Painting Folds using Expansion Textures

Jean Combaz Fabrice Neyret
iMAGIS - GRAVIR / IMAG - INRIA †

Figure 1. Examples of created folds.

Abstract

We present a method relying on expansion textures *to add details such as folds on a surface. The user paints the expansion attributes on the surface (amount and direction of expansion, wavelength and regularity of folds), using either interactive or procedural tools. The modeling system generates the folds by calculating the new surface equilibrium. Results show that this tool allows CG artists to easily control the aspect of folds and drapes by adding surface locally, which is close to the way sculptors think. Our original contribution does not lie in the equilibrium solver, but in the very principle of texturing expansions to specify shapes details.*

Keywords: *surface modeling, user interaction, procedural modeling, folds, growth, details, imperfections.*

1. Introduction

Drape and folds occur in many situations (see Figure 2), from gravity or friction acting on cloth material to growth of elastic surfaces with constraints (e.g. aged coat layers, biological or geological developing surfaces). The sequence of actions that have shaped a given folded surface can be very complex or unknown (e.g. for an unmade bed sheet), and the original state of the surface can be much artificial (e.g. for a cloth). As a consequence the physical simulation of these objects, which supposes that initial state and acting forces are known, is often difficult to apply in practice.

However, traditional artists know how to paint or sculpt drapes, without having to define state and forces nor having to simulate the physics. Moreover, different situations can lead to similar shapes, what allows artists to rely on various intuitions to interpret the shapes they have in mind or in front of their eyes. For instance, artists create new shapes or modify an existing one (either mentally or in reality) by adding or suppressing matter, rather than considering an ideal initial state and applying a series of physical forces to shape it.

Our purpose is to ease the work of CG artists by offering them a painting-oriented way of designing physical deformations such as drapes and folds. Our approach consists in starting with an initial state which is an approximation of the final state (e.g. a cylinder in the case of the tablecloth of Figure 2, or the body for drape clothes). This is similar to a sculptor or a painter first sketching the envelop or the silhouette of his work. Then we implement the 'adding matter' paradigm for shaping the surface by considering user-controlled expansion or contraction operators, which can be isotropic or anisotropic. The control can be either interac-

† *iMAGIS* is a joint research project of CNRS,INPG,INRIA,UJF.
{Jean.Combaz|Fabrice.Neyret}@imag.fr
http://www-imagis.imag.fr/Membres/Jean.Combaz/

tive through maps, or procedurally based. The modeling system produces the folds by solving the surface equilibrium, which is eased by the fact that the initial state is geometrically close to the final state.

Our engine is similar to a physical simulator, although it does not need to simulate the dynamics. Moreover we subdivide the surface only whenever needed, i.e. when and where the user adds folds. In this paper we show several applications of this principle in order to illustrate its convenience.

Figure 2. Some drapes and folds from the real world: a scrunchy, real or sculpted clothes, a tablecloth, an aged paint coat, folds on macadam and a plastic cover lying on the ground.

This paper is structured as follows: Section 2 reviews the existing work related to the modeling of details and the simulation of folds. Then, we describe our *expansion texture* technique from the user's point of view in Section 3, and from the programmer's point of view in Section 4. We show and discuss our results in Section 5, and conclude in 6.

2. Previous Work

There are different ways exist to shape the geometric details necessary to make CG objects look real. This includes two different aspects: how to define or control the *content* and how to *represent* the details.

The first aspect is addressed by:
- interactive tools allowing the user to define explicitly the details, such as free form deformations (FFD) [24, 7] or direct painting on surface [16].
- procedural tools which let the user control the parameters of an automatic detail generator, which can be generic [20, 11, 13] or specialized [12, 23, 22, 1, 32].
- simulation tools which reproduce physical laws to be solved, used in particular for cloth material [25, 5, 2] and for some biological patterns [14, 31, 27].

The second aspect, i.e. the way to represent the details, may involve:
- polygonal meshes and other 3D surface encodings,
- displacement maps, i.e. texture encoded relief that can be translated into geometry at rendering time [29, 15],
- volumetric textures [17, 18] and hypertextures [21], encoding the details on a non-surfacic way,
- bump maps [4] which fake geometry by modifying the shading of the surfaces.

Transition between these representations are defined in [3, 6].

The idea of simulating growth has been introduced in the scope of biological objects [28, 23, 22]. The shape of elastic surfaces such as cloth material at equilibrium is generally obtained through physical simulations [25, 5, 2] although some geometric tools are also used to fake the physics [9]. As D'Arcy Thompson [26] suggests for natural objects, there are several possible approaches to explain a given shape.

Interactive tools like Maya Artisan^{tm} are convenient for the user, but they require the explicit design of shapes which can be very tedious for details. Simulation tools are accurate and generate natural looking shapes, but obtaining the final shape can take a long time. Moreover the user has to define an initial state and to explicitly provide the material parameters and the acting forces which might not be known: a sculptor reproducing a shape he has in mind or in front of his eyes cannot easily describe it as a mechanical experiment. Procedural tools allow for a high level control of the shape, which is very useful for the user. However such tools have been proposed mainly in the scope of texturing [20, 11] and only few of them are used to create geometry.

Our goal is to model geometrical shapes looking similar to the result of physical simulation while providing the user with high level of control of the shape details, as if he were

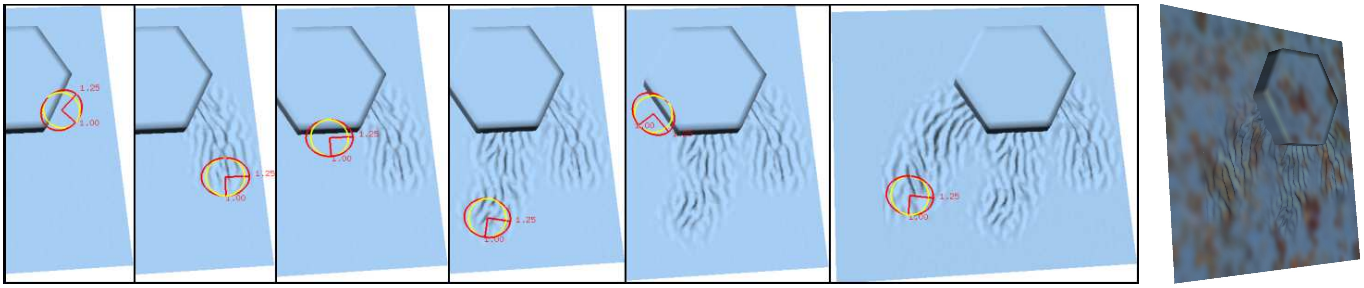

Figure 3. Interactive painting of folds in the paint coat around a bolt (see Figure 2). Here, the dark ellipse on the tool figures the amount and direction of expansion as compared to the identity (figured by the light circle). The tool orientation follows the direction of the mouse.

using a procedural tool. Our main concern is thus the first aspect reviewed above, i.e. the way the generated details are created. On the second aspect, though our current implementation is limited to generating polygonal meshes, it should be possible to adapt our scheme to the generation of other representations such as displacement maps or bump maps. In this paper, we concentrate on drape and fold patterns which have been largely used in classical art, but are very difficult to model using today's CG tools.

3. *Expansion textures* from the user's point of view

The principle is to let the user control the high-level parameters either globally or by painting their variations on the surface (through various means, either interactive or not). The solver (which is part of the modeling system) modify the mesh to generate the new surface equilibrium, thus adding details – the folds – to the main surface. This result will be computed either interactively or off-line depending of the complexity of the task. Figure 3 illustrates an interactive session.

The handles correspond to the expansion: the user controls the amount and direction of surface expansion, and provides extra information such as the desired wavelength and the amount of regularity. The expansion is given by a tensor field, which can be figured locally by an ellipse. In the special case of an unidirectional expansion, this tensor field collapses into a vector field.

The secondary handles correspond to the surface degrees of freedom. In our implementation, we account for the freezing of surface locations in 1 to 3 directions, extra forces (representing sticking or gravity), and collision detection (e.g. to simulate sticky coat or molded shapes).

The parameters corresponding to the primary handles are similar to the shaders in powerful rendering software. They can be input in different ways depending whether they are uniform or locally variable, and how the user prefers to tune them. Moreover, these parameters can be a scalar, a vector or a tensor. In our implementation, we allow the user to control the parameters interactively, procedurally or through maps. Non-scalar maps can be considered either as coupled scalar maps (which can be specified using regular paint programs) or vector maps. Although we implemented a GUI for parameters specification, our contribution is not in the human-computer interaction: numerous articles and softwares already deal with the painting of attributes, including vectorial or tensorial elements (e.g. the brush orientation in a paint program).

As for texture shaders, the expected complexity and the required quality of the result conditions the context of use: simple surface editing can be processed interactively while highly complex shapes are better specified by relying on maps or procedural definitions.

4. From the programmer's point of view

As stated in the introduction, defining the surface equilibrium knowing the constraints is a standard mechanical problem, for which various classical solutions exist (e.g., Finite Elements). We describe in the following our simplified implementation, which is based on existing techniques.

We rely on a triangular mesh to represent the free form surface on which details are to be created. We define the *reference state* as the length l_0 of each edge and the mean curvature κ_0 at each vertex. We consider the l_0 values as the rest length of the edges, which will be updated according to the growth. Thus this is a *virtual* reference state, since it is only locally attainable. After having applied the user defined expansion or contraction acting on the l_0, we rely on an iterative solver to compute the new surface equilibrium. Each iteration moves the vertices in order to decrease surface stress. The stress is obtained by comparing the present location to the local reference state. In the following l is the edge length and κ is the mean curvature over the iterations. Our surface can be oriented, allowing surface growth on a preferred side (*e.g.,* if the surface corresponds to the boundary of a filled volume). This is done by providing oriented normals $\vec{N}$.

Here is the overview of our algorithm, which we will describe in detail in the following sections:

- apply expansion or contraction on the surface by modifying the rest-length values l_0;
- optimize the mesh (possibly by adding or deleting edges);
- iterate small displacements $\epsilon\vec{F}$ that decreases the stress.

In case of huge expansion, this scheme has to be adapted. We deal about this in Section 4.5.

4.1. Expansion

The expansion acts on the edge lengths: given the expansion tensor field $\tau(u,v)$, an edge $\vec{l}$ is lengthened by a factor $\sqrt{\vec{l}^t\tau\vec{l}}$. In practice we integrate $\tau(u,v)$ along the edge to get the mean value. MIP-mapping of the expansion texture could be used as well.

4.2. Optimizing the mesh

While the initial mesh can be sparse, modeling small details requires a dense final mesh. We rely on mesh subdivision to fit with the required wavelength and the scale of local parameters gradients. Moreover we re-triangulate deformed triangles adaptively in order to maintain a good triangulation quality. Mesh subdivision and retriangulation are important in practice for stability and performance issues. We implemented edge swap, edge splitting and edge collapse (see Figure 4) using Delaunay inspired criterion applied to free surfaces (see for instance [30]). We flip edges to maximize the minimum angle triangles. We simply split an edge when its length or when the angle between the normals at its vertices exceeds a limit (these limits are defined by the user) Moreover, we insert vertices where the expansion gradient is important in order to maintain accuracy of our simulation. Edge collapse is done on the contrary case using smaller limits to ensure an hysteresis between these two operations (this keeps the stability of the mesh configuration).

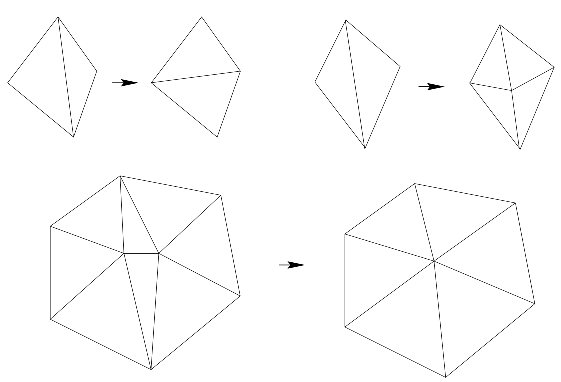

Figure 4. Edge swap, edge splitting and edge collapse.

4.3. Evaluating the stress

As for classical Finite Elements the strain can be obtained from the location of vertices and the rest lengths. We rely on the Green-Lagrange tensor (1) rather than the Cauchy tensor since it is more adapted to large deformation and not too complicated to evaluate (see discussions about this choice in [8]). We evaluate this tensor at each triangle. It is represented by a 2×2 matrix:

$$(\epsilon)_{ij} = \left(\frac{\partial x}{\partial \Omega_i} \cdot \frac{\partial x}{\partial \Omega_j}\right) - \delta_{ij} \tag{1}$$

where δ is the Kronecker delta function [1] and (Ω_1,Ω_2) is a local coordinate system of the element. This tensor gives the deformation between the triangle in the reference state (defined by its rest lengths) and its present state.

Then the stress tensor σ can be deduced using the Hook's law (assuming isotropic linear elasticity), yielding the forces $\vec{F}$ at the vertices:

$$\sigma = \lambda tr(\epsilon)I + 2\mu\epsilon \tag{2}$$

where μ represents the surface rigidity and λ measures its incompressibility. See for instance [19], where the problem is exposed in 3D. Since we deal with a 2D free form surface, this has to be restricted to flat 2D, then adapted to curved shapes.

The forces which express the tangential deformation of the surface are deduced from σ. At each node i we sum the contributions $\vec{F_{ki}}$ of triangles k which shared the vertex i where:

$$\vec{F_{ki}} = -\frac{a_{0_k}}{2}\sum_{j=1}^{3}\vec{x_j}\sum_{\alpha=1}^{2}\sum_{\beta=1}^{2}(L^{ki})_\alpha(L^{kj})_\beta(\sigma)_{\alpha\beta} \tag{3}$$

a_{0_k} is the area of the triangle k, and L^{kj} its linear basis vector for the node j. This vector depends only on the rest lengths. $\vec{x_j}$ is the vertex j 3D position. The sum of $\vec{F_{ki}}$ is first projected in the tangent plane.

Then surface curvature forces $\vec{F_\kappa}$ and normal stress $\vec{F_N}$ have to be added. $\vec{F_\kappa}$ is a restoring force which tends to smooth the surface by limiting the difference of curvature as compared to the reference state. $\vec{F_N}$ is a folding forces which translates the triangle compression into a displacement along the normal. We also use it to control the fold shapes. In order to avoid the complexity of the shell theory, we use simplifications inspired by [10]: we chose

$$\vec{F_\kappa} = -k_\kappa.(\kappa - \kappa_0)\vec{N} \tag{4}$$

$$\vec{F_N} = (k_p \mathrm{f}(\kappa - \kappa_0) + k_{p_i})C_a\vec{N} \tag{5}$$

κ and $\vec{N}$ are computed using the same neighbor vertices interpolation as [10]. k_κ and k_p are two constants. k_{p_i} is a

[1] δ_{ij} equals 1 if i=j and equals 0 otherwise.

bias allowing to push the surface in a preferred direction ($\vec{N}$ or -$\vec{N}$). If no direction is preferred k_{p_i} could be zero, but it is better to set it at each vertex using small random values. This avoid flat surfaces to get stuck in 2D despite having a high internal constraint. C_a is the area compression rate of the dilated surface. Considering the cell around a vertex, we have:

$$C_a = \frac{\sum_{cell} a_{0_i} - a_i}{\sum_{cell} a_{0_i}} \tag{6}$$

where a_0 and a are the rest and actual triangle areas (deduced from the edge length l_{0_i} and l_i). $f(\kappa)$ is a function which controls the shape of folds. We explain how to choose it in the next paragraph.

Note that all the forces are relatively independent of the discretization. They are more accurate in case of almost equilateral and smaller triangles but there is no bias. If an equilibrated surface is subdivided the same shape will be kept except for small numerical errors.

4.4. Controlling the folds shape

We introduce $\kappa^* = \kappa + L(\kappa)$ where L is a smoothing kernel. We chose $f(\kappa) = \sigma(\kappa^*)$ where $\sigma()$ is a sigmoid [2]. For high κ^* we only want to know in which direction to push the surface. Let us illustrate this in 1D with $L = \nu \frac{\partial^2}{\partial x^2}$. At equilibrium we have $F = 0$, thus $\kappa^* = 0$. So $\kappa + \nu \frac{\partial^2 \kappa}{\partial x^2} = 0$, yielding $\kappa = \cos(\frac{x}{\sqrt{\nu}})$. Thus to get folds with a wavelength λ, one has to choose $\nu = (\frac{\lambda}{2\pi})^2$ in the smoothing kernel. In fact, we use an anisotropic operator: the value in the folding direction is chosen as above, and the coefficient in the orthogonal direction (*i.e.,* along the folds) is chosen according to the desired fold regularity. Similarly, we chose $\nu_y = (\frac{\Lambda}{2\pi})^2$ with Λ the desired 'coherency length' of the folds (see Figure 5).

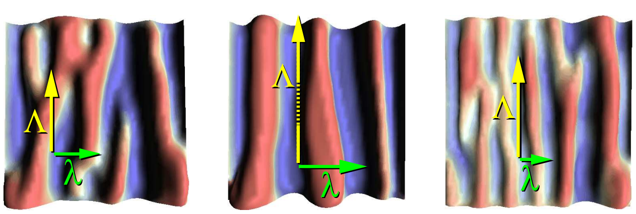

Figure 5. Tuning of the wavelength λ and the regularity Λ.

[2] A sigmoid is a monotonous function varying from -1 to 1 with a fast transition around 0.

4.5. Modification of the algorithm in case of huge expansion

If the amount of expansion is reasonable, enlarging the rest lengths in one step is sufficient. Otherwise it is better to iterate, applying successive steps of partial expansion and surface equilibrium.

For very huge expansion (for morphogenesis which is beyond the scope of this paper) it is necessary to add plasticity in order to avoid accumulating undissipated stress in the surface. Plasticity corresponds to fading the memory of the initial state, which is done by relaxation of the rest length: at each iteration l_0 is replaced by $(1-\varepsilon)l_0 + \varepsilon l$. It can be useful to add some plasticity even in the scope of modeling surface details, especially if the expansion field is complex enough so that some stress cannot be released. Otherwise folds could look distorted).

4.6. Optimizations

In our implementation we rely on several classes of optimization:

- As mentioned in Section 4.2, we adapt the mesh to the needs of the simulation.
- Our solver uses a different time step for each vertex in order to spend time only where needed. This adaptation is done in the spirit of [8]. We calculate the maximum time step allowable for each point. The smallest of them defines dt. If a vertex i requires a time step $dt_i \in [2^n dt, 2^{n+1} dt[$, we update its forces only when the time is a multiple of $2^n dt$.
- In the case of interactive design, we maintain an active area outside which no calculation is done. The interactive tool expands or contracts a limited circular area on the surface. We consider a larger circular area around it, where the surface aspect is likely to be affected. We activate the simulation only in this area. An iteration counter attached to the included vertices is set, then a vertex is deactivated when its counter reach zero.

5. Results

In Figure 7 (on the color page at the end of the paper), we show an uniform unidirectional expansion with small wavelength and low regularity, and with large wavelength and higher regularity. For comparison, we show on the left the folds of a real plastic cover. On the right of the figure we present gathers simply obtained by painting the expansion as a narrow band in the map. On the right image of Figure 1, we used an increasing amount of expansion from the top to the bottom of the surface, and activated sticking. The sticking is equivalent to gravity force plus collision detection with the floor of an horizontal surface. It results in positive folds separated by relatively flat areas.

In Figure 8, a circular unidirectional expansion is applied in a reduced ring area (the map is on the left end), with various tunings of wavelength and regularity: On the left the wavelength is very small and the regularity almost zero so that the pattern looks random. On the right end the wavelength is large and the regularity is higher so that the pattern shows more organized folds.

Figure 9 shows a CG scrunchy with various amount of expansion. It is obtained from a torus shape, by dilating largely along the large circle direction and slightly along the small circle direction as figured in the left. Like in reality, the surface has to be kept narrow to avoid getting simply a smooth inflated torus. In reality this is done by inserting a rubber band inside. We achieve this by inserting a small rigid torus inside the large soft torus (i.e. collision detection prevents the large torus to grow wide instead of folding). Other solutions can be used to achieve this purpose: in the middle image of Figure 1, we defined several rigid bands on the surface (which is much equivalent to real clothes with underwire). Despite the fact that no self-intersecting test is done, the curvature smoothing is able to prevent from local self-intersections (a pure geometric deformation method such as displacement mapping would not achieve this). Without any test distant folds can intersect, however. But it appears that this generally occurs for an amount of expansion greater than three (see Figure 6). Further expansion on Figure 9 would probably shows such self-intersections.

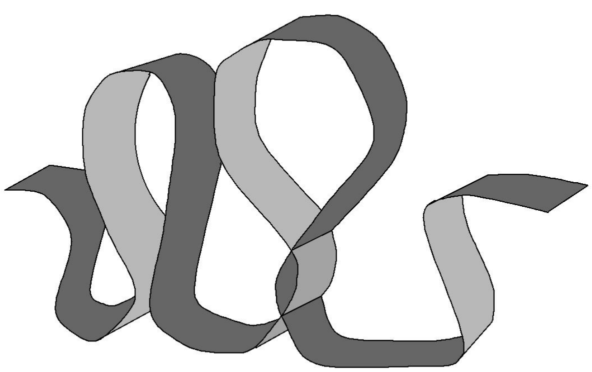

Figure 6. For large expansion (typically more than three) neighbor folds might intersect.

Computation time with our implementation varies from a few seconds for interactive editing (see Figure 3) to about half an hour on a 700 MHz PIII for the scrunchy at the middle of Figure 1 (about 50000 vertices). Better performances could be obtained with a more efficient solver, e.g. based on implicit simulation (see [2, 10]).

Although the purpose of this paper is limited to modeling surface details, we experimented with more fancy expansion textures which affect the shapes globally. In future work, this should allow us to go towards morphogenesis methods for modeling complex shapes. We present here some preliminary results: Figure 10 shows a brain shape obtained by the uniform isotropic expansion of a sphere, and a set of blisters obtained by using a expansion map (shown on the right) with smooth spots of isotropic expansion. Figure 11 shows square surfaces getting curved after applying isotropic expansion on limited parts of their surfaces. The map yielding the first shape is figured at the middle.

6. Conclusion

We have introduced the new paradigm of *expansion textures* which allows the user to specify the aspect of folded surfaces at a higher level of control, i.e. without having to shape precisely the folds as one has to do using displacement maps or direct modeling. The user designs the folded areas, the amount and direction of folding and possible constraints on the surface. To do this, the user uses interactive parameter painting tools or maps (either explicit or procedural), following a workflow much similar to texturing. Although the calculation of equilibrium surface has strong similarities with physical simulation solvers, our approach is far more compatible with the kind of knowledge and wishes of an artist: the users need to know only the kind of shape they want, and not the history of forces that should be applied in reality to get the result (moreover, numerous famous sculptures show very exaggerated folds that are probably non-physical).

For future work we will introduce self-collisions in our implementation to prevent neighboring folds from intersecting. We would also like to study the direct generation of bump maps. Our long-term goal is to experiment with morphogenesis, i.e. the modeling of shapes which are mainly due to the result of growth. Since this implies large orders of magnitude in the amount of expansion, we are especially interested in the definition of procedural expansion textures having fractal properties.

Acknowledgments

We wish to thank Laks Raghupathi who carefully reread this paper. Thanks are also due to Lionel Reveret and Marie-Paule Cani for rereading an early version of the paper.

References

[1] N. I. Badler and W. Becket. Imperfection for realistic image synthesis. *Journal of Visualization and Computer Animation*, 1(1):26–32, August 1990.

[2] D. Baraff and A. P. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54, Orlando, Florida, July 1998. ACM SIGGRAPH / Addison Wesley.

[3] B. G. Becker and N. L. Max. Smooth transitions between bump rendering algorithms. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 183–190, Aug. 1993.

[4] J. F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12(3), pages 286–292, Aug. 1978.

[5] D. E. Breen, D. H. House, and M. J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of SIGGRAPH 94*, pages 365–372, Orlando, Florida, July 1994. ACM SIGGRAPH / ACM Press.

[6] J. Cohen, M. Olano, and D. Manocha. Appearance-preserving simplification. *Proceedings of SIGGRAPH 98*, pages 115–122, July 1998. Held in Orlando, Florida.

[7] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In F. Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 187–196, Aug. 1990.

[8] G. Debunne, M. Desbrun, M.-P. Cani, and A. H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of ACM SIGGRAPH 2001*, pages 31–36. ACM Press / ACM SIGGRAPH, August 2001.

[9] T. D. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 85–94, Orlando, Florida, July 1998. ACM SIGGRAPH.

[10] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of SIGGRAPH 99*, pages 317–324, Los Angeles, California, August 1999. ACM SIGGRAPH.

[11] D. Ebert, K. Musgrave, D. Peachey, K. Perlin, and Worley. *Texturing and Modeling: A Procedural Approach*. Academic Press, Oct. 1994. ISBN 0-12-228760-6.

[12] K. W. Fleischer, D. H. Laidlaw, B. L. Currin, and A. H. Barr. Cellular texture generation. *Computer Graphics*, 29(Annual Conference Series):239–248, 1995.

[13] A. Fournier and D. Fussell. Stochastic modeling in computer graphics. *Computer Graphics (Proceedings of SIGGRAPH 80)*, 14(3):108, July 1980. Held in Seattle, Washington.

[14] D. R. Fowler, H. Meinhardt, and P. Prusinkiewicz. Modeling seashells. In Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 379–388, July 1992.

[15] I. Guskov, W. Sweldens, and P. Schröder. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 325–334, Los Angeles, California, August 1999. ACM SIGGRAPH / Addison Wesley Longman.

[16] P. Hanrahan and P. E. Haeberli. Direct WYSIWYG painting and texturing on 3D shapes. In F. Baskett, editor, *Computer Graphics (SIGGRAPH 90 Proceedings)*, volume 24, pages 215–223, Aug. 1990.

[17] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In J. Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 271–280, July 1989.

[18] F. Neyret. Modeling animating and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):55–70, Jan.–Mar. 1998. ISSN 1077-2626.

[19] J. O'Brien and J. Hodgins. Graphical modeling and animation of brittle fracture. In *SIGGRAPH'99 Conference Proceedings*, pages 137–146. ACM SIGGRAPH, 1999.

[20] K. Perlin. An image synthesizer. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19(3), pages 287–296, July 1985.

[21] K. Perlin and E. M. Hoffert. Hypertexture. In J. Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 253–262, July 1989.

[22] P. Prusinkiewicz. Modelling and visualization of biological structures. *Graphics Interface '93*, pages 128–137, May 1993. Held in Toronto, Ontario, Canada.

[23] P. Prusinkiewicz, M. Hammel, and R. Mech. Visual models of morphogenesis: A guided tour. `http://www.cpsc.ucalgary.ca/Redirect/bmv/vmm-deluxe/`.

[24] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In D. C. Evans and R. J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 151–160, Aug. 1986.

[25] D. Terzopoulos and K. Fleisher. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *SIGGRAPH'88 Conference Proceedings*, pages 269–278, 1988.

[26] D. W. Thompson. *On Growth and Form*. Cambridge University Press, Cambridge, 1917.

[27] G. Turk. Generating textures for arbitrary surfaces using reaction-diffusion. In T. W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 289–298, July 1991.

[28] M. Walter, A. Fournier, and D. Menevaux. Integrating shape and pattern in mammalian models. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 317–326. ACM Press / ACM SIGGRAPH, August 2001.

[29] X. C. Wang, J. Maillot, E. L. Fiume, V. Ng-Thow-Hing, A. Woo, and S. Bakshi. Feature-based displacement mapping. *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 257–268, June 2000.

[30] W. Welch and A. Witkin. Free-form shape design using triangulated surfaces. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, pages 247–256, Orlando, Florida, July 1994. ACM SIGGRAPH / ACM Press.

[31] A. Witkin and M. Kass. Reaction-diffusion textures. In T. W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 299–308, July 1991.

[32] T.-T. Wong, W.-Y. Ng, and P.-A. Heng. A geometry dependent texture generation framework for simulating surface imperfections. In *Eurographics Rendering Workshop 1997*, pages 139–150, St. Etienne, France, June 1997. Eurographics / Springer Wien. ISBN 3-211-83001-4.

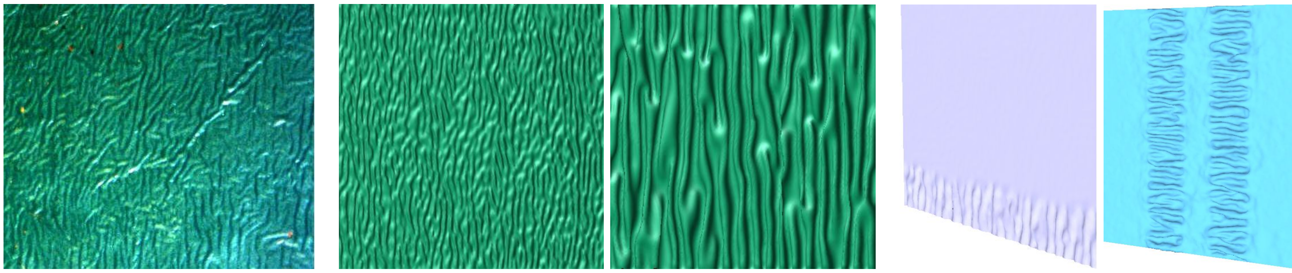

Figure 7. Regular folds (left is a real image of a plastic cover), gathers.

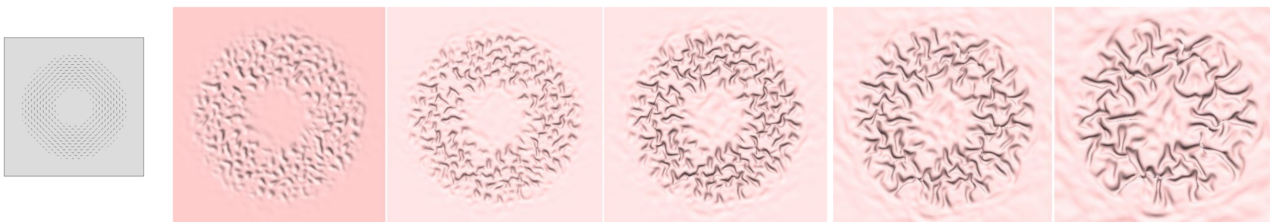

Figure 8. Coat folded in a ring area.

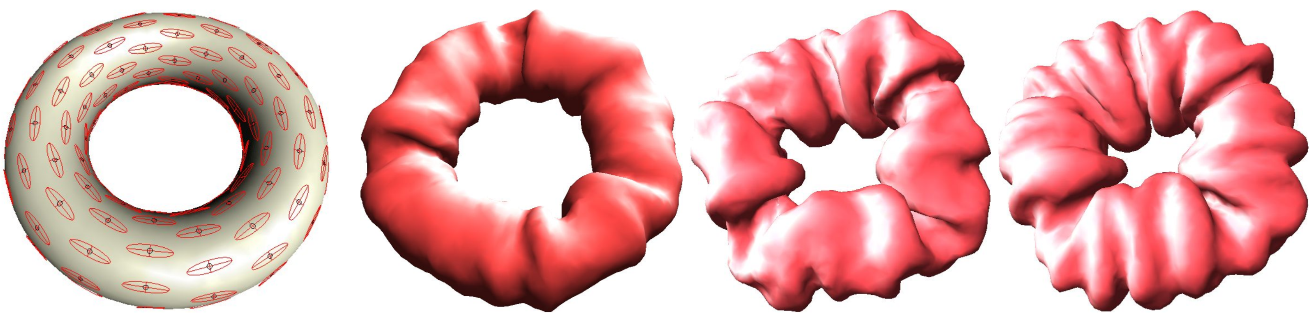

Figure 9. Scrunchy obtained from a torus.

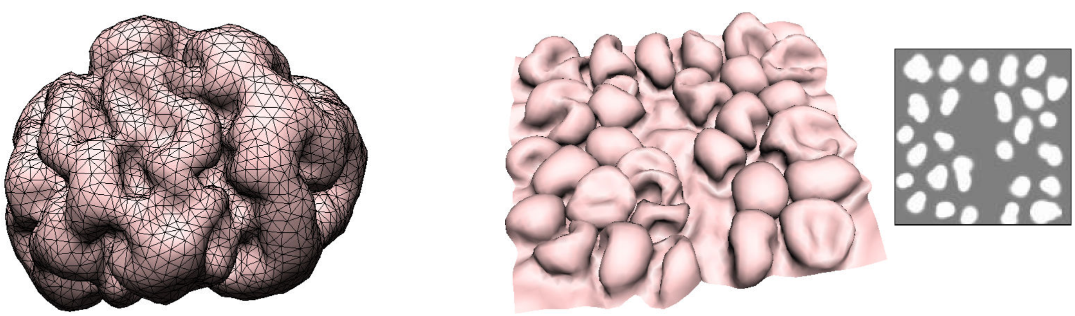

Figure 10. Circumvolution shapes.

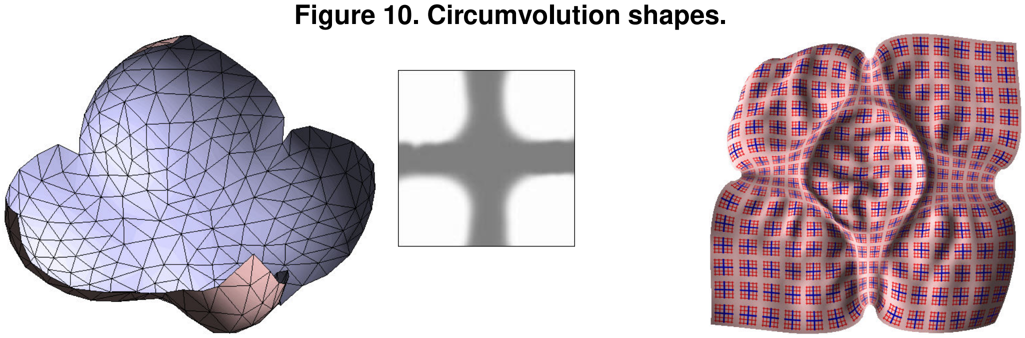

Figure 11. Non-uniformly growing squares.