Cords: Geometric Curve Primitives for Modeling Contact

Patrick Coleman and Karan Singh
University of Toronto

The authors present a geometric curve primitive, known as a *cord*, which allows for interactive modeling of curves that contact complex geometry.

Curves are perhaps the most versatile of modeling primitives in computer graphics. They define a rough structure for many surface-generation algorithms and are often fit to meaningful surface features for further shape modeling. Deformable objects such as hair and fur are simulated on finite element curve discretizations. Motion paths for planning and animation applications are tied to underlying curves. The internal structure of tubular objects, such as the laces in Figure 1a, are typically built and manipulated using an underlying curve. Artistic strokes for rendering are often defined in terms of an underlying implicit form that follows a curve; Figures 1b and 1c show two real-world examples. Despite this versatility, existing curve models are difficult to work with in applications that require precise control, and where interpenetration with complex objects is objectionable (see Figure 1b). Here, we describe cords, geometric curve primitives designed to appropriately contact geometry while providing a user with precise control.

Design criteria for cords

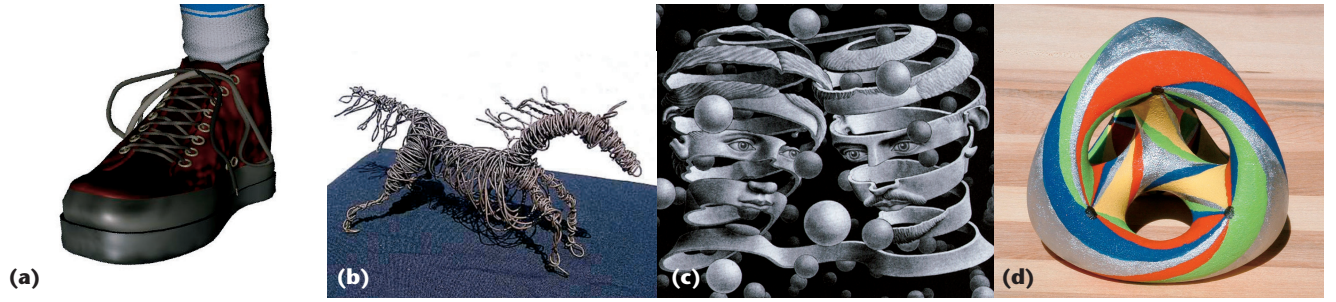
For creative design and animation applications, we have identified three primary criteria for algorithmic models of curves in contact. First, the curves should appropriately contact geometry, as object interpenetration can be quite objectionable from both user-control and aesthetic standpoints. Second, users should have precise control over the curve's shape, as this is often a creative necessity. Third, as animation is an important application for curves in contact with geometry, the curve shape should be continuous with respect to control properties.

Current algorithms for curve modeling do not meet all these requirements. Parametric curves specified with control points, by far the most common curve model, offer precise shape and continuity control. However, users must explicitly model any interpenetration constraints. This is a particularly arduous task in complex scenes, as users must create a large number of control points. If objects or the curve are in motion, contact relationships can change, and the number of control points must be sufficient to model all contacts. Simulation models provide collision response to avoid geometric interpenetration. However, they are notoriously difficult to control, with small changes to the control parameters often resulting in large, unpredictable changes to the resulting motion. Variational models can incorporate various types of constraints, but they do not guarantee continuity of shape with respect to the constraints. Such models are more appropriate for applications where exact geometric requirements take precedence over precise control (see Figure 1d).

In our approach, a user creates an arbitrary parametric curve—the guide curve—to specify a rough approximation of a cord's shape, relative to scene geometry. The cord is then generated using this curve to guide it through the scene. The appearance properties and ray intersection tests are incorporated into an efficient local growth algorithm, ensuring appropriate object contact while providing a desired shape. Rather than generate the entire cord with a global algorithm, which can lead to many ray intersection tests, we locally add sections to the cord. Figure 2 illustrates an example guide curve positioned among geometric primitives and the resulting cord.

The appearance properties of cords lets users modify either shape or length. The stiffness and slack properties affect the cord shape. Stiffness models the cord's resistance to bending, similarly to wire. Slack causes a cord to bend into local surface concavities where the cord would otherwise wrap over the concavities. The length and elasticity properties modify only the cord's length. Length represents the length of a nonstretchable cord, a required feature for many practical applications. Elasticity allows the cord to partially stretch or shrink along its length to meet the guide curve's endpoints. Figure 3 illustrates how these properties alter the cord's shape. Some of these properties, while analogous to physical material properties, do not have the same definitions that apply under physical simulation. We have formulated our definitions instead to provide precise control of visual appearance.

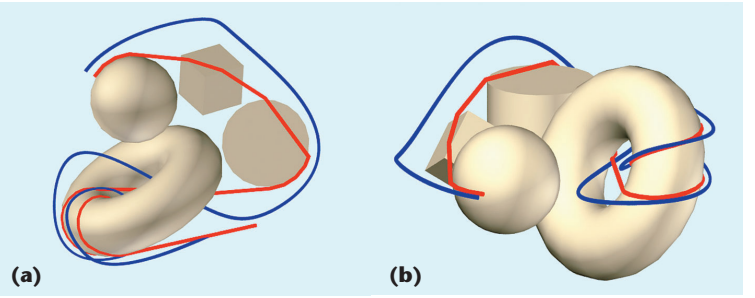
Published by the IEEE Computer Society



1 Examples of curve modeling applications appropriate for cords. (a) A 3D curve underlying a shoelace model. (b) Wire sculpture by Jill Abrams. (© 2002 Dover Abrams.) (c) M.C. Escher's *Bond of Union*. (© 2006 The M.C. Escher Company B.V., the Netherlands. All rights reserved. Used by permission; http://www.mcescher.com.) (d) K_{12} graph embedded in a genus 6 surface. (© 2004 Carlo Sequin.)

By presenting design criteria and algorithms for contact-constrained curve modeling for creative applications, we

- present a geometric approach to modeling curves in contact with complex scenes that offers users precise control;
- describe algorithms for generating such curves in terms of user properties that allow them to specify a rough path through a scene and visual appearance characteristics;
- present an analytic form for stiff cords that allows for decoupling of rendering resolution from generation resolution; and
- describe an approach to guarantee higher-order transitions between regions of bending and linear regions of stiff cords in static scenes, which can be important for design applications.

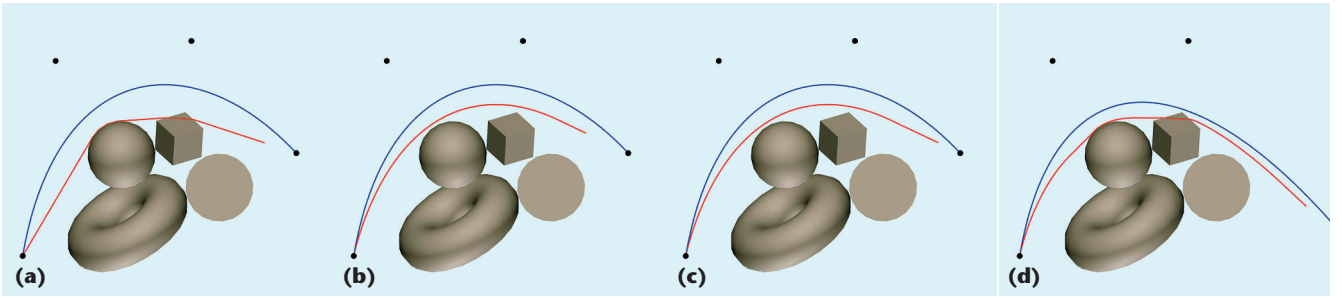


2 Example guide curve (blue) and cord (red) in a 3D scene.

Cord generation

Given a guide curve and scene geometry, we can generate cords using one of three algorithms, each a generalization capable of capturing a wider variety of shapes. Nonstiff cords are made up of a sequence of linear segments and have the appearance of string. Stiff cords introduce regions of bending, which model the bending resistance of some materials such as wire. Slack cords generalize stiff cords, causing the regions of bending to locally fall into surface concavities. We can generalize any one of these forms to have length constraints and elasticity qualities. As slack cords generalize stiff cords and stiff cords generalize nonstiff cords, an implementation of slack cords with elasticity is capable of representing all forms.

The guide curve, appearance properties, and the geometry the cord should contact all determine a cord's shape. Cords are inherently directional, as they are generated with local growth algorithms that progressively look forward along the guide curve. As such, we require the guide curve to have a parametric form $\mathbf{f}(t)$. In practice, we use nonuniform cubic B-splines, as they provide local control and allow for endpoint interpolation, thus being popular for shape modeling. However, any parametric guide curve could be used. We precompute a parametric discretization of the guide curve $\mathbf{f}_i = \mathbf{f}(i\Delta t)$ such that $i = \{0, \ldots, N\}$ and $\Delta t = (t_{\max} - t_{\min})/(N-1)$. In



3 (a) Fixed-length, nonstiff cords will wrap around geometry with the appearance of string. (b) Adding stiffness causes the cord to resist bending, modeling the appearance of wire. (c) Elasticity can be increased to allow for stretching toward the guide curve's endpoint. (d) Moving this guide curve endpoint then causes the cord to stretch further toward it.

Related Work

Curve and surface design is one of the oldest and most mature areas of computer graphics. Most relevant work in geometric design and deformation deals with the modeling of scene geometry to conform to the shape of specified parametric curves. Variational models allow users to specify constraints such as positions and tangents and solve a constrained optimization problem to generate a curve or surface.[1] Surfaces can also be modeled using constraints expressed as energy functions, and gradient descent in parameter space can be used to find shapes that best meet the constraints.[2] Curves can be generated on surfaces to meet certain geometric criteria,[3] and curve networks can be retargeted to surface manifolds.[4] Parametric curves can also be generated to meet length constraints.[5]

Simulation models, introduced by Terzopoulos and colleagues,[6] are set up as networks of point masses connected by springs for finite element or finite difference simulation. Direct embedding of physical properties increases realism for modeling applications.[7] Most hair simulation models are also built upon finite element techniques.[8] Particular to real-world curves, Pai incorporated a thin-solid model to increase accuracy for the surgical simulation of threads.[9] Empirical simulation has also been investigated to reduce computational overhead.[10] Constraints on physical behavior can be applied to simulation models through optimization.[11] Geometric constraints among rigid bodies can be enforced through the use of inverse dynamics.[12] Appropriate object-to-object contact under physical forces can be ensured by tracking contact points and finding a stable configuration with respect to the forces.[13]

Barzel provides a unique approach to curve modeling for animation production. His model achieves the appearance of dynamic motion while providing precise user control.[14] He models dynamic motion as parameterized changes in shape that can be keyframe animated, avoiding the complexities of simulation while providing believable motion. In contrast, we provide precise control of shape in contact rather than shape in dynamic-like motion.

References

1. W. Welch and A. Witkin, "Variational Surface Modeling," *Proc. Siggraph*, ACM Press, 1992, pp. 157-166.
2. A. Witkin, K. Fleischer, and A. Barr, "Energy Constraints on Parameterized Models," *Proc. Siggraph*, ACM Press, 1987, pp. 225-232.
3. V. Surazhsky et al., "Fast Exact and Approximate Geodesics on Meshes," *ACM Trans. Graphics*, vol. 24, no. 3, 2005, pp. 553-560.
4. K. Singh, H.K. Pedersen, and D Krishnamurthy, "Feature Based Retargeting of Parameterized Geometry," *Proc. Geometric Modeling and Processing 2004*, IEEE CS Press, 2004, p. 163.
5. E. Fiume, "Isometric Piecewise Polynomial Curves," *Computer Graphics Forum*, vol. 14, no. 1, 1995, pp. 47-58.
6. D. Terzopoulos et al., "Elastically Deformable Models," *Proc. Siggraph 87*, ACM Press, 1987, pp. 205-214.
7. D. Terzopoulos and H. Qin, "Dynamic NURBS with Geometric Constraints for Interactive Sculpting," *ACM Trans. Graphics*, vol. 13, no. 2, 1994, pp. 103-136.
8. N. Magnenat-Thalmann, S. Hadap, and P. Kalra, "State of the Art in Hair Simulation," *Proc. Int'l Workshop Human Modeling and Animation*, Korea Computer Graphics Soc., 2000, pp. 3-9.
9. D.K. Pai, "Strands: Interactive Simulation of Thin Solids Using Cosserat Models," *Proc. Eurographics*, Eurographics Assoc., 2002.
10. J. Brown, J.-C. Latombe, and K. Montgomery, "Real-Time Knot Tying Simulation," *The Visual Computer*, vol. 20, nos. 2-3, 2004, pp. 165-179.
11. J.C. Platt and A.H. Barr, "Constraints Methods for Flexible Models," *Proc. Siggraph*, ACM Press, 1988, pp. 279-288.
12. R. Barzel and A.H. Barr, "A Modeling System Based on Dynamic Constraint," *Proc. Siggraph*, ACM Press, 1988, pp. 179-188.
13. J. M. Snyder, "An Interactive Tool for Placing Curved Surfaces Without Interpenetration," *Proc. Siggraph*, ACM Press, 1995, pp. 209-218.
14. R. Barzel, "Faking Dynamics of Ropes and Springs," *IEEE Computer Graphics and Applications*, vol. 17, no. 3, 1997, pp. 31-39.

the following discussions, we assume the parameterization t has been remapped to [0, 1].

Nonstiff cords

Nonstiff cords, which have the shape of string, are made up of a series of linear segments that wrap around objects while following the guide curve. We thus construct nonstiff cords by searching forward toward guide curve samples $\mathbf{f}_i$ and locating rays to the guide curve that graze geometry. We append a linear segment whenever such a grazing point is found.

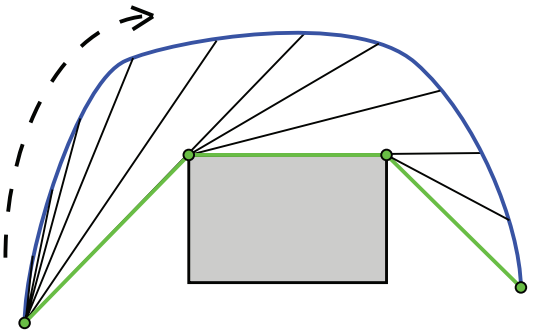
We first initialize the cord to the starting point of the guide curve, $\mathbf{f}_0$. We then repeatedly consider subsequent guide curve samples $\mathbf{f}_i$ and cast rays from the cord's current endpoint to these sample points, until we find an intersection (see Figure 4). We then look for a grazing point along a ray toward the guide curve at point $\mathbf{f}(t)$, such that a ray toward $\mathbf{f}(t^-)$, has no intersection and a ray toward $\mathbf{f}(t^+)$ does have an intersection. We know that the ray to the prior sample $\mathbf{f}_{i-1}$ has no intersection and that the ray to the sample $\mathbf{f}_i$ does. We isolate the grazing intersection point by using interval halving on the corresponding guide curve parameter values.

When we isolate a grazing point to within some user-adjustable numerical tolerance (typically 0.01 cm for scenes of dimension 10 cm × 10 cm × 10 cm), we append a segment from the cord's endpoint to this grazing point. The grazing point, offset by some small ray bias to avoid self-intersection, then becomes the new cord endpoint. Note, the offset must not be near scene geometry to avoid self-intersection. Given the rays to the bounding samples $\mathbf{r}_{i-1}$ and $\mathbf{r}_i$ and the ray $\mathbf{r}_t$ along which we found the grazing point, we offset along $(\mathbf{r}_i \times \mathbf{r}_{i-1}) \times \mathbf{r}_t$, which will have a direction away from the surface at the grazed point.

We then know that a ray moving toward the guide curve point that we isolated at a halfway interval will not intersect (due to the ray bias) and will continue searching with the subsequent guide curve sample. If at any point we find multiple grazing points along one ray,

we consider only the furthest point, as this lets us append a longer segment to avoid redundant intermediate points. We also ignore grazing points beyond the guide curve, as cords that grow to such points are not consistent with the wrapping metaphor. When the last sample $\mathbf{f}_N$ is reached and does not lead to an intersection, we append a linear segment to this endpoint.

Users should avoid positioning the guide curve so that it passes through geometry, as this creates an ambiguity about how a cord should wrap around that geometry. Should users choose to do this, we alert them to the error and grow the cord through the shape by appending a segment to each intersection of the guide curve with the geometry.

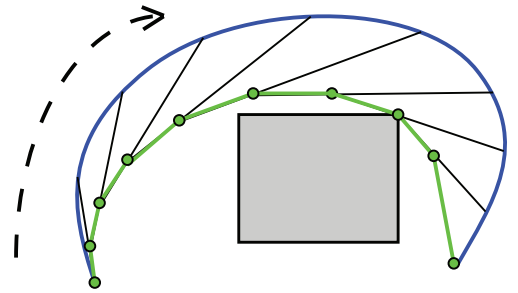


4 The simplest type of cords, nonstiff cords, have the appearance of string. Linear segments (green) are appended toward points that graze geometry; these grazing points are located by repeatedly casting rays (black) to the guide curve (blue).

Stiff cords

Stiff cords provide an extension to nonstiff cords so that we can model resistance to bending. Stiff cords alternate between the linear regions of nonstiff cords and regions of bending, in which the cord gradually curves toward a new grazing point. Users work with a stiffness parameter, such that when stiffness equals zero, bending regions have zero length and the cord appears nonstiff. When stiffness equals one, the cord becomes an approximation to the guide curve that approaches it in shape as the sample count approaches infinity. If the guide curve is polynomial, we can express each bending region as an analytic function of the guide curve control points. This lets us render them using common curve rendering algorithms, independently of the guide curve discretization.

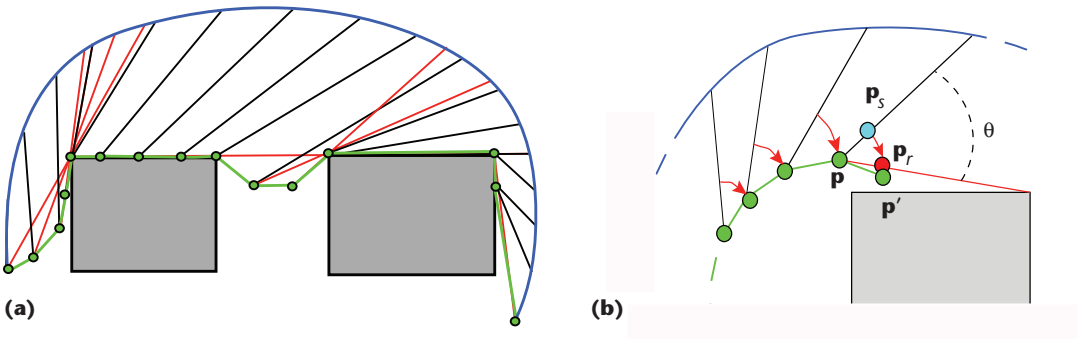
To generate stiff cords, rather than ignoring failed intersection tests, we append small linear segments for every candidate ray toward the initial guide curve sample set. Let s = stiffness/N be a proportional step size, where N is the number of guide curve samples. This step size is not arbitrary; it specifically leads to the analytic form that we will derive later in this article. Whenever a ray is cast toward one of the initial guide curve samples, we still locate a grazing intersection and append a linear segment if it intersects. If it does not, we compute a point $\mathbf{p}' = s(\mathbf{f}_i - \mathbf{p})$, where $\mathbf{p}$ is the cord's current endpoint. Then we append a new segment from $\mathbf{p}$ to $\mathbf{p}'$, setting $\mathbf{p}'$ as the new cord endpoint. We illustrate this process in Figure 5. Usually, many small linear segments will be appended in sequence before appending a (typically longer) linear segment to a grazing point. Each such sequence is one bending region.



5 Stiff cords, which are a generalization of nonstiff cords, can model materials that resist bending, such as wire. To construct stiff cords, we always append a short linear segment along rays used for intersection testing.

Slack cords

Slack cords are extended stiff cords capable of falling into surface concavities (see Figure 6a). Users can adjust this property to cause cords to fall onto a surface when necessary, such as when controlling the shape of curves



6 (a) Slack cords generalize stiff cords by allowing them to fall into local surface concavities. (b) Each short linear segment (toward $\mathbf{p}_s$) is rotated beyond a ray that reaches the next grazing intersection ($\mathbf{p}_r$), resulting in the segment being appended to $\mathbf{p}'$.

7 **Cords were used in the production of the animated film *Ryan*. Each colored bundle of hair is procedurally generated from an underlying cord, which was keyframe animated to establish contact relationships with the geometry of the character's body. (Images rendered by the authors.)**

on surfaces. We define a slack property, such that when slack is zero, the cord otherwise behaves like a stiff cord. If slack is one, the cord will have a sagging appearance. If slack is sufficiently greater than one, it will generate a cord that lies close to the surface in regions of local concavity. If stiffness is zero, slack has no effect on the cord's shape.

We modify the stiff cord algorithm so that the small linear segments of bending regions grow in an altogether different direction. As with stiff cords, we calculate a proportional step point $\mathbf{p}_s$, in the direction of the guide curve sample (blue in Figure 6b). Additionally, the algorithm finds the next grazing ray (red). We then calculate a step along that ray that is equal in length to the step toward the guide curve sample (the red point, $\mathbf{p}_r$). We calculate the new cord point $\mathbf{p}'$ as a rotational interpolation or extrapolation from $\mathbf{p}_s$ to $\mathbf{p}_r$. The slack parameter parameterizes this interpolation, so that if the angle from $\mathbf{p}_s - \mathbf{p}$ to $\mathbf{p}_r - \mathbf{p}$ is θ, the angle from $\mathbf{p}_s - \mathbf{p}$ to $\mathbf{p}' - \mathbf{p}$ is slack $* \theta$. If $\mathbf{p}'$ should cause the cord to pass through the surface, it is truncated at the intersection point.

If slack is sufficiently large, the cord will lie in the convex regions of a surface that it would otherwise wrap over. Slack is also not appropriate for large, near-closed concave regions and will not appropriately move to lie in the surface that is internal to them, as this can be ambiguous. Instead, the slack cords handle subtle concavities as would appear in a bumpy surface, where user intent is clear. Regions of bending in slack cords do not admit an analytic form, as each proportional step incorporates ray intersection tests that are often discontinuous with respect to a parameterization.

Length constraints and elasticity

The nonstiff, stiff, and slack cords interpolate both guide curve endpoints. Often, we want to enforce length constraints along a curve to model materials with various stretching properties. It's also useful to allow a cord to stretch or shrink as the guide curve stretches or shrinks, without necessarily interpolating its endpoints. We allow users to modify properties of length and elasticity to accomplish this. Rather than physically model elasticity and its effect on the cord's shape, which would be difficult to control, we use elasticity to stretch or shrink the cord only along its length. This decoupling allows users to more easily animate cords in a keyframe environment, one of our target applications. We also model elasticity so that when its value is zero, the cord has a fixed length equal to the length property. When elasticity is one, these properties have no effect on the appearance, and the cord interpolates both endpoints of the guide curve. We constrain elasticity to have a value in [0, 1].

We first generate a cord using any of the previously described algorithms; let its length be l_c. We then compute a target length $l_t = \text{elasticity} * l_c + (1 - \text{elasticity}) * \text{length}$, an interpolation between the length of a cord interpolating both endpoints and the length appearance property. If $l_t < l_c$, we simply truncate the cord to the target length. If $l_t > l_c$, we extend the cord along its end tangent, adding a linear segment of length $l_t - l_c$. This constrains the generated cord relative to one endpoint of the guide curve. To constrain to a general position along the guide curve, we could apply the truncation or extension to both ends of the cord. Figure 7 illustrates the use of cords with zero elasticity whose length has been animated.

Analytic extensions

The algorithms we just described generate cords efficiently, are easy to code, and capture the desired design properties. For stiff cords, however, we can improve upon these algorithms while generating the same shapes. First, we generate the bending regions of stiff cords as sequences of small linear segments. We will derive an analytic form for these bending regions, which allows their continuity properties to be expressed in terms of the guide curve's continuity properties. Users can then achieve desired continuity properties by using an appropriate form for the guide curve. This analytic form also allows rendering of stiff cords at a resolution independent of the guide curve discretization. In addition, for polynomial guide curves specified with control points or other geometric constraints, we can analytically compute the bending regions using only the guide curve control points.

Stiff cords possess tangent continuity at join points between bending regions and linear segments. Some design applications require higher-order continuity at join points. Here we describe an approach to replacing the linear regions of nonanimated stiff cords with quintic spans, so that the entire cord is G^2 continuous.

These extensions don't apply to slack cords, as slack cords can have arbitrary discontinuities due to the incorporation of the grazing ray direction into every cord segment.

Stiff cords: regions of bending

Regions of bending for nonslack cords converge to an analytic form as the number of guide curve samples approaches infinity. Given an analytic form, if the guide curve is a polynomial specified with control points, we can express the regions of bending in terms of the guide curve control points. Without loss of generality, let the start of a bending region be some point $\mathbf{p}_0$. We can think of this point as generated by some ray cast to a guide curve point $\mathbf{f}(t_0)$. Similarly, the last point in the bending region is generated by a ray to $\mathbf{f}(t_1)$. Without loss of generality, let $t_0 = 0$ and $t_1 = 1$.

Recall, from the definition of stiffness in the "Stiff cords" section, that s = stiffness Δt. We can write a single step of the generation algorithm in the form

$$\mathbf{p}_i = \mathbf{p}_{i\text{-}1} + a\Delta t\,(\mathbf{f}\,(i\Delta t) - \mathbf{p}_{i\text{-}1})$$

using a as shorthand notation for stiffness. Dividing by Δt and taking the limit as $\Delta t \to 0$, we arrive at the following differential equation involving the bending region $\mathbf{g}(t)$:

$$\frac{d\mathbf{g}}{dt} = a\left(\mathbf{f}\left(t\right) - \mathbf{g}\left(t\right)\right) \tag{1}$$

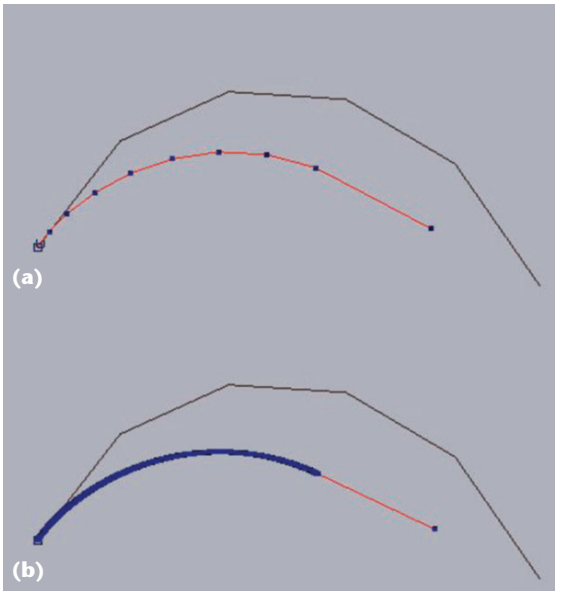
This is a first-order differential equation, and given the initial condition $\mathbf{g}(0) = \mathbf{p}_0$, it has the solution

$$\mathbf{g}\left(t\right) = e^{-at}\mathbf{p}_0 + ae^{-at}\int_0^t \mathbf{f}\left(x\right)e^{ax}dx \tag{2}$$

Equation 1 also illustrates why we chose the particular stiffness value, as it's equivalent to an approximate solution using Euler's method. Figure 8 illustrates this empirically. As the number of guide curve samples increases, the discretization approaches the underlying analytic form of the bending region.

For each bending region, we still need the two points on the guide curve that map to the first and last point of the bending region, as well as the first point $\mathbf{p}_0$. To do this, we use the generative algorithm from the "Stiff cords" section to identify these points for each bending region. We can directly use Equation 2 by remapping each bending region parameter range to [0, 1].

If the guide curve is a polynomial, we can express each bending region of a stiff cord in terms of its control points (or other constraints). Consider the common matrix decomposition of a polynomial spline with parameterization x, where $\mathbf{x}$ is the power basis, $\boldsymbol{B}$ is the basis matrix of the particular curve form, and $\boldsymbol{Q}$ is a matrix of constraints (for example, control point coefficients):



8 As the number of guide curve samples increases, the generated cord with positive stiffness approaches the corresponding limit curve, made up of a region of bending and a linear segment: (a) 10 steps and (b) 1,000 steps. The convex hull of the guide curve is dark gray, and the cord is red. Each blue dot represents a point on the generated cord.

$$\mathbf{f}(x) = \mathbf{x}\boldsymbol{BQ}$$

Let $\mathbf{k}(x)$ denote the integral in Equation 2, without bounds:

$$\mathbf{k}\left(x\right) = \int \mathbf{f}\left(x\right)e^{ax}dx$$

If we plug the matrix form of $\mathbf{f}(x)$ into $\mathbf{k}(x)$, we can evaluate the indefinite integral, resulting in the following form:

$$\mathbf{k}\left(x\right) = \frac{1}{a}e^{ax}\mathbf{x}\boldsymbol{ABQ}$$

For an order n polynomial guide curve, $\boldsymbol{A}$ is an $n \times n$ upper triangular matrix with the following entries:

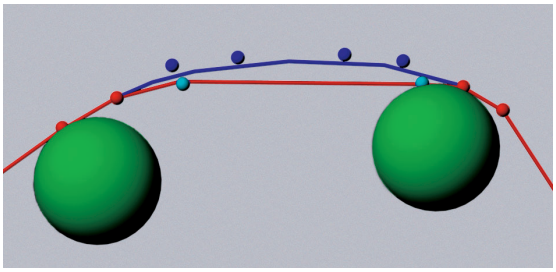
$$A_{ij} = \begin{cases} 0 & i > j \\ \left(-\frac{1}{a}\right)^{j-1}\frac{j!}{i!} & i <= j \end{cases}$$

where $i, j \in [1, n]$. Inserting this form of $\mathbf{k}(x)$ into Equation 2, we can evaluate the definite integral and achieve the following form for the bending region, with $\mathbf{t}$ being a power basis in terms of t:

$$\mathbf{g}(t) = e^{-at}\,(\mathbf{p}_0 - [1\ 0\ 0\ \ldots\ 0]\boldsymbol{ABQ}) + \mathbf{t}\boldsymbol{ABQ}$$

Higher-order continuity for static stiff cords

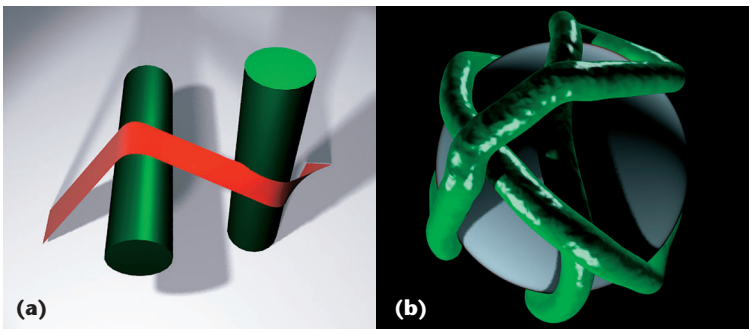
A stiff cord's bending regions are guaranteed to have the same continuity properties of the guide curve, as we



9 Static cords used for design can be altered to ensure G^2 continuity. Each linear segment is replaced with a quintic curve (dark blue). To do this, we move the control points where the regions join (light blue) and set the new control points to ensure the regions join with the desired curvature continuity.

can express these regions as an integral operator over the corresponding region of a guide curve (see Equation 2). At join points between bending regions and linear segments, G^1 continuity is guaranteed, as the adjacent small linear segment making up the end of the bending region becomes parallel with the linear segment as the number of samples approaches infinity. For cords defined by piecewise polynomial guide curves, the cord's analytic form allows us to ensure higher-order continuity between connected regions, to G^2 or higher. Figure 9 illustrates the approach.

On the left of Figure 9, a bending region of the cord is joined to a linear segment grown to the sphere on the right. By stepping back from the end points of the connecting line segment in either direction by a small value, we can analytically evaluate cord position, tangent, and curvature. We can similarly evaluate these attributes for curve points on the wrapped geometry. Then we can fit a quintic spline to match the tangent and curvature values at both end points. This construction makes a limited local change along the connecting linear segments, but it does not change the cord's overall shape. For the iterative cord generation algorithm, the step size can limit the practical ability to take this approach. This approach can also be discontinuous for animated cords, however, as pairs of bending regions and linear segments can appear or disappear as contact relationships change.



10 (a) A wide cord used to model ribbon. (b) Thick cords wrapped around geometry.

Applications and results

We implemented cords as a plug-in to Autodesk's Maya modeling and animation system. We typically use nonuniform cubic B-splines as guide curves for their ease of use in shape modeling and keyframe animation. To calculate ray intersections, we use the appropriate API for each geometry type, along with the built-in acceleration structures. Users can animate the guide curve, appearance properties, and geometry, and Maya's computational architecture lets us update the cord both interactively and as other keyframe animation plays back. If many complex geometric primitives are used, users can temporarily reduce the guide curve sample rate to avoid objectionable computational lag.

Our notation in the "Cord generation" section assumes all shape properties have constant values. In practice, we allow some to vary along the length of the curve, and users can use a painting interface to specify how shape properties change along the length of the guide curve. The cord's parameterization is directly mapped from the guide curve, as formally expressed in Equation 2.

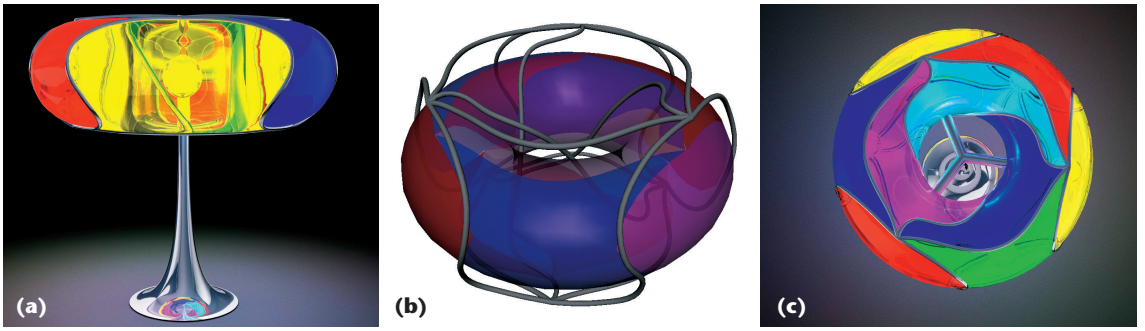
Wide cords and thick cords

Wide cords are a variation of cords used to model primitives that can be represented as long 2D manifolds, such as flat ribbons or straps. Thick cords can represent objects such as thick tubes or ropes. In these cases, we can extend the cord model to have more width or thickness, which might vary along the curve's length, as well as a local orientation that varies along the cord. Orientation is expressed as a vector perpendicular to the curve tangent, which defines the surface normal of a flat cord or the radial orientation of a thick cord. For flat surfaces, we define this normal vector so that the cord surface lies flat on the geometric surface. The Frenet frame definition of a curve normal has singularities and is not well suited for this task. We use the geometric surface normal to define the orientation for contact points and interpolate this orientation vector for other segments of the cord, linearly between neighboring values. Users can specify orientation vectors at the endpoints. Figure 10a demonstrates this technique.

In addition, constraining wide cords against locally convex geometry or thick cords against any geometry requires that the intersection test be replaced with a proximity test in the cord-generation algorithm. The geometry that these cords create is fed back into the algorithm as growth continues to avoid self-intersection. Figure 10b shows an example of a thick cord with self-overlap.

Artistic modeling

We can define artistic primitives such as paint strokes or procedural geometric curves with an underlying curve primitive.[1] Existing software packages allow such artistic primitives to be defined on 2D canvases or embedded within parametric 3D surfaces. These techniques can be extremely limiting, as general 3D curves are difficult to manipulate by hand. Cords can be used to easily specify the locations of such curves in space or about arbitrary geometry.



11 (a) Tiffany lamp K_6 graph. (b) Guide curves have been positioned by an artist to generate (c) aesthetically pleasing cords on the surface of a torus, in the form of a K_6 graph embedding. We use the cords to partition the torus and (c) model geometry connecting the torus partitions.

Motivated by Figure 1d, Figure 11 illustrates the use of cords to interactively embed a nonplanar six-vertex clique (the K_6 graph) on a torus. Emphasis on the curve layout in this example is based on visual aesthetics under direct artist control, rather than using a procedural approach that attempts to follow geodesic paths on the geometry. The curves on the inside of the torus wrap around regions of concavity, where an element of slackness in the cord can help ensure that the generated curve lies on the given geometry. This example illustrates how artists and designers can benefit from having precise control over curves in geometric contact to create expressive shapes.

Cords were also extensively used in the production of the 2004 Oscar-winning animated film *Ryan* (directed by Chris Landreth). Without the ability to interactively specify a curve that maintains continuity in time with a simple interface, various sequences would have never been possible given the time constraints of production.

Future work

Cords used in conjunction with physical simulation models can leverage the advantages that each provides; this is the subject of ongoing work. Another area of investigation is nondirectional cords in which the direction of guide curve parameterization has no effect on the cord shape. Straightforward interpolation of cords grown in each direction—a commonly suggested approach—does not maintain contact relationships correctly. The use of cords for curve design in surfaces, shape modeling of flat and tubular structures, and as an animation tool for a number of shots in the animated film *Ryan*, has demonstrated that cords are a compelling curve primitive for use in complex production environments. ■

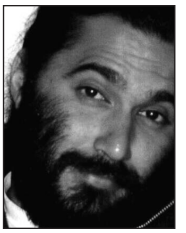
Acknowledgments

We thank Chris Landreth and the *Ryan* production team for pointing out the exceptional difficulty of modeling curves in complex, changing-contact relationships and for creating excellent examples demonstrating the capabilities of cords. The National Film Board of Canada, Copper Heart Entertainment, and Seneca College supported the production of *Ryan*. The Mathematics of Information Technology and Complex Systems Network provided research funding, Tim Hanson helped produce the lamp images, Keith Conrad helped derive the analytic form of stiff cords, and Joe Laszlo provided editorial feedback. Alias and the *Ryan* production team provided a number of models, and we rendered images with software donated by Alias and Pixar Animation Studios.

Reference

1. A. Hertzmann, "Curve Analogies," *Proc. 13th Eurographics Workshop Rendering*, Eurographics Assoc., 2002, pp. 233-246.

Patrick Coleman *is a PhD student at the University of Toronto. His research interests include algorithms for editing character motion, character construction, and animation interfaces. Coleman has a BS in computer science and engineering from The Ohio State University and an MS in computer science from the University of Toronto. Contact him at patrick@dgp.toronto.edu.*



Karan Singh *is an associate professor of computer science at the University of Toronto. His research interests include artist-driven interactive graphics, spanning character animation, anatomic modeling, and geometric shape design. Singh has a BTech from the Indian Institute of Technology, Madras, and an MS and PhD in computer science from The Ohio State University. Contact him at karan@dgp.toronto.edu.*