Skinning Cubic Bézier Splines and Catmull-Clark Subdivision Surfaces

Songrun Liu* George Mason University
Alec Jacobson† Columbia University
Yotam Gingold‡ George Mason University

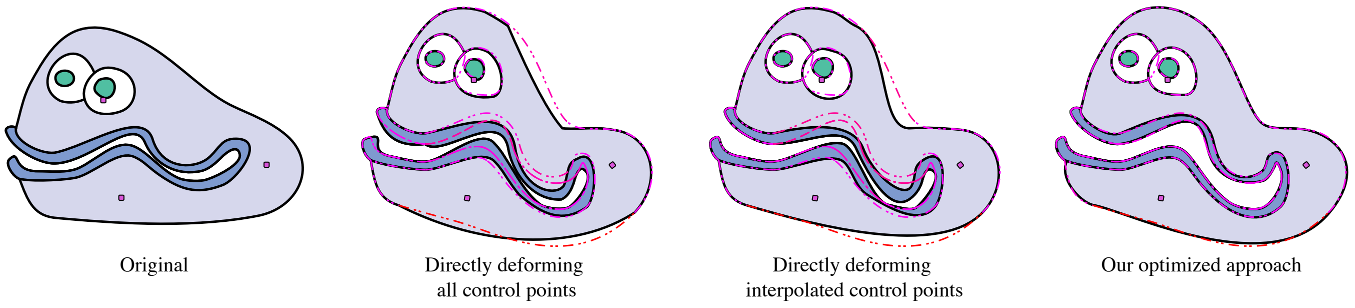

Original
Directly deforming all control points
Directly deforming interpolated control points
Our optimized approach

Figure 1: *Splines stay splines. Our efficient solution for linear blend skinning of splines deforms the* Clam *without destroying its Bézier curve representation. Our approach matches the target deformation (dashed line) better than naive approaches. Here, skinning weights are Bounded Biharmonic Weights [Jacobson et al. 2011].*

Abstract

Smooth space deformation has become a vital tool for the animation and design of 2D and 3D shapes. Linear methods, under the umbrella term of "linear blend skinning", are the *de facto* standard for 3D animations. Unfortunately such approaches do not trivially extend to deforming *vector graphics*, such as the cubic Bézier splines prevalent in 2D or subdivision surfaces in 3D. We propose a variational approach to reposition the control points of cubic Bézier splines and Catmull-Clark subdivision surfaces—or any linear subdivision curves or surfaces—to produce curves or surfaces which match a linear blend skinning deformation as closely as possible. Exploiting the linearity of linear blend skinning, we show how this optimization collapses neatly into the repeated multiplication of a matrix per handle. We support C^0, C^1, G^1, and fixed-angle continuity constraints between adjacent Bézier curves in a spline. Complexity scales linearly with respect to the number of input curves and run-time performance is fast enough for real-time editing and animation of high-resolution shapes.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling —Splines

Keywords: spline, subdivision, linear blend skinning, skeletal shape deformation, image warping, vector graphics

Links: DL PDF

*e-mail:sliu11@gmu.edu
†jacobson@cs.columbia.edu
‡ygingold@gmu.edu

1 Introduction

Smooth shape deformation brings animated characters to life and facilitates virtual design and sculpture. Due to their simplicity, predictability, and real-time performance, linear methods are the *de facto* standard for 3D animations. These methods, under the umbrella term of "linear blend skinning," describe the deformation function at each point $\mathbf{p}$ on the shape as a weighted combination of a small number of affine transformations:

$$\mathbf{p}' = s_{w,T}(\mathbf{p}) = \sum_{i=1}^{h} w_i(\mathbf{p}) T_i \mathbf{p} \tag{1}$$

where T_i is the i-th transformation, $w_i(\mathbf{p})$ is its associated weight function evaluated at $\mathbf{p}$, and $\mathbf{p}'$ is the new position of $\mathbf{p}$. Recently, a surge of works have perfected the automatic computation of such weight functions [Baran and Popović 2007; Joshi et al. 2007; Lipman et al. 2008; Jacobson et al. 2011] and assisted with the specification of affine transformations [Der et al. 2006; Weber et al. 2007]. Though most of these 3D techniques generalize to 2D—and some are specialized for 2D [Weber et al. 2009; Weber and Gotsman 2010; Weber et al. 2011]—their success and impact has not permeated 2D animation and design as thoroughly. In 2D, illustrations are typically defined in terms of piecewise cubic Bézier curves (e.g. `PDF`'s, modern drawing API's, and fonts).

Space deformations, a general class of deformations containing linear blend skinning, are pointwise functions mapping a region of space to new locations. They lend themselves nicely to piecewise linear mesh deformation or *raster* image deformation, where pixels in the output image are computed via a texture on a deformed mesh [Igarashi et al. 2005; Schaefer et al. 2006; Jacobson et al. 2011] or inversely as the result of warping the texture coordinate space [Beier and Neely 1992; Hormann and Floater 2006; Adobe Systems Inc. 2014; Schaefer et al. 2006]. Unfortunately such approaches do not trivially extend to deforming *vector graphics*.

The core problem is that when applying the skinning formula in Equation (1) to an e.g. Bézier curve, the deformed result is, in general, no longer a Bézier curve. This is only acceptable if skinning is the last operation just before rendering. Otherwise it hinders the application of skinning techniques to design and other editing contexts. Artists and designers precisely place control points for

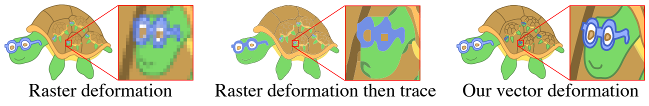
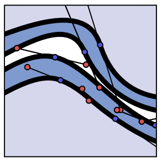


Figure 2: *The* Turtles *deformed using linear blend skinning as a raster image have the obvious pitfall of fixed resolution. Tracing the deformed raster image back into a vector graphic performs poorly due to raster ambiguities and tracing parameters. Our method deforms vector graphics directly and the* Turtles *stay crisp.*

their splines and subdivision surfaces and expect to continue editing them.



Naive solutions to skinning have drastic failure modes (Section 5). Deforming the control points of a spline or subdivision surface as if they were ordinary points in space is problematic. Applying the space deformation to the control structure is not the same as applying it to the shape (see Figure 1 or 8). Non-interpolated control points may lie far from the curve or surface itself and thus receive a drastically different weight function (close-up of the clam mouth, inset). More generally, any (naive) approach that evaluates the deformation function only at control points is unable to fully capture variations within a single curve or surface patch (Figure 4). Our approach evaluates the deformation function along the entire curve or surface, and does not suffer from such problems (see Figures 1 and 7; see Section 5 for details). Alternatively, one could sample and retrace a deformed curve (e.g. with [Schneider 1990] or [Noris et al. 2013]), but this leads to the control points swimming, jumping, or changing quantity, in addition to unwanted parameters and related ambiguities (see Figures 2 and 3). This destroys the editability of the input.

Instead, we propose a variational approach that repositions the control points of an input vector graphics shape (curves or surface) to match a given skinning deformation applied to the input shape as closely as possible. Exploiting the linearity of Equation (1), we show how this optimization collapses neatly into a single matrix multiplication per weight function. We demonstrate our approach on cubic Bézier splines and Catmull-Clark subdivision surfaces, though our approach generalizes to any spline or linear subdivision curve or surface. Complexity scales linearly with respect to the number of control points and run-time performance supports interactive editing and animation of high-resolution shapes composed of up to 10,000 Bézier curves.

Powerful vector graphics editors such as ILLUSTRATOR and

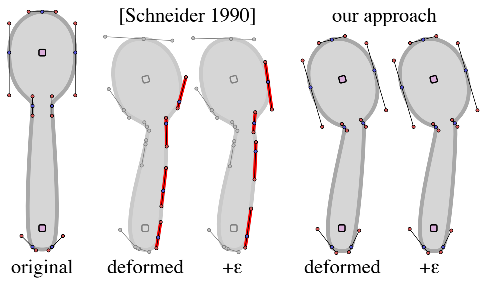


Figure 3: *A Bézier spline spoon bent using Equation (1), and then infinitesimally bent again. Sampling and refitting a new Bézier spline leads to swimming and jumping artifacts (highlighted in red). Our approach preserves the editability of the original artwork. See also accompanying video.*

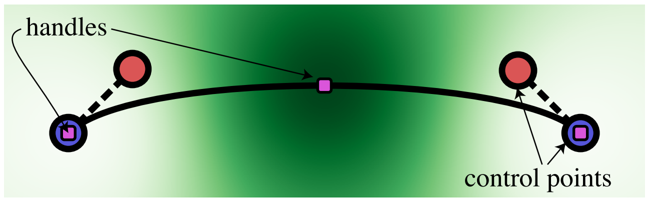


Figure 4: *The influence of a handle (purple) placed at the midpoint of a cubic Bézier curve may not extend to any of its control points. The midpoint handle's influence is shown in green. Naive approaches that only evaluate the deformation at the curve's control points cannot fully capture the influence of such handles.*

INKSCAPE support different levels of continuity at curve junctures along cubic Bézier splines. We complement this by introducing optimization constraints to preserve C^0, C^1, and G^1 continuity. As Bézier splines are C^∞ everywhere else, our output is thus guaranteed to be at least G^1. We support similar constraints to maintain the angle at sharp features (such as 90° corners). In addition to *preserving* the existing G^1 continuity of the input, we also support G^1 *flexibility*, which does not preserve the ratio of tangent magnitudes on either side of a curve juncture. We enable this by enforcing nonlinear constraints via a simple and efficient solving routine. This extra flexibility is optional, however, as it requires a few iterations.

By optimizing the sparse set of control points visible to—and in many cases created by—designers, our output stays just as sparse and editable. We evaluate our algorithm on a wide range of inputs and application scenarios. We animate cartoon characters, warp graphical designs, and apply fine-grained edits to text. We highlight how our approach generalizes to a wide variety of deformation techniques and complements a range of popular weighting functions.

2 Related Work

Bézier curve and subdivision surface fitting is nearly as old and well studied as Bézier curves and subdivision surfaces in general. Therefore, we focus on works closely related to our formulation and place our work in context with other shape deformation systems.

Schneider [1990] provided a thorough and still-used G^1-continuous solution to the general problem of fitting a cubic Bézier spline to a digitized curve. More recently, Frisken [2008] introduced an approach suitable for fitting a G^1-continuous spline to a live data source such as handwriting. In contrast to these and other approaches for curve and surface fitting [Schmitt et al. 1986; Plass and Stone 1983; Krishnamurthy and Levoy 1996; Wang et al. 2006], our input is a known Bézier spline or subdivision surface undergoing a linear blend skinning deformation. We exploit the additional structure in our setting to (a) preserve the topology of the typically hand-designed shape and (b) create a more efficient solution.

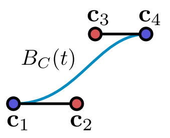
Also related to our problem are approaches for editing splines beyond manually moving control points. Fowler and Bartels [1993] use a similar energy formulation and introduce techniques for on-curve constraints, allowing for local and direct manipulation of individual curves. These constraints may be extended to deal with more general constraints such as high level deformation operations [Zheng et al. 1998] and *over-sketching* [Olsen et al. 2009]. Zhou et al. [2007] presented a technique for the direct manipulation of subdivision surfaces by minimizing the Laplacian Editing Energy. Our approach is a general technique supporting any linear blend skinning deformation.

Recently, Liao et al. [2012] proposed an alternative editing paradigm for vector graphics. By contrast, our solution enhances into the industry standard editing pipeline, readily taking standard vector graphics as input and producing standard vector graphics as output for further processing.

The reverse problem of fitting a skinning deformation to an existing, possibly smooth, animation has been well studied since James and Twigg [2005]. These formulations use similar energies to ours, but are often augmented with regularization terms for ensuring smoothness [Kavan et al. 2011]. With appropriate continuity constraints and by working in the space of smooth Bézier curves, we ensure smoothness by construction.

We focus on deformations created by linear blend skinning. Written as in Equation (1), skinning is quite general, also encompassing free-form deformation [Milliron et al. 2002], coordinate-based cage deformations [Joshi et al. 2007], quadratic energy-based modeling [Botsch and Kobbelt 2004], model-reduced non-linear variational modeling and physically-based simulations [Hildebrandt et al. 2011; Barbič et al. 2012; Jacobson et al. 2012]. Our system bootstraps any linear skinning deformation, regardless of whether the user is explicitly painting the weights and choosing the transformations. Therefore, we bring all of these works to the domain of vector graphics.

3 Derivation



Our input is a parametric vector graphics shape $G_C(p)$ and a linear blend skinning deformation function $s_{w,T}(\mathbf{p}) : \mathbb{R}^d \to \mathbb{R}^d$, as in Equation (1). The linear blend skinning deformation function $s_{w,T}$ takes the form of a set of weight functions, painted manually or computed using any technique in the literature, and a corresponding set of linear transformations. $G_C(p)$ could be a 3D Catmull-Clark subdivision surface (with $p = (\textit{face index}, u, v)$), or $G_C(p)$ could be a 2D Bézier curve $B_C(t)\colon [0,1] \to \mathbb{R}^2$. C is the matrix whose columns are the control points; for a cubic Bézier curve in 2D, $C = [\mathbf{c}_1\ \mathbf{c}_2\ \mathbf{c}_3\ \mathbf{c}_4] \in \mathbb{R}^{3\times4}$ (the third dimension is always 1 for homogeneous transformations).

A cubic Bézier curve can be expressed as the product of three matrices:

$$B_C(t) = CM\bar{t} = C \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} t^3 \\ t^2 \\ t \\ 1 \end{bmatrix} = Cm_t, \quad (2)$$

where m_t is a column vector. Other types of splines have different M matrices. Similarly, a linear subdivision surface's G_C evaluated at a point with parameterization p can be expressed as $G_C(p) = Cm_p$, where m_p is the sparse column vector of control point weights defined by the specific subdivision scheme.

In general, a deformed spline curve or subdivision surface $s_{w,T}(G_C(p))$ will not be exactly representable by any spline curve or subdivision surface, respectively. Thus, we find the *closest fitting* shape $G_{C'}(p)$ with control point matrix C'.

3.1 Energy

Treating the control point matrix C' of our fit shape as unknown, we cast our problem as minimization of the energy

$$E(C') = \int_D \left\| G_{C'}(p) - \sum_{i=1}^{h} w_i(G_C(p)) T_i G_C(p) \right\|^2 \mathrm{d}p, \quad (3)$$

where $E(C')$ measures the integrated distance between the deformed shape and the shape generated with so far unknown control points C', over the entire parameter space D.

Since $G_{C'}(p)$ is linear in C', our energy is quadratic in C'. We set its derivative with respect to C' equal to 0 to find the unique minimizer. For brevity, let $s(t) = s_{w,T}(G_C(p))$ be the original shape deformed by the given skinning deformation function. By the linearity of integration we have

$$\begin{aligned} 0 = \frac{\partial E(C')}{\partial C'} &= \frac{\partial}{\partial C'} \int_D \|C'm_p - s(t)\|^2 \,\mathrm{d}p, \\ &= \int_D \frac{\partial}{\partial C'} (C'm_p - s(t))^\mathsf{T} (C'm_p - s(t)) \,\mathrm{d}p, \\ &= 2C' \int_D m_p m_p^\mathsf{T} \,\mathrm{d}p - 2 \int_D s(t) m_p^\mathsf{T} \,\mathrm{d}p, \end{aligned}$$

and, rearranging terms and letting $A(p) = m_p m_p^\mathsf{T}$,

$$C' \int_D A(p) \,\mathrm{d}p = \sum_{i=1}^{h} T_i C \int_D w_i(Cm_p) A(p) \,\mathrm{d}p.$$

We can compute the matrix $\hat{A} = \int_D A(p)\,\mathrm{d}p$ and the matrices $\hat{W}_i = C \int_D w_i(Cm_p)A(p)\,\mathrm{d}p$, *independently of the actual transformations* T_i. The solution to our energy minimization problem, the control points C' of the best fitting shape, may then be found at the cost of matrix multiplication:

$$C' = \sum_{i=1}^{h} T_i \hat{W}_i \hat{A}^{-1} \quad (4)$$

The $\hat{W}_i$ matrices have the same dimensions as C. $\hat{A}$ is an $N \times N$ symmetric matrix, where N is the number of control points. For a single cubic Bézier curve, its inverse is:

$$\hat{A}^{-1} = \begin{bmatrix} 16 & -24 & 16 & -4. \\ -24 & 69\frac{1}{3} & -57\frac{1}{3} & 16 \\ 16 & -57\frac{1}{3} & 69\frac{1}{3} & -24 \\ -4 & 16 & -24 & 16 \end{bmatrix}$$

The $\hat{W}_i$ matrices—and indeed even $\hat{W}_i\hat{A}^{-1}$—can be precomputed once as soon as the input shape and skinning weights are defined.[1] Since only the skinning transformations T_i are manipulated interactively, this is the key result that allows for efficiently skinning vector graphics. This exact formulation is suitable for subdivision surfaces or B-splines, where the m_p matrices themselves ensure continuity or other desirable features.

4 Cubic Bézier Spline Constraints

Cubic Bézier splines, the spline format most commonly used in 2D design, are not naturally continuous. However, due to the ubiquity of cubic Bézier splines, we support a variety of continuity constraints at the junctions between individual curves (see Figure 5). Basic continuity conditions manifest as linear constraints in our energy minimization problem and so an equally efficient solution. In the following, let $C = \begin{bmatrix}\mathbf{c}_1 & \mathbf{c}_2 & \mathbf{c}_3 & \mathbf{c}_4\end{bmatrix}$ and $D = \begin{bmatrix}\mathbf{d}_1 & \mathbf{d}_2 & \mathbf{d}_3 & \mathbf{d}_4\end{bmatrix}$ be the control points of two cubic Bézier curves sharing an endpoint, such that $B_C(1) = B_D(0)$, and let C' and D' be the control points of the fitted curves.

Continuity The linear constraint $\mathbf{c}'_4 = \mathbf{d}'_1$ enforces C^0 continuity.

[1]If the W_i are integrated numerically, then $\hat{A}$ should also be to avoid artifacts resulting from numerical inconsistencies.

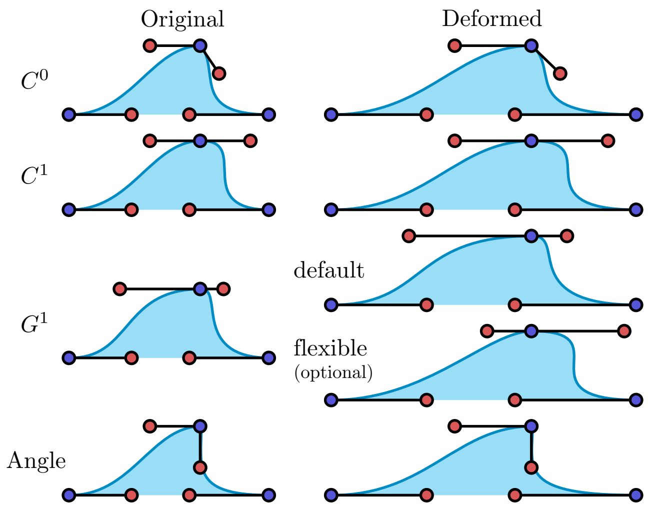


Figure 5: *On import, curves are analyzed for continuity at shared endpoints (left). Continuity is maintained during deformation via appropriate constraints (right).*

Continuous derivatives C^1 continuity requires that $dB_C(1)/dt = dB_D(0)/dt$ in addition to C^0 continuity. We enforce this via the linear constraint $\mathbf{c}'_4 - \mathbf{c}'_3 = \mathbf{d}'_2 - \mathbf{d}'_1$.

Continuous tangents G^1 continuity is a generalization of C^1 continuity [DeRose 1985]. The derivatives at $B_{C'}(1)$ and $B_{D'}(0)$ must point in the same direction, but there need not be any relationship between their magnitudes. In practice, the control points of the undeformed curves are often such that $\mathbf{c}_4 - \mathbf{c}_3$ and $\mathbf{d}_2 - \mathbf{d}_1$ are parallel but have differing magnitudes. We can preserve this pre-existing G^1 continuity by preserving the undeformed tangent magnitude ratio, implemented as the (still linear) constraint

$$\mathbf{c}'_4 - \mathbf{c}'_3 = (\mathbf{d}'_2 - \mathbf{d}'_1)\frac{\|\mathbf{c}_4 - \mathbf{c}_3\|}{\|\mathbf{d}_2 - \mathbf{d}_1\|}. \tag{5}$$

Complete G^1 flexibility can be expressed as a constraint enforcing the dot product of the curves' tangents to be one:

$$\frac{(\mathbf{c}'_4 - \mathbf{c}'_3)}{\|\mathbf{c}'_4 - \mathbf{c}'_3\|} \cdot \frac{(\mathbf{d}'_2 - \mathbf{d}'_1)}{\|\mathbf{d}'_2 - \mathbf{d}'_1\|} = 1. \tag{6}$$

This constraint is nonlinear. In our optimization, we rewrite this single nonlinear constraint in two different (linear) forms, and alternate enforcement of one and then the other. The first form is Equation 5, using the most recently computed values of $\mathbf{c}'_4, \mathbf{c}'_3, \mathbf{d}'_2$, and $\mathbf{d}'_1$ for the ratio term. This allows the tangent directions to change, but not the ratio of tangent magnitudes. The second form fixes the direction of the tangent while allowing the magnitudes to change. We substitute

$$\mathbf{c}'_3 = \mathbf{c}'_4 + \alpha\mathbf{u}, \tag{7}$$

$$\mathbf{d}'_2 = \mathbf{d}'_1 + \beta\mathbf{v}, \tag{8}$$

where $\mathbf{u}$ and $\mathbf{v}$ are the tangent direction

$$\mathbf{v} = \frac{\mathbf{d}'_2 - \mathbf{d}'_1}{\|\mathbf{d}'_2 - \mathbf{d}'_1\|}, \tag{9}$$

$$\mathbf{u} = -\mathbf{v}, \tag{10}$$

and α and β are new degrees of freedom subject to

$$\alpha > 0, \tag{11}$$

$$\beta > 0. \tag{12}$$

By fixing $\mathbf{v}$ and $\mathbf{u}$ to the most recently computed tangent direction, we are left with linear equality constraints on the remaining variables and constant bound constraints on α and β.

Preserving angles We also support constraints that maintain a fixed angle θ between $B_{C'}(1)$ and $B_{D'}(0)$. Let R be the 2D rotation matrix that rotates $\mathbf{c}_3 - \mathbf{c}_4$ by θ to $\mathbf{d}_2 - \mathbf{d}_1$. Then the linear constraint is (cf. Equation 5):

$$R(\mathbf{c}'_3 - \mathbf{c}'_4) = (\mathbf{d}'_2 - \mathbf{d}'_1)\frac{\|\mathbf{c}_4 - \mathbf{c}_3\|}{\|\mathbf{d}_2 - \mathbf{d}_1\|}, \tag{13}$$

Just as with G^1 continuity, this preserves the pre-existing ratio of tangent magnitudes. We can similarly relax this ratio at the cost of nonlinear constraints. The first form of the G^1 constraints is replaced with Equation 13, and the second form is modified such that $R\mathbf{u} = \mathbf{v}$.

4.1 Implementation

Putting all constraints together, our complete Bézier spline energy optimization problem for fixed tangent magnitude ratios is:

$$\begin{array}{lll}
\underset{C'}{\text{argmin}} & \sum_{j=1}^{n} \ell_j E(C'_j) & \\
\text{s.t.:} & \mathbf{c}'_4 = \mathbf{d}'_1, & \forall\,(B_C, B_D)\; C^0, \\
& \mathbf{c}'_4 - \mathbf{c}'_3 = (\mathbf{d}'_2 - \mathbf{d}'_1), & \forall\,(B_C, B_D)\; C^1 \\
& \mathbf{c}'_4 - \mathbf{c}'_3 = (\mathbf{d}'_2 - \mathbf{d}'_1)\frac{\|\mathbf{c}_4 - \mathbf{c}_3\|}{\|\mathbf{d}_2 - \mathbf{d}_1\|}, & \forall\,(B_C, B_D)\; G^1 \\
& R(\mathbf{c}'_4 - \mathbf{c}'_3) = (\mathbf{d}'_2 - \mathbf{d}'_1)\frac{\|\mathbf{c}_4 - \mathbf{c}_3\|}{\|\mathbf{d}_2 - \mathbf{d}_1\|}, & \forall\,(B_C, B_D)\; \text{Angle.}
\end{array}$$

When solving for multiple cubic Bézier curves simultaneously, we weight the energy term for each curve C_j according to its undeformed length ℓ_j. We enforce these linear equality constraints via the Lagrange multiplier method. The upper left (non-Lagrange multiplier) portion of the system matrix Q is block diagonal, composed of 4×4 blocks equal to the $\hat{A}$ matrix from 4. Due to the Angle constraints, which involve the x and y dimensions simultaneously, we solve for all dimensions of C' at once as a vectorized (reshaped) column vector. In order to still compute C' via simple matrix multiplication as handle transformations change, we perform the following precomputation. As soon as the curves and their constraints are known, we calculate the Cholesky factorization of Q and solve for the matrices $O_i = Q^{-1}\begin{bmatrix} I_{3\times3} \otimes \hat{W}_i^{\mathsf{T}} \\ 0 \end{bmatrix}$, where $\otimes$ is the Kronecker product. C' (and the Lagrange multipliers λ) are obtained at runtime with the expression: $\begin{bmatrix} \text{vec}(C') \\ \lambda \end{bmatrix} = \sum_{i=1}^{h} O_i \text{vec}(T_i)$.

To allow for flexible tangent magnitude ratios, we alternate the above *even* iterations with *odd* iterations in which we replace the G^1 and angle constraints with their second forms. By making appropriate substitutions, we enforce the linear equality constraints for the odd iterations also via the Langrange multiplier method. Changing the fixed magnitude ratios in the odd iterations and the fixed directions in the even iterations affects a small number of entries in the system matrix for each constraint. Thus, the odd and even systems must be refactored anew for every iteration, though the symbolic factorization could be reused.[2] In this case, the O_i

[2]Several techniques exist for quickly updating the inverse of a matrix after a low-rank update [Hager 1989; Davis and Hager 1999]; we did not use them.

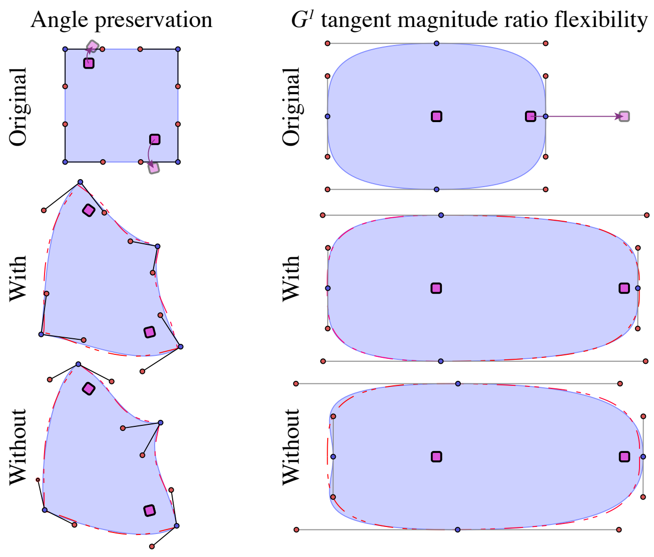


Figure 6: *Left column: A box with two handles imposing rotations. The solution with angle-preservation maintains right angles; the solution without does not. Right column: An undeformed shape whose right side is stretched. The solution with G^1 tangent magnitude ratio flexibility closely matches the target deformation by adjusting tangent magnitudes on the right side of the shape; the solution without such flexibility must scale tangent magnitudes uniformly, leading to buckling on the left side.*

precomputation is unnecessary. By default, we preserve tangent magnitude ratios during live manipulation and relax them when the mouse button is released.

Note that one could use a quadratic programming solver to force α and β to be positive, but this rarely occurs in practice and we simply clamp them. We have not found this to be a problem. We have not experienced problems with convergence, because each iteration will not increase the energy. Figures 6 shows the effect of angle preservation (left column) and G^1 tangent magnitude ratio flexibility (right column). Angle preservation is a feature unique to our construction and difficult to achieve in raster constructions. G^1 tangent magnitude ratio flexibility allows deformations that are asymmetric around control points more degrees of freedom for optimization, manifesting in lower energy and a better fitting solution.

5 Evaluation

We evaluate our approach with cubic Bézier splines and Catmull-Clark subdivision surfaces. We use the "unconstrained" formulation of our energy (Section 3.1) for Catmull-Clark surfaces (implemented in C++), and the constrained formulation (Section 4) for cubic Bézier splines (implemented in Python).

We support loading arbitrary SVG (Scalable Vector Graphics) files, which natively support cubic Bézier splines. Upon loading, we convert straight lines and circular arcs to cubic Bézier curves and analyze the junctions between curves to determine the continuity constraint to impose (C^0, C^1, G^1, or fixed 90° angle). When creating splines in ILLUSTRATOR, smooth junctions are always C^1 upon creation, but relaxed to G^1 when editing. We mirror this concept. Treating our operation as an edit, we default to G^1 constraints along smooth input splines.

In 2D, to obtain a mesh for computing Harmonic Coordinates [Joshi et al. 2007] or Bounded Biharmonic Weights [Jacobson et al. 2011]

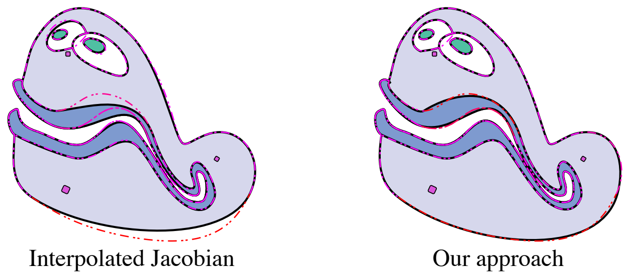


Figure 7: *The* Clam *from Figure 1 deformed using linear blend skinning with a more sophisticated "naive" Jacobian-based approach (left) versus our approach (right). In this Interpolated Jacobian approach, the linear blend skinning deformation is directly applied to the Bézier splines' interpolated control points. Each off-curve control point is transformed by the Jacobian of the deformation at its adjacent interpolated control point. This example uses Shepard weights.*

(or any other automatic weights requiring spatial discretization), we triangulate the user-defined cage or the largest closed curve (or a manually created union curve in Adobe Illustrator if no such curve exists) using Triangle [Shewchuk 1996]. We sample every curve uniformly 100 times or once per pixel, whichever is sparser, and submit the positions as additional meshing constraints for the triangulation. In 3D, to obtain a Bounded Biharmonic Weights, we used OpenSubdiv [Pixar 2014] to generate a refined mesh, generalized winding numbers [Jacobson et al. 2013] to compute an intersection-free bounding volume, and TetGen [Si 2003]. In 2D and 3D, our examples using Shepard weights compute the weights analytically during numerical integration of $\hat{W}$ (Equation 4). We compute Catmull-Clark subdivision matrices using OpenSubdiv.

In our figures, when dashed target curves are visible, they are color-coded to display normalized energy along the curve, with red as the maximum and purple as the minimum.

Comparisons to Naive Approaches We compare our approach to four "naive" alternatives. In the first, we apply the deformation directly to all control points (Figure 1 for Bézier splines and Figure 8 for Catmull-Clark subdivision surfaces). In the second and third, we apply the deformation directly to the interpolated control points and indirectly to the off-curve points by treating each off-curve point as a vector whose origin is the adjacent interpolated point. (Note that this is only possible for Bézier or Hermite splines.) For the second approach, we apply the interpolated point's transformation (Figure 1, interpolated control points). For the third approach, we apply the Jacobian of the interpolated point's transformation (Figure 7). This Jacobian-based approach requires computation of the weight function derivative, which cannot be analytically computed for manually painted weights or many commonly used automatic weight functions. (Our example uses Shepard weights.) Unlike our approach, the naive approaches do not evaluate the deformation along the entire curve (Figure 4). As a result, the naive approaches' control points produce curves that stray far from the target deformation.

Our fourth comparison is with the still-popular cubic Bézier spline fitting algorithm of Schneider [1990]. Figure 3 demonstrates swimming and jumping artifacts that often occur between a deformation and an infinitesimally small perturbation of it. These artifacts can be seen more vividly in the accompanying video.

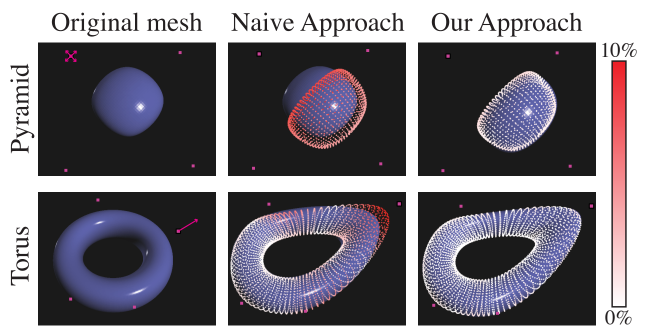


Figure 8: *A pyramid and torus Catmull-Clark subdivision surface are deformed via linear blend skinning (Shepard weights). Naively applying the deformation to the control points matches the target shape much less accurately than our approach. Here, the target shape is depicted point-wise; the color of each point corresponds to its distance from the target shapes as a percentage of the bounding box diagonal.*

Performance. Data regarding the performance of our approach is presented in Tables 1 (Bézier splines) and Table 2 (Catmull-Clark subdivision surfaces). Our timing information was gathered on a dual-core, 2 GHz Intel Core i7 for Bézier splines and on a 2.4 GHz Intel Core i5 for Catmull-Clark subdivision surfaces. Precomputation time measures computation of $\hat{W}_i\hat{A}^{-1}$ (subdivision surfaces) or O_i (Bézier splines). For Bézier splines, it does not include the computation of the weight functions w_i themselves. We sampled every curve in the spline or face in the surface 100 times uniformly. In 2D, our precomputation typically takes less time than computing e.g. Bounded Biharmonic Weights.

For subdivision surfaces (resp. Bézier splines), the performance of our approach is the cost of multiplying an $N \times 4$ matrix by a 4×4 matrix (resp. $3N \times 9$ matrix by a 9×1 matrix) for each handle and then summing the result. (N is the number of control vertices.) We are able to deform complex vector graphics extremely efficiently. Our approach computes updated control point positions for all of our 3D models, and all but the 10,000 Bézier curve Coat of Arms, at over 30fps. The butterfly, rabbit, and squirrel models are production subdivision surfaces [Blender Foundation 2008]. Run-time memory usage is the cost of storing these matrices. Computation is independent across splines, but we do not currently take advantage of this embarrassing parallelism.

For Bézier splines, when tangent magnitude ratio flexibility is desired, the even & odd optimization requires several iterations per update, each of which entails solving a $3N + 2L \times 3N + 2L$ system, where L is the number of constraints. We typically perform this non-linear optimization when the user finishes dragging a handle (e.g. on mouse-up).

6 Results

A variety of edited 2D vector graphics using our technique can be seen in Figure 10. We deformed illustrations and typography. The Turtles, stacked seven tall, demonstrate the resolution independence of our approach. Our construction is general with respect to weights. Applications can choose fast, low quality analytic weights or more expensive numerically-optimized weights. The Penguin and Lion were deformed using a cage and Harmonic Coordinates [Joshi et al. 2007]. The Clam (Figure 1) and Worm were deformed using Bounded Biharmonic Weights (BBW) [Jacobson et al. 2011]. The Coat of Arms, which contains nearly 10,000 Bézier curves, was deformed using Shepard weights [Shepard 1968]. Note that while BBW weights and Harmonic Coordinates are computed numerically on a piecewise linear—that is, C^0— domain, the deformations are restricted by our construction to the space of smooth splines. Thus the deformation is always C^1 regardless of the discrete integration of these C^0 weights.

example	# curves	# handles	# G^1 & angle constraints	precomp. secs	seconds per update: regular	seconds per update: flexible
Boxes (Fig. 6)	4	2	4	0.03	2e-4	0.008
Spoon (Fig. 3)	7	2	7	0.08	2e-4	0.02
Clam (Fig. 1)	56	3	50	0.1	8e-4	0.1
Boy	226	5	118	0.7	5e-3	0.4
Zapfino	379	3	316	1.4	4e-3	0.65
Worm	395	2	261	8.1	3e-2	2.75
Penguin on Lion	400	9	168	2.5	1e-2	0.64
Man	516	4	240	1.3	1e-2	0.78
Seven Turtles	1324	5	914	3.2	3e-2	2.5
Octopus	1706	8	1181	11	2e-2	4.2
Coat of Arms	9496	4	8159	42	1e-1	16.1

Table 1: *The performance of our cubic Bézier spline deformation. Precomputation measures our own computation for $\hat{O}_i$; it does not include linear blend skinning weight computation. Flexible tangent magnitude ratio updates are optional, and can be computed on e.g. mouse-up.*

example	# control vertices	# control faces	precompute time (secs)	seconds per update
pyramid	5	5	1e-3	6e-6
torus	32	32	8e-3	6e-6
butterfly	1216	1222	0.3	5e-5
squirrel	4307	4278	4	2e-4
rabbit	4139	4150	9	2e-4

Table 2: *The performance of our Catmull-Clark subdivision surface deformation. All examples used 4 handles. Precomputation measures computation of $\hat{W}_i\hat{A}^{-1}$.*

In Figure 9, our approach is used to edit 3D Catmull-Clark subdivision surfaces with Shepard weights (the car) and Bounded Biharmonic Weights. To achieve the same result without our approach would require the tedious repositioning of multiple control vertices. By construction, our optimization maintains the desired continuity (including the car's sharp edges).

7 Conclusion

Our method bridges a long-standing gap between mesh-based deformation techniques and vector graphics. We hope our work's success encourages more thorough treatment of vector graphics in future research on image deformation and warping techniques.

Our work could even offer a more efficient alternative to vertex-based linear blend skinning for animation. For 2D vector graphics on the web, where Bézier spline drawing is highly optimized, our approach can be implemented in JavaScript to animate characters created out of Bézier curves. In 3D, vertex-based linear blend skinning with h bones requires $12(h+1)$ floating-point fused multiply-add (`FMA`) operations in a vertex shader. For a Catmull-Clark subdivision surface with N control points, each refined vertex is the weighted sum of 16 control points. Our approach therefore requires $16hN$ `FMA` and $3N(h-1)$ `add` operations to compute the control

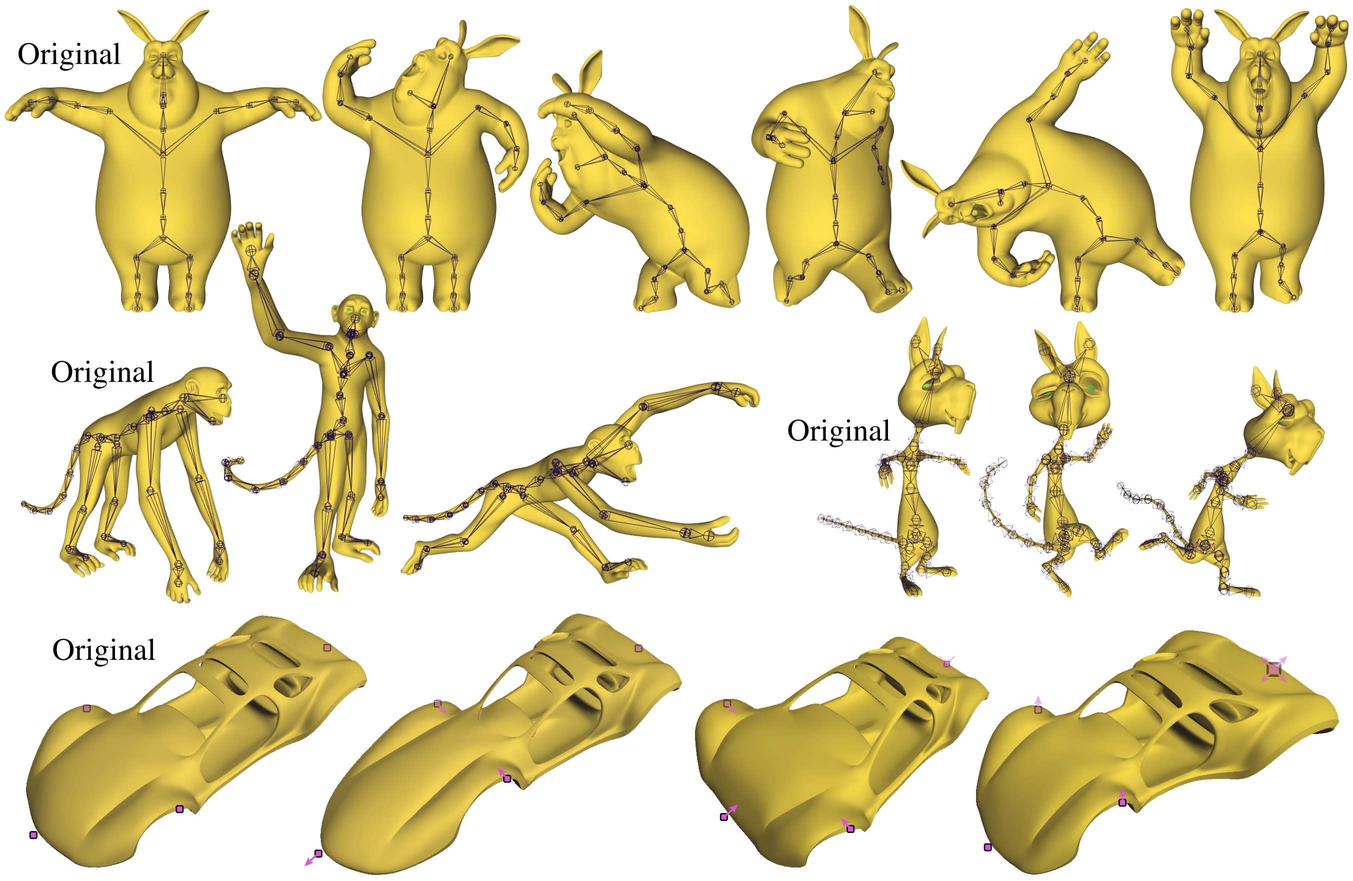


Figure 9: *A variety of linear blend skinning deformations applied to Catmull-Clark subdivision surfaces. The car uses Shepard weights; the remainder use Bounded Biharmonic Weights. The sharp edges along the bottom of the car and along the wheel wells are maintained by our optimization. The rabbit and squirrel are from the film Big Buck Bunny [Blender Foundation 2008]. The car is copyright Pixar [Pixar 2014].*

points once per-frame, and only $16 \cdot 3$ `FMA` operations in a vertex shader. Thus, our approach offers potential performance advantages when linear blend skinning with 4 or more bones. In this way, subdivision surfaces could be animated and directly rendered [Niessner et al. 2012] on the GPU.

7.1 Limitations and Future Work

The most immediate future work is building support for other primitives common in vector graphics. Many editing tools already convert rectangles, circular arcs and polylines seamlessly to cubic Bézier curves during editing. However, maintaining these shapes would also be an interesting feature better suited for vector graphics than raster editing where discrete optimization over a mesh would be necessary [Bouaziz et al. 2012].

Features like gradients are linear or radial, and cannot be smoothly deformed by anything more interesting than an affine or similarity transform, respectively, to stay linear or radial. Perhaps exploiting not-yet standardized diffusion curves could alleviate this.

When there are insufficient control points to represent a desired deformation, our optimization may produce undesirable undulations (Figure 11). These can be eliminated with the insertion of additional control points. In the future, we wish to automatically insert control points to maintain a desired deformation accuracy.

We plan to improve the precomputation efficiency of our algorithm. We can adaptively sample the curve and surface integral. Each spline in an illustration can be precomputed (and deformed) in parallel. The repeated block diagonal structure of our system matrix invites so far unexplored benefits of SIMD parallelism and a possible GPU solver.

We would also like to explore efficient approaches to skinning other types of curves and surfaces. Our approach could be applied *mutatis mutandis* to the "linear" portion of NURB curves and surfaces, but optimizing the knot vector and the rational control point weights is an avenue of future work. Though non-standard, clothoid splines promise higher order continuity [McCrae and Singh 2009; Baran et al. 2010].

Finally, while we highlight the usefulness of editing splines for visual design and animation tasks, splines have many other uses. An obvious extension of our work is to consider curve editing in other contexts such as silhouette editing [Zimmermann et al. 2007] or motion path planning [Kim et al. 2009].

Acknowledgements

We are grateful to Kenshi Takayama, Romain Prévost, Chaim Gingold, and the anonymous reviewers for helpful feedback; Michelle Lee for artwork; Denis Zorin for an illuminating discussion; and Jianchao Tan for moral support. Authors Liu and Gingold are supported in part by the NSF (IIS-1451198) and a Google research award. The Columbia University Computer Graphics Group is supported by the NSF, Intel, The Walt Disney Company, and Autodesk.

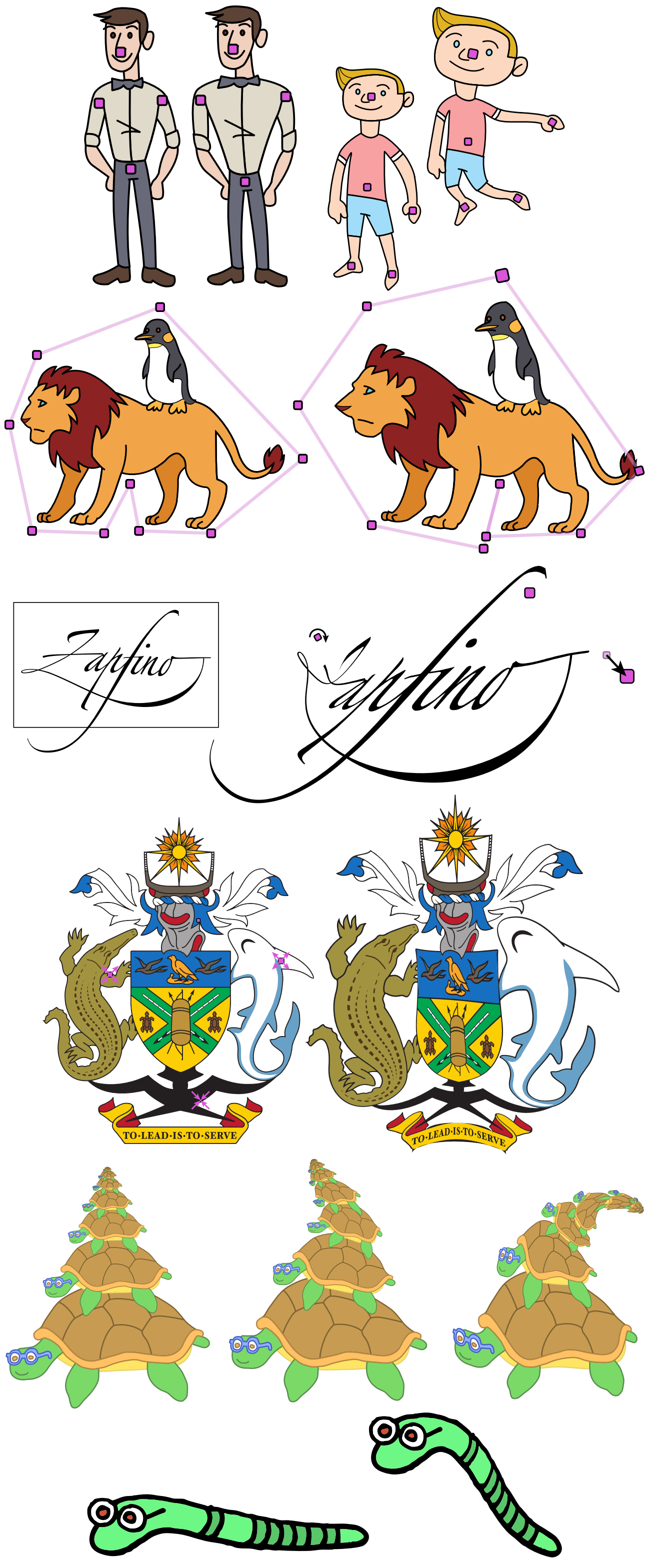


Figure 10: *A variety of linear blend skinning deformations applied to vector graphics using our approach. The left column depicts undeformed shapes. The boy, man, lion, and penguin are copyright Michelle Lee. The coat of arms of the Solomon Islands are copyright Wikimedia Commons user Prez001.*

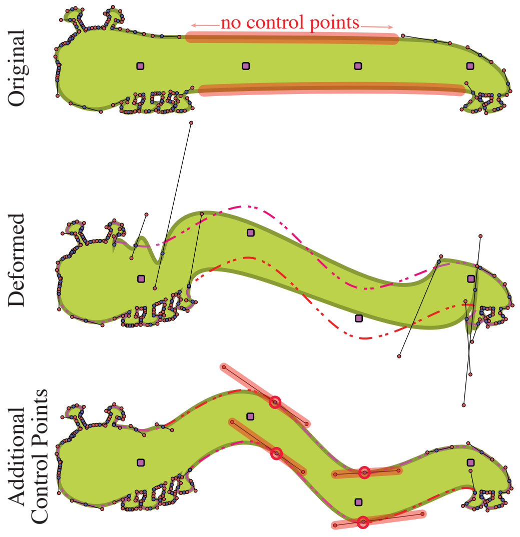


Figure 11: *Top: A caterpillar whose long back and belly are each composed of a single Bézier curve. Middle: Deforming the long body results in highly undesirable undulations. Bottom: Splitting the back and belly into three Bézier curves allows the target deformation to be closely matched.*

References

ADOBE SYSTEMS INC., 2014. Photoshop CS6's Liquify tool. http://www.adobe.com/photoshop/.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3D characters. *ACM Trans. Graph. 26*, 3, 72:1–72:8.

BARAN, I., LEHTINEN, J., AND POPOVIĆ, J. 2010. Sketching clothoid splines using shortest paths. In *Comput. Graph. Forum*.

BARBIČ, J., SIN, F., AND GRINSPUN, E. 2012. Interactive editing of deformable simulations. *ACM Trans. Graph.*.

BEIER, T., AND NEELY, S. 1992. Feature-based image metamorphosis. In *Proc. SIGGRAPH*.

BLENDER FOUNDATION, 2008. Big Buck Bunny. `http://www.bigbuckbunny.org`.

BOTSCH, M., AND KOBBELT, L. 2004. An intuitive framework for real-time freeform modeling. *ACM Trans. Graph. 23*, 3, 630–634.

BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. 2012. Shape-up: Shaping discrete geometry with projections. In *Comput. Graph. Forum*.

DAVIS, T. A., AND HAGER, W. W. 1999. Modifying a sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications 20*, 3, 606–627.

DER, K. G., SUMNER, R. W., AND POPOVIĆ, J. 2006. Inverse kinematics for reduced deformable models. *ACM Trans. Graph.*.

DEROSE, A. D. 1985. *Geometric Continuity: A Parametrization Independent Measure of Continuity for Computer Aided Geometric Design*. PhD thesis, EECS Department, University of California, Berkeley.

FOWLER, B., AND BARTELS, R. 1993. Constraint-based curve manipulation. *IEEE Comput. Graph. Appl. 13*, 5, 43–49.

FRISKEN, S. 2008. Efficient curve fitting. *Journal of Graphics, GPU, and Game Tools 13*, 2, 37–54.

HAGER, W. W. 1989. Updating the inverse of a matrix. *SIAM Review 31*, 2, 221–239.

HILDEBRANDT, K., SCHULZ, C., TYCOWICZ, C. V., AND POLTHIER, K. 2011. Interactive surface modeling using modal analysis. *ACM Trans. Graph. 30*, 5, 119:1–119:11.

HORMANN, K., AND FLOATER, M. S. 2006. Mean value coordinates for arbitrary planar polygons. *ACM Trans. Graph.*.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*.

JACOBSON, A., BARAN, I., POPOVIĆ, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4, 78:1–78:8.

JACOBSON, A., BARAN, I., KAVAN, L., POPOVIĆ, J., AND SORKINE, O. 2012. Fast automatic skinning transformations. *ACM Trans. Graph.*.

JACOBSON, A., KAVAN, L., , AND SORKINE, O. 2013. Robust inside-outside segmentation using generalized winding numbers. *ACM Trans. Graph. 32*, 4, 33:1–33:12.

JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *ACM Trans. Graph. 24*, 3, 399–407.

JOSHI, P., MEYER, M., DEROSE, T., GREEN, B., AND SANOCKI, T. 2007. Harmonic coordinates for character articulation. *ACM Trans. Graph. 26*, 3, 71:1–71:9.

KAVAN, L., GERSZEWSKI, D., BARGTEIL, A., AND SLOAN, P.-P. 2011. Physics-inspired upsampling for cloth simulation in games. *ACM Trans. Graph. 30*, 4, 93:1–93:9.

KIM, M., HYUN, K., KIM, J., AND LEE, J. 2009. Synchronized multi-character motion editing. *ACM Trans. Graph. 28*, 3.

KRISHNAMURTHY, V., AND LEVOY, M. 1996. Fitting smooth surfaces to dense polygon meshes. In *Proc. SIGGRAPH*.

LIAO, Z., HOPPE, H., FORSYTH, D., AND YU, Y. 2012. A subdivision-based representation for vector image editing. *IEEE TVCG 18*, 11, 1858–1867.

LIPMAN, Y., LEVIN, D., AND COHEN-OR, D. 2008. Green coordinates. *ACM Trans. Graph. 27*, 3, 78:1–78:10.

MCCRAE, J., AND SINGH, K. 2009. Sketching piecewise clothoid curves. *Computers & Graphics 33*, 4, 452–461.

MILLIRON, T., JENSEN, R. J., BARZEL, R., AND FINKELSTEIN, A. 2002. A framework for geometric warps and deformations. *ACM Trans. Graph. 21*, 1, 20–51.

NIESSNER, M., LOOP, C., MEYER, M., AND DEROSE, T. 2012. Feature-adaptive gpu rendering of catmull-clark subdivision surfaces. *ACM Trans. Graph. 31*, 1, 6:1–6:11.

NORIS, G., HORNUNG, A., SIMMONS, M., SUMNER, R., AND GROSS, M. 2013. Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics 32*, 1, 4:1–4:11.

OLSEN, L., SAMAVATI, F. F., SOUSA, M. C., AND JORGE, J. A. 2009. Sketch-based modeling: A survey. *Computers & Graphics*.

PIXAR, 2014. Opensubdiv. `http://graphics.pixar.com/opensubdiv/`.

PLASS, M., AND STONE, M. 1983. Curve-fitting with piecewise parametric cubics. *SIGGRAPH Comput. Graph.*.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. *ACM Trans. Graph.*.

SCHMITT, F. J. M., BARSKY, B. A., AND DU, W.-H. 1986. An adaptive subdivision method for surface-fitting from sampled data. *SIGGRAPH Comput. Graph. 20*, 4, 179–188.

SCHNEIDER, P. J. 1990. Graphics gems. ch. An Algorithm for Automatically Fitting Digitized Curves, 612–626.

SHEPARD, D. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, ACM, 517–524.

SHEWCHUK, J. R. 1996. Triangle: Engineering a 2D quality mesh generator and delaunay triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, 203–222.

SI, H., 2003. TETGEN: A 3D delaunay tetrahedral mesh generator. `http://tetgen.berlios.de`.

WANG, W., POTTMANN, H., AND LIU, Y. 2006. Fitting b-spline curves to point clouds by curvature-based squared distance minimization. *ACM Trans. Graph. 25*, 2, 214–238.

WEBER, O., AND GOTSMAN, C. 2010. Controllable conformal maps for shape deformation and interpolation. *ACM Trans. Graph. 29*, 4, 78:1–78:11.

WEBER, O., SORKINE, O., LIPMAN, Y., AND GOTSMAN, C. 2007. Context-aware skeletal shape deformation. *Comput. Graph. Forum 26*, 3, 265–274.

WEBER, O., BEN-CHEN, M., AND GOTSMAN, C. 2009. Complex barycentric coordinates with applications to planar shape deformation. *Comput. Graph. Forum 28*, 2, 587–597.

WEBER, O., BEN-CHEN, M., GOTSMAN, C., AND HORMANN, K. 2011. A complex view of barycentric mappings. In *Proc. SGP*, vol. 30, 1533–1542.

ZHENG, J. M., CHAN, K. W., AND GIBSON, I. 1998. A new approach for direct manipulation of free-form curve. *Comput. Graph. Forum 17*, 3, 327–334.

ZHOU, K., HUANG, X., XU, W., GUO, B., AND SHUM, H.-Y. 2007. Direct manipulation of subdivision surfaces on gpus. *ACM Trans. Graph. 26*, 3 (July).

ZIMMERMANN, J., NEALEN, A., AND ALEXA, M. 2007. Silsketch: automated sketch-based editing of surface meshes. In *Proceedings of the 4th Eurographics workshop on Sketch-based interfaces and modeling (SBIM)*, 23–30.