Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 281 (2011) 159–173

www.elsevier.com/locate/entcs

A User-editable C^1-Continuous 2.5D Space Deformation Method For 3D Models

Elisa de Cássia Silva Rodrigues[1,2] Anamaria Gomide[3]
Jorge Stolfi[4]

Institute of Computing
State University of Campinas (UNICAMP)
Campinas-SP, Brazil

Abstract

Shape deformation methods are important in such fields as geometric modeling and computer animation. In biology, modeling of shape, growth, movement and pathologies of living microscopic organisms or cells require smooth deformations, which are essentially 2D with little change in depth. In this paper, we present a 2.5D space deformation method. The 3D model is modified by deforming an enclosing control grid of prisms. Spline interpolation is used to satisfy the smoothness requirement. We implemented this method in an editor which makes it possible to define and modify the deformation with the mouse in a user-friendly way. The experimental results show that the method is simple and effective.

Keywords: Space deformation, biological modeling, 3D spline, C^1 continuity, deformation editing, deformation modeling.

1 Introduction

Many applications require arbitrarily complex but smooth user-editable deformations. An important example, that we use to illustrate and validate our work, is the modeling of shape, growth, movement and pathologies of living microscopic organisms or cells. Our goal is to define mathematical and software tools to describe such deformations in a user-friendly way.

Generally, at any given stage of its life, each specie of organism has a relatively well-defined morphology that makes it possible to model its basic form. However, many biological structures have elastic or gelatinous consistency and may undergo

[1] Supported by CAPES, FAPESP and CNPq.

[2] Email: elisa.rodrigues@students.ic.unicamp.br

[3] Email: anamaria@ic.unicamp.br

[4] Email: stolfi@ic.unicamp.br

doi:10.1016/j.entcs.2011.11.032

considerable deformations due to motion or even pathologies. A realistic modeling of these deformations is crucial in tasks such as the generation of synthetic images and videos for study and teaching, and automatic recognition of organisms in clinical analysis.

We assume that the basic shape of the target organism is given as a dense triangular mesh with tens of thousands of triangles. See Figure 1.

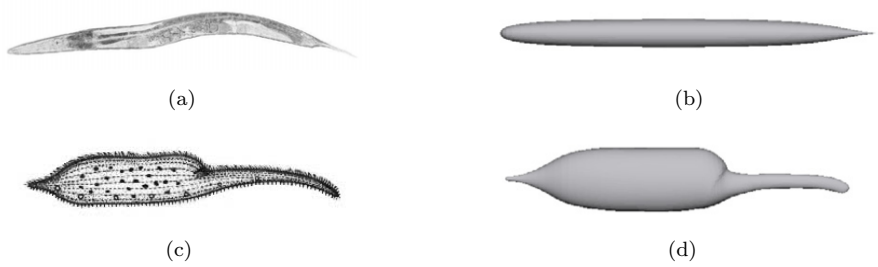


Fig. 1. Images of the morphology of the *Caenorhabditis elegans* (a) and of the *Dileptus anser* (c). And the 3D models of their basic shapes (b) and (d), respectively.

Having the model, we need a method to deform it. We use here the space deformation approach which modifies the model by deforming the space it is embedded in. Space deformation methods are highly interactive and intuitive for user-editable deformations. Generally, the space deformation methods embed the target model in a coarser 3D mesh, called *control grid* [5,18,29,34]. The embedded shape can be changed by manipulating the control grid to obtain a *deformed grid*, and, by using spline interpolation techniques, extend the control grid deformation to the whole space inside it.

Many existing space deformation techniques are continuous (C^0) but not smooth (C^1), i.e., they often introduce corners or creases in the initial model. See Figure 2. The C^1 techniques are very hard to edit because they have a very large number of free parameters.

Fig. 2. A comparison between space deformations with C^0 (left) and C^1 (right) continuity.

In this paper, we focus on deformation modeling techniques that are C^1 and allow convenient editing of the control grid by the user. To achieve this goal, we propose a special subset of space deformations that can be described by a 2D deformation in the x, y directions with a position-dependent 1D stretching map. Therefore we may call it a "2.5D" space deformation method. In general, although the models for this application are 3D, the deformations are essentially 2D with little change in depth, especially because the third dimension can not be easily perceived through a microscope.

Specifically, our coarse grid consists of a single layer of triangular prisms with

curved top and bottom faces. Our editor ensures that the space deformation implied by the deformed grid is C^1-smooth.

We implemented the method in an editor which makes it possible to define and modify the deformation with the mouse in a user-friendly way. The choice of prismatic cells reduces the number of points of the control grid, making the deformation easier to edit and visualize. The experimental results show that the proposed method is simple and effective.

The paper is organized as follows. We will discuss the related works in Section 2 and introduce concepts on spline surfaces, C^0 and C^1 continuity and local control in Section 3. In Section 4, we present the methodology of this work. Experimental results are presented in Section 5. In Section 6 we conclude the paper with some discussions and future works.

2 Related Work

Shape deformation editors can be classified as surface methods [7,33], where the user manipulates the object mesh directly; or space deformation methods [14], where the user modifies the object mesh indirectly by deforming the space it is embedded in. Gain and Bechmann [14] describe some space deformation methods and classify them according to the dimension of the *control objects* used to define the deformation: control points [1,2,3,6,21], curves [4,26], surfaces [9,31,32] or three-dimensional grids [5,8,17,18,23,27,29,34].

Space deformation editors that use 3D control grids seem better suited to our application. For one thing, the deformed control grid provides an immediate intuitive understanding of the general nature of the deformation, and of the scope of each control parameter. In particular, it makes it easier to notice and avoid singularities in the deformation (places where the deformation is not injective, meaning that space is being folded over itself). Compared to surface editors, 3D grid methods also tend to have fewer adjustable parameters, and are independent of the model's resolution. For these reasons, we have opted for a 3D control method in this work.

Most 3D space deformation methods described in the literature use either hexahedral [23,29] or tetrahedral [5,18,34] control meshes. These methods can be used with spline interpolation techniques to maintain the continuity of the inner 3D model [5]. However, the requirement of C^1 continuity makes this approach very hard to edit. For example, with a tetrahedral mesh, the degree of the interpolating spline must be at least 5 to allow localized editing of the grid. Therefore, to specify each tetrahedron in the mesh the user must specify the position of 56 control points [5]. With an hexahedral mesh one can have C^1 splines of degree 3, however each cell requires 64 control points [14].

With so many control points, it becomes difficult to identify and select the ones that must be edited to achieve a desired effect. Furthermore, the editing software must automatically move many additional control points in order to satisfy the C^1 continuity constraints, increasing the user's confusion. Complex user interfaces, with high-level abstractions, have been developed to address this problem [5,14],

but they do not solve it completely.

Some space deformation methods attempt to get around this problem by the use non-spline interpolating functions, which are determined by the control mesh vertices and/or faces only. One early approach was the *mean value coordinates* of [12,13,20,24]; these are infinitely smooth almost everywhere, but are not C^1 at the vertices of the mesh. The *harmonic coordinates* of [19] are smooth everywhere, but do not have closed formulas, and are expensive to compute numerically. The most recent approach in this direction is based on *Green coordinates* [25]. These interpolants have a closed form, but are still expensive to compute. They also yield quasi-conformal deformations, which partially preserve the object's shape; which is an advantage in some cases, but a drawback in others. In any case, since these methods have fewer control points than spline methods, they require control meshes with many more cells. We believe that our 2.5D method provides a reasonable balance between control grid size, naturalness of editing, and computation speed.

Finally we may cite deformation techniques based on physical simulations, such as [28]. We did not consider this approach since they would require accurate physical models of the interior of the micro-organisms, including the elasticity and viscosity of tissues – information that is unlikely to be available, even for the best-studied organisms.

3 Basic Concepts

This section introduces concepts on spline surfaces, C^0 and C^1 continuity and local control used in the implementation of the space deformation method presented in the paper.

3.1 Splines

Splines are widely used to model functions of unlimited complexity with local control [14]. Typically, a spline is defined as a piecewise-defined function whose pieces are polynomials [10].

More precisely, we define a *spline function on a mesh T in $\mathbb{R}^n$* as $f : \mathbb{R}^n \to \mathbb{R}$ such that the restriction f^u of f to each part u of the mesh is a polynomial on the coordinates of the argument. See Figure 3.

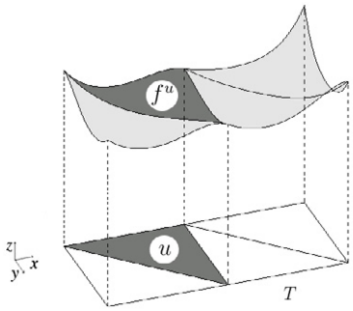


Fig. 3. A spline function from $\mathbb{R}^2$ to $\mathbb{R}$.

If $n = 2$ and the cells of the mesh are triangles, then each polynomial f^u can be conveniently expressed as a linear combination of the *Bernstein-Bézier polynomials* of the corresponding triangle u. Let p be a point of $\mathbb{R}^2$. Let β_0, β_1 and β_2 be barycentric coordinates of p relative to the vertices of u. Let $d \in \mathbb{N}$ be a degree, and i, j and k be non-negative integers such that $i+j+k = d$. Then the two-dimensional Bernstein-Bézier polynomial of degree d with indices i, j and k is defined as

$$B_{ijk}(p) = \frac{d!}{i!j!k!}\beta_0^i\beta_1^j\beta_2^k. \tag{1}$$

Note that B_{ijk} is a homogeneous polynomial function with total degree d of the barycentric coordinates β_0, β_1 and β_2. The set of all polynomials B_{ijk} with $i+j+k = d$ is a basis for the polynomials of degree d defined on $\mathbb{R}^2$. That is, every polynomial g of degree d defined on $\mathbb{R}^2$ can be written uniquely as

$$g(p) = \sum_{i+j+k=d} c_{ijk}B_{ijk}(p) \tag{2}$$

for all $p \in \mathbb{R}^2$ where the coefficients c_{ijk} depend on g and u.

The numbers c_{ijk} are called *Bézier coefficients of* g (*relative to* u); there are $(d + 1)(d + 2)/2$ of them. Each coefficient c_{ijk} can be associated to a *nominal position* u_{ijk} in the triangle u, whose barycentric coordinates are, by definition, $(i/d, j/d, k/d)$ relative to u. See Figure 4.

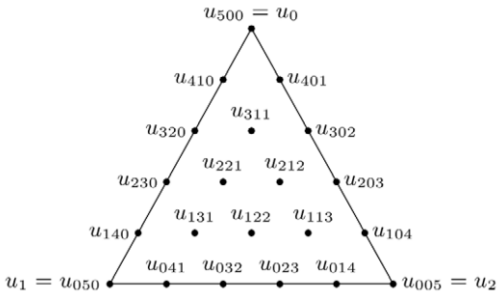


Fig. 4. Nominal positions u_{ijk} of the Bézier coefficients c_{ijk} for a piece f^u of degree 5.

3.2 Using Splines to Model Space Deformations

A deformation of the space $\mathbb{R}^n$ can be defined as a function $\phi : \mathbb{R}^n \to \mathbb{R}^n$. A convenient way of modeling such functions if to let each coordinate of $\phi(x)$ be a spline function $\phi_r(x)$, with all these splines being of the same degree and defined on the same mesh T. The function ϕ deforms T, the *domain grid*, into a new mesh $\phi(T)$ with curved edges, the *deformed grid*. See Figure 5.

The Bernstein-Bézier polynomial representation can be used to describe the deformation ϕ. Let u be a triangle of T and ϕ^u be the part of ϕ with domain u. For each coordinate r (0 for x or 1 for y), the Bézier coefficient $c^u_{ijk;r}$ of ϕ^u_r can be viewed as

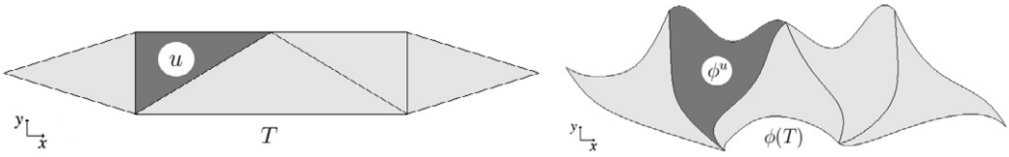


Fig. 5. A deformation of $\mathbb{R}^2$ of the domain grid T (left) in the deformed grid $\phi(T)$ (right).

coordinate r of a point q^u_{ijk}, the *Bézier control point* of ϕ^u with indices i, j and k. The function ϕ can be modified by moving the points q^u_{ijk}. See Figure 6.

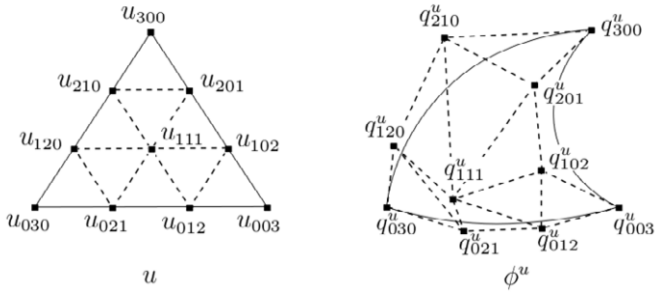


Fig. 6. Bézier control points q^u_{ijk} (right) of a degree 3 patch ϕ^u from $\mathbb{R}^2 \to \mathbb{R}^2$ and their nominal positions u_{ijk} on the domain triangle u (left). The curved triangle on the right is the image of u under the deformation ϕ^u.

Note that the control points q^u_{ijk} are distinct from their nominal positions u_{ijk}. They are also distinct from the images $\phi^u(u_{ijk})$ of the nominal positions, except at the corners, that is, $\phi^u(u_{d00}) = q^u_{d00}$, $\phi^u(u_{0d0}) = q^u_{0d0}$ and $\phi^u(u_{00d}) = q^u_{00d}$.

3.3 Ensuring C^0 Continuity

A spline function has C^0 continuity when there are no discontinuities across cell boundaries. For splines defined on a triangulation the C^0 condition can be easily expressed in terms of the Bézier coefficients. More precisely, let $f : \mathbb{R}^2 \to \mathbb{R}$ be a spline function defined on a triangulation T. Let u and v be two adjacent triangles of T with Bézier coefficients c^u_{ijk} and $c^v_{i'j'k'}$. The condition for f to be continuous across the common edge of u and v is that $c^u_{ijk} = c^v_{i'j'k'}$ for all i, j, k, i', j', k' such that the nominal positions coincide, that is, such that $u_{ijk} = v_{i'j'k'}$.

The same criterion determines when a deformation $\phi : \mathbb{R}^2 \to \mathbb{R}^2$ is C^0-continuous across that edge. Namely, we must have $q^u_{ijk} = q^v_{i'j'k'}$ whenever $u_{ijk} = v_{i'j'k'}$. See Figure 7.

3.4 Ensuring C^1 Continuity

A spline function is smooth when it has at least C^1 continuity at the meeting of its parts, i.e. given adjacent triangles u and v, the first derivatives of the corresponding polynomials f^u and f^v with respect to the domain coordinates are equal at the boundary between u and v. Theorem 3.1 [22] expresses this condition in terms of the Bézier coefficients of f^u and f^v.

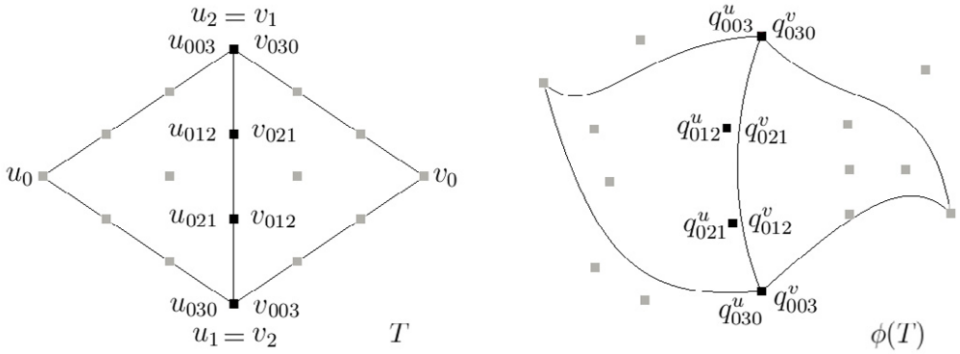


Fig. 7. Bézier control points (right) of a spline ϕ of degree 3 from $\mathbb{R}^2 \to \mathbb{R}^2$ which satisfy C^0 continuity constraints, and their nominal positions (left).

Theorem 3.1 *Let u and v be two adjacent triangles with vertices u_0, u_1, u_2 and v_0, v_1, v_2, respectively, where $u_1 = v_1$ and $u_2 = v_2$ belong to the shared edge e. Let f^u and f^v be polynomials of degree d with Bézier coefficients c^u_{ijk} and c^v_{ijk} relative to u and v, respectively, i.e.*

$$f^u(p) = \sum_{i+j+k=d} c^u_{ijk} B^u_{ijk}(p)$$
$$f^v(p) = \sum_{i+j+k=d} c^v_{ijk} B^v_{ijk}(p). \tag{3}$$

Then, f^u and f^v and all their partial derivatives up to order r are joined smoothly at the edge e, if and only if

$$c^v_{ijk} = \sum_{i'+j'+k'=i} c^u_{i',j+j',k+k'} B^u_{i'j'k'}(v_0) \tag{4}$$

for all $i = 0, ..., r$ and all j, k such that $j + k = d - i$.

Corollary 3.2 *In particular, the polynomials f^u and f^v of the Theorem 3.1 are C^1-continuous along e if and only if*

$$c^v_{0jk} = c^u_{0jk} \tag{5}$$

for all j, k such that $j + k = d$ and

$$c^v_{1jk} = \beta_0 c^u_{1,j,k} + \beta_1 c^u_{0,j+1,k} + \beta_2 c^u_{0,j,k+1} \tag{6}$$

for all j, k such that $j + k = d - 1$, where β_0, β_1 and β_2 are barycentric coordinates of v_0 relative to u_0, u_1 and u_2.

A C^1-continuous spline deformation $\phi(T) : \mathbb{R}^2 \to \mathbb{R}^2$ with control points q^u_{ijk} and q^v_{ijk} follows analogous conditions to those given in (5) and (6). Namely, let u and v be two triangles of T sharing vertices $u_1 = v_1$ and $u_2 = v_2$. The deformation

ϕ is continuous across the shared edge if and only if

$$q^v_{0jk} = q^u_{0jk} \tag{7}$$

$$q^v_{1jk} = \beta_0 q^u_{1,j,k} + \beta_1 q^u_{0,j+1,k} + \beta_2 q^u_{0,j,k+1} \tag{8}$$

where i, j, k, β_0, β_1 and β_2 are as above.

Note that the barycentric coordinates of v_{1jk} relative to $u_{1,j,k}$, $u_{0,j+1,k}$ and $u_{0,j,k+1}$ are the same as those of v_0 relative to u_0, u_1 and u_2. Namely, these four nominal positions form a quadrilateral in the domain of ϕ that is similar to the quadrilateral v_0, u_0, u_1, u_2. Equation (8) then says that the corresponding quadrilateral formed by the control points $q^v_{1jk}, q^u_{1jk}, q^u_{0,j+1,k}, q^u_{0,j,k+1}$ must be an affine image of that domain quadrilateral. See Figure 8.

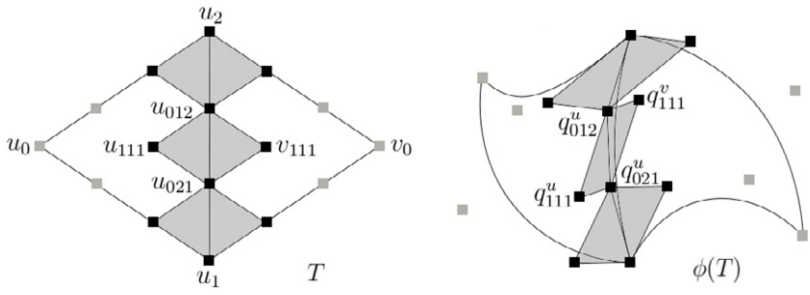


Fig. 8. Bézier control points (right) of a spline ϕ of degree 3 from $\mathbb{R}^2 \to \mathbb{R}^2$ which satisfy C^1 continuity constraints, and their nominal positions (left).

Theorem 3.1 and its consequences also hold with permuted indices when the two domain triangles share other edges (e. g. when $u_0 = v_1$ and $u_1 = v_2$).
We call (8) the *quadrilateral condition* on those four control points. We say that points q^v_{1jk} and q^u_{1jk} are the *extreme* members of the condition, and $q^u_{0,j+1,k}$ and $q^u_{0,j,k+1}$ are *shared* members.

3.5 Local Control

The main advantage of splines over other function approximation methods is that they allow local control: if we change only one control point, the spline changes only within the corresponding domain triangle and perhaps a few other triangles surrounding it.

However, to maintain C^0 and C^1 continuity we often have to change two or more control points at the same time, so as to preserve (7) and (8). If the degree d is too low, these constraints are interconnected in such a way that the changes propagate from triangle to triangle all over the domain grid, so that local control is not possible. In particular, for triangle grids in $\mathbb{R}^2$, one can easily obtain local control and C^1 continuity with splines of degree $d \geq 5$. For $d \leq 4$ these features cannot be obtained simultaneously [22].

Therefore, for the application we consider here (C^1 deformations on a triangular mesh) we use polynomial splines of degree at least 5, which means that each triangle has 21 control points. See Figure 9.

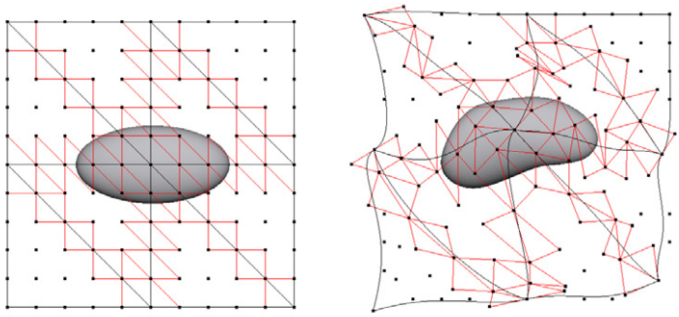


Fig. 9. The domain grid for a spline deformation of degree 5 (left), and the deformed grid (right), showing the control points and the quadrilateral conditions.

4 Methodology

This section presents our 2.5D space deformation method. We describe it in the context of a typical interactive editing session.

4.1 Constructing the Domain Grid and the 3D Model

First, the program loads the 3D model M to be deformed from a given triangular mesh file. Then the user chooses a degree $d \geq 5$ and defines a domain grid P in $\mathbb{R}^3$ that surrounds M. The grid consists of a collection of prisms in $\mathbb{R}^3$ whose projection on the xy plane is a triangulation T of $\mathbb{R}^2$, and whose projection on the z axis is an interval $[a, b]$ adjustable by the user. See Figure 10.

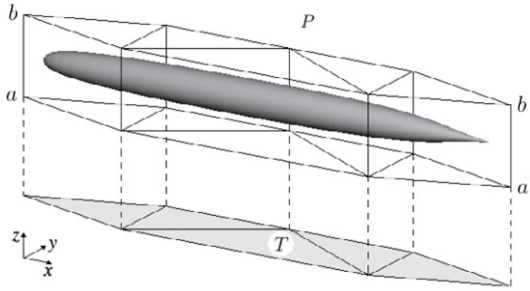


Fig. 10. A 3D biological model M surrounded by a 3D domain grid P, and the corresponding 2D domain grid T.

4.2 Computing Barycentric Coordinates

Since the user is satisfied with the domain grid P, the program finds the prism U of P that contains each vertex p of the model, and computes the barycentric coordinates β_0, β_1 and β_2 of p with respect to the triangle u of the triangulation T that corresponds to the prism U. It also computes the vertical position of p relative to the two triangular faces u^0 and u^1 of U, namely the two numbers $\alpha_0 = (b - z)/(b - a)$ and $\alpha_1 = (z - a)/(b - a)$. See Figure 11.

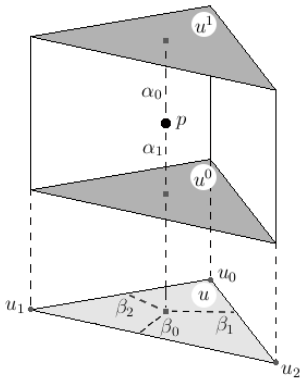


Fig. 11. The barycentric coordinates α_0, α_1, β_0, β_1 and β_2 of a 3D model point p relative to the prism U.

4.3 The Space Deformation

The deformed grid $\psi(P)$ consists of another collection of prisms with vertical walls, whose top and bottom faces are curved triangles. These curved triangles of chosen degree d constitute two spline surfaces, that are the top and bottom surfaces of the deformed grid. Both surfaces are based on the same domain grid T on $\mathbb{R}^2$. See Figure 12.

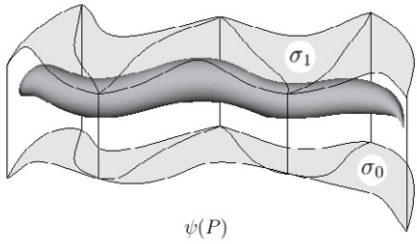


Fig. 12. A deformed 3D biological model $\psi(M)$ surrounded by a deformed grid $\psi(P)$.

The space deformation ψ is a function from the interior of the domain grid P to the interior of the deformed grid $\psi(P)$. The deformed model $\psi(M)$ is another dense triangular mesh with the same topology as M obtained by mapping every vertex of M through the function ψ.

Since the walls of ψ are restricted to be vertical, ψ can be separated into a 2D deformation $\phi : T \to \mathbb{R}^2$ and two spline functions $\sigma_0 : T \to \mathbb{R}$ and $\sigma_1 : T \to \mathbb{R}$. Namely,

$$\psi(p) = (\psi(p).x; \psi(p).y; \psi(p).z) \tag{9}$$

where $p = (x, y, z) \in \mathbb{R}^3$ and

$$\begin{aligned}
\psi(x, y, z).x &= \phi(x, y).x \\
\psi(x, y, z).y &= \phi(x, y).y \\
\psi(x, y, z).z &= \alpha_0(x, y)\sigma_0(x, y) + \alpha_1(x, y)\sigma_1(x, y).
\end{aligned}$$

Note that for each position p inside P, the z coordinates of the point $\psi(p)$ range between $\sigma_0(x, y)$ and $\sigma_1(x, y)$.

4.4 The Bézier Control Points

The splines σ_0, σ_1 and ϕ are defined by their Bézier control points. The user can modify the deformation by moving each control point with the mouse. In this phase the user can switch back and forth between two editing modes: *xy-mode* and *z-mode*.

In the xy-mode the user sees a top view of the deformed grid (that is, a view of the deformed triangulation $\phi(T)$) and can only modify the x and y coordinates of each control point q^u_{ijk} of ϕ as described in Section 3.2. Each triangle has $(d+1)(d+2)/2$ control points; except that two or more control points, with the same nominal positions (along shared edges), are presented and edited as a single point. See Figure 13.

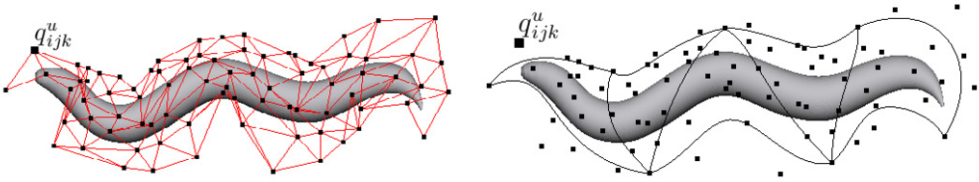


Fig. 13. The xy-mode view of a deformation, showing the control points of the spline ϕ and their adjacency relations in the domain grid (red lines, left), and the deformed triangulation $\phi(T)$ (black lines, right).

When editing in the z-mode the user sees an oblique view of the deformed grid $\psi(P)$ and can edit the coefficients $c^u_{ijk;r}$ of the two spline functions σ_0 and σ_1. For each triangle $u \in T$ and each set of indices i, j, k with $i + j + k = d$ there are two coefficients: $c^u_{0;ijk}$ for the bottom surface and $c^u_{1;ijk}$ for the top surface. Therefore, there are $2(d+1)(d+2)/2 = (d+1)(d+2)$ control points for each prism of P but only their z coordinates can be changed by the user. Each of the two splines σ_0 and σ_1 is edited independently. See Figure 14.

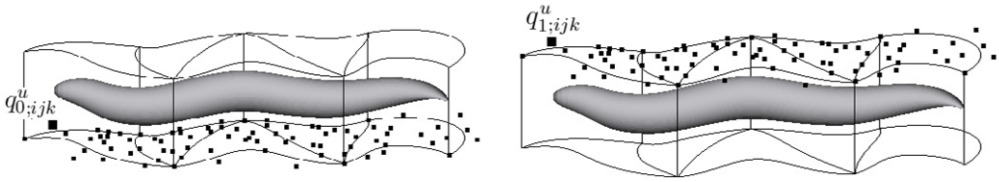


Fig. 14. The z mode view of a deformed grid P, showing the control points of the splines σ_0 (left) and σ_1 (right).

4.5 Enforcing Continuity

As observed in Section 4.4 the C^0 continuity condition is enforced by giving the user only one control point for all Bézier control points whose nominal positions

coincide, namely the control points whose nominal positions lie on edges or vertices of T shared by two or more triangles.

As explained in Section 3.4, the C^1 continuity condition is expressed by a number of quadrilateral conditions (which can be optionally displayed; see Figure 9). To enforce C^1 continuity, whenever the user modifies a control point the program determines one or more additional control points that must be modified in order to preserve the quadrilateral conditions, and automatically applies to them the necessary adjustments. This computation is carried out independently for each spline; in xy-mode it is repeated for both splines, while in z-mode it affects only the spline σ_0 or σ_1 that is being edited at the time.

When the user modifies a control point q, the program finds all quadrilateral conditions that involve q; and then selects, among the other control points that enter into these conditions, a small subset that can be adjusted so as to preserve the conditions. If any of those points enter into additional conditions, the process is iterated as needed. When $d \geq 5$, this subset will include points from the triangle that owns q and only a few adjacent triangles. At the same time, the program computes the displacements for those points that preserve all their quadrilateral conditions. See Figure 15.

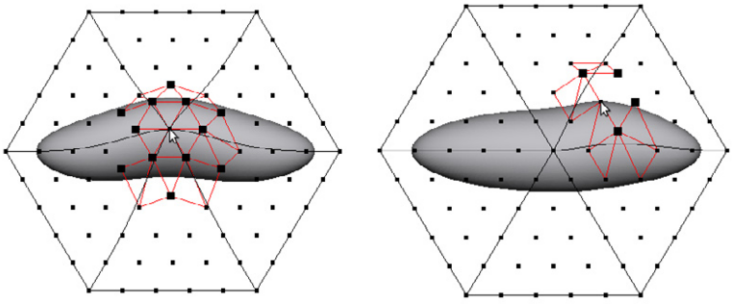


Fig. 15. Editing a control point (arrow), showing the control points (black squares) that are automatically adjusted to maintain the C^1 continuity conditions (red lines).

5 Experiments

To test the suitability of our formulation we used our editor to reproduce deformations of some microorganisms observed in actual microscope images.

We used two organisms in the experiments, the worm *Caenorhabditis elegans* and the protozoan *Dileptus anser*. For each organism we built a mesh model of its 'rest' (undeformed) shape using the Blender 3D editor [30]. See Figure 1. We used our editor to choose an appropriate control grid for each model. See Figure 16. Actual microscope images of these organisms, deformed in various ways, were obtained from the Internet. See Figure 17 (left). We used our editor to deform the model until it matched the actual images. Some results are shown in Figure 17 (right).

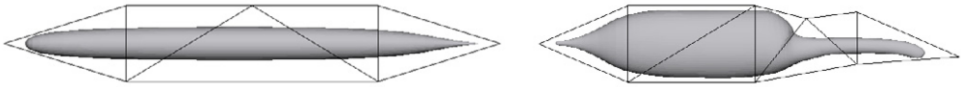
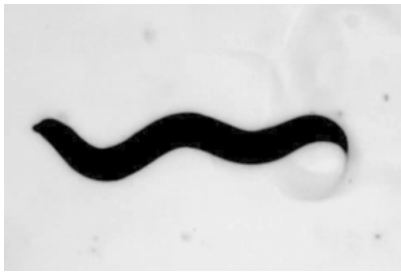
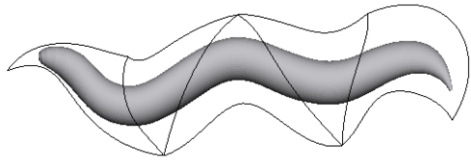


Fig. 16. 2D view of the domain grids and of the undeformed 3D models of the *Caenorhabditis elegans* (left) and of the *Dileptus anser* (right) used in our tests.

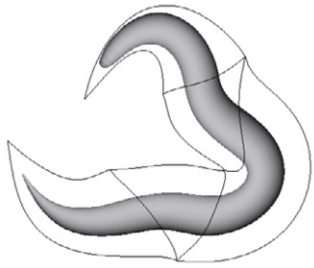


(a) *Caenorhabditis elegans* [16].



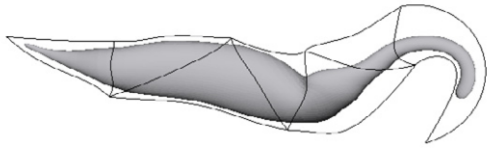
(b) Deformed model.

(c) *Caenorhabditis elegans* [15].



(d) Deformed model.

(e) *Dileptus anser* [11].



(f) Deformed model.

Fig. 17. Images of the test organisms (left) and 2D view of deformed models (right).

6 Conclusions and Future Work

We described an efficient 2.5D space deformation method for 3D models of micro-organisms, based on a control grid of prisms and splines of arbitrary degree. The method allows convenient editing of the deformation while preserving the C^1 continuity of the surface during deformation. The experimental results demonstrated that is possible to simulate various kinds of deformations that can occur in chosen biological structures.

In future works, we will focus in control mesh editing and deformation techniques to facilitate the acquisition the most common coarse deformations. One idea to extend the method to provide multiscale edition, another is to implement a space deformation method based on curves where each point of the curve represents one group of control points. Another direction for future research is the incorporation of volume-preserving constraints and avoidance of self-intersection.

Acknowledgement

This work is supported by Brazilian government grants CAPES 01 P-04388-2010, FAPESP 2007/52015-01 and CNPq 301016/92-5.

References

[1] Angelidis, A., M.-P. Cani, G. Wyvill and S. King, *Swirling-sweepers: Constant-volume modeling*, Graph. Models **68** (2006), pp. 324–332.

[2] Angelidis, A., G. Wyvill and M.-P. Cani, *Sweepers: Swept user-defined tools for modeling by deformation*, in: *SMI '04: Proceedings of the Shape Modeling International 2004* (2004), pp. 63–73.

[3] Aubert, F. and D. Bechmann, *Volume-preserving space deformation*, Comput. Graph. **21** (1997), pp. 625–639.

[4] Barr, A. H., *Global and local deformations of solid primitives*, SIGGRAPH Comput. Graph. **18** (1984), pp. 21–30.

[5] Bechmann, D., Y. Bertrand and S. Thery, *Continuous free form deformation*, in: *COMPUGRAPHICS '96: Proceedings of the fifth international conference on computational graphics and visualization techniques on Visualization and graphics on the World Wide Web* (1997), pp. 1715–1725.

[6] Botsch, M. and L. Kobbelt, *Real-time shape editing using radial basis functions*, in: *Computer Graphics Forum*, 2005, pp. 611–621.

[7] Botsch, M. and O. Sorkine, *On linear variational surface deformation methods*, IEEE Transactions on Visualization and Computer Graphics **14** (2008), pp. 213–230.

[8] Coquillart, S., *Extended free-form deformation: a sculpturing tool for 3d geometric modeling*, in: *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (1990), pp. 187–196.

[9] Decaudin, P., *Geometric deformation by merging a 3D-object with a simple shape*, in: *GI '96: Proceedings of the conference on Graphics interface '96* (1996), pp. 55–60.

[10] Farin, G., "Curves and surfaces for CAGD: A practical guide," Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002, 5th edition.

[11] Ferry, D. L., *Vidcaps from the Lake near Mud Lake*, available at: http://wolfbat359.com/crpvc.html. Accessed on May 27, 2011.

[12] Floater, M. S., *Mean value coordinates*, Comput. Aided Geom. Des. **20** (2003), pp. 19–27.

[13] Floater, M. S., G. Kós and M. Reimers, *Mean value coordinates in 3d*, Comput. Aided Geom. Des. **22** (2005), pp. 623–631.

[14] Gain, J. and D. Bechmann, *A survey of spatial deformation from a user-centered perspective*, ACM Trans. Graph. **27** (2008), pp. 1–21.

[15] Goldstein, B., *The Goldstein Lab*, available at: http://www.bio.unc.edu/Faculty/Goldstein/lab/wormMO.gif. Accessed on May 27, 2011.

[16] Halderman, J. A., *Worm patterns*, available at: http://www.youtube.com/watch?v=7WOxyVvMp8s. Accessed on May 27, 2011.

[17] Hirota, G., R. Maheshwari and M. C. Lin, *Fast volume-preserving free form deformation using multi-level optimization*, in: *SMA '99: Proceedings of the fifth ACM symposium on Solid modeling and applications* (1999), pp. 234–245.

[18] Huang, J., L. Chen, X. Liu and H. Bao, *Efficient mesh deformation using tetrahedron control mesh*, in: *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling* (2008), pp. 241–247.

[19] Joshi, P., M. Meyer, T. DeRose, B. Green and T. Sanocki, *Harmonic coordinates for character articulation*, ACM Trans. Graph. **26** (2007).

[20] Ju, T., S. Schaefer and J. Warren, *Mean value coordinates for closed triangular meshes*, ACM Trans. Graph. **24** (2005), pp. 561–566.

[21] Kojekine, N., V. Savchenko, M. Senin and I. Hagiwara, *Real-time 3D deformations by means of compactly supported radial basis functions*, in: *Proceedings of Eurographics 2002 (Short Papers)*, 2002, pp. 35–43.

[22] Lai, M.-J. and L. L. Schumaker, "Spline Functions On Triangulations," Cambridge University Press, New York, NY, USA, 2007.

[23] Lamousin, H. J. and W. N. Waggenspack Jr., *NURBS-based free-form deformations*, IEEE Comput. Graph. Appl. **14** (1994), pp. 59–65.

[24] Langer, T., A. Belyaev and H.-P. Seidel, *Spherical barycentric coordinates*, in: *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06 (2006), pp. 81–88.

[25] Lipman, Y., D. Levin and D. Cohen-Or, *Green coordinates*, ACM Trans. Graph. **27** (2008), pp. 78:1–78:10.

[26] Llamas, I., A. Powell, J. Rossignac and C. D. Shaw, *Bender: A virtual ribbon for deforming 3D shapes in biomedical and styling applications*, in: *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, SPM '05 (2005), pp. 89–99.

[27] MacCracken, R. and K. I. Joy, *Free-form deformations with lattices of arbitrary topology*, in: *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), pp. 181–188.

[28] Nealen, A., M. Müller, R. Keiser, E. Boxerman and M. Carlson, *Physically based deformable models in computer graphics*, Comput. Graph. Forum **25** (2006), pp. 809–836.

[29] Sederberg, T. W. and S. R. Parry, *Free-form deformation of solid geometric models*, SIGGRAPH Comput. Graph. **20** (1986), pp. 151–160.

[30] The Blender Foundation, *Blender*, available at: `http://www.blender.org`. Accessed on May 27, 2011.

[31] von Funck, W., H. Theisel and H.-P. Seidel, *Vector field based shape deformations*, ACM Trans. Graph. **25** (2006), pp. 1118–1125.

[32] von Funck, W., H. Theisel and H.-P. Seidel, *Explicit control of vector field based shape deformations*, in: *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (2007), pp. 291–300.

[33] Xu, W.-W. and K. Zhou, *Gradient domain mesh deformation: A survey*, J. Comput. Sci. Technol. **24** (2009), pp. 6–18.

[34] Zhao, Y., X. Liu, C. Xiao and Q. Peng, *A unified shape editing framework based on tetrahedral control mesh*, Comput. Animat. Virtual Worlds **20** (2009), pp. 301–310.