Semi-interactive morphogenesis
Jean Combaz, Fabrice Neyret

To cite this version:

HAL Id: inria-00537441
https://inria.hal.science/inria-00537441
Submitted on 18 Nov 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Semi-interactive morphogenesis

Jean Combaz
Evasion-GRAVIR / IMAG-INRIA

Fabrice Neyret
Evasion-GRAVIR / IMAG-INRIA
Fabrice.Neyret@imag.fr

Abstract

This paper presents a method to simulate growth phenomena, and its application to the modeling of complex organic shapes (e.g., plants organs) and folded surfaces.

Our main contribution is the interactive and stable resolution of the mechanical problem of growth-induced deformations, based on the minimization of the energy due to the various constraints in the shell. From this, we propose a new modeling approach based on a set of growing tools: The user can apply ‘hot spots’, ‘hot curves’, or paint growing parameters on the surface to grow. Growth can be simulated either simultaneously to the user interaction, or once all parameters have been settled on the surface (which allows the use of textures of parameters and procedural operations). The main parameters are the intensity and anisotropy of growth, as well as their variations over time. Geometric constraints and plasticity can also be considered.

As our results show shapes can fold, bend, and curl as in nature, which deforming tools such as displacement map could not achieve. We demonstrate our tool with an interactive session and a gallery of shapes easily produced.

1. Introduction and Previous Work

Natural shapes are often complex, globally as well as in their details. Modeling them using classical geometric modelers can thus be especially tedious. A number of these natural shapes result from a growth phenomena [22] (e.g., plants, organic shapes) or can be modeled as such (e.g., folded surfaces, geological shapes). Even though growth-induced deformations can be physically simulated similarly to cloth animation or deformable objects [2, 21], real-world parameters are usually unknown. Moreover, artists want to keep enough control so a blind off-line simulation is not desirable.

Various tools have been introduced to model or enrich a shape by deforming it either at global (e.g., Free-Form Deformations [19, 8, 3]) or local scale (e.g., displacement maps [7, 26]). Similarly, several virtual sculpting tools have been proposed either for the overall shaping (e.g., [4, 11, 1]) or to edit and enrich an existing shape (e.g., [20], which inspired Maya Artisan). Although the user interaction is more intuitive with these tools than using a classical geometric modeler, the user still has to explicitly specify where each bulge or pit should be. For shapes that are complex (either globally or locally) this can be very tedious.

We would like to let the user specify the global and local aspect of the shape without having to define every little detail. [5] introduces the expansion texture paradigm (summarized in the next section) to shape folded surfaces: the user specifies expansion (i.e. growth) parameters using either a brush or a texture map. The surface folds under the local constraints caused by the growth. Since the parameters are attached to the surface, even large amplitude of growth can be simulated and still yields natural looking results. Oppositely, surface displacement tools cannot be applied for huge amplitudes: the displacement is done along the normal and cannot curl the shapes. Tools like Maya Artisan (inspired from [20]) or [14] can iterate displacement relying on the new normals. But such a geometric growth quickly degenerates into unnatural and self-intersecting surfaces. This does not occur with expansion textures due to the use of a mechanical model to simulate the deformations. Moreover, a mechanical model reproduces the behavior of natural growing surfaces making folds and bulges under constraints.

The [5] model has similarities with physical models such as the ones used for deformable surfaces (e.g. [21]). However, the mechanical model is tuned to give controllable results (e.g. fold wavelength) and the solving is quasi-static (the purpose is to simulate the shape at equilibrium, not the dynamics).

Our model draws on [5] and extends it to permit huge
deformations at interactive rate (i.e., we are not limited to adding details) and to increase the flexibility. Thus, our tool applies to the modeling of a large variety of shapes (and not simply folds). For that we revisit the mechanical model in order to make it more general, steady and efficient as detailed in section 3.

From this new engine we propose a new modeling approach based on a set of growing tools. These allow both local and global shaping, and both interactive and semi-procedural specification of growth. We describe these tools in section 4.

In section 5, we demonstrate our tool with interactive sessions, as well as samples of typical shapes that are easily produced (e.g., plant organs, and folded shapes like a tablecloth or an unmade bed).

2. The expansion texture formalism

Since we draw on and expand the approach of [5], we share the same formalism. We present here a summary (see the paper for more details).

The growth is modeled by local anisotropic expansions which act on a free form surface discretized by a triangular mesh. The growth does not modify the surface explicitly but a reference state defined by the rest length $l_v$ of each edge $e$, the rest mean curvature $k_e$ of each vertex $v$ and the graph connectivity of the mesh (see figure 1). This virtual reference state (updated by the growth) defines locally the ideal shape the surface should attain. The ‘real’ shape is obtained by a mechanical solver which minimizes the constraints, i.e., the differences between the mesh and the reference state. Note that the surface has to be dynamically remeshed (see [5]).

![Figure 1. The virtual reference state is defined by the rest lengths $l_v$ (a), the rest mean curvatures $k_e$ (b) and the graph connectivity (c) of the mesh. The solver finds the shape (d) that minimizes the difference with the reference state (modified by the growth).](image)

The local expansion is described by an expansion tensor field over the surface. An expansion tensor can be represented by a $2 \times 2$ positive definite symmetric matrix $D$. It can be defined by three parameters: two expansion rates $D_1$ and $D_2$ relative to the eigenvector directions, and an angle $\theta$ which defines the orientation of the eigenvectors basis relative to the local 2D frame. The expansion along a vector $\hat{l}$ is then given by $G_D(\hat{l}) = \sqrt{\hat{l}^T D \hat{l}}$.

The growth parameters the user can tune are: the rate and the anisotropy of the expansion (i.e., $D_1, D_2, \theta$) at every active location; and desired wavelength and regularity $\lambda_1, \lambda_2$ of folds or curls. External constraints can be added by the user, such as static collision with solid objects and partial or total attachment (i.e., 1D, 2D or 3D constraints) of some surface locations. The local growth parameters interactively painted by the user or predefined using a paint program are stored in a map—the expansion texture. A parameterization of the surface is thus required.

3. Our mechanical model and solver

We want to allow huge expansion in order to deal with a large range of morphogenesis situations, and we want to provide more precise tools to the user (see section 4). For this, our model must not depend on any global parameterization (which would be soon overstretched). A very stable model is also required: we rely on an energetic formalism close to thin shells mechanical model. We also want to provide the user with higher level controls (i.e., semi-procedural, or semi-interactive), by allowing him to tune how long a growth will last, according to which activation pattern (e.g., delayed starting, progressive decay...).

3.1. Expansion model

To avoid a global $(u,v)$ parameterization we store the various local data at the vertices or at extra nodes within the faces. For extra nodes we define a radius of influence $r$ and an influence kernel $w(x)$. We define a local 2D frame for each vertex using a local geodesic polar parameterization [27].

For growth data defined at vertices we model the expansion of the reference length $l_v$ of edge $e$ as the average of the expansion at the tips: $G(e) = \frac{1}{2}(GG_{P_1}(l_e) + GG_{P_2}(l_e))$ where $D_{P_1}$ and $D_{P_2}$ are the tensors at the edge tips $P_1$ and $P_2$. For growth data defined at an arbitrary node $N$ we define $D_P = w(||P - N||)D_N$. For growth data defined on an arbitrary curve $C$ (based on control points $\{N_i\}$) we define $D_P = w(d(P, C))D_N$ where $d(P, C)$ is the distance from a point $P$ to the curve $C$.

As stated above and detailed in section 4 we account for the duration of growth in time. The expansion of an edge $e$ between iteration steps corresponding to time $t_1$ and $t_2$ is $G(e) = G(e)^{t_2/t_1}$.

We also take plasticity into account if desired, which avoids accumulating undissipated stress in the surface. This permit smoother shapes in case of huge expansion. When the difference between the reference length $l_e$ and the corresponding real length exceeds a threshold $\alpha_{\text{plast}}$ we operate a relaxation over $\tilde{l}_e$:
We opted for a finite element expression of the stress. We adapt this term to our formulations could be used in the scope of off-line complex growth simulation. The expression we propose below for these two energies is easily to calculate and derive. It produces realistic deformations. However, more accurate (and thus time-consuming) formulations could be used in the scope of off-line complex growth simulation. The pressure energy is not a classical notion here and was introduced by [5] to control the wavelength of the triangles compression into a displacement along the normal. This force succeeds in modulating the fold wavelength and the growth preferred direction. We adapted this term to define a pressure energy:

**Pressure energy**

where

\[ E_{pressure} = \sum_v \tilde{v} \]

\[ \tilde{v} = \frac{1}{V_t} (v - \bar{v}) \]

\[ v \]

is the 3D position of the vertex at a previous sub-step in the optimization solver.

3.3. The solver

To generate a new global equilibrium of the shape after a growth step, we solve the energy minimization using a classical Conjugate Gradient Optimization (see [17] for details). The optimality of the steps provides a more robust convergence than the simple displacement used in [5] (corresponding to a constant step Gradient Optimization). We stop the optimization when the difference between the energies in two consecutive steps or the maximum stress along the surface are below a small threshold.

4. Growing tools

In nature, growth exists in a variety of modalities: in plants, cellular proliferation mostly occurs at apexes (i.e., tip of branches). This causes the 1D nature of branches. At some stage of the leaves development the surface enlarges due to the cellular division along the leaf edge. Geological rifts are another situation showing an expansion occurring along a curve. Animal tissues grow more homogeneously, but often with an important anisotropy: body proportions change from embryo to adult [25]. The distribution of active and passive (expansion-wise) areas on a surface strongly influences the pattern of deformation due to mechanical constraints: an expanding spot can only bulge out of the surface. An expanding linear area crossing the surface might simply enlarge it (e.g., rifts). When the whole
surface is expanding, any heterogeneity or local constraint will result in folds or curling.

In order to provide the user with most flexibility, we account for the same variety of modalities: Our tools can be hot spots (0D), hot curves (1D) or hot surfaces (2D). Formally these three kinds of active regions could all be simply painted in an expansion texture. The difference mostly lies in their specification and control: Spots and curves control points are located precisely on the mesh by relying on their barycentric coordinates within faces (and updating this data during remeshings). Hot surface data is defined at the mesh vertices.

4.1. Hot spots (0D tools)

A hot spot is defined by the growth parameters \((D_1, D_2, \ldots)\), a radius of influence \(r\) and an activation curve \(I(t)\) (growth intensity in time, or by default a constant and a duration). Since a hot spot generates a bulge, a long activation results in the growth of a branch.

We implemented various spot tools such as rotating the frame over time, and splitting a spot into several spots within the radius of influence. As shown in the results section, this results in branching. It would be easy to connect this mechanism to a procedural event generator such as L-systems [18] or [10]. We tried simple ones as illustrated in the result section.

4.2. Hot curves (1D tools)

A hot curve can share common growth parameters, whose orientation can follow the local frame along the curve. We denote \(D_T\) the expansion along the curve (i.e., tangential) and \(D_N\) the expansion in the orthogonal direction (i.e., normal). To model an heterogeneous growth, growth parameters can be painted and attached to the control points (and thus to the surface), or specified by a 1D texture \(I(u)\) (relying on a curvilinear parameterization \(u\)). The last case can typically be used for controlling the activation of growth in time: A user-defined 1D texture defines the growth intensity along a normalized range of time, and the local time offset and local time pace are interpolated along the curve.

The curve is attached to the surface and thus is distorted together with it (by the effect of its activity and that of other growing tools). For instance, this allows the user to define a very short curve across an apex (i.e., a hot spot) and to let it enlarge before activating it. This can be used to generate a leaf from a stem. It is easy to create automatically such a small curve from geometric parameters (e.g., the apex normal – which corresponds to the stem axis –, or the vertical), so a procedural tool can generate and trigger hot curves as illustrated in the results.

4.3. Hot surfaces (2D tools)

The user interface for the 2D tool draws on and extends the [5] one: growth parameters are defined either by interactive painting (parameters are sampled at mesh vertices, orientation is given by the mouse direction) or by a 2D texture (a temporary \((u,v)\) parameterization – e.g., a projection – is then defined on the targeted area). From this, we added several features. In the interactive case we store a time offset \(t_0\) and a time pace \(dt\) at each painted vertex: this allows the user to control how long the growth will apply (i.e., while \((t - t_0)dt < 1\)). This also permits to spread huge growth along time, thus allowing stable huge deformations. We also implemented various painting tools in order to apply growth ‘brushes’ and interpolation. For the latter, we rely on adapted Laplace interpolation [12]: the user paints values (spots or curves) which are spread by solving a diffusion equation (we used an implicit solver for efficiency). Interpolating tensors is not as simple as for scalars. Similarly to what [24] does for vectors, we first interpolate the tensor coefficients. Then we rebuild an orthonormal frame and we reconstruct the interpolated tensor by scaling it with the separately interpolated eigenvalues.

5. Results

Our results are presented on figure 2. Animated sequences\(^1\) are framed.

- (a): Using a hot spot (the initial shape is a sphere). Isotropic growth; anisotropic growth (folds appear in the most expanded side); with twisted frame; branchings by splitting the spot.
- (b): Using a hot curve (the initial shape is a triangle). Mushrooms (the initial hot curve is a circle at the stem tip): without normalization; with the normalization (see Section 4.2). Leaves (the initial hot curve is a half-circle crossing the apex at the stem tip; its control points are figured on the first two images).

\(^1\)Videos and more data are available on our web site http://www-evasion.imag.fr/Publications/2006/CN06/ and in [6].
• (c): Using a hot surface (and isotropic growth). Pumpkin: rigid lines (in grey) were painted on a sphere before the homogeneous growth. Leaves: growth is painted only on the edge, using various tunings (frequency, gradient). Flowers: painting growth only on the corners grows and curls them into petals.
• (d): Modeling an unmade bed. A physical cloth simulation would require knowing the initial state, the material parameters, and the history of forces. Using our model, the artist simply ‘paints’ on the areas where to add ‘extra matter’ (resulting in folds) much like a sculptor would. The frequency is tuned by the user and the folds are oriented by the mouse direction.
• (e): Table cloth. Starting with a cylinder, the user paints a totally tangential anisotropic expansion on the bottom, and a zero-expansion circle on the top. Then he interpolates the growth parameters in between. Note that the user can exaggerate the folds (as for the drapes in ancient paintings) if desired.
• (f): A cabbage leaf modeled using an expansion texture (much like in [5]).
• (g): Procedural growth. Apexes are split randomly. Note that our model produces a continuous growth, and that mechanical constraints move branches apart naturally.

6. Conclusion and Future Work

We have presented a new modeling paradigm based on the control of surface growth. The engine consists on a new efficient and stable mechanical model and solver able to simulate growth-induced deformations. Compared to physical simulations we are able to provide the user with fine control of the shape. Compared to traditional modeling tools and to sculpting tools we do not require the user to
explicitly define every little detail. However, by tuning the radius of influence the user can indeed edit details whenever he wants. Most of the result we show could not have been modeled easily with any other modeling tool or physical simulation.

Our approach is able to model either global shapes or complex surface details, relying on interactive or semi-interactive operations. However, the user may sometime want an even higher level of control. Our model could be connected to procedural tools such as L-systems [18] to generate complex structures. Similarly, expansion texture could be procedurally evaluated or simulated using noise [16] or reaction-diffusion [23]. Finally, it would be interesting to add to our model a scheme for the semi-procedural synthesis of surface colors and textures.

Acknowledgments

Thanks are due to Hector Briceno for rereading the paper.

References