INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Editing C^1-Continuous 2D Spline Deformations by Constrained Least Squares

E. C. S. Rodrigues *J. Stolfi* *A. Gomide*

Technical Report - IC-14-05 - Relatório Técnico

February - 2014 - Fevereiro

Editing C^1-Continuous 2D Spline Deformations by Constrained Least Squares

Elisa de Cássia Silva Rodrigues [*] Jorge Stolfi [*] Anamaria Gomide [*]

Abstract

We describe a method for interactive editing of C^1-continuous deformations of the plane defined by splines of degree 5 on an arbitrary triangular mesh. The continuity constraints and the user editing actions are combined using weighted and constrained least squares instead of the usual finite element approach. We worked with exact integer arithmetic in order to detect and eliminate redundancies in the set of constraints. To validate our method, we used it as part of an friendly interactive editor for "2.5D" space deformations. This editor has been used to deform 3D models of non-rigid living microscopic organisms as seen in actual optical microscope images.

1 Introduction

The manual editing of arbitrarily complex but smooth two-dimensional deformations is a necessary tool in many applications such as the deformation of geometric models, image morphing and image registration. In this paper, our goal is to define mathematical and software tools to enable the edition of deformations in a user-friendly way by using the mouse.

Specifically we consider *mesh-based* space deformation techniques; where the region to be deformed is covered by a *control mesh* whose cells have simple shapes (such as triangles, tetrahedral, squares, cubes etc.). The desired deformation is specified by manipulating the control mesh to obtain a *deformed mesh*. Spline interpolation techniques can then be used to extend the control grid deformation to the whole space inside it.

Because the deformation is applied to a control grid, this approach enables the reuse of the deformation modeled on other models. In addition, it ensures that the method is independent of the resolution and representation of the model. In this paper, we assume that the embedded model is given as a dense triangular mesh with tens of thousands of triangles. Another advantage is that the deformed control grid provides an immediate intuitive understanding of the general nature of the deformation, and of the scope of each control parameter.

Many existing mesh-based space deformation methods provide continuity (C^0) but not smoothness (C^1), and therefore often introduce corners or creases in embedded smooth

[*]Institute of Computing, University of Campinas, Campinas, SP, Brazil. Supported by CAPES (process 01-P-04554-2013), CNPq (processes 140780/2013-0 and 301016/92-5) and FAPESP (process 2007/52015-01).

figures. See Figure 1. The few existing techniques for C^1 deformation either provide little control, or are hard to edit because they have a very large number of free parameters with non-intuitive effects and constraints, or yield deformations whose representations that become increasingly complex as they are edited.

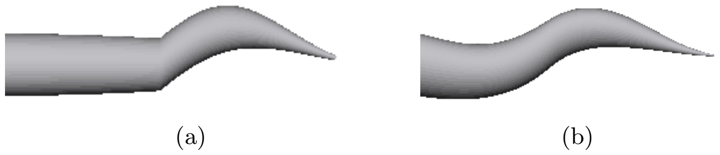

Figure 1: A comparison between space deformations with (a) C^0 and (b) C^1 continuity.

In this paper, we focus on editing algorithms for C^1-continuous 2D deformations that are defined by polynomial splines. We use triangular splines, specifically, because the popular quadrangular tensor-product splines constrains the topology of the mesh [35]. We also assume that the deformations are edited by moving the positions of one or more the Bézier control points of the splines. Each editing operation requires the automatic adjustment of several other nearby points in order to preserve the smoothness of the spline. The set of adjusted points is determined at the time of editing by a general algorithm. We use a weighted and constrained least squares criterion in order to make the adjustment symmetric and natural.

Our algorithm has been implemented as part of an editor which makes it possible to define and modify a "2.5D" deformation with the mouse in a user-friendly way. In order to illustrate and validate our work, we used that editor to model the deformation of living microscopic organisms or cells as seen in optical microscope images. See Figure 2.

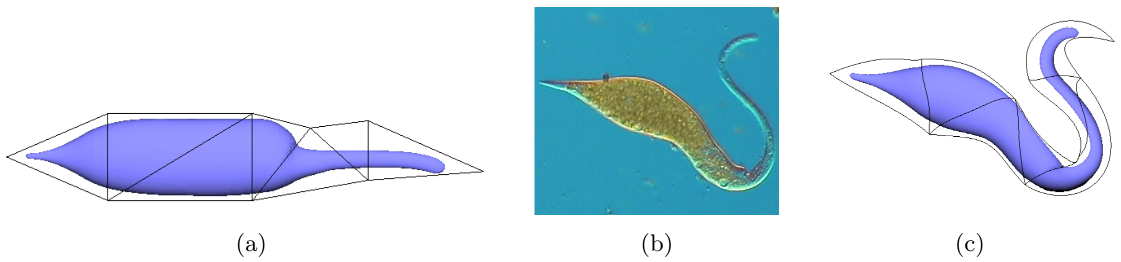

Figure 2: (a) A control mesh surrounding a 3D model of the protozoan *Dileptus anser*; (b) an actual microscopic image of the same *[8]*; and (c) the control mesh and model deformed so as to match the image using our editor.

The paper is organized as follows. We will discuss related work in Section 2, and introduce concepts on spline surfaces, C^0 and C^1 continuity, local control and theoretical limitations in Section 3. We describe our editing algorithm for 2D splines in Section 4, and the interactive 2.5D editor in Section 5. Section 6 has some concluding remarks.

2 Related work

Smooth space deformations have been extensively studied in the context of three-dimensional shape editing. A survey focusing on interactivity is presented by Gain and Bechmann [11]. Space deformation techniques such as radial basis functions and free-form deformations have also been studied in the context of two-dimensional image morphing and registration. Wolberg [33] surveys on image morphing techniques and, more recently, Islam et al. [16] also describe an overview about these techniques. Zitova and Flusser [37] provide a review of image registration. A recent survey is presented by Sotiras et al. [32] emphasizing techniques applied to medical images.

2.1 Non-spline methods

Some mesh-based space deformation methods attempt to obtain C^1 smoothness by the use of non-polynomial interpolating functions, which are determined only by the control mesh vertices and/or faces.

One early approach in this direction was based on *mean value coordinates* [9, 10, 18, 21]; these are infinitely smooth almost everywhere, but are not C^1 at the vertices of the mesh. The *harmonic coordinates* [17] 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* [23]. 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.

Another popular approach to non-spline modeling of deformation uses a linear combination of *radial basis* functions[5]. Each time the deformation is edited one radial element is added to its description and its coefficient is manipulated directly by the user. This approach is very flexible but has the drawback that the complexity of the deformation increases without bound as editing goes on. It is also difficult to ensure that the deformation remain one-to-one, without "fold-over".

2.2 Spline methods

Many deformation modeling methods use *polynomial splines*, that is, piecewise-defined functions where each piece is a polynomial on the domain coordinates.

The theory of multivariate polynomial splines was developed by Paul de Casteljau between 1959 and 1963, but it was popularized by Pierre Bézier after about 1966. De Casteljau initially developed the theory of a triangular surface patches (simplicial) but his work was never published. Bézier's name is most often associated to quadrangular (tensor-product) patches [7]. Two of their most important contributions were the idea of specifying each piece by a set of *Bézier control points*, with fairly intuitive meaning, and the development of relatively simple constraints between such points to ensure C^r continuity for any $r \geqslant 0$.

Barr [1] and Sederberg and Parry [31] were the pioneers in the development of a space deformation method using splines as interpolation technique, namely Free-Form Deformation (FFD). Due to its advantages, the FFD approach has been widely investigated allowing several extensions and variations [3, 24, 11].

Spline-based deformation editors for modeling of 3D objects generally use control meshes consisting of either hexahedra [20, 31] or tetrahedra [2, 15, 36] defined by Bézier control points. As we shall see in Section 5 one can also use prismatic elements [29]. In the context two-dimensional, most spline-based deformation techniques use quadrangular or triangular patches. Some applications involve morphing [26, 22] registration [30] and vectorization [34].

Simplicial (triangular or tetrahedral) Bézier patches have the advantage that they can be joined with almost arbitrary topology. However, the constraint equations may become quite complicated.

With respect to non-spline methods, splines use more control points but require control meshes with fewer cells. An important advantage of the spline approach is that the complexity of the deformation is independent of its editing history. Namely, the number of patches and control points is fixed by the choice of the control mesh . It is also easier to guarantee the one-to-one property if necessary.

2.2.1 Editing techniques

One popular approach for editing with prescribed continuity requires determining a finite element basis for the space of such splines with required continuity, and choosing a subset of the Bézier control points such that there is exactly one independent control point for each basis element. This approach was extensively studied by Schumaker [19] and others; an exposition can be found in Xu's doctoral thesis [35]. In this approach, the editing interface allows the user to specify those independent control points, and the remaining Bézier control points are computed from them.

One particular case of the finite element technique is the B-spline approach [7], where there is only one control point for each patch, and the resulting spline is automatically continuous to the maximum possible order. B-splines are well defined for tensor (quadrangular or hexahedral) type meshes with regular topology [6, 4]; extending them to irregular and simplicial meshes is possible, but rather complicated [14].

Another technique for editing the Bézier control points (which we use) uses the criterion of least squares with constraints [28]. This technique was used, for instance, by Masuda et al. [25] for direct editing of surface meshes. It allows editing any control point and computes the remaining points solving a linear system that combines the criterion of least squares and the constraint equations.

3 Basic concepts on splines

A *mesh* is a partition of some domain $\Omega \subseteq \mathbb{R}^n$ into simple *parts* (or *cells*) with pairwise disjoint interiors. We define a *polynomial spline function* on a mesh T with domain Ω as a function $f : \Omega \to \mathbb{R}$ such that the restriction f^u of f to each part u of the mesh is a polynomial on the coordinates of the argument.

As is well known [19], if the cells of the mesh are simplices (intervals, triangles, tetrahedra, etc.), then each polynomial f^u can be conveniently expressed as a linear combination of the *Bernstein-Bézier polynomials* of the corresponding simplex u. In this article, we consider specifically the case $n = 2$, so the cells are triangles.

3.1 Triangular Bézier splines

Let p be a point of $\mathbb{R}^2$, and β_0, β_1 and β_2 be the barycentric coordinates of p relative to the vertices of a triangle 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

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

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 such polynomial can be written uniquely as

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

for all $p \in \mathbb{R}^2$. The coefficients c_{ijk} are called the *Bézier coefficients of g* (*relative to the triangle u*) [19].

There are $(d+1)(d+2)/2$ Bernstein-Bézier polynomials (and hence Bézier coefficients) for each triangle. 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 3. The triangular grid defined by those points will be called the *local Bézier net* of the triangle; the collection of all those local nets is the *global Bézier control net.*

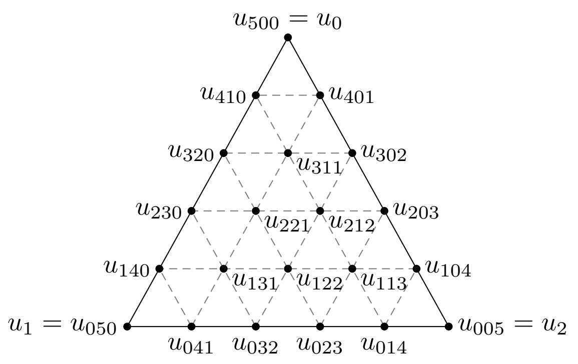

Figure 3: Nominal positions u_{ijk} (dots) of the Bézier coefficients c_{ijk} for a piece f^u of degree 5, and the local Bézier net (solid and dashed lines).

3.2 Using splines to model deformations

A deformation of a region $\Omega \subseteq \mathbb{R}^n$ can be defined as a function $\phi : \Omega \to \mathbb{R}^n$. 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. We call such a function a *spline deformation.* The function ϕ deforms T, the *domain grid*, into a new mesh $\phi(T)$ with curved boundaries, the *deformed grid.* See Figure 4.

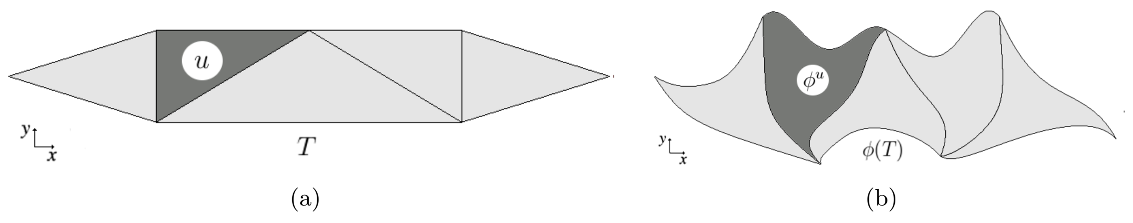

Figure 4: A deformation of $\mathbb{R}^2$ of the (a) domain grid T in the (b) deformed grid $\phi(T)$.

The Bernstein-Bézier polynomial representation can be used to describe the deformation ϕ. For $n = 2$, 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 coordinate r of a point q^u_{ijk}, the *Bézier control point* of ϕ^u with indices i, j and k. The function ϕ^u can be modified by moving the points q^u_{ijk}. See Figure 5.

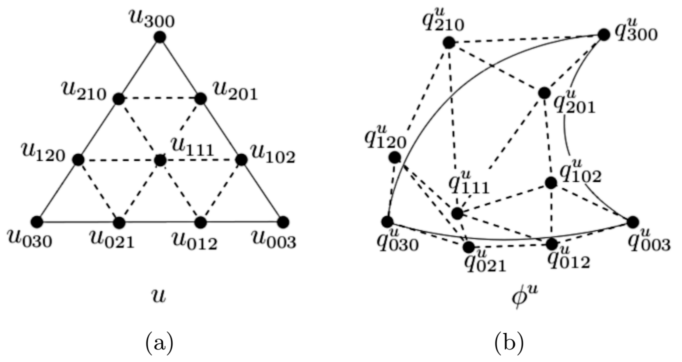

Figure 5: (b) Bézier control points q^u_{ijk} of a degree 3 patch ϕ^u from $\Omega \to \mathbb{R}^2$ and; (a) their nominal positions u_{ijk} on the domain triangle u. The curved triangle on the right is the image of u under the deformation ϕ^u.

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 $\phi^u(u_{00d}) = q^u_{00d}$, but otherwise $\phi^u(u_{ijk}) \neq q_{ijk})$ in general.

3.2.1 Ensuring C^0 continuity

A polynomial spline function has C^0 continuity when there are no discontinuities across cell boundaries. For a spline function f defined on a triangulation T the C^0 condition can be easily expressed in terms of the Bézier coefficients. 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 $u_{ijk} = v_{i'j'k'}$. See Figure 7.

We say that a deformation $\phi : \Omega \to \mathbb{R}^2$ is *continuous to order* r (C^r) if each coordinate of ϕ is continuous to order r. This is so called *parametric continuity* criterion which is distinct from the *geometric continuity* criterion (G^r) sometimes used in computer graphics [7]. The latter is not appropriate here since the parametrization of the deformed grid is relevant, not just its shape.

A deformation $\phi : \Omega \to \mathbb{R}^2$ defined by spline is C^0-continuous across an edge if have $q^u_{ijk} = q^v_{i'j'k'}$ whenever $u_{ijk} = v_{i'j'k'}$. See Figure 6.

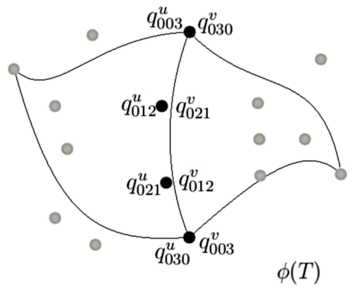

Figure 6: Bézier control points of a deformation ϕ of degree 3 which satisfy C^0 continuity constraints.

3.2.2 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 in any direction coincide at the boundary between u and v. This requirement translates into a set of linear constraints on the Bézier coefficients of f^u and f^v. Specifically, f has C^1 continuity along that edge if and only if

$$c^v_{0jk} = c^u_{0jk} \tag{3}$$

$$c^v_{1jk} = \beta_0 c^u_{1,j,k} + \beta_1 c^u_{0,j+1,k} + \beta_2 c^u_{0,j,k+1} \tag{4}$$

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 [19]. We call Equation (4) the *quadrilateral condition* on those four coefficients. See Figure 7.

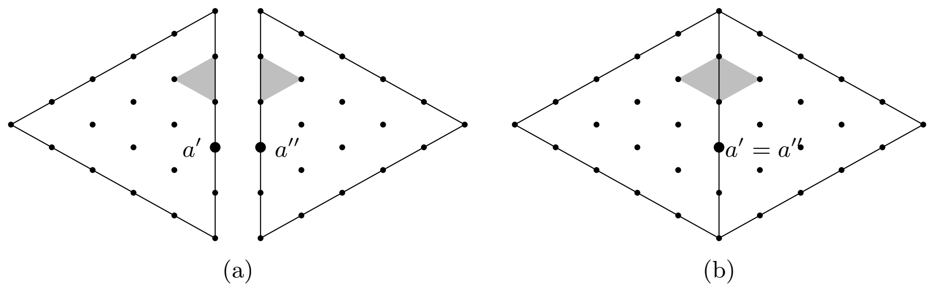

Figure 7: Nominal positions of the Bézier coefficients of a spline of degree 5. (a) Patches before union. (b) Ensuring C^0 continuity by unification of corresponding control points (a', a'', for example), and C^1 continuity by four-points constraints (the gray diamond, for example).

The C^1 continuity of a spline deformation $\phi(T) : \Omega \to \mathbb{R}^2$ with control points q^u_{ijk} and q^v_{ijk} follows analogous conditions to equations in (3) and (4). In other words, 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

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

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

Equation (6) says that the 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 the quadrilateral formed by their nominal positions. See Figure 8.

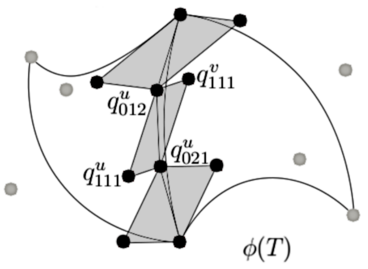

Figure 8: Bézier control points of a spline deformation ϕ of degree 3 which satisfy C^1 continuity constraints.

3.3 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.

The theory of C^1-continuous two-dimensional splines with triangular cells has been extensively studied for example by Schumaker [19]. It is known that the degree d of the interpolating spline must be at least 5 to allow localized editing of the grid and smoothness of the deformed surface [12, 35]. However, to maintain C^0 and C^1 continuity we often have to change one or more control points at the same time, so as to preserve (5) and (6).

If the degree d is too low, the 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 $d \geq 5$. However, for $d \leq 4$ these features cannot be obtained simultaneously [19].

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.

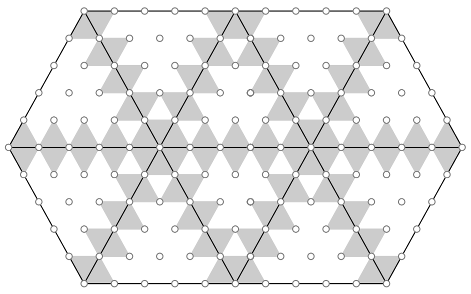

Figure 9: A regular domain grid for a spline deformation of degree 5, showing the control points and the quadrilateral conditions.

3.4 Non-degeneracy

A deformation ϕ is *locally degenerate* at a point $p \in \mathbb{R}^2$ if its derivative is zero in some direction at p. This condition is equivalent to the determinant of the Jacobian matrix $\nabla\phi$ is zero at p, that is,

$$\begin{vmatrix} \dfrac{\partial \phi_x}{\partial x} & \dfrac{\partial \phi_x}{\partial y} \\ \\ \dfrac{\partial \phi_y}{\partial x} & \dfrac{\partial \phi_y}{\partial y} \end{vmatrix} = 0. \tag{7}$$

If the Jacobian is positive everywhere, the mapping ϕ is locally one-to-one (that is, one-to-one when restricted to a sufficiently small neighborhood of any point).

Another advantage of the spline approach is that the condition for non-degeneracy is easier to express, because the derivatives that define the Jacobian are polynomials of degree $d-1$. Moreover if all triangles of the deformed Bézier net have the same orientation as in the undeformed Bézier control net, then the Jacobian is positive. (The converse is not true however).

3.5 Why not tensor splines?

Many modeling systems for 2D splines use quadrilateral tensor patches instead of triangular patches. As in the case of triangular patches, C^0 continuity is assumed if the corresponding control points along the shared edges coincide. For C^1 continuity, each control point along a shared edge must be the average of the control points nearest to it in the interior of each patch; and each corner point must be the average of its four diagonal neighbors. See Figure 10.

As observed in Section 3.3 the degree of a triangular spline must be at least 5. With quadrilateral tensor, one can obtain locality of the deformation and C^1 (or even C^2) continuity with patches of degree 3. However, high degree of the triangular spline is compensated by effectiveness of the deformation generated by the movement of each point.

Considering triangular splines of degree 5 (Figure 9) and quadrilateral splines of degree 3 (Figure 10), both with regular domain meshes of n cells. If we consider adding one patch

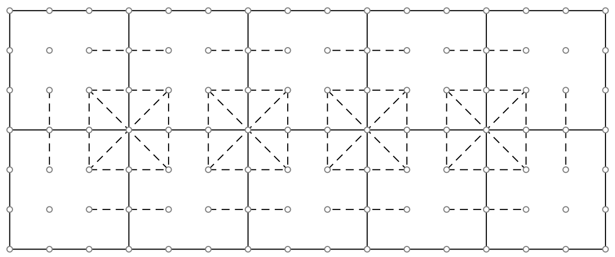

Figure 10: A quadrilateral tensor spline of degree 3. The dotted lines indicate the its C^1 continuity constraints.

at a time, row by row, we conclude that a triangular spline with n patches has $\frac{25}{2}n + O(L)$ distinct control points and $\frac{13}{2}n + O(L)$ degrees of freedom, where L is the number of edges in the free border of the mesh; whereas a quadrangular spline with n patches has $9n + O(L)$ control points and $4n + O(L)$ degrees of freedom. Therefore, a large degree 5 triangular spline will have about $25/13 \approx 1.9$ control points for each degree of freedom, whereas a bicubic tensor spline has about $9/4 = 2.25$ control points for each degree of freedom.

4 Editing a 2D spline

We now describe our iterative spline editing algorithm. This algorithm can be used to edit a spline function from a two-dimensional domain Ω to $\mathbb{R}$ (where the user modifies the values of Bézier coefficients) or to $\mathbb{R}^2$ (where the user modifies the position of Bézier control points). For definitiveness, we will consider the second case in what follows.

4.1 Point updating algorithm

The core of our editor is an iterative algorithm for *Deformation Editing by Constrained Least Squares* which we call *DECLeS*. This algorithm updates the positions of the Bézier control points by the least squares criterion with C^1 continuity constraints. In this way, the algorithm preserves the smoothness of the spline while trying to obey the changes indicated by the user.

In contrast to the finite element technique, the set of points is determined at the time of editing. Specifically, in each editing action the control points are divided into four sets:

- $\mathcal{A}$ *(anchor points):* one or more Bézier control points whose position is entirely set by the user.

- $\mathcal{S}$ *(slave points):* zero or more additional points, chosen by the user, that may also be displaced;

- $\mathcal{D}$ *(derived points):* the set $\mathcal{S}$ plus to zero or more control points that must be adjusted to maintain C^0 and C^1 continuity;

- $\mathcal{F}$ *(fixed points):* set of all other control points of the spline.

The anchor points $\mathcal{A}$ and the slave points $\mathcal{S}$ are specified by the user through the user interface (see Section 5.1). The DECLeS algorithm defines which points must be added to $\mathcal{S}$ to obtain the derived points $\mathcal{D}$ and which constraints (set $\mathcal{E}$) must be satisfied during the deformation.

For each control point p in $\mathcal{A} \cup \mathcal{D}$ the algorithm receives also somehow a new value p'. These new positions can be specified directly by the user with mouse gestures, or by some more complicated editing operation such as translation or rotation of a whole set of control points at once. For the points in $\mathcal{A}$, the new positions are mandatory. For the points in $\mathcal{D}$, the new positions are only suggestions. The DECLeS algorithm then computes for each control point p in $\mathcal{D}$ a new position p'' that is close or equal to p', in such away that all C^0 and C^1 constraints are satisfied. See Figure 11.

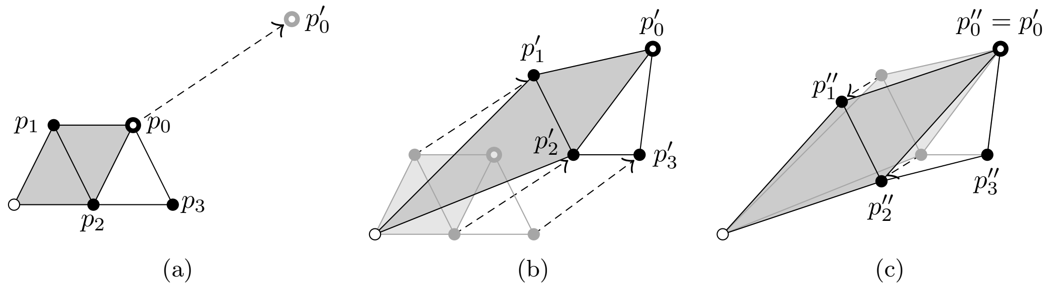

Figure 11: Steps of the deformation editing cycle: (a) user-specified translation of anchor point p_0 to p_0'; (b) desired positions p_1', p_2' and p_3' of derived points p_1, p_2 and p_3 computed by interface; (c) final positions p_1'', p_2'' and p_3'' computed by the algorithm.

The C^0 continuity condition can be trivially enforced by giving the user only one "handle" (editable dot) for all Bézier control points q^u_{ijk} whose nominal positions u_{ijk} coincide. These shared control points lie on edges or vertices of the mesh T shared by two or more triangles. Then any editing to that handle is automatically applied to all control points that it represents. The C^1 continuity condition is expressed by a number of quadrilateral conditions which must be preserved during the edition.

The main updating algorithm is given below and its steps are described in Sections 4.2 and 4.3.

Algorithm 1: DECLeS

```
1 Obtain the set $\mathcal{A}$ and the set $\mathcal{S}$ from the user interface.
2 Determine the set $\mathcal{E}$ and the set $\mathcal{D}$.
3 Obtain the new position $p'$ for each point $p \in \mathcal{A} \cup \mathcal{D}$.
4 Compute new positions $p''$ for each control point in $\mathcal{D}$, so as to satisfy the constraints
  in $\mathcal{E}$.
5 Update $p \leftarrow p'$ for all $p \in \mathcal{A}$ and $p \leftarrow p''$ for all $p \in \mathcal{D}$.
```

4.2 Determining the constraints and derived points

In step 2 the algorithm has to determine a set $\mathcal{E}$ of relevant constraints, and expand the set $\mathcal{S}$ of additional slave points to obtain the set $\mathcal{D}$ of derived points. Initially the algorithm sets $\mathcal{D} \leftarrow \mathcal{S}$. Then, for each control point $p \in \mathcal{A} \cup \mathcal{D}$, the algorithm finds all quadrilateral conditions that involve p; and then adds zero or more control points that enter into these conditions to the set $\mathcal{D}$. If any of those control points enter into additional conditions, the process is iterated as needed. The final sets $\mathcal{D}$ and $\mathcal{E}$ must satisfy the following properties:

1. Every constraint that includes any control point from $\mathcal{A} \cup \mathcal{D}$ is included in $\mathcal{E}$;
2. For any combination of new positions assigned to the control points in $\mathcal{A}$ it is possible to satisfy all constraints in $\mathcal{E}$ by assigning appropriate new positions to all control points in $\mathcal{D}$.

When $d \geq 5$, the final set of derived control points $\mathcal{D}$ can be confined to the triangles that own the control points in $\mathcal{A} \cup \mathcal{S}$ and only a few adjacent triangles. These conditions are a set of linear equations that must be solved to determine the new positions p''.

For the purpose of this step, each Bézier control point $p = q^u_{ijk}$ is classified into six types according to its nominal position u_{ijk} in the triangle u. See Figure 12 and Table 1.

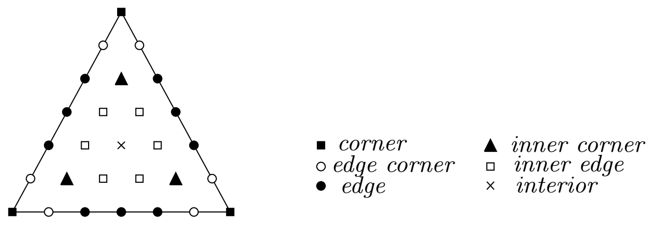

Figure 12: Classification of control points of a Bézier patch of degree 6.

Type	Description
corner	$i = d$ or $j = d$ or $k = d$.
edge corner	$i = 0$ and ($j = 1$ or $k = 1$) or $j = 0$ and ($i = 1$ or $k = 1$) or $k = 0$ and ($i = 1$ or $j = 1$).
edge	none other above and ($i = 0$ or $j = 0$ or $k = 0$).
inner corner	$i = j = 1$ or $i = k = 1$ or $j = k = 1$.
inner edge	none other above and ($i = 1$ or $j = 1$ or $k = 1$).
interior	$i \geq 2$ and $j \geq 2$ and $k \geq 2$.

Table 1: Description of each type of control point.

The type of the point p determines the set of quadrilateral constraints that apply to that point, that are added to $\mathcal{E}$, and the additional control points inserted in the set $\mathcal{D}$. A point p of type *interior* does not take part in any quadrilateral condition, so it does not contribute to the set of derived points $\mathcal{D}$ or to the set of constraints $\mathcal{E}$. A point p of any other type contributes to the constraints and additional derived points according to rules are shown in Figure 13.

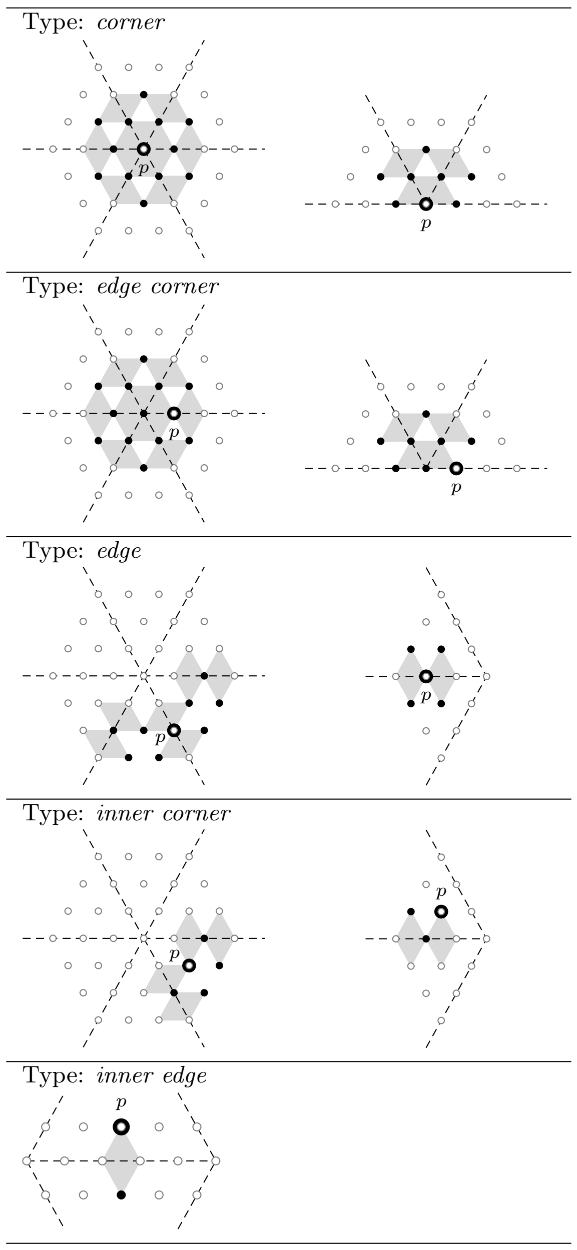

Figure 13: Continuity constraints which belong to the set $\mathcal{E}$ (gray diamonds) and additional derived points (solid dots) added to $\mathcal{S}$ in step 2 depending on the type of the control point $p \in \mathcal{A} \cup \mathcal{S}$ (open dot). For each type, the figure on left side shows a typical situation where the point p is sufficiently far from the triangulation's border. The right side shows a situation near the border where some of those control points and constraints are missing. These diagrams are generalized to vertices of arbitrary degree in the obvious way.

When applying these rules the algorithm skips any control points that would lie on non-existing triangles, and any quadrilateral conditions that would depend on them. These rules ensure that the number of derived points n is greater than or equal to the number of quadrilaterals m and that there is at least one derived point for each quadrilateral chosen.

4.3 Computing new positions of the derived points

Once the derived points and the constraints are chosen, the algorithm solves the set $\mathcal{E}$ of C^1 continuity equations and adjusts the derived points of $\mathcal{D}$ to the computed positions p''_s.

For example, suppose that the set $\mathcal{A}$ is the single point $p = q^u_{023}$ of type *edge* between triangles u (right) and v (left), shown in Figure 14, and $\mathcal{S} = \varnothing$. According to Figure 13, the algorithm will select the four derived control points $\mathcal{D} = \{q^u_{113}, q^u_{122}, q^v_{113}, q^v_{122}\}$, and the two quadrilateral conditions

$$(-1)q^u_{113} + (\beta_0)q^v_{113} + (\beta_1)q^u_{014} + (\beta_2)q^u_{023} = 0 \tag{8}$$
$$(-1)q^v_{122} + (\beta_0)q^u_{122} + (\beta_2)q^u_{023} + (\beta_1)q^u_{032} = 0 \tag{9}$$

where β_0, β_1, and β_2 are the barycentric coordinates of v_0 relative to u_0, u_1, and u_2.

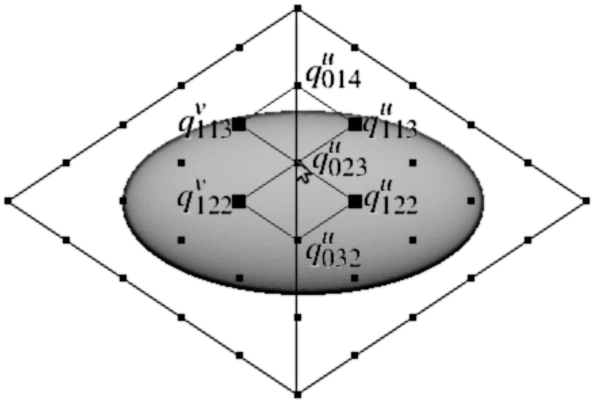

Figure 14: Editing a point $p = q^u_{023}$ of type *edge*.

The computation can be done separately for the x and y coordinates of the points. Therefore, let X and Y be column vectors such that X_s and Y_s are the x and y coordinates, respectively, of each derived point p_s; X'_s, Y'_s are the coordinate vectors of its desired position; and X''_s, Y''_s are the coordinate vectors of its final position.

4.3.1 Determining matrix form of the equations

We can write the continuity equations in matrix form

$$AX'' = -B\overline{X} \tag{10}$$
$$AY'' = -B\overline{Y} \tag{11}$$

where A is a matrix $m \times n$ of coefficients of the derived points in the equations $\mathcal{E}$; X'' and Y'' are coordinate vectors of the n derived points; B is the matrix $m \times t$ of the coefficients of the t anchors and fixed points in those equations; and $\overline{X}$ and $\overline{Y}$ are coordinate vectors of those anchors and fixed points.

In the example of the Figure 14, the matrices A and B are

$$A = \begin{bmatrix} -1 & \beta_0 & 0 & 0 \\ 0 & 0 & -1 & \beta_0 \end{bmatrix} \text{ and } B = \begin{bmatrix} \beta_1 & \beta_2 & 0 \\ 0 & \beta_2 & \beta_1 \end{bmatrix}. \tag{12}$$

As observed in section 4.2, the editing algorithm must ensure that the set of derived points is large enough to allow all constraints to be satisfied.

4.3.2 Eliminating redundant equations

When the number n of derived points is equal to the number m of equations, $n = m$, and these are linearly independent, we can solve the systems in Equations (10) and (11) directly, obtaining a single solution (X'', Y'').

However it often happens that the quadrilateral constraints included in $\mathcal{E}$ are linearly dependent. For example, around a vertex of T with degree g there are g constrains, of which two are implied by the other $g - 2$.

To remove those redundancies, we perform Gaussian elimination on the matrices A and B with exact multi-word integer arithmetic, using the coefficients that vertex coordinates of T are 16-bits integers and then we discard rows of A that are entirely zero. The resulting arrays are then converted to double-precision floating point.

4.3.3 Solving indeterminate systems

The number n of derived points can be greater than the number m of constraints in $\mathcal{E}$. In those case, Equations 10 and 11 are indeterminate linear system S with m constraints and n unknowns where $n > m$ and there are infinitely many solutions.

To overcome this problem, we use the *least squares* approach to find the "best" displacements for the derived points that satisfy the equations. More precisely, let be p_s a derived point, such that $p_s \in \mathcal{D}$. We want to minimize the distance between the new position p''_s of each Bézier control point p_s and the desired position p'_s specified by the user interface. That is, we want minimize the value

$$S(X'', Y'') = \sum_{s=1}^{n} w_s[(X''_s - X'_s)^2 + (Y''_s - Y'_s)^2] \tag{13}$$

where the coefficient w_s is a weight that indicates the importance of the point: the bigger the weight value w_s is, compared to the other weights, the more the algorithm will try to honor the desired position p'_s. In the current implementation, we set $w_s = 1$ for all the derived points.

In order to minimize S, satisfying the constraints of C^1 continuity in Equations (10) and (11), it is necessary that the gradient ∇S be perpendicular to the solution set of the constraints. The gradient consists of the derivatives

$$\frac{\partial S}{\partial X''_s} = 2w_s(X''_s - X'_s) \tag{14}$$

$$\frac{\partial S}{\partial Y''_s} = 2w_s(Y''_s - Y'_s) \tag{15}$$

for $s = 1, 2, \ldots, n$. To be perpendicular to the constraint solution space, the gradient ∇S must satisfy

$$\frac{\partial S}{\partial X''_s} = -\sum_{k=1}^{n} \lambda_k A_{ks} \tag{16}$$

$$\frac{\partial S}{\partial Y''_s} = -\sum_{k=1}^{n} \gamma_k A_{ks} \tag{17}$$

where the variables λ_k and γ_k are indeterminate real coefficients, the *Lagrange multipliers* [27]. Therefore

$$2w_s X''_s + \sum_{k=1}^{n} \lambda_k A_{ks} = 2w_s X'_s \tag{18}$$

$$2w_s Y''_s + \sum_{k=1}^{n} \gamma_k A_{ks} = 2w_s Y'_s. \tag{19}$$

Equations (18) and (19) can be written in matrix form

$$2WX'' + A^\top \Lambda = 2WX' \tag{20}$$

$$2WY'' + A^\top \Gamma = 2WY' \tag{21}$$

where W is the $n \times n$ diagonal matrix of the weights (with $W_{ss} = w_s$), therefore the matrix W is the $n \times n$ identity matrix; and Λ and Γ are column vectors with the Lagrange multipliers $\lambda_1, ..., \lambda_m$ and $\gamma_1, ..., \gamma_m$.

We can combine the Equations (20) and (21) with the constraints of C^1 continuity in Equations (10) and (11) obtaining the linear systems

$$\begin{bmatrix} 2W & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} X'' \\ \Lambda \end{bmatrix} = \begin{bmatrix} 2WX' \\ -B\overline{X} \end{bmatrix} \tag{22}$$

$$\begin{bmatrix} 2W & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} Y'' \\ \Gamma \end{bmatrix} = \begin{bmatrix} 2WY' \\ -B\overline{Y} \end{bmatrix}. \tag{23}$$

To solve the systems (22) and (23), we use the method of Gaussian elimination [27].

5 Application

We have used the 2D editing algorithm described in this paper, as part of an editor of C^1-continuous 2.5D space deformations [29]. This editor was used to deform 3D models of cells and other organisms viewed through optical microscopes.

Although the models for this application are three-dimensional, the deformations are essentially two-dimensional with little change in depth, especially because the third dimension can not be easily perceived through a microscope. Thus, the deformations allowed by this system consist of a 2D deformation in the x and y directions combined with a (x, y)-dependent 1D stretching map in the z direction.

Specifically, we define a 3D control grid around the object consisting of a single layer of triangular prisms with curved top and bottom faces and vertical walls. The editor has separate modes for adjusting the (x, y) deformation (a spline mapping $\phi : \mathbb{R}^2 \to \mathbb{R}^2$) and the top and bottom surfaces (two spline functions $\sigma_0, \sigma_1 : \mathbb{R}^2 \to \mathbb{R}$). Each function has pieces of degree 5 and is defined by Bézier control points (21 per triangle) according to Section 3. Our algorithm is used in all three editing modes to ensure ϕ, σ_0 and σ_1 are C^1-smooth. A video showing the implemented editor in action is available at `http://www.youtube.com/watch?v=X-vKoOrrlTA`. Figures 2 and 15 show the kind of results obtained.

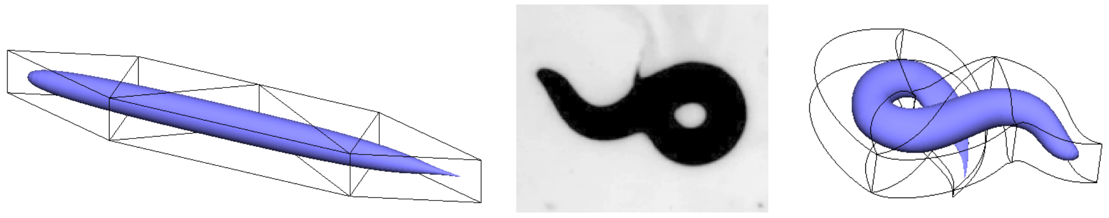

Figure 15: (a) A prism grid and 3D model of the nematode *Caenorhabditis elegans*; (b) a microscope image *[13]*; and (c) a 3D view of the model deformed with our editor.

5.1 Local translation

In our editor, a typical xy editing operation is a *variable neighborhood soft translation* of control points. The user selects one or more control points (that will be the anchors of the action) and drags them to new positions. The application then selects additional slave points around those anchors, and suggests that they undergo a translation in the same direction but with smaller magnitudes.

Specifically, the slave points $\mathcal{S}$ are all control points that lie within a user specified maximum graph distance $\delta_{\max}$ in the global Bézier control net. See Figure 16.

The user interface then suggests for each slave point a translation that decreases in magnitude as one goes away from the anchor points. The new positions are computed only in step 4 of the Algorithm 1, after the set $\mathcal{S}$ has been augmented to yield the set $\mathcal{D}$ of derived points. See Figure 17.

More precisely, the suggested position p'_s for a control point $p_s \in \mathcal{D}$ will be

$$p'_s = p^0_s + \theta_s \vec{v} \tag{24}$$

where p^0_s is the current position of the derived point p_s, $\vec{v}$ is the displacement vector applied by the user to the anchors, and θ_s is a number between 0 and 1 that depends on the

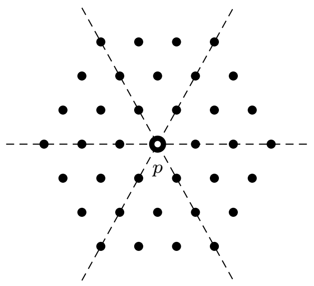

Figure 16: Selected slave points $\mathcal{S}$ (solid dots) for the anchor p (open dot) with a maximum graph distance $\delta_{\max} = 3$.

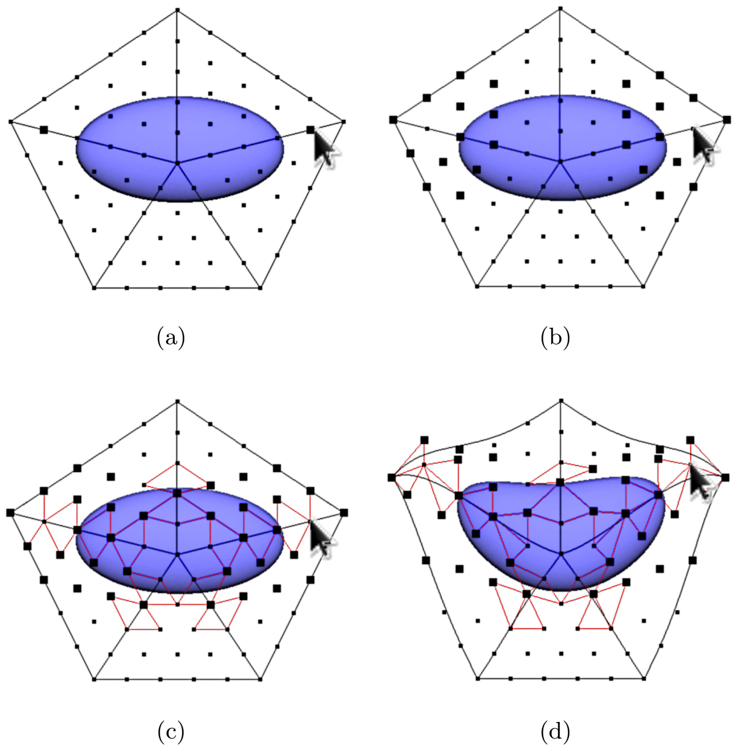

Figure 17: Editing of two anchor points with $\delta_{max} = 2$. (a) Anchor points specified by the user (set $\mathcal{A}$); (b) slave points specified by the user interface (set $\mathcal{S}$); (c) derived points specified by the updating algorithm (set $\mathcal{D}$); and (d) derived points updated.

distances δ'_s and δ''_s from the control point p_s to the sets $\mathcal{A}$ and $\overline{\mathcal{A} \cup \mathcal{D}}$, respectively in the Bézier control mesh, that is,

$$\theta_s = \frac{\delta''_s}{\delta'_s + \delta''_s} \tag{25}$$

where δ'_s is a distance between the derived point p_s and the nearest anchor point; and δ''_s is a distance between the derived point p_s and the nearest fixed point.

Our editor also supports a local soft rotation and scale of one or more control points about an arbitrary center $c \in \mathbb{R}^2$ where the desired angle of rotation of each derived point is determined in a similar way. See Figure 18.

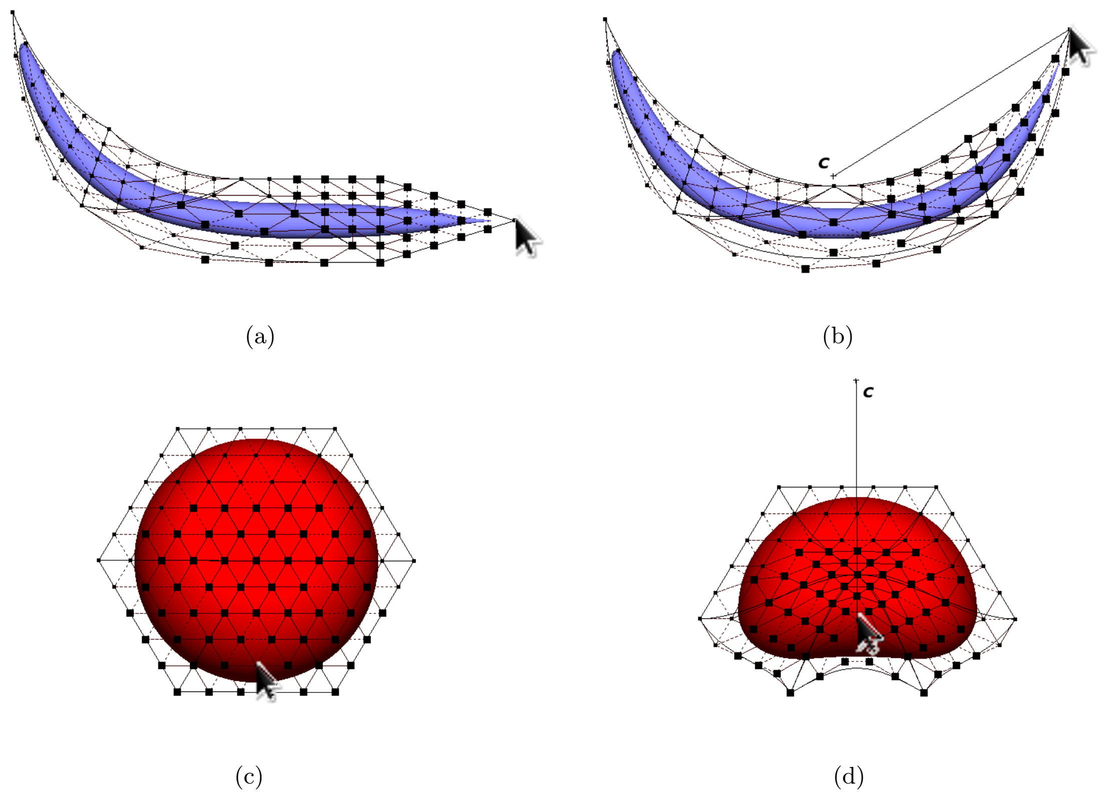

Figure 18: (a) Editing of an anchor point with $\delta_{max} = 8$. (b) Soft rotation around the center c. (c) Editing of an anchor point with $\delta_{max} = 5$. (d) Scale around the center c.

6 Conclusions

We described a general modeling technique for interactive editing of C^1-continuous two-dimensional deformations using triangular elements with Bézier control nets.

The method described supports splines of degree 5 or higher and allows convenient editing of the deformation while preserving the C^1 continuity of the surface. In order to achieve this, the user editing actions and the continuity constraints are combined using the constrained least squares criterion. The algorithm DECLeS developed was used as part of an effective and user-friendly editor of 2.5D space deformations that can be used, for example, to reproduce the deformation of 3D models of non-rigid cells and other organisms viewed through optical microscopes.

References

[1] Alan H. Barr. Global and local deformations of solid primitives. *SIGGRAPH Comput. Graph.*, 18(3):21–30, January 1984.

[2] D. Bechmann, 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*, pages 1715–1725, New York, NY, USA, 1997. Elsevier Science Inc.

[3] Sabine Coquillart. 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*, pages 187–196, New York, NY, USA, 1990. ACM.

[4] Yuanmin Cui and Jieqing Feng. Technical section: Real-time b-spline free-form deformation via gpu acceleration. *Comput. Graph.*, 37(1-2):1–11, February 2013.

[5] A. de Boer, M.S. van der Schoot, and H. Bijl. Mesh deformation based on radial basis function interpolation. *Computers & Structures*, 85(11–14):784–795, 2007. Fourth MIT Conference on Computational Fluid and Solid Mechanics.

[6] Carl de Boor. Splines as linear combinations of B-splines. A Survey, 1976.

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

[8] Donald L. Ferry. Vidcaps from the Lake near Mud Lake. Available at: `http://wolfbat359.com/crpvc.html`. Accessed on Jan 09, 2013.

[9] Michael S. Floater. Mean value coordinates. *Comput. Aided Geom. Des.*, 20:19–27, March 2003.

[10] Michael S. Floater, Géza Kós, and Martin Reimers. Mean value coordinates in 3d. *Comput. Aided Geom. Des.*, 22:623–631, October 2005.

[11] James Gain and Dominique Bechmann. A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph.*, 27(4):1–21, 2008.

[12] Stefanie Hahmann and Georges-Pierre Bonneau. Polynomial surfaces interpolating arbitrary triangulations. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):99–109, January 2003.

[13] J. Alex Halderman. Worm patterns. Available at: `http://www.youtube.com/watch?v=7WOxyVvMp8s`. Accessed on May 27, 2011.

[14] Ying He, Xianfeng Gu, and Hong Qin. Fairing triangular B-splines of arbitrary topology. In *In Proceedings of Pacific Graphics '05*, pages 153 – 156, 2005.

[15] Jin Huang, Lu Chen, Xinguo Liu, and Hujun Bao. Efficient mesh deformation using tetrahedron control mesh. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 241–247, New York, NY, USA, 2008. ACM.

[16] Md. Baharul Islam, Md. Tukhrejul Inam, and Balaji Kaliyaperumal. Overview and challenges of different image morphing algorithms. *International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE)*, 2(4), 2013.

[17] Pushkar Joshi, Mark Meyer, Tony DeRose, Brian Green, and Tom Sanocki. Harmonic coordinates for character articulation. *ACM Trans. Graph.*, 26, July 2007.

[18] Tao Ju, Scott Schaefer, and Joe Warren. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph.*, 24:561–566, July 2005.

[19] Ming-Jun Lai and Larry L. Schumaker. *Spline Functions On Triangulations.* Cambridge University Press, New York, NY, USA, 2007.

[20] Henry J. Lamousin and Warren N. Waggenspack Jr. NURBS-based free-form deformations. *IEEE Comput. Graph. Appl.*, 14(6):59–65, 1994.

[21] Torsten Langer, Alexander Belyaev, and Hans-Peter Seidel. Spherical barycentric coordinates. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 81–88, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[22] Seung-Yong Lee, Kyung-Yong Chwa, and Sung Yong Shin. Image metamorphosis using snakes and free-form deformations. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 439–448, New York, NY, USA, 1995. ACM.

[23] Yaron Lipman, David Levin, and Daniel Cohen-Or. Green coordinates. *ACM Trans. Graph.*, 27:78:1–78:10, August 2008.

[24] Ron MacCracken and Kenneth 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*, pages 181–188, New York, NY, USA, 1996. ACM.

[25] H. Masuda, Y. Yoshioka, and Y. Furukawa. Interactive mesh deformation using equality-constrained least squares. *Comput. Graph.*, 30(6):936–946, December 2006.

[26] Tomoyuki Nishita, Toshihisa Fujii, and Eihachiro Nakamae. Metamorphosis using bézier clipping, 1993.

[27] Carl E. Pearson. *Handbook of applied mathematics: Selected results and methods.* Van Nostrand Reinhold, New York, NY, US, 2th edition, 1990.

[28] William H. Press. *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Nova Iorque, NY, EUA, 3 edition, set 2007.

[29] Elisa C. S. Rodrigues, Anamaria Gomide, and Jorge Stolfi. A user-editable C^1-continuous 2.5D space deformation method for 3D models. *Electronic Notes in Theoretical Computer Science*, 281(0):159 – 173, 2011. Selected papers of the 2011 Latin American Conference in Informatics (CLEI).

[30] D. Rueckert, L. I. Sonoda, C. Hayes, D. L G Hill, M. O. Leach, and D.J. Hawkes. Nonrigid registration using free-form deformations: application to breast mr images. *Medical Imaging, IEEE Transactions on*, 18(8):712–721, 1999.

[31] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph.*, 20(4):151–160, 1986.

[32] A. Sotiras, C. Davatzikos, and N. Paragios. Deformable medical image registration: A survey. *Medical Imaging, IEEE Transactions on*, 32(7):1153–1190, 2013.

[33] George Wolberg. Image morphing: a survey. *The Visual Computer*, 14(8-9):360–372, 1998.

[34] Tian Xia, Binbin Liao, and Yizhou Yu. Patch-based image vectorization with automatic curvilinear feature alignment. *ACM Trans. Graph.*, 28(5):115:1–115:10, December 2009.

[35] Dianna Xu. *Incremental algorithms for the design of triangular-based spline surfaces.* PhD in computer and information science, Faculties of the University of Pennsylvania, Philadelphia, PA, USA, 2002.

[36] Yong Zhao, Xinguo Liu, Chunxia Xiao, and Qunsheng Peng. A unified shape editing framework based on tetrahedral control mesh. *Comput. Animat. Virtual Worlds*, 20(2-3):301–310, 2009.

[37] Barbara Zitová and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977 – 1000, 2003.