Interactive Editing of C^1-Continuous 2D Spline Deformations using ECLES Method

Elisa de Cássia Silva Rodrigues and Jorge Stolfi
Institute of Computing, University of Campinas (UNICAMP)
Campinas, Brazil
Email: {erodrigues, stolfi}@ic.unicamp.br

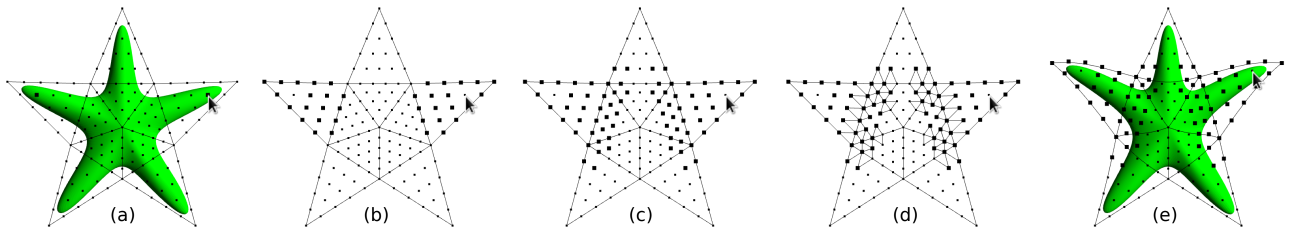


Fig. 1. A control mesh with 10 triangles showing the Bézier control points (small dots) and some specific subsets (larger dots): (a) Anchor points (set $\mathcal{A}$) specified by the user; (b) initial derived points (set $\mathcal{S}$) specified by setting $\delta_{max} = 3$ by the user interface; (c) derived points (set $\mathcal{D}$) specified by the 2DSD; (d) Relevant and non-redundant constraints specified by ECLES; and (e) modified derived control points.

***Abstract*—We describe 2DSD, a general algorithm for interactive editing of smooth deformations of the plane defined by polynomial splines of degree 5 on an triangular mesh. 2DSD is based on ECLES, a flexible and general exact method for interactive editing of numeric parameters subject to linear or affine constraints. Each editing action is abstracted as setting a set of anchor points to new positions, and recomputing some derived points so as to satisfy the constraints. When there are multiple solutions, ECLES uses least squares to choose the closest solution to a suggested hint. The advantages of 2DSD include immunity to rounding errors, automatic selection of the derived points for each editing action, and a representation of the deformation whose complexity does not grow with as editing progresses. To validate our algorithm, we re-implemented an editor of space deformations designed to assist the matching of 3D models of flexible microorganisms to microscope images of the same.**

***Keywords*-Editing; Deformation; Splines; Exact Arithmetic; Least Squares; Morphing.**

I. Introduction

In this paper, we define a general method for interactive editing of smooth simplicial spline deformations of the plane, which we call 2DSD. This is a necessary tool in many applications such as editing of surfaces, deformation of geometric models, image morphing, registration and vectorization.

We implemented our algorithm in an editor which makes it possible to define and modify a 2D deformation with the mouse in a user-friendly way. See Fig. 1.

a) Deformation model: In our method, the object to be deformed is covered by a *control mesh*, a planar triangulation with straight edges. The deformation is a function ϕ from the region Ω covered by a mesh in the plane $\mathbb{R}^2$, defined by a single polynomial map inside each cell. We require ϕ to be *smooth* (C^1), that is, to be continuous and have continuous derivatives, even along the shared cell boundaries. We chose to use triangular splines because the popular quadrangular tensor-product splines constrain the topology of the mesh.

Inside each cell, the deformation is specified by the Bézier control points of the corresponding polynomial. The desired deformation is edited by manipulating one or more control points of the spline. 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 the 2DSD.

b) Contributions: The main contributions are the use of ECLES [1], a flexible and general exact method for interactive editing of numeric parameters subject to linear or affine constraints, to select and to solve the smoothness constraints, and an automatic method to select the control points to be modified at each editing action. Standard floating-point solvers may fail to identify possible redundancies due to rounding errors, which justifies the use of ECLES, since its fraction-free integer solver efficiently and reliably identifies redundant constraints, and to ensure that the constraints are exactly satisfied even after many editing actions.

Some techniques for C^1 deformation, such as radial basis methods, yield deformations whose representations become increasingly complex as they are edited. In our approach, the description of the deformation remains simple and of finite size. Since the deformation is applied to a control mesh, this approach enables the reuse of the edited deformation on other models, and ensures that the method is independent of the resolution and representation of the target object. Another advantage is that the deformed control mesh provides an immediate visualization of the scope of each control parameter and of the deformation, even before it is applied to the object.

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 cell of the mesh and perhaps a few other triangles surrounding it. Also, at each point of the domain, except in cell boundaries, the deformation has a simple analytic formula that allows the efficient computation of derivatives.

A. *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 [2]. 2D space deformation techniques such as radial basis functions and free-form deformations have also been studied in the context of image morphing and registration [3], [4], [5], [6].

Our 2DSD algorithm is an evolution and generalization of the one previously described by Rodrigues et al. [7]. The improvements include using ECLES instead of floating-point linear algebra packages, a more flexible and general method for control point selection (allowing multiple anchors), and a different goal function for the least squares method. Also, the 2DSD approach can easily accommodate higher degree splines and other first degree constraints, such as C^2 smoothness, vertical or horizontal alignment, fixed points, etc.

a) 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* [8], [9], [10]; these are infinitely smooth almost everywhere, but are not C^1 at the vertices of the mesh. The *harmonic coordinates* [11] 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* [12]. These interpolants have a closed form, but are still expensive to compute.

Another popular approach to non-spline modeling uses a linear combination of *radial basis* functions[13]. 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.

b) Spline methods: Many deformation modeling methods use polynomial splines, that is, piecewise-defined functions where each piece is a polynomial on the domain coordinates, developed by Paul de Casteljau and Pierre Bézier [14].

Barr [15], and Sederberg and Parry [16] pioneered a space deformation method using splines as interpolation technique, namely Free-Form Deformation (FFD) [17], [18], [2].

Spline-based deformation editors for modeling of 3D objects generally use a control mesh consisting of either hexahedra [19], [16] or tetrahedra [20], [21], [22] defined by Bézier control points. As we shall see in Section V, one can also use prismatic elements [7].

In the two-dimensional context, most spline-based deformation techniques use quadrangular or triangular patches. Some applications include morphing [23], [24], registration [25], and vectorization [26]. Simplicial (triangular or tetrahedral) Bézier patches have the advantage over quadrangular ones that they can be joined with almost arbitrary topology. On the other hard, their continuity constraints are more complicated.

Compared to non-spline methods, splines generally use more control points, but can use a control mesh 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.

B. *Technique overview*

We can write a set of C^1 continuity constraints of a spline as the matrix equation

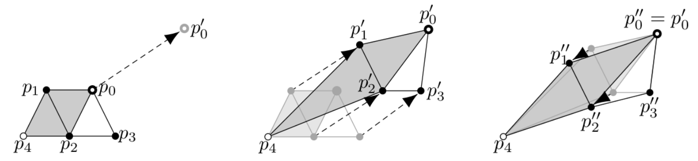
$$RP = Q \tag{1}$$

where P is an $c \times 2$ matrix of the coordinates of all Bézier control points, R is a constant $l \times c$ coefficient matrix, and Q is a constant $l \times 2$ matrix (usually zero).

a) Parameter sets: Let $\mathcal{P}$ be the set of indices of all control points of the mesh, and $\mathcal{R}$ the set of indices of the continuity constraints. For any $\mathcal{X} \subseteq \mathcal{P}$, we denote by $p_{\mathcal{X}}$ the subvector p whose indices are the elements of $\mathcal{X}$. For each editing action, two disjoint subsets of $\mathcal{P}$ should be defined:

- $\mathcal{A}$ (*anchor*): indices of one or more control points whose values will be set by the user;
- $\mathcal{D}$ (*derived*): indices of zero or more control points that may be adjusted if necessary to satisfy the constraints.

For each $s \in \mathcal{A}$, the user specifies a new position p'_s which is mandatory. For each $s \in \mathcal{D}$, a new position p'_s is also suggested. The 2DSD method will compute a new position p''_s for each parameter. If $s \in \mathcal{A}$, p''_s is equal to the given position p'_s. If $s \in \mathcal{D}$, p''_s is close to p'_s, but not necessarily equal to it. For every $s \in \mathcal{P} \setminus (\mathcal{A} \cup \mathcal{D})$, the desired position p'_s and final position p''_s are equal to the current value p_s, that is, the value does not change. See an example in Fig. 2.



(a) First step. (b) Second step. (c) Result.

Fig. 2. Steps of the 2D spline deformation editing cycle with set $\mathcal{A} = \{0\}$, and set $\mathcal{D} = \{1, 2, 3\}$ subject to one relevant constraint (the shaded parallelogram). (a) User-specified translation of anchor point p_0 to p'_0; (b) desired positions p'_1, p'_2, and p'_3 of the derived points; and (c) final positions p''_1, p''_2, and p''_3 of the derived points.

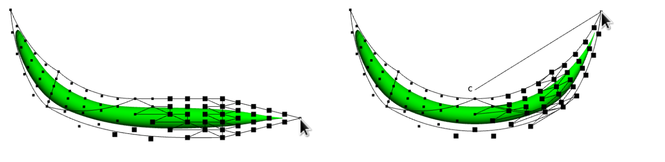
The constraints that involve control points of $\mathcal{A}$ or $\mathcal{D}$ are called *relevant constraints*, whose indices comprise the set $\mathcal{E}$, a subset of $\mathcal{R}$. There are often redundant constraints, especially when some of the control points are considered fixed. Therefore, one of the subproblems that we need to solve is to identify and ignore such redundant constraints. For this, the ECLES method is used by the 2DSD algorithm.

b) Solvability condition: A pair ($\mathcal{A}$, $\mathcal{D}$) is said to be *strongly solvable* if for any assignment of values to $p_\mathcal{A}$ there is $p''_\mathcal{D}$ that satisfies all constraints. If $p''_\mathcal{D}$ only exists for a particular $p'_\mathcal{A}$, we say that ($\mathcal{A}$, $p'_\mathcal{A}$, $\mathcal{D}$) is *weakly solvable*.

c) A user editing action: A typical user editing action which we consider is a *variable-neighborhood soft translation* of one or more control points (see Fig. 1). In the initial part, the user selects with the mouse a set $\mathcal{A}$ of anchor points. Then, the set $\mathcal{S}$ is automatically selected by the interface as having all points that lie within a user specified maximum distance of $\mathcal{A}$ in the global Bézier control net (see Section II-A). Then 2DSD determines the final set $\mathcal{D}$ of derived points from the set $\mathcal{S}$, and computes a factor θ for each point. Finally, the set of non-redundant constraints is identified.

The user then defines a displacement $\vec{v}$ for the anchor points by dragging them to new positions. The 2DSD algorithm gets called at each new position of the anchors, and computes for each point p_s a suggested translation $\theta_s\vec{v}$ that decreases in magnitude as one goes away from the anchor points. The second part of the editing action is concluded when 2DSD computes the new positions of the anchor and derived points.

Our prototype editor also supports the operations of *local soft rotation* (see Fig. 3) and *local soft scaling* of one or more anchors. In rotation, the user defines an angle α and center $c \in \mathbb{R}^2$. The suggested angle α'_s of each point p_s is $\theta_s\alpha$. In scaling, the user defines a center c and a scale factor σ. The suggested scaling for each point p_s is σ^{θ_s}.



(a) First step. (b) Second step.

Fig. 3. Soft rotation of an anchor point (arrow) with $\delta_{max} = 8$ around the center point c.

II. Triangular Spline Deformations

In this section, we review the theory of Bézier splines defined on *simplicial meshes*, whose cells are geometric simplices (intervals, triangles, tetrahedra, etc.). In this paper, we consider specifically the case $t = 2$, so the cells are triangles.

As is well known, any polynomial f from $\mathbb{R}^2$ to $\mathbb{R}$ of degree d can be conveniently expressed as a linear combination of the $(d+1)(d+2)/2$ *Bernstein-Bézier simplicial polynomials* of degree d relative to any simplex u, denoted B^u_{ijk} [27]. That is, let p be a point of $\mathbb{R}^2$, every such polynomial f can be written uniquely as

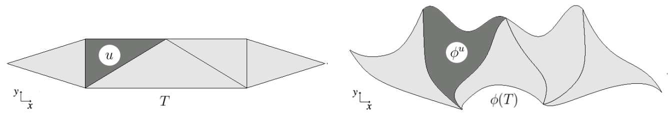
$$f(p) = \sum_{i+j+k=d} c_{ijk} B^u_{ijk}(p) \tag{2}$$

where c_{ijk} are the *Bézier coefficients of f relative to u* [27].

Each coefficient c_{ijk} can be associated to a *nominal position* u_{ijk} in the triangle u, whose barycentric coordinates β_0, β_1 and β_2 of p relative to the vertices of u are, by definition, $(i/d, j/d, k/d)$.

A. Using splines to model deformations

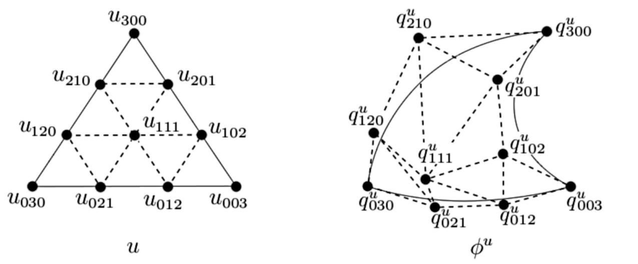
A deformation of a region $\Omega \subseteq \mathbb{R}^t$ can be defined as a function $\phi : \Omega \to \mathbb{R}^t$. A convenient way of modeling such functions is to let each coordinate of $\phi(x)$ be a spline function $\phi_r(x)$, with $0 \le r \le n$; all these splines being of the same degree and defined on the same mesh T. We call such function a *spline deformation*. The function ϕ deforms T, the *domain mesh*, into a new mesh $\phi(T)$ with curved boundaries, the *deformed mesh*. See Fig. 4.



(a) Domain mesh. (b) Deformed mesh.

Fig. 4. A deformation of $\mathbb{R}^2$ of the domain mesh T in the deformed $\phi(T)$.

The Bernstein-Bézier polynomial representation can be used to describe the deformation ϕ. For $t = 2$, 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 Fig. 5. The triangular grid defined by those points and the edges, shown in Fig. 5, will be called the *local Bézier control net* of the triangle; the union of all those local nets is the *global Bézier control net*.



(a) Nominal positions. (b) Bézier control points.

Fig. 5. Bézier control points q^u_{ijk} of a degree 3 patch ϕ^u from $\Omega \to \mathbb{R}^2$, and their nominal positions u_{ijk} on the domain triangle u, showing the local Bézier control net (dashed lines). 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 those 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.

B. Continuity constraints

We say that a deformation ϕ is *continuous to order* r (C^r) if each coordinate of ϕ is continuous to order r. This is called *parametric continuity* which is distinct from the *geometric continuity* (G^r) sometimes used in computer graphics [14]. The latter is not appropriate here since the parametrization of the deformed mesh is relevant, not just its shape.

Let u and v be two adjacent triangles of T. It is well known that the condition for the spline deformation ϕ to be continuous across the common edge of u and v is that $q^u_{ijk} = q^v_{i'j'k'}$ for all i, j, k, i', j', k' such that the nominal positions coincide, that is, such that $u_{ijk} = v_{i'j'k'}$. Therefore, C^0 continuity can be achieved by representing those point pairs by a single point.

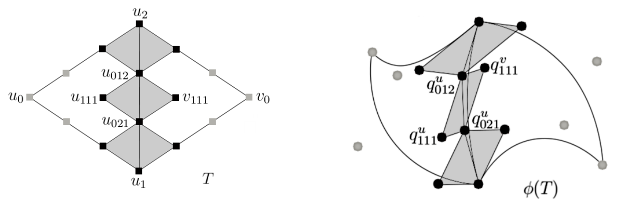
A spline is smooth along the common edge between u and v if the first derivatives of the corresponding polynomial maps ϕ^u and ϕ^v, in any direction, coincide at any point on that boundary. For simplicial polynomial splines, this requirement translates into a set of linear constraints on the Bézier points of ϕ^u and ϕ^v. Specifically, ϕ has C^1 continuity if and only if

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

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

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 [27].

We call Equation (4) the *quadrilateral condition*. It 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 Fig. 6.



(a) Nominal positions. (b) Bézier control points.

Fig. 6. A deformation ϕ of degree 3 which satisfy C^0 and C^1 continuity constraints (the gray diamonds).

C. Local control

The theory of C^1-continuous 2D splines with triangular cells has been extensively studied, for example, by Schumaker [27]. It is known that there is a minimum degree d of the interpolating spline that allows local editing of the spline while maintaining its smoothness [28], [29]. If the degree d is too low, the constraints are interconnected in such a way that the required changes propagate from triangle to triangle over all the domain mesh, so that local control is not possible.

In particular, for triangle meshes in $\mathbb{R}^2$, the smallest degree that allows local control with C^1 continuity is $d = 5$, which we use in the examples that follow. In this case, each triangle has 21 Bézier control points.

Let l be the number of edges in the free border of the mesh, it can be proved that a spline with n parts has $\frac{25}{2}n + O(l)$ distinct control points and $\frac{13}{2}n + O(l)$ degrees of freedom; that is, about $\frac{25}{13} = 1.9$ control points for each degree of freedom.

In comparison, a quadrangular bicubic tensor spline of degree 3 (the smallest degree that provides local control and C^1 continuity) with n patches has $9n + O(l)$ control points and $4n + O(l)$ degrees of freedom; that is, about $9/4 = 2.25$ control points for each degree of freedom. Therefore, despite requiring a higher degree than tensor splines to obtain locality, the triangular spline deformation generated by each control point is more effective.

III. The 2DSD Deformation Editing Algorithm

The editing of smooth 2D spline deformations can be considered a special case of the general problem of editing a set of parameters with linear (or affine) constraints. In this section, we describe the 2DSD algorithm for this problem using the ECLES method [1], described in Section IV.

Fig. 7 shows the interaction model between the user, the application's user interface, our interactive algorithm 2DSD, and the ECLES general method. This process has two steps: the first occurs once in each editing action when the user chooses the points to be adjusted; and the second step occurs one or more times when the user modifies the position those points, e.g. by dragging them with the mouse. To simplify, only one editing operation is shown (translation of the anchor points). Other operations, such as rotation and scaling of the anchor points can be implemented in similar ways.

Initially, through appropriate gestures of the user interface, the user selects a set of anchors ($\mathcal{A}$) and a set of initial derived points ($\mathcal{S}$). These sets are transmitted by the interface method (*UI.Select*) to the first part of the algorithm, *2DSD.Select* (see Section III-A). This procedure chooses the final derived points (set $\mathcal{D}$) based on the set $\mathcal{S}$. Then, the sets $\mathcal{A}$ and $\mathcal{D}$ of control points and the constraints matrix R are given to *ECLES.Initialize* (see Section IV-A). This procedure constructs the coefficient matrix of the linear system, in factored form (Π_{R}, L, D, U, Π_{C}; see Section IV-A), and computes the rank r of that matrix. Optionally, *ECLES.Initialize* checks the strong solvability, and, if it is not satisfied, fails and notifies the application.

The second part is executed when the user moves the anchors to new positions, summarized by a displacement vector $\vec{v}$, in Fig. 7. This data is passed by the corresponding interface method (*UI.Drag*) to *2DSD.Translate* (see Section III-B) which computes the new positions $p'_{\mathcal{A}}$ of the anchors and the suggested positions $p'_{\mathcal{D}}$ for the derived points, and passes that and other informations to *ECLES.Update* (see Section IV-B).

If *ECLES.Initialize* did not check the strong solvability, then *ECLES.Update* verifies the weak solvability condition for the given $p'_{\mathcal{A}}$. If it is not satisfied, it returns an error message to *2DSD.Translate*. Otherwise, *ECLES.Update* computes the new positions $p''_{\mathcal{D}}$ of the derived points satisfying the constraints. Then, *2DSD.Translate* updates the current position p_s of each control point $s \in (\mathcal{A} \cup \mathcal{D})$, and gives that information to

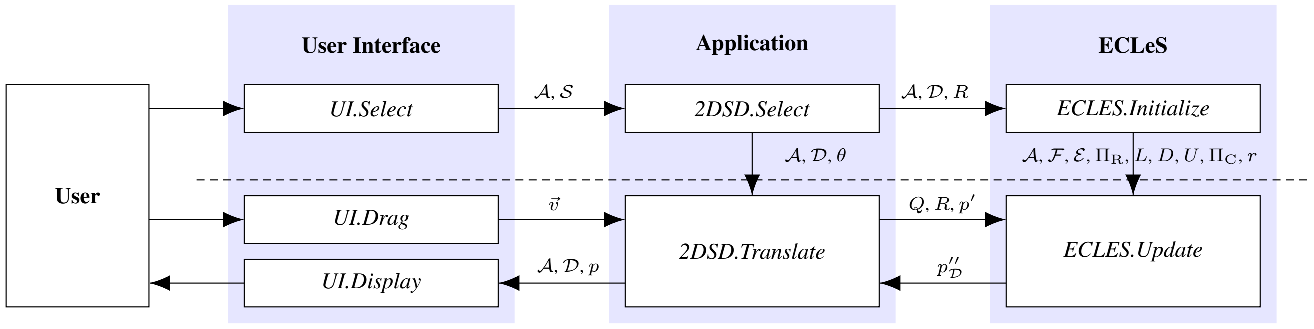

User Interface
Application
ECLeS
User
UI.Select
$\mathcal{A}, \mathcal{S}$
2DSD.Select
$\mathcal{A}, \mathcal{D}, R$
ECLES.Initialize
$\mathcal{A}, \mathcal{D}, \theta$
$\mathcal{A}, \mathcal{F}, \mathcal{E}, \Pi_{\mathrm{R}}, L, D, U, \Pi_{\mathrm{C}}, r$
UI.Drag
$\vec{v}$
2DSD.Translate
Q, R, p'
ECLES.Update
UI.Display
$\mathcal{A}, \mathcal{D}, p$
$p''_{\mathcal{D}}$

Fig. 7. Control flow for a typical editing action (translation of anchor points).

the user interface to display the effects of the change on the deformed mesh and/or the deformed object.

A. *The 2DSD.Select procedure*

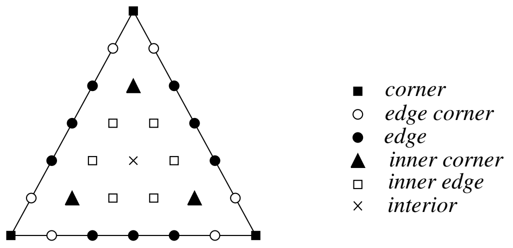
In step 2, the *2DSD.Select* procedure, described in Algorithm 1, expands the initial set $\mathcal{S}$ of derived points to the set $\mathcal{D}$, using *2DSD.ExpandDerivedSet*. Then, in step 3, *2DSD.Select* uses *2DSD.ComputeRelMagnitude* to compute a real coefficient θ_s for each control point p_s, which will be used by *2DSD.Translate* to compute p'_s. Finally, in step 4, *2DSD.Select* calls the *ECLES.Initialize* procedure to identify the relevant and non-redundant C^1 continuity constraints for the editing action, and to obtain a collection of matrices Π_{R}, L, D, U, Π_{C} and the rank r.

Algorithm 1: *2DSD.Select*

```
Data: $\mathcal{A}, \mathcal{S}$: set of anchor and initial derived points;
      $G$: the control graph;
      $R$: the constraint coefficient matrix.
Result: $\mathcal{D}, \mathcal{F}$: set of derived and fixed points;
      $\theta$: relative magnitude to the displacement of point;
      $\Pi_{\mathrm{R}}, L, D, U, \Pi_{\mathrm{C}}, r$: matrices and rank $r$ obtained
      from ECLES.Initialize.
1 begin
2     $\mathcal{D} \leftarrow$ 2DSD.ExpandDerivedSet ($\mathcal{A}$, $\mathcal{S}$, $G$)
3     $\theta \leftarrow$ 2DSD.ComputeRelMagnitude ($\mathcal{A}$, $\mathcal{D}$, $G$)
4     $(\mathcal{F}, \Pi_{\mathrm{R}}, L, D, U, \Pi_{\mathrm{C}}, r) \leftarrow$ ECLES.Initialize
      $(\mathcal{A}, \mathcal{D}, R)$
```

Initially, *2DSD.ExpandDerivedSet* sets $\mathcal{D} \leftarrow \mathcal{S}$. Then, for each $s \in (\mathcal{A} \cup \mathcal{D})$, the algorithm finds all quadrilateral conditions that involve the control point p_s; and then adds the indices of zero or more control points that enter into these conditions to the set $\mathcal{D}$. The process is iterated until all points in $\mathcal{A} \cup \mathcal{D}$ have been examined. 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.

For the purpose of this step, each Bézier control point $p_s = q^u_{ijk}$ is classified into six types according to its nominal position u_{ijk} in the triangle u. See Fig. 8. The type of the point p determines the set of quadrilateral constraints that apply to that point and the additional 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 $\mathcal{D}$. A point p of any other type contributes additional derived points according to rules are shown in Fig. 9.



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 $k \geq 2$.

Fig. 8. Classification of control points of a Bézier patch of degree 6.

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 there is at least one derived point for each quadrilateral involved in the editing action. The set $\mathcal{D}$ obtained ensures the strong solvability condition of the ECLES.

The *2DSD.ComputeRelMagnitude* procedure computes the value θ_s, which defines the relative magnitude of the desired displacement of each point p_s, that is, how much the suggested position p'_s is affected by the specified displacement $\vec{v}$ of the anchor points. The value θ_s is a number between 0 and 1 given by the formula

$$\theta_s = \frac{\delta''_s}{\delta'_s + \delta''_s} \tag{5}$$

where δ'_s is the distance between the point p_s and the nearest anchor point; and δ''_s is the distance between the point p_s and the nearest fixed point. Both δ'_s and δ''_s are graph-theoretical distances measured on the global Bézier control net G. The distances are computed by Dijkstra's algorithm [30]. The value θ_s computed for anchor points is 1, and for fixed points is 0.

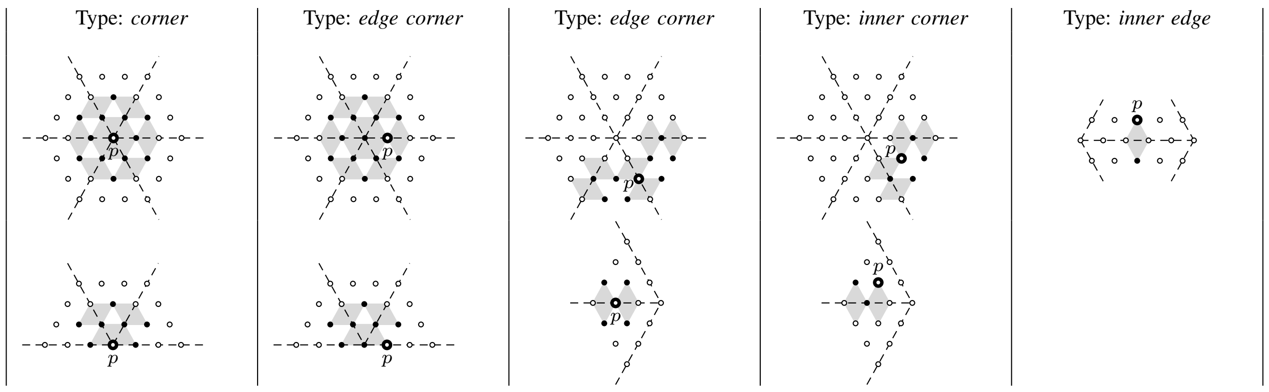


Fig. 9. Relevant constraints (gray diamonds) and derived points added to $\mathcal{D}$ (solid dots) for a control point p_s with $s \in \mathcal{A} \cup \mathcal{D}$ (open dot). For each type, the upper figure shows a typical situation where the point p is sufficiently far from the triangulation's border. The lower figure shows a situation near the border where some of those points and constraints are missing. These diagrams are generalized to vertices of arbitrary degree in the obvious way.

B. The 2DSD.Translate procedure

In step 2, the *2DSD.Translate* procedure, described in Algorithm 2, computes the suggested positions p'_s of each control point using the value θ_s. Then, in step 3, the new positions p''_s of the derived points are computed by *ECLES.Update*. Finally, in steps 4 and 5, *2DSD.Translate* sets the final position p_s of each control point.

Algorithm 2: *2DSD.Translate*

Data: $\vec{v}$: displacement applied to the anchor points;
Q: independent terms of the constraints;
$\mathcal{A}, \mathcal{D}, \mathcal{F}$: set of anchor, derived and fixed points;
θ: relative magnitude to the displacement of points;
$\Pi_{\rm R}$, L, D, U, $\Pi_{\rm C}$, r: matrices and rank r returned by *ECLES.Initialize*;
p: matrix of coordinates of the current positions of the control points.
Result: p: updated matrix of coordinates of the control points.

```
1 begin
2     for each $s \in \mathcal{P}$ do $p'_s \leftarrow p_s + \theta_s \vec{v}$
3     $p''_{\mathcal{D}} \leftarrow$ECLES.Update ($\mathcal{A}$, $\mathcal{F}$, $\mathcal{E}$, $Q$, $R$, $\Pi_{\rm R}$, $L$, $D$, $U$, $\Pi_{\rm C}$, $r$, $p'$)
4     for each $s \in \mathcal{D}$ do $p_s \leftarrow p''_s$
5     for each $s \in \mathcal{A}$ do $p_s \leftarrow p'_s$
```

IV. The ECLES Method

In this section, we describe the ECLES method specialized for editing the Bézier control points of the spline (defined on a mesh T), subjected to the C^1 quadrilateral constraints.

If the coordinates of the vertices of the domain mesh are rational, each constraint can be expressed by a linear equation with integer coefficients. Therefore, we can use the ECLES general method, which consists of the *ECLES.Initialize* and *ECLES.Update* procedures, as part of our algorithm 2DSD for editing of 2D spline deformations, in order to adjust the control points preserving the C^1 continuity of the spline while trying to obey the changes indicated by the user.

A. The ECLES.Initialize procedure

The sets $\mathcal{A}$ and $\mathcal{D}$ of control points and the constraints matrix R are given by *2DSD.Translate* to *ECLES.Initialize*. Based on these informations, the *ECLES.Initialize* procedure identifies the set $\mathcal{E}$ of m relevant constraints, and the set $\mathcal{F}$ of the *relevant fixed points*, that occur in some equation of $\mathcal{E}$ but are not in $\mathcal{A}$ or $\mathcal{D}$. Then, *ECLES.Initialize* extracts a linear system $AP''_{\mathcal{D}} = B$ that express the set $\mathcal{E}$ written in matrix form, namely

$$R_{\mathcal{E}\mathcal{D}} P''_{\mathcal{D}} = Q_{\mathcal{E}} - R_{\mathcal{E}\mathcal{A}} P'_{\mathcal{A}} - R_{\mathcal{E}\mathcal{F}} P'_{\mathcal{F}} \qquad (6)$$

where the matrices $R_{\mathcal{E}\mathcal{D}}$, $R_{\mathcal{E}\mathcal{A}}$ and $R_{\mathcal{E}\mathcal{F}}$ are $m \times n$, $m \times a$ and $m \times f$ matrices with the coefficients of the points $\mathcal{D}$, $\mathcal{A}$ and $\mathcal{F}$ in the equations $\mathcal{E}$, respectively. The $n \times 2$ matrix $P''_{\mathcal{D}}$ consists of the n computed positions p''_s of each derived point. The $m \times 2$ matrix $Q_{\mathcal{E}}$ contains the m elements of the constant matrix Q corresponding to the equations $\mathcal{E}$. The $a \times 2$ matrix $P'_{\mathcal{A}}$ contains the a user-specified positions p'_s of each anchor point; and $P'_{\mathcal{F}}$ is a $f \times 2$ matrix with the f current positions $p'_s = p_s$ of each relevant fixed points.

In general, there may be redundancies in system (6). The *ECLES.Initialize* procedure finds the rank r of the matrix $R_{\mathcal{E}\mathcal{D}}$ and factors the matrix $R_{\mathcal{E}\mathcal{D}}$ into integer matrices $L_{m\times r}, D_{r\times r}, U_{r\times n}$, and permutation matrices $\Pi_{\rm R}$ (rows) and $\Pi_{\rm C}$ (columns), such that $R_{\mathcal{E}\mathcal{D}} = \Pi_{\rm R} L D^{-1} U \Pi_{\rm C}$, using the fraction-free Gaussian LU factoring method [31]. The first r rows of $\Pi_{\rm R}^{-1} R_{\mathcal{E}\mathcal{D}}$ are a set of non-redundant equations.

If the strong solvability condition is required but is not satisfied, then an error flag is returned and the *ECLES.Update* is not called.

B. The ECLES.Update procedure

If the system (6) has redundant equations ($r < m$), the problem is reduced to the non-redundant linear system

$$\hat{A} P''_{\mathcal{D}} = \hat{B} \qquad (7)$$

where $\hat{A}$ is the first r rows of $\Pi_{\rm R}^{-1} A = L D^{-1} U \Pi_{\rm C}$, and $\hat{B}$ is the first r rows of $\Pi_{\rm R}^{-1} B = \Pi_{\rm R}^{-1}(Q_{\mathcal{E}} - R_{\mathcal{E}\mathcal{A}} P'_{\mathcal{A}} - R_{\mathcal{E}\mathcal{F}} P'_{\mathcal{F}})$ [1].

If the system (7) has a single solution, it is solved directly. Otherwise, if it is indeterminate ($r < n$), *ECLES.Update* uses the least squares criterion to minimize the distance between each new position p''_s and the desired position p'_s, while satisfying the non-redundant constraints. That is, the goal function that we want minimize is

$$S(p'') = \sum_{s=1}^{n} |p''_s - p'_s|^2. \tag{8}$$

In order to minimize the goal function (8) while satisfying the constraints (7), it is necessary the gradient of the function be perpendicular to solution of the constraints, that is

$$2P''_{\mathcal{D}} + \hat{A}^{\top}\lambda = 2P'_{\mathcal{D}} \tag{9}$$

where λ is a column vector with the Lagrange multipliers $\lambda_1, ..., \lambda_r$. Equation (9) combined with the constraints (7) yields a least squares linear system which can be solved in two steps; namely, solving

$$\frac{1}{2}\hat{A}\hat{A}^{\top}\lambda = \hat{A}P'_{\mathcal{D}} - \hat{B} \tag{10}$$

for λ, and then computing $P''_{\mathcal{D}}$ by solving

$$2P''_{\mathcal{D}} = 2P'_{\mathcal{D}} - \hat{A}^{\top}\lambda. \tag{11}$$

C. An example of user editing action

Suppose that the set $\mathcal{A}$ is the point $p = q^u_{032}$ between the triangles u (right) and v (left), shown in Fig. 10.

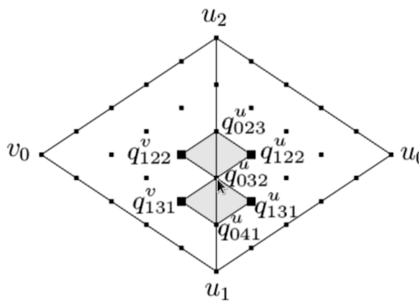


Fig. 10. Editing point q^u_{032} with the derived points q^v_{122}, q^u_{122}, q^v_{131}, q^u_{131}.

According to Fig. 9, the algorithm will select the four derived control points $\mathcal{D} = \{q^v_{122}, q^u_{122}, q^v_{131}, q^u_{131}\}$ (large black dots in Fig. 10). Based on these points and on the anchor point q^u_{032}, *ECLES.Initialize* identifies two relevant constraints, which are included in the set $\mathcal{E}$.

$$-(\beta)q^v_{122} + (\beta_0)q^u_{122} + (\beta_1)q^u_{032} + (\beta_2)q^u_{023} = 0 \tag{12}$$
$$-(\beta)q^v_{131} + (\beta_0)q^u_{131} + (\beta_1)q^u_{041} + (\beta_2)q^u_{032} = 0 \tag{13}$$

where $\beta = \beta_0 + \beta_1 + \beta_2$, and β_0, β_1, and β_2 are the barycentric coordinates of v_0 relative to u_0, u_1, and u_2. The set $\mathcal{F}$ has the two points $q^u_{023} = q^v_{023}$ and $q^u_{041} = q^v_{041}$.

The matrix form of equations (12) and (13) is

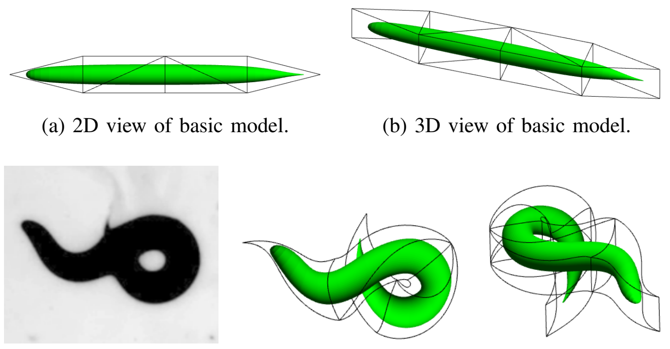
$$\begin{bmatrix} -\beta & 0 & \beta_0 & 0 \\ 0 & -\beta & 0 & \beta_0 \end{bmatrix} P''_{\mathcal{D}} = -\begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} P'_{\mathcal{A}} - \begin{bmatrix} \beta_2 & 0 \\ 0 & \beta_1 \end{bmatrix} P'_{\mathcal{F}} \tag{14}$$

where the constant matrix $Q_{\mathcal{E}}$ is zero.

In this example, the equations are linearly independent and the system is indeterminate. Therefore, the new positions $P''_{\mathcal{D}}$ are computed by *ECLES.Update* which solves the system (14) using the least squares criterion.

V. Application: 2.5D Space Deformations

We have used the 2DSD deformation algorithm, described in Section III, to improve the editor of C^1-continuous 2.5D space deformations, described by Rodrigues et al [7]. This editor was used to deform 3D models of cells and other organisms to match them with images of real specimens viewed through an optical microscope. A typical result is shown in Fig. 12.

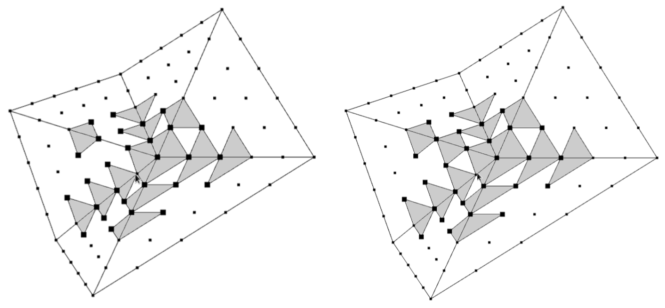


(a) 2D view of basic model. (b) 3D view of basic model.

(c) An actual image [32]. (d) 2D view of result. (e) 3D view of result.

Fig. 11. Prismatic control mesh and 3D model of the nematode *Caenorhabditis elegans* (provided by Rodrigues et al. [7]) deformed by our editor.

The improvements include the use of the ECLES method and the extension of editing actions. The use of ECLES removed the rounding errors and numerical instabilities of the previous floating-point implementation. See Fig. 12.



(a) Floating-point version. (b) ECLES version.

Fig. 12. Example where a floating-point version of 2DSD (a) eliminate one constraint that is not redundant and is preserved by ECLES version (b).

The use of ECLES also allows more general constraints. As observed in Section I-A the enhancements afforded by 2DSD included additional editing operations (such as local rotation and scaling), multiple anchor points, semiautomatic selection of derived points (explicitly by distance and implicitly by the solvability condition) and an improved goal function for the least squares.

In this application, we assume that the embedded model is given as a dense triangular mesh with tens of thousands of triangles. Although the models for this application are three-dimensional, the deformations are essentially two-dimensional with little change in depth, because the third dimension cannot 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. 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 II.

Our algorithm 2DSD is used in the editor's "XY mode", to edit the ϕ mapping, and in the "Z^0 mode" and "Z^1 mode", to edit the functions σ_0 and σ_1.

VI. Conclusion

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, the 2DSD algorithm, supports splines of degree 5 or higher and allows convenient editing of the deformation while preserving the C^1 continuity of the surface. The main contribution of this paper is the use of the integer-based ECLES general method to combine the user editing actions and the continuity constraints in a reliable and efficient way, avoiding the fatal failures that could arise from floating-point rounding errors. The 2DSD algorithm 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.

Acknowledgment

This work is supported by Brazilian government grants CNPq 140780/2013-0 and 310706/2015-7 (NV).

References

[1] E. C. S. Rodrigues and J. Stolfi, "ECLeS: A Flexible and General Method for Local Editing of Parameters with Linear Constraints," in *SIBGRAPI '15: Proc. of Workshop of Works in Progress*, 2015, pp. 1–4.

[2] J. Gain and D. Bechmann, "A survey of spatial deformation from a user-centered perspective," *ACM Trans. Graph.*, vol. 27, no. 4, pp. 1–21, 2008.

[3] G. Wolberg, "Image morphing: a survey," *The Visual Computer*, vol. 14, no. 8-9, pp. 360–372, 1998.

[4] M. B. Islam, M. T. Inam, and B. Kaliyaperumal, "Overview and challenges of different image morphing algorithms," *International Journal of Advanced Research in Computer Science and Electronics Engineering (IJARCSEE)*, vol. 2, no. 4, 2013.

[5] B. Zitová and J. Flusser, "Image registration methods: a survey," *Image and Vision Computing*, vol. 21, no. 11, pp. 977 – 1000, 2003.

[6] A. Sotiras, C. Davatzikos, and N. Paragios, "Deformable medical image registration: A survey," *Medical Imaging, IEEE Transactions on*, vol. 32, no. 7, pp. 1153–1190, 2013.

[7] E. C. S. Rodrigues, A. Gomide, and J. Stolfi, "A user-editable C^1-continuous 2.5D space deformation method for 3D models," *Electronic Notes in Theoretical Comp. Sci.*, vol. 281, no. 0, pp. 159 – 173, 2011.

[8] M. S. Floater, "Mean value coordinates," *Comput. Aided Geom. Des.*, vol. 20, pp. 19–27, March 2003.

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

[10] T. Langer, A. Belyaev, and H.-P. Seidel, "Spherical barycentric coordinates," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, ser. SGP '06. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 81–88.

[11] P. Joshi, M. Meyer, T. DeRose, B. Green, and T. Sanocki, "Harmonic coordinates for character articulation," *ACM Trans. Graph.*, vol. 26, July 2007.

[12] Y. Lipman, D. Levin, and D. Cohen-Or, "Green coordinates," *ACM Trans. Graph.*, vol. 27, pp. 78:1–78:10, August 2008.

[13] A. de Boer, M. van der Schoot, and H. Bijl, "Mesh deformation based on radial basis function interpolation," *Computers & Structures*, vol. 85, no. 11–14, pp. 784–795, 2007, fourth MIT Conference on Computational Fluid and Solid Mechanics.

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

[15] A. H. Barr, "Global and local deformations of solid primitives," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 21–30, Jan. 1984.

[16] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *SIGGRAPH Comput. Graph.*, vol. 20, no. 4, pp. 151–160, 1986.

[17] S. 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*. New York, NY, USA: ACM, 1990, pp. 187–196.

[18] R. MacCracken 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*. New York, NY, USA: ACM, 1996, pp. 181–188.

[19] H. J. Lamousin and W. N. Waggenspack Jr., "NURBS-based free-form deformations," *IEEE Comput. Graph. Appl.*, vol. 14, no. 6, pp. 59–65, 1994.

[20] 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*. New York, NY, USA: Elsevier Science Inc., 1997, pp. 1715–1725.

[21] J. Huang, 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*. New York, NY, USA: ACM, 2008, pp. 241–247.

[22] Y. Zhao, X. Liu, C. Xiao, and Q. Peng, "A unified shape editing framework based on tetrahedral control mesh," *Comput. Animat. Virtual Worlds*, vol. 20, no. 2-3, pp. 301–310, 2009.

[23] T. Nishita, T. Fujii, and E. Nakamae, "Metamorphosis using bézier clipping," 1993.

[24] S.-Y. Lee, K.-Y. Chwa, and S. Y. Shin, "Image metamorphosis using snakes and free-form deformations," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '95. New York, NY, USA: ACM, 1995, pp. 439–448.

[25] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. G. Hill, M. O. Leach, and D. Hawkes, "Nonrigid registration using free-form deformations: application to breast mr images," *Medical Imaging, IEEE Transactions on*, vol. 18, no. 8, pp. 712–721, 1999.

[26] T. Xia, B. Liao, and Y. Yu, "Patch-based image vectorization with automatic curvilinear feature alignment," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 115:1–115:10, Dec. 2009.

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

[28] S. Hahmann and G.-P. Bonneau, "Polynomial surfaces interpolating arbitrary triangulations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 1, pp. 99–109, Jan. 2003.

[29] D. 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.

[30] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[31] D. J. Jeffrey, "LU Factoring of Non-invertible Matrices," *ACM Commun. Comput. Algebra*, vol. 44, no. 1/2, pp. 1–8, Jul. 2010.

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