EUROGRAPHICS 2013 / I. Navazo, P. Poulin
(Guest Editors)

Volume 32 (2013), Number 2

Circular Arc Snakes and Kinematic Surface Generation

M. Bartoň [1] L. Shi [1] M. Kilian [2,3] J. Wallner [3,4] H. Pottmann [1,3]

[1] King Abdullah University of Science and Technology, Saudi Arabia [2] Evolute GmbH, Vienna, Austria
[3] Vienna University of Technology, Austria [4] Graz University of Technology, Austria

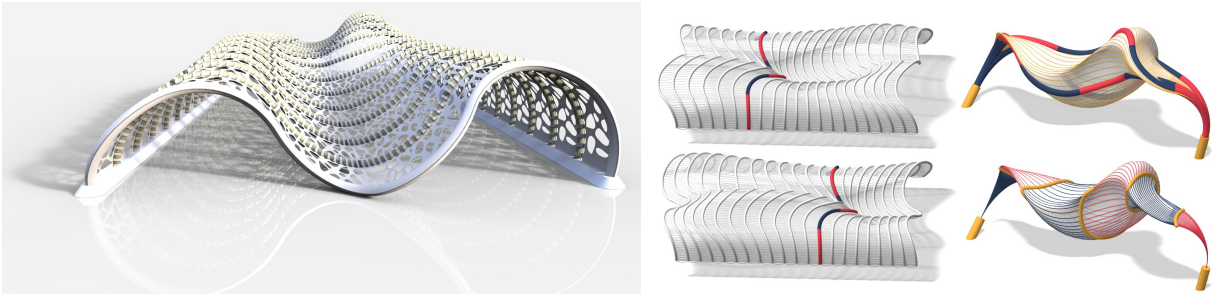

Figure 1: *Non-static architecture: A flexing structure composed of many instances of the same flexible 6-snake of circular arcs, each of which in turn is composed of equal pieces. This fact is very relevant for the cost of manufacturing. In the center column one can see two stages of this flexion: two different snakes which happen to be at the same instant of their own respective flexion are highlighted.* Right: *The motion of the flexible kinematic chain of arcs which this structure is based on (see Section 4.2).*

Abstract

We discuss the theory, discretization, and numerics of curves which are evolving such that part of their shape, or at least their curvature as a function of arc length, remains unchanged. The discretization of a curve as a smooth sequence of circular arcs is well suited for such purposes, and allows us to reduce evolution of curves to the evolution of a control point collection in a certain finite-dimensional shape space. We approach this evolution by a 2-step process: linearized evolution via optimized velocity fields, followed by optimization in order to exactly fulfill all geometric side conditions. We give applications to freeform architecture, including "rationalization" of a surface by congruent arcs, form finding and, most interestingly, non-static architecture.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—

1. Introduction

Several problems in geometry processing are *kinematic* in nature, meaning that they have to do with the motion of rigid bodies. An example is the unsolved problem of deciding if a given surface can be generated by the motion of a certain curve. This paper is concerned with a question of this kind, namely the evolution of curves such that their curvature (as a function of arc length) is unchanged. If the curve in question is treated as an object of differential geometry, of course such an evolution does not have anything to do with the motion of rigid bodies, but for numerical and algorithmic treatment, we discretize a curve as a smooth union of circular arcs, and let it evolve such that the single pieces move in a rigid manner.

Remarkably these discrete curves, which we call *arc snakes*, or shortly snakes, have applications in freeform architectural design. Some are already known, such as the ones shown by Figure 2 (pipe-like covering of facades) and Figure 3 (easy construction of substructures from repetitive el-

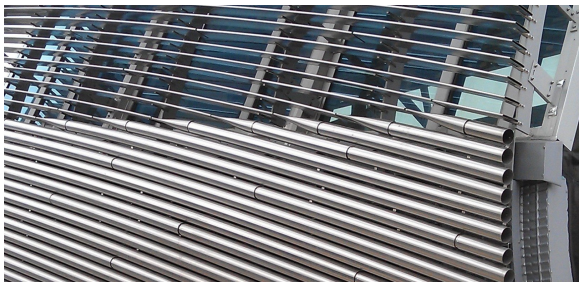

Figure 2: *Tubular and laminar covering of a freeform hull based on the evolution of a curve (photo of a mock-up for a real-world project).*

ements). The most interesting application, however, is non-static architecture.

Previous work. There is quite some work on the approximation of smooth curves by a smooth sequence of arcs ("arc splines"). We refer to the many publications by D. Walton and D. Meek, e.g. [WM95], and also to [SACJ09], [Leo01]. It should be mentioned that the term *snake* was coined by [KWT87] and meant an evolving energy-minimizing curve.

For flexible elements in freeform architecture we refer to current academic work [Tac10], and practical work by Hoberman Associates [Hob]. Flexible structures in general have fascinated people for a long time. Notable examples of flexible surfaces composed of rigid elements are polyhedral surfaces of Voss type [Sau70], or foldable Miura-Ori structures [Miu85]. For the geometric foundations of kinematics we refer to [PW01] and [BR79]. For kinematic surfaces in general we refer to the textbook [PAHK07].

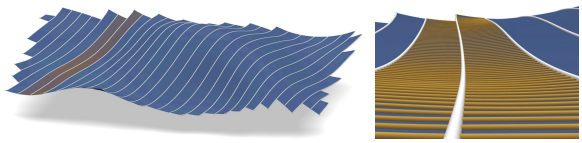

Figure 3: *A roof surface from the Louvre's Islamic Art exhibition is covered by constant radius arcs, which are however not congruent* [BPK*11].

From the viewpoint of architectural geometry, work related to this paper has been done by [BPK*11], who cover surfaces by families of circular arcs whose radius is constant (whose size however is not, see Figure 3). Another example of patterns of curves with special curvature properties is the 1-parameter families of geodesics studied by [PHD*10].

Contributions and overview. This paper is structured as follows: We begin with a general discussion of curves, discrete curves (i.e., snakes), and their evolution (Section 2).

- In Section 3 we show how to compute arc snakes and manipulate them via a first or second order kinematic analysis. We can design surfaces swept by a moving snake with various options of guiding the motion of the snake, e.g. moving in close proximity to a given reference surface, or being guided by rails.
- Section 4 discusses our two main applications. One is rationalization in freeform architecture (Section 4.1): Surfaces traced by the movement of a snakes are efficiently manufacturable because their geometry is represented by many instances of a few different arcs. This especially applies to the manufacturing of formworks for concrete shells.
- Finally, Section 4.2 deals with non-static architecture like actively changing roofs or facades, based on kinematic linkages formed by arc snakes of suitable length.

2. Evolution of smooth and discrete curves.

This section discusses curves and a discrete version of curves which is useful for handling curves with prescribed curvature. Further we look at the evolution of such curves. We start with discretization and then proceed to general curves.

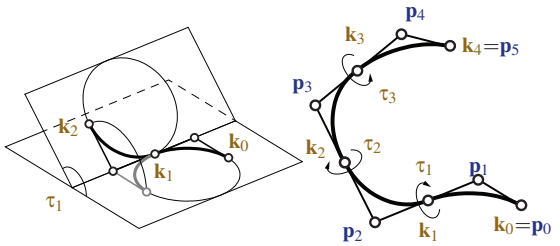

Figure 4: *Left: Torsion angle τ_1 between arcs. Right: A snake consists of N arcs and is defined by control points $\mathbf{p}_i$ and contact points $\mathbf{k}_i$. At each interior contact point $\mathbf{k}_i$ there is a torsion angle τ_i.*

Discrete curves and snakes. For the purposes of this paper, a discrete curve is a smooth union of N circular arcs, which will be called an *arc snake*, or *snake* for short. Much like a quadratic B-spline curve which consists of parabolic arcs, such a snake is encoded by control points $\{\mathbf{p}_i\}$, such that the i-th arc touches the edges $\mathbf{p}_{i-1}\mathbf{p}_i$ and $\mathbf{p}_i\mathbf{p}_{i+1}$ in contact points $\mathbf{k}_{i-1}$ and $\mathbf{k}_i$, respectively. If the curve under consideration is closed, indices are taken modulo N. Otherwise we require $\mathbf{p}_0 = \mathbf{k}_0$ and $\mathbf{p}_{N+1} = \mathbf{k}_N$ (see Figure 4).

This collection of points is not arbitrary, since the two segments emanating in a control point $\mathbf{p}_i$ which are both tangent to the i-th arc must be of equal length:

$$\|\mathbf{p}_i - \mathbf{k}_{i-1}\| = \|\mathbf{p}_i - \mathbf{k}_i\|, \quad i = 1,\dots,N. \tag{1}$$

Curvatures of snakes. Generally a curve in space which is traversed by unit speed defines two functions of arc length: the *curvature* which is the angular velocity of the rotation of the tangent, and the *torsion* which is the angular velocity of the rotation of the osculating plane. For a snake, the curvature is piecewise-constant along each arc, its value being the inverse of the arc's radius. The movement of the osculating plane is not continuous, so it makes sense to consider

an amount "τ_i" of torsion to be concentrated in the contact points $\mathbf{k}_i$, see Fig. 4.

Evolution of snakes. The goal we have in mind is the evolution of a curve such that its curvature, as a function of arc length, remains unchanged, see Fig. 5. Letting a snake evolve with this side condition means that each arc moves as a rigid body, and the smooth join of successive arcs is maintained. The torsion angles τ_i change during evolution.

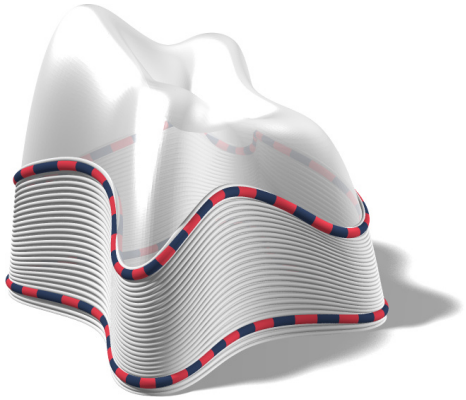

Figure 5: *Evolution of a snake within a surface, modeling a curve whose total arc length and also whose curvature as a function of arc length remains unchanged. See Example 3.1.*

Assume that the initial position of a snake is given by control points $\mathbf{p}_i^0$ together with contact points $\mathbf{k}_i^0$. A new position of the snake is determined by the new positions $\mathbf{p}_i$ of its control points alone, since the location of the new contact points $\mathbf{k}_i$ follows from the distance constraint (1). We write

$$\mathbf{x}^0 = (\mathbf{p}_0^0, \ldots, \mathbf{p}_{N+1}^0), \quad \mathbf{x} = (\mathbf{p}_0, \ldots, \mathbf{p}_{N+1}) \tag{2}$$

for initial position and new position of control points. The evolution of the $\mathbf{p}_i$'s is constrained by the condition that each arc moves in the manner of a rigid body:

$$\|\mathbf{p}_i - \mathbf{p}_j\| = \|\mathbf{p}_i^0 - \mathbf{p}_j^0\|, \quad i-j = 1,2. \tag{3}$$

This amounts to $2N+1$ distance constraints which have to be fulfilled by the point $\mathbf{x} \in \mathbb{R}^{3(N+2)}$.

The shape space of an evolving snake. In the language of robot kinematics, a snake evolves like a serial hinge mechanism, the hinges being associated with the torsion angles. The number of degrees of freedom experienced by an evolving snake is therefore $N+5$ which results from 6 d.o.f. for the position of the snake in space, plus the $N-1$ torsion angles which function as shape parameters.

We get the same result if we start counting from the $3(N+2)$ coordinates of control points and subtract the $2N+1$ distance constraints of (3) (this also means that when formulated in terms of control points, a snake's evolution is described by a bar and joint mechanism.)

Anyway we conclude that the *shape space* of a snake, i.e., the set of all admissible "$\mathbf{x}$", is an algebraic variety, generically of dimension $N+5$, in $\mathbb{R}^{3(N+2)}$. That shape space will be denoted by $\mathcal{M}$. A similar discussion shows that for closed snakes the shape space is N-dimensional, provided $N \geq 6$.

Evolution velocities. Later we need the individual speed $\dot{\mathbf{p}}_i$ of a control point $\mathbf{p}_i$ while a snake is evolving smoothly with time t. Differentiating the square of the constraints (3) yields

$$\langle \mathbf{p}_i - \mathbf{p}_j, \dot{\mathbf{p}}_i - \dot{\mathbf{p}}_j \rangle = 0, \quad i-j = 1,2. \tag{4}$$

Conversely, if the evolution velocity satisfies (4) for all times t, then the corresponding distances of control points remain unchanged. The collection of velocities,

$$\dot{\mathbf{x}} = (\dot{\mathbf{p}}_0, \ldots, \dot{\mathbf{p}}_{N+1})$$

represents a tangent vector of the shape space $\mathcal{M}$. If a snake's evolution is not fully specified, but only the tangent vector $\dot{\mathbf{x}}$ at time t is known, then the state of the snake at time $t+h$ approximately equals $\mathbf{x}+h\dot{\mathbf{x}}$, but this *tangential evolution* (Fig. 6) is only a 1st order approximation to a true evolution, as the constraints (3) are obeyed only in a linearized manner.

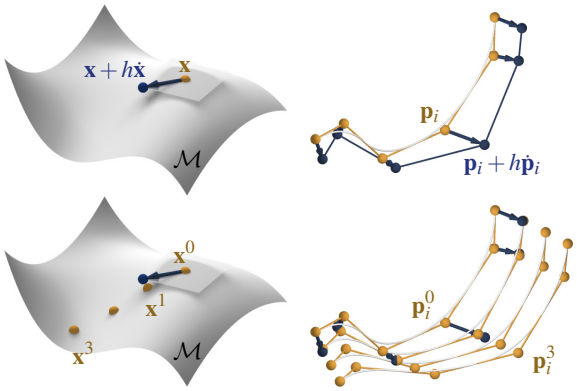

Figure 6: *The shape space viewpoint of snakes. The right hand images show control polygons and the corresponding arc snakes; the left hand column illustrates shape space.* Top: *Tangential evolution of snakes. The collection* $\mathbf{x} = (\mathbf{p}_0, \ldots, \mathbf{p}_{N+1})$ *of control points is updated by* $\mathbf{x} \mapsto \mathbf{x}+h\dot{\mathbf{x}}$. *This first order evolution is not exact and causes the control polygon to leave the shape space* $\mathcal{M}$. Bottom: *Exact evolution requires backprojection onto* $\mathcal{M}$, *see Sec. 3.2.*

Evolution of a snake within a surface. In the course of collecting useful properties of snakes we now consider the evolution of a snake such that it remains as close as possible to a given reference surface Φ (see Figures 7 and 5 for examples). It turns out that we can expect three essential degrees of freedom for the evolution of an open snake.

This is seen by the following argument: An evolution of a snake is characterized by contact points and also the midpoints of arcs moving tangentially to Φ. Since those points

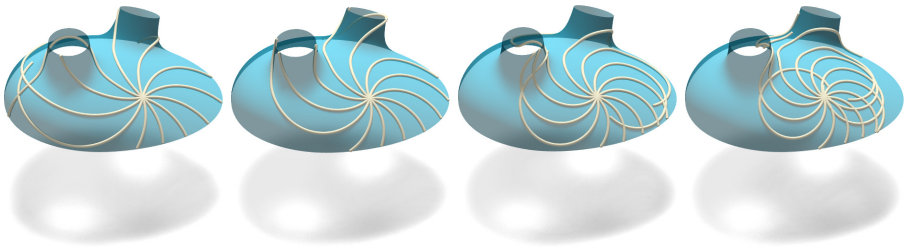

Figure 7: *Curves of constant curvature, determined by initial point and tangent vector (visualizing the three degrees of freedom stated by Prop. 2.1). The curvature increases from left.*

can be expressed as linear combinations (with constant coefficients) of the $\mathbf{p}_i$'s, the evolution is actually characterized by $\mathbf{p}_0, \ldots, \mathbf{p}_{N+1}$ moving tangentially to Φ. This amounts to $N+2$ conditions. Since unconstrained evolution has already been shown to have $N+5$ d.o.f., this leaves 3 d.o.f. for evolution constrained to Φ.

Evolution of a general curve within a surface. The previous paragraph established that evolution of a snake within a surface has 3 degrees of freedom. The same d.o.f. count occurs for rigid motions in $\mathbb{R}^2$, and for the motion of geodesics in surfaces [dC76]. Below we show the interesting fact that the evolution of a curve within a surface has also 3 degrees of freedom if the curvature is prescribed (see Fig. 7, this can be seen as a point in favour for using snakes as discrete curves).

Proposition 2.1. *For a given surface Φ, curvature function $\kappa(s) \geq 0$, initial point $\mathbf{c}(s_0)$ and initial unit tangent vector $\mathbf{c}_s(s_0)$ there exist generically and locally two curves $\mathbf{c}(s)$ in Φ which are parametrized by arc length s and which fit both the initial values and the given curvature function.*

In conclusion, evolution of a curve with invariant curvature function has 3 degrees of freedom.

Proof. We show that once we have decided if the sought-after curve makes a left turn or a right turn, it obeys a 2nd order differential equation. It thus smoothly depends on the initial conditions, which amount to 2 d.o.f. for the location of the initial point $\mathbf{c}(s_0)$ and one d.o.f. for the unit tangent vector $\mathbf{c}_s(s_0)$ (using subscript notation for differentiation w.r.t. s).

To find this equation we observe that geodesic curvature κ_g, normal curvature κ_n, and curvature κ of the curve $\mathbf{c}$ obey $\kappa_g^2 + \kappa_n^2 = \kappa^2$ and $\mathbf{c}_{ss}^{\text{tang}} = \kappa_g J \mathbf{c}_s$, where J is rotation about $+90$ degrees in the tangent plane, and the superscript "tang" indicates the tangential component of a vector. Further, κ_n depends on $\mathbf{c}_s$ only [dC76]. This implies $\mathbf{c}_{ss}^{\text{tang}} = \pm\sqrt{\kappa^2 - \kappa_n(\mathbf{c}_s)^2} J\mathbf{c}_s$, where the sign decides if the curve makes a left turn or a right turn. □

It would be nice if a curve could evolve such that its curvature remains constant, and at the same time the evolution velocity of each point is orthogonal to the curve. Unfortunately this is not possible in general:

Proposition 2.2. *Assume an evolution of curves $\mathbf{c}^{(t)}(s)$ in time t, where all curves are traversed with unit speed and the evolution velocity is perpendicular to the curve. Then these curves are geodesics.*

Proof. We use a dot to indicate differentiation w.r.t. evolution time t. We observe the conditions $\langle \mathbf{c}_s, \mathbf{c}_s \rangle = 1$ of unit speed and $\langle \mathbf{c}_s, \dot{\mathbf{c}} \rangle = 0$ of orthogonality and differentiate them w.r.t. arc length s and also w.r.t. evolution time t. This yields $\langle \mathbf{c}_{ss}, \mathbf{c}_s \rangle = 0$, $\langle \dot{\mathbf{c}}_s, \mathbf{c}_s \rangle = 0$, $\langle \mathbf{c}_{ss}, \dot{\mathbf{c}} \rangle + \langle \mathbf{c}_s, \dot{\mathbf{c}}_s \rangle = 0$, adding up to $\mathbf{c}_{ss}$ being orthogonal to the independent tangent vectors $\dot{\mathbf{c}}$ and $\mathbf{c}_s$. This property characterizes geodesics [dC76]. □

These remarks conclude the section on definitions and properties of snakes and their evolution. We proceed to Section 3, which deals with implementation and algorithms.

3. Implementation and numerics of snake evolution.

Our numerical approach to the evolution of a snake is based on two basic ingredients:

- One is to compute evolution velocities, which enables us to approximately do one evolution step. Obviously this computing of velocities depends on the application.
- The other one is to optimize control points which are not quite admissible in order to make them so (see Fig. 6).

We let snakes evolve by iterating between these two procedures. If evolution is constrained to a surface, then a third ingredient is important for finding snakes in the first place:

- Initialization: computing snakes of prescribed curvature which are constrained to surfaces.

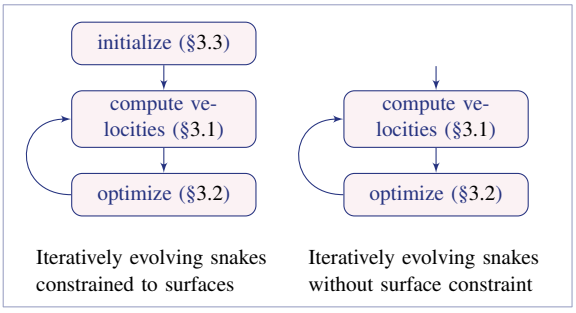

Figure 8: *Snake evolution flowchart*

From the abstract viewpoint of shape spaces (see Figure 6), finding velocities is the computation of tangent vectors of

the shape space $\mathcal{M}$, while optimization amounts to applying closest-point projection onto $\mathcal{M}$ to points which almost, but not quite, lie in $\mathcal{M}$. The above-mentioned three ingredients are discussed in the following subsections. In any concrete example we may have to employ them in a different order, e.g. starting with initialization, and subsequently performing evolution, alternating between projection and the computation of velocities, see Fig. 8.

3.1. Computing evolution velocities.

We describe how to compute evolution velocities $\dot{\mathbf{x}} = (\dot{\mathbf{p}}_0, \ldots, \dot{\mathbf{p}}_{N+1})$ for various applications, starting with the evolution of a snake which is constrained to a surface.

1st order evolution of snakes in surfaces. We have already discussed that for all practical purposes, evolution of a snake within a surface is characterized by single evolution velocities $\dot{\mathbf{p}}_i$ being tangential to Φ. We also make the model assumption that the speed of evolution is given or at least suggested by the application at hand: Around the control point $\mathbf{p}_i$, it is desired that the snake moves sideways with the velocity d_i (measured orthogonal to the current position of the snake). We find a suitable $\dot{\mathbf{x}}$ by minimizing an appropriate target functional

$$\mathcal{F}_{\text{gliding}}(\dot{\mathbf{x}}) = \mathcal{F}_{\text{tang}}(\dot{\mathbf{x}}) + \lambda \mathcal{F}_{\text{offset}}(\dot{\mathbf{x}}), \tag{5}$$

where $\mathcal{F}_{\text{tang}}$ penalizes deviation from the reference surface and $\mathcal{F}_{\text{offset}}$ strives to achieve the desired sideways speed. The factor $\lambda > 0$ steers the importance of $\mathcal{F}_{\text{offset}}$. We let

$$\mathcal{F}_{\text{tang}}(\dot{\mathbf{x}}) = \sum\nolimits_i \langle \mathbf{n}_i, \dot{\mathbf{p}}_i \rangle^2, \quad \mathcal{F}_{\text{offset}}(\dot{\mathbf{x}}) = \sum\nolimits_i (\langle \dot{\mathbf{p}}_i, \mathbf{d}_i \rangle - d_i)^2.$$

Here $\mathbf{n}_i$ is unit normal vector of Φ at the footpoint of $\mathbf{p}_i$, and $\mathbf{d}_i$ is a unit vector tangential to the surface and orthogonal to the snake. Minimization of $\mathcal{F}_{\text{gliding}}$ is performed within the linear space of admissible $\dot{\mathbf{p}}_i$'s which obey (4). Note that even if no sideways velocities are specified, we can use $\mathcal{F}_{\text{offset}}$ as a regularizer, by choosing $\lambda \ll 1$, $d_i = 1$.

Further objectives when modeling evolution. Controlling the speed of evolution via $\mathcal{F}_{\text{offset}}$ is not the only aim we might have. By adding appropriate terms to the target functional (5) we achieve various properties of the evolution:

- We might require the velocities $\dot{\mathbf{p}}_i$ to be close to given vectors $\dot{\mathbf{p}}_i^0$, or to planes with normal vectors $\mathbf{n}_i$. This is done by adding $\mathcal{F}_{\text{prox}}(\dot{\mathbf{x}}) = \sum_i w_i \|\dot{\mathbf{p}}_i - \dot{\mathbf{p}}_i^0\|^2 + \tilde{w}_i \langle \dot{\mathbf{p}}_i, \mathbf{n}_i \rangle^2$ to (5). Here weights $w_i, \tilde{w}_i$ encode the importance which is attached to the individual proximity conditions.
- Our aim might be that velocities are orthogonal to the snake (i.e., $\dot{\mathbf{p}}_i$ is orthogonal to the plane carrying the arc belonging to $\dot{\mathbf{p}}_i$, which is assumed to be spanned by basis vectors $\mathbf{e}_i^+, \mathbf{e}_i^-$). We do that by augmenting (5) by

  $$\mathcal{F}_{\text{ortho}}(\dot{\mathbf{x}}) = \sum\nolimits_i \langle \dot{\mathbf{p}}_i, \mathbf{e}_i^+ \rangle^2 + \langle \dot{\mathbf{p}}_i, \mathbf{e}_i^- \rangle^2.$$

  Prop 2.2 states that orthogonality can in general not be achieved, but anyway $\mathcal{F}_{\text{ortho}}$ may act as a regularizer.
- We might require the differential distance constraint (4) also for control points further apart, by adding

  $$\mathcal{F}_{\text{rot-min}}(\dot{\mathbf{x}}) = \sum\nolimits_i \langle \mathbf{p}_i - \mathbf{p}_{i+3}, \dot{\mathbf{p}}_i - \dot{\mathbf{p}}_{i+3} \rangle^2.$$

  The name 'rotation minimizing functional' is not directly related to the well known rotation minimizing frames associated with space curves [Bis75], but to the fact that $\mathcal{F}_{\text{rot-min}} \to 0$ causes the snake to move as rigidly as possible, with torsion angles changing minimally.

3.2. Closest-point projection onto shape space.

Similar to the text around Equation (2), we assume a snake is given and has control points $\mathbf{x}^0 = (\mathbf{p}_0^0, \ldots, \mathbf{p}_{N+1}^0)$. The corresponding shape space $\mathcal{M}$ is the set of possible control points of snakes which are reachable via evolution of the given snake. It is defined by the constraints (3).

Assume further that a control point collection $\mathbf{y}$ is given which does not lie in $\mathcal{M}$. We wish to find $\mathbf{x} \in \mathcal{M}$ closest to $\mathbf{y}$. This *closest point projection* of ambient space onto $\mathcal{M}$ is formulated as least squares optimization of a nonlinear target functional $\mathcal{F}$, constructed so as to penalize deviation of $\mathbf{x}$ from $\mathbf{y}$, and to penalize violation of (3):

$$\mathcal{F}(\mathbf{x}) = \mathcal{F}_{\text{geom}}(\mathbf{x}) + \mu \|\mathbf{x} - \mathbf{y}\|^2 \to \min, \quad \text{where}$$
$$\mathcal{F}_{\text{geom}}(\mathbf{x}) = \sum\nolimits_{i-j \in \{1,2\}} \left(\|\mathbf{p}_i - \mathbf{p}_j\| - \|\mathbf{p}_i^0 - \mathbf{p}_j^0\| \right)^2.$$

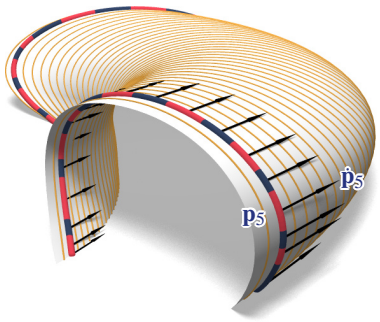

Figure 10: *Evolution of a snake via optimization of velocity vectors $\dot{\mathbf{p}}_i$ at each discrete time step. Evolution updates control points by the rule $\mathbf{p}_i \longmapsto \mathbf{p}_i + h\dot{\mathbf{p}}_i$, followed by back-projection onto the shape space. For more details on this particular figure see Ex. 4.1.*

Using closest point projection for evolution. Assume that we have, from Section 3.1, a procedure for computing a collection of evolution velocities $\dot{\mathbf{x}}$ for a control point collection $\mathbf{x}$. How do we now move $\mathbf{x}$ in direction of $\dot{\mathbf{x}}$? The simplest way is by choosing a time increment h, updating control points by $\mathbf{x} \longmapsto \mathbf{x} + h\dot{\mathbf{x}}$, and applying one round of closest-point projection onto the shape space according to the formulae in the previous paragraph. Now we can compute velocity vectors again and iterate (see Figs. 6, 10).

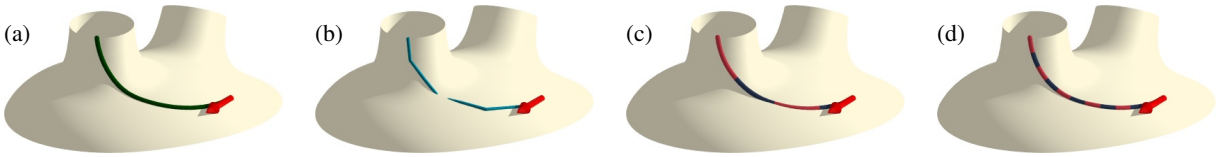

Figure 9: *Illustration of the initialization procedure of Section 3.3. (a) Dense polyline approximating a curve of constant curvature in a given mesh. (b) Initial guess at control points for a snake which follows that curve. (c) The result of optimization is a snake with 4 arcs. (d) An analogous procedure produces a 12-arc snake of constant curvature.*

3.3. Initialization

This section discusses how to find a snake of given curvature which lies in a surface Φ, which itself is given as a triangle mesh. This is a rather lengthy procedure. Our discussion first deals with the smooth case, then with a numerical integration scheme on the triangle mesh Φ, and finally describes how to make use of the optimization techniques described above.

Computing curves of prescribed curvature in surfaces. Consider a parametric surface $\mathbf{x}(u,v)$ and a curve $\mathbf{c}(s) = \mathbf{x}(u(s),v(s))$ in that surface. For the following discussion, subscripts indicate differentiation. The point $\mathbf{c}(s)$ is to travel with unit speed, i.e., $\|\mathbf{c}_s\|^2 = 1$. This condition expands to

$$\|u_s\mathbf{x}_u + v_s\mathbf{x}_v\|^2 = 1. \tag{6}$$

For such curves the curvature simply equals the length of the 2nd derivative, i.e., $\kappa = \|\mathbf{c}_{ss}\|$. The chain rule yields

$$\kappa^2 = \left\|u_{ss}\mathbf{x}_u + v_{ss}\mathbf{x}_v + u_s^2\mathbf{x}_{uu} + 2u_sv_s\mathbf{x}_{uv} + v_s^2\mathbf{x}_{vv}\right\|^2. \tag{7}$$

Our numerical scheme discretizes the curve $\binom{u(s)}{v(s)}$ as a polyline in the uv parameter domain. We show how to iteratively compute its vertices. Suppose vertices $\binom{u_i}{v_i}$, $\binom{u_{i+1}}{v_{i+1}}$ are already known. The parameter increment h_i between these two instances of the curve is not yet known, but in any case the discrete derivative equals $\binom{u_{s,i}}{v_{s,i}} = \frac{1}{h_i}\binom{u_{i+1}-u_i}{v_{i+1}-v_i}$. Thus we can determine h_i by requiring that $\binom{u_{s,i}}{v_{s,i}}$ fulfills (6).

The derivative $\binom{u_{s,i+1}}{v_{s,i+1}}$ is also unknown, but fulfills (6) and is related to 2nd derivatives via $\binom{u_{ss,i}}{v_{ss,i}} = \frac{1}{h_i}\binom{u_{s,i+1}-u_{s,i}}{v_{s,i+1}-v_{s,i}}$. We plug this relation into (7). Obviously, (6) reduces Equation (7) to a linear equation for the unknowns $\binom{u_{s,i+1}}{v_{s,i+1}}$. Omitting the details, (7) describes a straight line whose distance from the origin goes to zero as $h_i \to 0$. Since (6) describes an ellipse centered in the origin, we conclude that for small h_i we have 2 solutions (in accordance with Prop. 2.1). Which one to choose depends on our choice if the curve is supposed to make a left hand turn or a right hand turn. This procedure is now iterated. Once an entire sequence of parameter values u_i, v_i is known, we let $\mathbf{c}_i = \mathbf{x}(u_i,v_i)$ and we are done.

Computing curves of prescribed curvature in meshes. In order to apply the procedure described in the previous paragraph to a triangle mesh we choose a local coordinate system in each face and fit a cubic polynomial to a mesh-neighbourhood of that face. Thus we replace the mesh by a parametric surface and can apply the procedure described above as long as we work over the face under consideration. After leaving that face, we repeat the procedure for the next face, and so on (see Fig. 9a).

Fitting snakes to curves. Having computed a numerical polyline representation $\{\mathbf{c}_i\}$ of a curve of prescribed curvature, we now fit a snake to that curve. S We want to approximate the given polyline $\{\mathbf{c}_i\}$ by a snake consisting of N congruent arcs, where the j-th arc has radius r_j, with $1/r_j$ as an average curvature of the original curve in the interval being replaced by the j-th arc.

Uniform sampling of the given polyline yields initial guesses of $N-1$ interior contact points $\mathbf{k}_j$ (see Fig. 4), and the polyline's unit tangent vectors $\mathbf{t}_j$ there yield initial guesses for the control points, as follows: The distance d_j of control point and contact point is known from the arcs' radius r_j and length L_j as $d_j = r_j \tan\frac{L_j}{2r_j}$, so both $\mathbf{k}_{j-1} + d_j\mathbf{t}_{j-1}$ and $\mathbf{k}_j - d_j\mathbf{t}_j$ should result in the same control point. In our computations we initialize the control points $\mathbf{p}_j$ as the arithmetic mean of these expressions (Figure 9b).

A subsequent round of optimization applied to $\mathbf{x} = (\mathbf{p}_0,\ldots,\mathbf{p}_{N+1})$ achieves the exact desired geometric properties of the control points (see Fig. 9c; very similar to Section 3.2), as well as proximity of the snake to the input data. In order to measure that deviation we introduce the center $\mathbf{o}_j$ of the j-th arc which is a fixed linear combination of the control points; and we use the notation $\mathbf{c}_{j,k}$ for those samples of the given polyline which belong the j-th arc. Summing up, we minimize $\mathcal{F}_{\text{geom}}(\mathbf{x}) + \nu\mathcal{F}_{\text{prox}}(\mathbf{x})$, where

$$\mathcal{F}_{\text{geom}}(\mathbf{x}) = \sum_{i-j\in\{1,2\}}(\|\mathbf{p}_i - \mathbf{p}_j\| - d_{ij}^{\text{intended}})^2,$$
$$\mathcal{F}_{\text{prox}}(\mathbf{x}) = \sum_j \tfrac{1}{\#\text{samples}}\sum_k(\|\mathbf{o}_j - \mathbf{c}_{j,k}\| - r)^2.$$

3.4. Improving Opimization

Computing evolution velocities as described in Sec. 3.1 allows us to perform a first order update, followed by a closest-point projection. It is not difficult to improve this first order method and do a second order update of the form

$$\mathbf{x} \longmapsto \mathbf{x} + h\dot{\mathbf{x}} + \frac{1}{2}h^2\ddot{\mathbf{x}},$$

which is based on a velocity vector $\dot{\mathbf{x}}$ and an acceleration vector $\ddot{\mathbf{x}}$. Such a second order update allows greater time

steps. To achieve this, we differentiate (4):

$$\langle \mathbf{p}_i - \mathbf{p}_j, \ddot{\mathbf{p}}_i - \ddot{\mathbf{p}}_j \rangle + \|\dot{\mathbf{p}}_i - \dot{\mathbf{p}}_j\|^2 = 0, \quad i-j = 1,2. \tag{8}$$

Given control points $\mathbf{x} = (\mathbf{p}_0, \ldots, \mathbf{p}_{N+1})$ and evolution velocities $\dot{\mathbf{x}} = (\dot{\mathbf{p}}_0, \ldots, \dot{\mathbf{p}}_{N+1})$, this is a linear system for the 2nd derivative vectors $\ddot{\mathbf{x}} = (\ddot{\mathbf{p}}_0, \ldots, \ddot{\mathbf{p}}_{N+1})$.

For the evolution of snakes constrained to surfaces we now show how to choose $\ddot{\mathbf{x}}$ subject to conditions (8). This is done by solving an optimization problem, which involves the following data:

- closest-point projection π onto the reference surface Φ,
- footpoints $\mathbf{q}_i = \pi(\mathbf{p}_i + h\dot{\mathbf{p}}_i)$,
- the tangent planes $T_{\mathbf{q}_i}$ in those footpoints.

We minimize the target functional

$$\begin{aligned} \mathcal{F}_{2\text{nd}}(\ddot{\mathbf{x}}) &= \sum\nolimits_i \operatorname{dist}(T_{\mathbf{q}_i},\ \mathbf{p}_i + h\dot{\mathbf{p}}_i + \tfrac{h^2}{2}\ddot{\mathbf{p}}_i)^2 \\ &\quad + \alpha \sum\nolimits_i \operatorname{dist}(\mathbf{q}_i,\ \mathbf{p}_i + h\dot{\mathbf{p}}_i + \tfrac{h^2}{2}\ddot{\mathbf{p}}_i)^2 \quad (\alpha \ll 1) \end{aligned} \tag{9}$$

within the linear space of solutions of the linear system (8). Here the function dist() refers to the Euclidean distance between the two arguments. The particular form of $\mathcal{F}_{2\text{nd}}$ is intended to penalize deviation from Φ in a nicely linearized manner, but still allowing tangential movement. This is achieved by measuring distances from tangent planes, with a small regularizing contribution which involves distances from footpoints.

3.5. Examples

Example 3.1. Figure 5 shows the evolution of a closed snake within the design surface of the Warszawa *Lilium tower*. Evolution endeavours to keep the distance of successive positions of the snake constant, but this is of course not possible, since the snake's arc length is also invariant. Evolution works from the bottom upwards and stops when the deviation from the reference surface exceeds a certain threshold. This happens when the surface becomes too highly curved to allow the snake to proceed.

Example 3.2. In order to visualize the kinematics of a snake on a surface and its 3 degrees of freedom of movement, we fix one end point of a snake and let the snake glide on the surface (by "rotation" around the fixed center). The result looks just like Figure 7.

Example 3.3. Figure 11 shows open curves of constant curvature (modeled by a snake) evolving along the outer offset of the mesh shown in the figure (*"Skipper Library"* design by Formtexx). For evolution, velocities have been optimized using $\mathcal{F}_{\text{offset}}$ and $\mathcal{F}_{\text{rot-min}}$ with weights 1 and 10, resp.

Example 3.4. Figure 12 illustrates the limitations of gliding of a snake in a surface. The evolution is optimized such that it is as rigid as possible, using $\mathcal{F}_{\text{rot-min}}$. One can observe that not in all cases the snake is able to reproduce surface features of high curvature, and deviates from the reference surface to some extent.

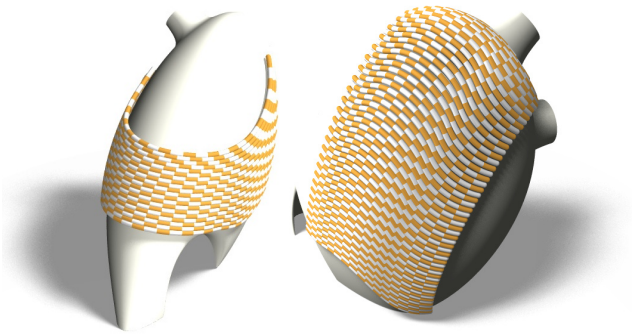

Figure 11: *Both figures show a curve of constant curvature evolving along a mesh Φ (Ex. 3.3). The evolution is optimized using the target functionals $\mathcal{F}_{\text{rot-min}}$. For regularization, $\mathcal{F}_{\text{offset}}$ is also used (otherwise the successive snakes would not cover Φ nicely).*

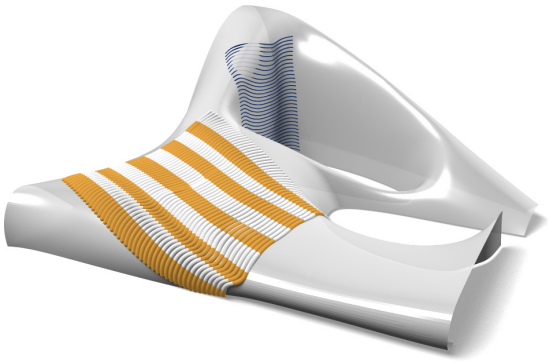

Figure 12: *Evolution of a snake within a surface (Ex. 3.4). The evolution of these curves is as rigid as possible, and curvature is unchanged. The reference mesh is rendered as a transparent surface, one can observe to which extent the evolution is able to reproduce the surface's features.*

Implementation details. The nonlinear optimization problems which occur in this paper are solved by a damped Gauss-Newton with line search. Our linear solver uses CHOLMOD for sparse Cholesky factorization [CDHR08]. As to the choice of weights $\lambda, \mu, \nu, \alpha$: In this academic implementation the weights were chosen manually. we used $\lambda = 0.1$ (seems to be not critical) and $\mu = \nu = 0.005$ (higher values won't work). The value α in the distance term was chosen as 0.01 (again, this is not critical and works just as well with $\alpha = 0.001$).

4. Applications

We have in mind mostly the application area of freeform architecture. There are several different particular applications where the concepts and algorithms developed in previous sections play a role.

4.1. Efficient manufacturing of auxiliary constructions.

It has been observed before that approximating a shape by a collection of circular arcs makes it possible to manufacture that shape by using only simple elements [BPK*11]. One application of this is supporting elements which are not visible in the final building, either because they are inside some outer skin, or they are present only during building construction (e.g. when realizing freeform shapes in concrete).

Previous sections demonstrated that we are typically able to nicely cover the greater part of a given reference surface by an evolving snake of constant curvature, see e.g. Figure 11. Such a covering would mean that we are able to approximate the given design surface by a smooth surface whose manufacturing is greatly simplified by the repetition of a *single* curved element, namely an arc of constant radius and length. We should mention that [BPK*11] also deals with the covering of a surface by circular arcs, but not with arcs having the same length.

Surface generation with snakes. *Functional* architectural design already takes manufacturing into account. The evolution of snakes is a valuable tool in this respect, since it allows us to generate surfaces which possess a simple intrinsic structure useful for manufacturing in the same ways as mentioned by the previous paragraph, with the difference that the entire surface is a priori generated in a kinematically simple way.

Example 4.1. We can guide certain points along curves (*rails*), by making evolution velocities equal to the derivatives of these curves, using $\mathcal{F}_{\text{prox}}$: In Figure 10 the snake's endpoints are guided along rails, and in addition $\mathcal{F}_{\text{rot-min}}$ has been used for the rotation-minimizing property. Guiding the snake along a rail without requiring the rail to be an evolution path can also be done with $\mathcal{F}_{\text{prox}}$: in each evolution step we constrain the evolution velocity of the contact point to lie in plane spanned by the tangents of snake and rail (see Figures 13 and 15).

Example 4.2. *Kinematic surfaces:* In continuation of Example 4.1 we consider a snake whose boundary runs along rails such that the distance of boundary points is constant; we have one additional interior guiding rail (see Figure 14). Here the rotation-minimizing property actually makes all instances at times t_i of the snake *congruent*: instead of evolving the initial snake until time t_i we could simply move it by a rigid body motion such that the boundary points match those at time t_i, and use the remaining degree of freedom of rotation about the boundary to put the snake in a position where it hits the interior guiding rail.

4.2. Flexible elements and non-static architecture.

Non-static architecture has great appeal and is a topic of current interest, even if it appears that little of the many available results on mechanisms and flexible structures has been

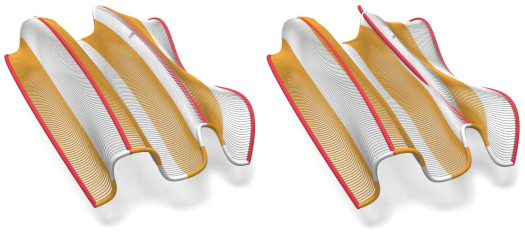

Figure 13: *Surface generation by snake evolution yields shapes which are manufacturable using many instances of a single curved element (shown here in yellow and white). Here a snake evolves such that its boundary moves along 2 rails, while the snake itself glides in a manner tangential to further guiding rails. The evolution is optimized towards the rotation-minimizing property.*

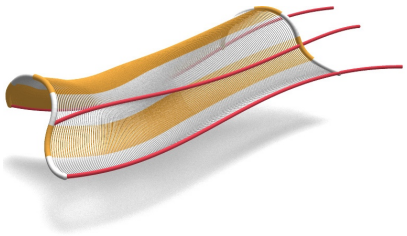

Figure 14: *Kinematic surfaces, continuing Figure 13: Since here we have only 1 interior guiding rail, the small number of constraints on the evolution lets optimization make instances of the snake congruent.*

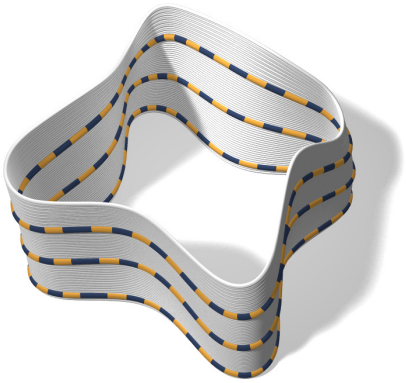

Figure 15: *Evolution of a snake using $\mathcal{F}_{prox}$ (in order to guide 2 vertices along rails) and $\mathcal{F}_{offset}$ with a constant vector $\mathbf{d}_i$ pointing upwards, in order to achieve equal spacing.*

systematically applied to architecture. We show how circular arc snakes can be employed in creating flexing roofs.

We consider an arc snake consisting of N successive arcs, constrained such that the arcs at the boundary are allowed only one rotational degree of freedom, instead of the usual six. We are left with $(N+5)-10$ degrees of freedom, and

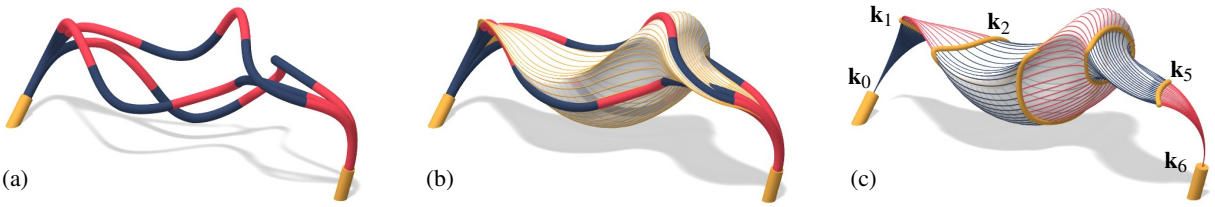

Figure 16: *Arc snake which is the basis of non-static architecture. (a) 3 instances in time of a 1-parameter evolution of a 6-arc snake with boundary constraints. (b) the same, together with the entire motion cycle. (c) The paths of the contact points during evolution:* $\mathbf{k}_1, \mathbf{k}_5$ *are moving back and forth (but not in a synchronized manner), while* $\mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4$ *travel around a loop. This fact is important for actuating the flexion.*

in case we use 6 arcs, as illustrated by Figure 16, we get the most interesting case of 1 degree of freedom. The following text shows how to compute the flexion of this snake and how to build non-static architecture from it.

Computing evolutions. A closer inspection of the flexion of snakes constrained in this way – see Figure 16 – shows that on a real-world model, the flexion cannot always be actuated by simply rotating the first arc: Here, A complete motion cycle of the snake causes the 1st arc simply to rock back and forth. The same is true for the last arc, but the two rocking motions are not synchronous. Figure 16c shows that only the interior part of the snake does anything like a 'rotation'. However, we can make the snake move by driving *either* the first *or* the last arc, and obviously we have to use less force if we drive that arc which is moving faster than the other. This observation is guiding our evolution algorithm:

0. As a preparation, choose whether to drive the snake by the 1st or by the 6th arc, by letting $i_0 = 2$ or $i_0 = 5$. Further choose which way to turn. We start from an admissible position $\mathbf{x} = (\mathbf{p}_0, \ldots, \mathbf{p}_7)$ of control points. Note that control points $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_6, \mathbf{p}_7$ are fixed during evolution.
1. The linearized distance constraints (4) together with the conditions $\dot{\mathbf{p}}_0 = \dot{\mathbf{p}}_1 = \dot{\mathbf{p}}_6 = \dot{\mathbf{p}}_7 = 0$ define a 1-dimensional linear space of possible velocities $\dot{\mathbf{x}}$. Choose a solution where $\dot{\mathbf{p}}_{i_0}$ has a prescribed magnitude, and from the 2 vectors with this property choose the one which points in roughly the same direction as in the previous time step.
2. Perform 1st order evolution, combined with backprojection onto the shape space according to Section 3, with the obvious additional constraints on the 1st and 6th arc.
3. Compare the angular velocites of the 1st and 6th arc: if $\|\dot{\mathbf{p}}_{i_0}\| < \|\dot{\mathbf{p}}_{7-i_0}\|$, switch to the other end of the snake for driving it (i.e., let $i_0 := 7 - i_0$). Repeat from step 1 until we reach a position we had before.

Surfaces from 6-arc snakes. We now go one step further and imagine a sequence of snakes which are composed of the same arcs, but which are at a different instance of its flexion, see Figure 17. If we cause all snakes to flex simultaneously we achieve a flexing series of arcs which e.g. constitute a flexing roof structure such as illustrated by Figures 1 and 17.

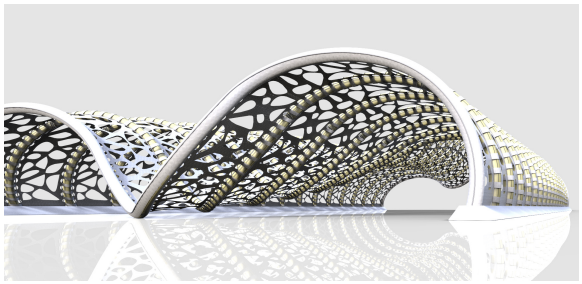

Figure 17: *Non-static architecture made from copies of the flexing arc snake shown in Fig. 16, each in a different state of its flexion. All snakes are actuated in a synchronous manner, creating a wave-like effect. See also Fig 1.*

Arc snakes as 6R linkages. A sequence of rigid bodies, starting with a zeroth "fixed" system and 6 further ones, is called a 6R kinematic chain, or 6R linkage, if each is connected to the previous one with a hinge. Our 6-arc snakes are just that, with hinge axes spanned by $\mathbf{k}_i, \mathbf{p}_{i+1}$ $(i = 0, \ldots, 5$, see Fig. 4); in addition the 6th arc is further constrained by another axis of rotation passing through $\mathbf{k}_6, \mathbf{p}_6$. Since the *inverse kinematic problem* for 6R linkages is solved, one can compute the position of arcs No. 1–5 from the position of the 6th arc [HPS07]. Unfortunately this procedure is multi-valued and of high algebraic degree, so it is not so useful for obtaining an explicit description of the flexion of our snakes.

Singularities and design of moving snakes. A classification of the types of motion which can possibly occur when one of our 6-arc snakes moves, is beyond this paper. In order get an impression, the interested reader is referred to the classification of 4-bar mechanisms instead, see [Hun78]. The same is true for designing the desired shape of flexions, where we cannot hope to have a systematic approach. A little bit of guidance is offered by geometric knowledge:

A *singular* position of a 6R linkage means that actuating the 6 d.o.f. of rotation yields not 6 but only 5 degrees of freedom in its 6th element. It is well known that this occurs exactly when the Plücker coordinates of rotation axes are

linearly dependent [PW01], i.e., when

$$\det \begin{pmatrix} \mathbf{p}_1 - \mathbf{k}_0 & \dots & \mathbf{p}_6 - \mathbf{k}_5 \\ \mathbf{k}_0 \times (\mathbf{p}_1 - \mathbf{k}_0) & \dots & \mathbf{k}_5 \times (\mathbf{p}_6 - \mathbf{k}_5) \end{pmatrix} = 0.$$

In this case we cannot hope to constrain the 6th arc in the way we do without halting the flexion altogether; such singular positions must therefore be avoided. Being close to a singularity will likely produce interesting motions, with large differences in the velocities of the individual arcs, such as shown by Figure 18.

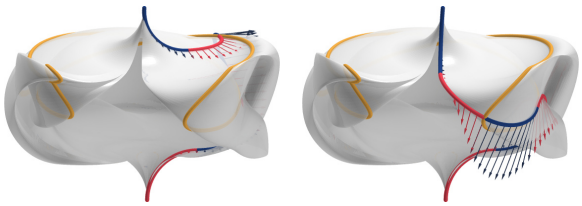

Figure 18: *Experimenting with 6-arc snakes. Here two time instances of the motion of an appropriately constrained 6-arc snake is shown, overlaid on the surface swept by the complete motion cycle. A path of a joint is shown in yellow.*

Directions of Future Research

An obvious direction of further research is to study the problem of automatic segmentation of a given surface into parts which can be nicely covered by an evolving curve of constant curvature (the analogous problem for geodesics has been solved by [PHD*10]). Another difficult geometric problem is an immmediate extension of our work: Finding out if a given surface can be generated by a single sweep of a curve, moving as a rigid body. Extensions of this question are the approximation with sweep surfaces, and the segmenation of surfaces into parts which can be well approximated in this way. Another direction of research is a more systematic approach to the design of snakes which have 1 d.o.f. and which exhibit interesting flexions. Here one can probably draw on the extensive literature on 6R linkages.

Conclusion

We have studied geometric and numerical properties of *arc snakes* which constitute an effective discretization of smooth curves if one is interested in evolution of curves such that the curvature is unchanged. We have shown applications in functional architectural design, e.g. for efficiently manufacturable underconstructions, and for surface generation. Finally we discussed the flexions of 6-arc snakes and their use for non-static architecture.

Acknowledgments

This research has in part been supported by the Austrian Science Fund (FWF, grant P23735). We also want to thank Florin Isvoranu for help with architectural realization.

References

[Bis75] BISHOP R. L.: There is more than one way to frame a curve. *Amer. Math. Monthly 82* (1975), 246–251. URL: http://www.jstor.org/stable/2319846.

[BPK*11] BO P., POTTMANN H., KILIAN M., WANG W., WALLNER J.: Circular arc structures. *ACM Trans. Graphics 30* (2011), #101,1–11. doi:10.1145/1964921.1964996.

[BR79] BOTTEMA O., ROTH B.: *Theoretical Kinematics*. North Holland, 1979. Reprinted by Dover Publications 1990.

[CDHR08] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S.: Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35*, 3, #22 (2008), 1–14. doi:10.1145/1391989.1391995.

[dC76] DO CARMO M.: *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[Hob] HOBERMAN: Transformable design. URL: http://www.hoberman.com.

[HPS07] HUSTY M., PFURNER M., SCHRÖCKER H.-P.: A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator. *Mech. Mach. Theory 42* (2007), 66–81. doi:10.1016/j.mechmachtheory.2006.02.001.

[Hun78] HUNT K. H.: *Kinematic Geometry of Mechanisms*. Clarendon Press, 1978.

[KWT87] KASS M., WITKIN A., TERZOPOULOS D.: Snakes: Active contour models. *Int. J. Comp. Vision 1* (1987), 321–331. doi:10.1007/BF00133570.

[Leo01] LEOPOLDSEDER S.: Algorithms on cone spline surfaces and spatial osculating arc splines. *Comput. Aided Geom. Design 18* (2001), 505–530. doi:10.1016/S0167-8396(01)00047-4.

[Miu85] MIURA K.: *Method of packaging and deployment of large membranes in space*. Tech. Rep. 618, Inst. of Space and Astronautical Science, 1985. 31st IAF congress, Tokyo, 1980. URL: http://airex.tksc.jaxa.jp.

[PAHK07] POTTMANN H., ASPERL A., HOFER M., KILIAN A.: *Architectural Geometry*. Bentley Institute Press, 2007.

[PHD*10] POTTMANN H., HUANG Q., DENG B., SCHIFTNER A., KILIAN M., GUIBAS L., WALLNER J.: Geodesic patterns. *ACM Trans. Graphics 29*, 4, #43 (2010), 1–10. doi:10.1145/1778765.1778780.

[PW01] POTTMANN H., WALLNER J.: *Computational Line Geometry*. Springer, 2001. 2nd printing 2010. doi:10.1007/978-3-642-04018-4.

[SACJ09] SONG X., AIGNER M., CHEN F., JÜTTLER B.: Circular spline fitting using an evolution process. *J. Comp. Appl. Math. 231* (2009), 423–433. doi:10.1016/j.cam.2009.03.002.

[Sau70] SAUER R.: *Differenzengeometrie*. Springer, 1970.

[Tac10] TACHI T.: Freeform rigid-foldable structure using bidirectionally flat-foldable planar quadrilateral mesh. In *Advances in Architectural Geometry 2010*. Springer, 2010, pp. 87–102. doi:10.1007/978-3-7091-0309-8_6.

[WM95] WALTON D. J., MEEK D. S.: Approximating smooth planar curves by arc splines. *J. Comp. Appl. Math 59* (1995), 221–231. doi:10.1016/0377-0427(94)00029-Z.