure, en plus des maillages associés à l'animation, de nombreuses informations se révélant utiles pour implémenter d'autres fonctionnalités telles que la synthèse de la déchirure ou du son.

## A.3 Modéliser la déchirure du papier

La méthode présentée dans la section précédente traite du cas où les contraintes tendent à compresser le modèle physique élastique et aboutit au froissement du papier de par son inextensibilité. À l'opposé, dans le cas où les contraintes extérieures tendent à étirer la surface, le papier finit par se déchirer. Basée sur des études physiques ([Rom13]), la courbe décrite par la déchirure est reproductible (la même déformation appliquée sur deux feuilles différentes donnera des courbes de déchirures similaires). Par contre, les détails issus de la structure fibreuse du papier visible le long de la déchirure semblent stochastiques et dépendent principalement de la répartition aléatoire des fibres. Nous séparons donc le calcul de la trajectoire de la déchirure, de la génération des détails le long de celle-ci. Pour reproduire le phénomène de déchirure de façon plausible nous devons donc résoudre les problèmes suivants:

- inextensibilité locale : comme précédemment la surface du papier ne peut être ni compressée ni étirée localement.
- propagation de la déchirure : nous devons calculer quand et comment une déchirure va se propager.
- détails de la déchirure : les détails très fins de la déchirure doivent être calculés et représentés de façon efficace.

Nous proposons d'abord une méthode procédurale traitant un cas particulier. Deux mains sont représentées chacune par deux points (qui sont les positions du pouce et de l'index). Les mains imposent un mouvement de rotation tout en restant toujours dans le même plan. La direction de propagation de la déchirure est calculée comme une combinaison linéaire des vecteurs représentant la direction entre chacun des pouces et l'extrémité de la déchirure, chacun pondéré par l'angle de rotation de chacune des mains. Le chemin détaillé entre deux extrémités successives de la déchirure est calculé comme le plus court chemin dans une texture combinant la résistance des fibres et le stress généré par la déchirure. Le papier est représenté par deux régions planes articulées par un cône généralisé dont le sommet est l'extrémité de la déchirure. Le cône est calculé de façon à préserver l'isométrie du papier pendant la déformation. Cette méthode donne des résultats probants en temps réel mais reste limitée à une situation très spécifique.

Nous généralisons cette méthode en la couplant avec notre modèle de papier froissé. O'Keefe [OKe94] décrit les principes physiques décrivant la direction de propagation de la déchirure que nous réutilisons dans notre approche. Pour cela, nous considérons l'hypothèse que les forces à l'origine de la déchirure peuvent être modélisées localement par deux forces opposées appliquées sur l'extrémité dela déchirure. Suivant cette hypothèse, les principes de propagation de la déchirure décrits par O'Keefe peuvent être appliqués de manière géométrique et procédurale. De plus, notre approche permet de déterminer efficacement les points de départ de la déchirure définis par les régions où le stress se concentre et caractérisés explicitement dans notre modèle précédent comme étant des points singuliers.

Notre algorithme fonctionne suivant l'approche décrite ci-après. À chaque instant :

- la déformation est obtenue par le biais du modèle de papier froissé décrit au chapitre précédent.
- une liste de points potentiels de départ de déchirure est récupérée à partir des informations géométriques du modèle de papier froissé.
- pour chacun de ces points de déchirures potentiels, nous approximons les forces 3D, calculées à partir de la simulation physique pour chacun des triangles adjacents, en deux forces opposées. Cela nous permet de réduire le problème à la situation décrite par O'Keefe [OKe94], situation pour laquelle la direction de propagation du pli peut être décrite géométriquement dans l'espace 2D du patron.
- le chemin entre deux extrémités successives est calculé procéduralement puis représenté à l'aide de texture et non d'une triangulation de façon à garder le maillage peu dense.

Nous obtenons des résultats largement prometteurs bien qu'encore en développement à l'heure actuelle, notamment nous pouvons d'ores et déjà valider les courbes trajectoires des déchirures correspondant à l'étude de O'Keefe.

## A.4 Générer le son du papier

La dernière partie de notre travail concerne la synthèse du son du papier s'appuyant sur les modèles géométriques décrits précédemment. Notons tout d'abord que le son du phénomène visualisé permet d'améliorer considérablement l'impression de réalisme et d'immersion de cette scène. Bien que la synthèse de son commence à attirer l'intérêt des chercheurs en informatique graphique, le son est encore très souvent créé par des

artistes dans les applications standards. En particulier, il n'existe pour le moment aucune méthode capable de synthétiser efficacement le son d'un papier virtuel froissé. Nous proposons ici une telle synthèse prenant en compte la géométrie de la surface.

Nous identifions deux types de sons différents produits par le papier : les sons de friction et les sons de froissement. Nous constatons également que les sons dépendent grandement de la géométrie de la région qui les produit. Nous introduisons donc la notion de *résonnateur* : un résonnateur représente une région dans laquelle les vibrations produites par un évènement de friction ou de froissement se propagent. Chaque évènement (friction ou froissement) produisant un son va donc activer un certain nombre de résonnateurs, qui vont chacun produire un son dépendant de leur géométrie. Nous utilisons les régions courbes et planes fournies par le modèle de papier froissé comme résonnateurs potentiels.

Nous caractérisons l'influence de la géométrie suivant deux paramètres, qui ont une incidence majeure sur le son, pour décrire un résonnateur : la longueur des bords libres (bord de la surface) du résonnateur et la longueur moyenne des règles. Les sons produits par les résonnateurs proviennent d'une base de données contenant des unités de son. Chaque unité de son correspond au son enregistré d'un évènement (friction ou froissement) pour un résonnateur dont les paramètres sont connus. Les sons peuvent donc être classés dans la base de données selon les deux paramètres permettant ainsi de pouvoir accéder efficacement à un son avec des paramètres spécifiques.

Le son est généré suivant une approche dite de *synthèse concaténation* [S*00]. À chaque instant :

- le modèle d'animation calcule la nouvelle surface.
- les évènements de froissement et de friction sont détectés sur la surface.
- pour chaque évènement, une liste de résonnateurs est calculée en collectant itérativement, à partir de la location de l'évènement, les régions (courbes et planes du modèle d'animation) voisines jusqu'à atteindre une région bloquant les vibrations. Chacune de ses régions est considérée comme un résonnateur.
- pour chacun des résonnateurs de chaque évènement, un son est récupéré dans la base de données dépendant du type d'évènement (friction ou froissement) et des paramètres du résonnateur. Ce son est ensuite spatialisé pour tenir compte de sa position dans l'espace et de l'environnement.
- tous les sons sont envoyés au système audio de l'ordinateur.

Nous obtenons ainsi le son correspondant à une animation de papier que nous validons qualitativement par rapport à des enregistrements réels de papier froissé. Le son obtenu dépend de la forme prise par la surface, il est synchronisé avec l'animation et il peut être généré en temps réel à l'exécution pendant qu'un utilisateur manipule interactivement le papier.

La principale limitation de notre approche provient de la large utilisation d'observations phénoménologiques. Une meilleure compréhension des phénomènes physiques impliqués spécifiquement dans la génération du son du papier, ou encore apporter une solution mathématique à la propagation des vibrations dans une surface développable seraient des pistes potentielles à explorer pour améliorer cette approche.

## A.5 Conclusion

Nous avons présenté un modèle permettant de créer de façon interactive des animations de papier froissé et déchiré incluant la génération automatique du son correspondant. Nous nous sommes attachés à comparer notre modèle à des situations simples où le papier commence à se déformer à partir d'un état planaire, et qui sont souvent négligées dans les travaux d'informatique graphique. Pourtant ce sont les situations qui, non seulement nous sont très familières et faciles à reconnaitre, mais aussi déterminent en grande partie la forme finale du papier, et souhaitons appuyer par le biais de ce travail qu'elles sont essentielles à étudier.

Plusieurs voies pourraient être explorées dans des travaux futurs. Par exemple, implémenter une gestion efficace des collisions pourrait élargir le nombre de scénarios possibles. Une autre extension pourrait consister à explorer davantage les principes physiques du comportement du papier lié à sa structure fibrée afin d'améliorer la précision des modèles de simulation.

Au final, l'utilisation de méthodes hybrides - combinant simulation physique et méthodes procédurales - fondées sur des connaissances préalables pourrait également être utile pour modéliser d'autres matériaux complexes à modéliser et liés à d'autres contraintes que celle d'inextensibilité.

# Bibliography


[Adr91] Adrien J.-M.: The missing link: Modal synthesis. In *Representations of Musical Signals*, De Poli G., Piccialli A., Roads C., (Eds.). MIT Press, Cambridge, MA, USA, 1991, pp. 269–298.

[AHS07] Andresen C. A., Hansen A., Schmittbuhl J.: Ridge network in crumpled paper. *Physical Review E 76* (Aug 2007), 026108.

[AJM12] An S. S., James D. L., Marschner S.: Motion-driven concatenative synthesis of cloth sounds. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* (2012).

[AN06] Alava M., Niskanen K.: The physics of paper. *Reports on Progress in Physics 69*, 3 (2006), 669.

[AP97] Amar M. B., Pomeau Y.: Crumpled paper. *Proceedings of the Royal Society. Mathematical, Physical and Engineering Sciences.* (1997).

[ARR05] Audoly B., Reis P. M., Roman B.: Cracks in thin sheets: When geometry rules the fracture path. *Physiscal Review Letters 95* (Jul 2005), 025502.

[Aum03] Aumann G.: A simple algorithm for designing developable bézier surfaces. *Computer Aided Geometric Design 20*, 8 (2003), 601–619.

[BDS*12] Bouaziz S., Deuss M., Schwartzburg Y., Weise T., Pauly M.: Shape-up: Shaping discrete geometry with projections. *Computer Graphics Forum 31*, 5 (Aug. 2012), 1657–1667.

[BDT*08] Bonneel N., Drettakis G., Tsingos N., Viaud-Delmon I. l., James D.: Fast modal sounds with scalable frequency-domain synthesis. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 27*, 3 (2008).

[BFM08] Bourdin B., Francfort G. A., Marigo J.-J.: The variational approach to fracture. *Journal of elasticity 91*, 1-3 (2008), 5–148.

[BGW06] Burgoon R., Grinspun E., Wood Z.: Discrete shells origami. *Proceedings of Computers And Their Applications* (2006), 180–187.

[BHTF07] Bao Z., Hong J.-M., Teran J., Fedkiw R.: Fracturing rigid materials. *IEEE Transactions on Visualization and Computer Graphics 13*, 2 (Mar. 2007), 370–378.

[BJLW*99] Bar-Joseph Z., Lischinski D., Werman M., Dubnov S., Elyaniv R.: Granular synthesis of sound textures using statistical learning. In *Proceedings of the International Computer Music Conference* (1999), pp. 178–181.

[BK05] Blair D. L., Kudrolli A.: Geometry of crumpled paper. *Physical Review Letters 94* (2005).

[BMF03] Bridson R., Marino R., Fedkiw R.: Simulation of clocloth with folds and wrinkles. *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003).

[BW98] Baraff D., Witkin A.: Large steps in cloth simulation. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 43–54.

[BW07] Bo P., Wang W.: Geodesic-controlled developable surfaces for modeling paper bending. *Computer Graphics Forum 26*, 3 (2007).

[CAJ09] Chadwick J. N., An S. S., James D. L.: Harmonic shells: A practical nonlinear sound model for near-rigid thin shells. *ACM Transactions on Graphics 28*, 5 (2009), 119:1–119:10.

[CBBJR03] Cardle M., Brooks S., Bar-Joseph Z., Robinson P.: Sound-by-numbers: Motion-driven sound synthesis. *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), 349–356.

[Cia97] Ciarlet P. G.: *Mathematical elasticity: theory of plates*. North-Holland, 1997.

[CK02] Choi K.-J., Ko H.-S.: Stable but responsive cloth. *ACM Transactions on Graphics 21*, 3 (July 2002), 604–611.

[CM98] Cerda E., Mahadevan L.: Conical surfaces and crescent singularities in crumpled sheets. *Physical Review Letters 80*, 11 (1998).

[CM11] Cambou A. D., Menon N.: Three-dimensional structure of a sheet crumpled into a ball. *Proceedings of the National Academy of Sciences.* (2011).

[Coo02] Cook P. R.: *Real Sound Synthesis for Interactive Applications.* A. K. Peters, Ltd., Natick, MA, USA, 2002.

[CS99] Cook P. R., Scavone G.: The synthesis toolkit (stk). *Proc. International Computer Music Conference* (1999), 164–166.

[CS02] Chu C.-H., Séquin C. H.: Developable bézier patches: properties and design. *Computer-Aided Design 34*, 7 (2002), 511–527.

[CYFW14] Chen Z., Yao M., Feng R., Wang H.: Physics-inspired adaptive fracture refinement. *ACM Transactions on Graphics 33*, 4 (July 2014), 113:1–113:7.

[DBJEY*02] Dubnov S., Bar-Joseph Z., El-Yaniv R., Lischinski D., Werman M.: Synthesizing sound textures through wavelet tree learning. *IEEE Computer Graphics and Applications 22*, 4 (2002), 38–48.

[DCDC76] Do Carmo M. P., Do Carmo M. P.: *Differential geometry of curves and surfaces*, vol. 2. Prentice-hall Englewood Cliffs, 1976.

[DDMS12] Dias M. A., Dudte L. H., Mahadevan L., Santangelo C. D.: Geometric mechanics of curved crease origami. *Physical Review Letters 109* (Sep 2012), 114301.

[DJW*06] Decaudin P., Julius D., Wither J., Boissieux L., Sheffer A., Cani M.-P.: Virtual garments: A fully geometric approach for clocloth design. *Computer Graphics Forum (Proc. of Eurographics) 25*, 3 (2006).

[DP98] Doel K. v. d., Pai D. K.: The sounds of physical shapes. *Presence: Teleoper. Virtual Environ. 7*, 4 (1998), 382–395.

[EB08] English E., Bridson R.: Animating developable surface using nonconforming elements. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 27*, 3 (2008).

[Fre98] Freund L. B.: *Dynamic fracture mechanics.* Cambridge university press, 1998.

[Fre04] Frey W. H.: Modeling buckled developable surfaces by triangulation. *Computer Aided Design 36* (2004).

[GHD*03] Grinspun E., Hirani A. N., Desbrun M., , Shröder. P.: Discrete shells. *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation 27* (2003), 62–67.

[GMD12] Glondu L., Marchal M., Dumont G.: Real-Time Simulation of Brittle Fracture using Modal Analysis. *IEEE Transactions on Visualization and Computer Graphics 19*, 2 (Aug. 2012), 201–209.

[Gri21] Griffith A. A.: The phenomena of rupture and flow in solids. *Philosophical transactions of the royal society of london. Series A, containing papers of a mathematical or physical character 221* (1921), 163–198.

[GS74] Goldstein R. V., Salganik R. L.: Brittle fracture of solids with arbitrary cracks. *International Journal of Fracture 10*, 4 (1974), 507–523.

[GSH*04] Gingold Y., Secord A., Han J. Y., Grinspun E., Zorin D.: A discrete model for inelastic deformation of thin shells. *Technical Report* (2004).

[GWV02] Gopinathan A., Witten T. A., Venkataramani S. C.: Trapping of vibrational energy in crumpled sheets. *Physical Review E 65* (Feb 2002), 036613.

[HHAL16] Hernandez E. A. P., Hartl D. J., Akleman E., Lagoudas D. C.: Modeling and analysis of origami structures with smooth folds. *Computer-Aided Design* (2016), –.

[How02] Howe M. S.: *Theory of Vortex Sound*. Cambridge University Press, 2002. Cambridge Books Online.

[HS93] Hodgdon J. A., Sethna J. P.: Derivation of a general three-dimensional crack-propagation law: A generalization of the principle of local symmetry. *Physical Review B 47* (Mar 1993), 4831–4840.

[HS96] Houle P. A., Sethna J. P.: Acoustic emission from crumpling paper. *Physical Review E 54*, 1 (1996), 278.

[Huf76] Huffman D. A.: Curvature and creases: A primer on paper. *IEEE Trans. Comput. 25*, 10 (Oct. 1976), 1010–1019.

[IO06] Iben H. N., O'Brien J. F.: Generating surface crack patterns. In *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Sept 2006), pp. 177–185.

[JBP06] James D. L., Barbic J., Pai D. K.: Precomputed acoustic transfer: Output-sensitive, accurate sound generation for geometrically complex vibration sources. *ACM Transactions on Graphics (Proc. of SIGGRAPH)* (2006), 987–995.

[JHR*15] Jung A., Hahmann S., Rohmer D., Begault A., Boissieux L., Cani M.-P.: Sketching Folds: Developable Surfaces from Non-Planar Silhouettes. *ACM Transactions on Graphics 34*, 5 (2015), 155:1–155:12.

[JKS05] Julius D., Kraevoy V., Sheffer A.: D-charts: Quasi-developable mesh segmentation. *Eurographics* (2005).

[JP02] James D. L., Pai D. K.: Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Transactions on Graphics 21*, 3 (2002), 582–585.

[KFC*08] Kilian M., Floery S., Chen Z., Mitra N., Sheffer A., Pottmann H.: Curved folding. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 27*, 3 (2008).

[KGK94] Kergosien Y., Gotoda H., Kunii T.: Bending and creasing virtual paper. *IEEE Computer Graphics and Applications 14*, 1 (1994).

[KL96] Kramer E. M., Lobkovsky A. E.: Universal power law in the noise from a crumpled elastic sheet. *Physical Review E 53*, 2 (1996), 1465.

[KZC09] Kang Y.-M., Zhang H.-G., Cho H.-G.: Plausible virtual paper for real-time application. *Computer Animation and Social Agents (CASA), Short Paper* (2009).

[LAJJ14] Langlois T. R., An S. S., Jin K. K., James D. L.: Eigenmode compression for modal sound models. *ACM Transactions on Graphics 33*, 4 (July 2014), 40:1–40:9.

[LBOK13] Liu T., Bargteil A., O'Brien J., Kavan L.: Fast simulation of mass-spring systems. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia) 32*, 6 (2013).

[LFD*15] Lejemble T., Fondevilla A., Durin N., Blanc-Beyne T., Schreck C., Manteaux P.-L., Kry P. G., Cani M.-P.: Interactive procedural simulation of paper tearing with sound. In *Motion In Games (MIG)* (Paris, France, Nov. 2015).

[LG03] Leblond J.-B., Germain P.: *Mécanique de la rupture fragile et ductile.* Hermès science publications, 2003.

[LP98] Leopoldseder S., Pottmann H.: Approximation of developable surfaces with cone spline surfaces. *Computer Aided Design 30* (1998).

[LPW*06] Liu Y., Pottmann H., Wallner J., Yang Y.-L., Wang W.: Geometric modeling with conical meshes and developable surfaces. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)* (2006).

[LW97] Lobkovsky A., Witten T.: Properties of ridges in elastic membranes. *Physical Review E 55* (1997).

[LW05] Liang T., Witten T. A.: Crescent singularities in crumpled sheets. *Physical Review E 71* (Jan 2005), 016612.

[MAKMV10] Marelli D., Aramaki M., Kronland-Martinet R., Verron C.: Time-frequency synthesis of noisy sounds with narrow spectral components. *IEEE Transactions on Audio, Speech and Language Processing 18*, 8 (2010), 1929–1940.

[MBCN09] Metaaphanon N., Bando Y., Chen B.-Y., Nishita T.: Simulation of tearing cloth with frayed edges. In *Computer Graphics Forum* (2009), vol. 28, Wiley Online Library, pp. 1837–1844.

[MC98] Mahadevan L., Cerda E.: Conical surfaces and crescent singularities in crumpled sheets. *Physical Review Letters 80*, 11 (1998).

[MCK13] Müller M., Chentanez N., Kim T.-Y.: Real time dynamic fracture with volumetric approximate convex decompositions. *ACM Transactions on Graphics 32*, 4 (July 2013), 115:1–115:10.

[MI11] Mitani J., Igarashi T.: Interactive design of planar curved folding by reflection. *Pacific Graphics (short paper)* (2011).

[NF99] Neff M., Fiume E.: A visual model for blast waves and fracture. In *Proceedings of the 1999 Conference on Graphics Interface '99* (San Francisco, CA, USA, 1999), Morgan Kaufmann Publishers Inc., pp. 193–202.

[NPO13] Narain R., Pfaff T., O'Brien J. F.: Folding and crumpling adaptive sheets. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 32*, 4 (2013).

[NSO12] Narain R., Samii A., O'Brien J. F.: Adaptive anisotropic remeshing for cloth simulation. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia) 31*, 6 (2012).

[OBH02] O'Brien J. F., Bargteil A. W., Hodgins J. K.: Graphical modeling and animation of ductile fracture. In *Proceedings of ACM SIGGRAPH* (Aug. 2002), ACM Press, pp. 291–294.

[OCE01] O'Brien J. F., Cook P. R., Essl G.: Synthesizing sounds from physically based motion. *Proc. of ACM SIGGRAPH* (2001), 529–536.

[OH99] O'Brien J. F., Hodgins J. K.: Graphical modeling and animation of brittle fracture. In *Proceedings of ACM SIGGRAPH* (Aug. 1999), ACM Press/Addison-Wesley Publishing Co., pp. 137–146.

[OKe94] OKeefe R.: Modeling the tearing of paper. *American Journal of Physics 62*, 4 (1994), 299–305.

[OSG02] O'Brien J. F., Shen C., Gatchalian C. M.: Synthesizing sounds from rigid-body simulations. *Proc. of ACM SIGGRAPH* (July 2002), 175–181.

[Pet04] Peternell M.: Developable surface fitting to point clouds. *Computer Aided Geometric Design 21*, 8 (Oct. 2004), 785–803.

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

[PNdJO14] Pfaff T., Narain R., de Joya J. M., O'Brien J. F.: Adaptive tearing and cracking of thin sheets. *ACM Transactions on Graphics 33*, 4 (July 2014), xx:1–9.

[Pro96] Provot X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. *Graphics Interface* (1996), 147–154.

[PW89] Pentland A., Williams J.: Good vibrations: Modal dynamics for graphics and animation. *SIGGRAPH Computer Graphics 23*, 3 (July 1989), 207–214.

[PW01] Pottmann H., Wallner J.: *Computational Line Geometry*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[PW10] Pele O., Werman M.: The quadratic-chi histogram distance family. In *Computer Vision – ECCV*, Daniilidis K., Maragos P., Paragios N., (Eds.), vol. 6312 of *Lecture Notes in Computer Science*. Springer, 2010, pp. 749–762.

[RCHbT11] Rohmer D., Cani M.-P., Hahmann S., boris Thibert: Folded paper geometry from 2d pattern and 3d contour. *Eurographics Short Paper* (2011), 21–24.

[RDRD92] Rodet X., Depall P., Rodet X., Depalle P.: Spectral envelopes and inverse fft synthesis. In *Proc. of Audio Engineering Society Convention* (1992).

[RL06] Raghuvanshi N., Lin M. C.: Interactive sound synthesis for large scale environments. In *Proc. Symp. on Interactive 3D Graphics and Games* (2006), I3D '06, ACM, pp. 101–108.

[Roa04] Roads C.: *Microsound*. The MIT Press, 2004.

[Rom13] Roman B.: Fracture path in brittle thin sheets: a unifying review on tearing. *International Journal of Fracture 182*, 2 (2013), 209–237.

[RPC*10] Rohmer D., Popa T., Cani M.-P., Hahmann S., Sheffer A.: Animation Wrinkling: Augmenting Coarse Cloth Simulations with Realistic-Looking Wrinkles. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia) 29*, 5 (December 2010).

[RSW*07] Rose K., Sheffer A., Wither J., Cani M.-P., Thibert B.: Developable surfaces from arbitrary sketched boundaries. *Symposium on Geometry Processing (SGP)* (2007).

[RYL13] Ren Z., Yeh H., Lin M. C.: Example-guided physically based modal sound synthesis. *ACM Transactions on Graphics 32*, 1 (2013), 1:1–1:16.

[S*00] Schwarz D., et al.: A system for data-driven concatenative sound synthesis. In *Digital Audio Effects (DAFx)* (2000), pp. 97–102.

[SC05] Scavone G., Cook P. R.: Rtmidi, rtaudio, and a synthesis toolkit (stk) update. *Proc. International Computer Music Conference* (2005).

[SHCB15] Shen Z., Huang J., Chen W., Bao H.: Geometrically exact simulation of inextensible ribbon. In *Computer Graphics Forum* (2015), vol. 34, Wiley Online Library, pp. 145–154.

[SKD11] Schroll R. D., Katifori E., Davidovitch B.: Elastic building blocks for confined sheets. *Physical Review Letters 106* (2011).

[SP74] Seth R., Page D.: Fracture resistance of paper. *Journal of Materials Science 9*, 11 (1974), 1745–1753.

[SRH*15] Schreck C., Rohmer D., Hahmann S., Cani M.-P., Jin S., Wang C. C., Bloch J.-F.: Non-smooth developable geometry for interactively animating paper crumpling. *ACM Transactions on Graphics 35*, 1 (2015).

[SRJ*16] Schreck C., Rohmer D., James D. L., Hahmann S., Cani M.-P.: Real-time sound synthesis for paper material based on geometric analysis. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2016)* (Zürich, Switzerland, July 2016).

[STA02] Salminen L., Tolvanen A., Alava M.: Acoustic emission from paper fracture. *Physical Review Letters 89*, 18 (2002), 185503.

[SvWC14] Souza M. S., von Wangenheim A., Comunello E.: Fast simulation of cloth tearing. *SBC Journal on Interactive Systems 5*, 1 (2014).

[SVWG12] Solomon J., Vouga E., Wardetzky M., Grinspun E.: Flexible developable surfaces. *Computer Graphics Forum (Proc. of Symposium on Geometry Processing) 31*, 5 (2012).

[Tac09] Tachi T.: Simulation of rigid origami. In *Origami 4* (October 2009), K. P. A., (Ed.), Editorial de la Universitat Politécnica de Valencia, p. 175.

[Tac10] Tachi T.: Origamizing polyhedral surfaces. *IEEE Transactions on Visualization and Computer Graphics 16*, 2 (2010), 298–311.

[Tac11] Tachi T.: Rigid-foldable thich origami. In *Origami 5:Fifth International Meeting of Origami Science Mathematics and Education* (2011), Wang-Iverson P. Lang R. J. Y. M., (Ed.), pp. 253–264.

[TÅT09] Tallinen T., Åström J., Timonen J.: The effect of plasticity in crumpling of thin sheets. *Nature materials 8*, 1 (2009), 25–29.

[TF88] Terzopoulos D., Fleischer K.: Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. *SIGGRAPH Comput. Graph. 22*, 4 (June 1988), 269–278.

[TH92] Takala T., Hahn J.: Sound rendering. *Proc. SIGGRAPH 92, ACM Computer Graphics* (1992), 211–220.

[TMT10] Tang M., Manocha D., Tong R.: Fast continuous collision detection using deforming non-penetration filters. *Symposium on Interactive 3D Graphics and Games* (2010).

[VCMT95] Volino P., Courchesne M., Magnenat Thalmann N.: Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1995), SIGGRAPH '95, ACM, pp. 137–144.

[vdDKP01] van den Doel K., Kry P. G., Pai D. K.: Foleyautomatic: Physically-based sound effects for interactive simulation and animation. *Proc. of ACM SIGGRAPH* (2001), 537–544.

[Wan08] Wang C.: Towards flattenable mesh surfaces. *Computer-Aided Design 40*, 1 (2008), 109–122.

[Wit07] Witten T.: Stress focusing in elastic sheets. *Review of Modern Physics 79* (2007).

[WOR10] Wang H., O'Brien J. F., Ramamoorthi R.: Multi-resolution isotropic strain limiting. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia) 29*, 6 (2010), 160.

[WT04] Wang C., Tang K.: Achieving developability of a polygonal surface by minimum deformation: a study of global and local optimization approaches. *The Visual Computer 20*, 8 (2004), 521–539.

[ZIM13] Zhu L., Igarashi T., Mitani J.: Soft folding. *Computer Graphics Forum (Proc. of Pacific Graphics) 32*, 7 (2013), 167–176.

[ZJ10] Zheng C., James D. L.: Rigid-body fracture sound with precomputed soundbanks. *ACM Transactions on Graphics (Proc. of SIGGRAPH) 29*, 3 (2010).