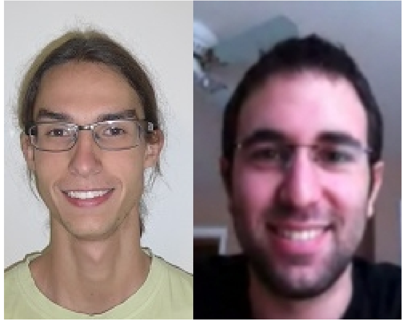


Curve Average

Project 3 for CS 6491: Computer Graphics

Sebastian Weiss, Can Erdogan
November 2, 2015

I. Objective

Formally, for two curves A and B defined by the control points $\{A_0...A_{n-1}\}$ and $\{B_0...B_{n-1}\}$, all points are in $\mathbb{R}^3$, and an interpolation scheme, the goal is to find a curve C such that for each point $c \in C$, and its closest projections

$$a^c = \operatorname*{argmin}_{a \in A} ||a - c||, \text{ and} \tag{1}$$

$$b^c = \operatorname*{argmin}_{b \in B} ||b - c|| \tag{2}$$

the following holds true:

$$c = \operatorname*{argmin}_{x \in L(a^c, b^c)} ||a^c - x|| \tag{3}$$

where for each point r on line $L(p,q)$, $||r-p|| = ||r-q||$. We call the curves A and B the control curves and C the average curve. Moreover, for any three consecutive sampled points c_i, c_j and c_k on curve C, the arc lengths of the curves between their closest projections are the same:

$$D^A(a^{c_i}, a^{c_j}) + D^B(b^{c_i}, b^{c_j}) = D^A(a^{c_j}, a^{c_k}) + D^B(b^{c_j}, b^{c_k}) \tag{4}$$

where $D^Z(x,y)$ is the distance along the curve Z between points $x, y \in Z$.

Semantically, we want to find the curve that is composed of the loci of the smallest spheres that touch the two inputs curves A and B, and sample it such that for each sample on the curve, the distances traveled by the matching samples along their curves is a constant.

It can also be seen as a generalization of the medial axis transformation from the 2D case to the 3D case: In every projection of the $\mathbb{R}^3$ in the $\mathbb{R}^2$, the average curve C is the medial axis of A and B.

II. Input

For simplicity, we only focus on a special case that each control curve is defined by six control points A_0 to A_5 and B_0 to B_5, such that the first and the last control points are the same.

III. Overview

The project is composed of the following three main parts: (1) the representation of the curves (see IV), (2) the computation of the average curve (see V), and (3) the visualization of the different curve properties (see VI). The challenge in the representation is that we want the spline curves to first meet at two end points and then, to have C^1 continuity at the intersection of the splines. For the curve average, we had to ensure the points satisfied the constraints in Equations 1-3. Lastly, the visualization contained multiple challenges including parallel transport, circular arc computation and etc.

IV. Curve Representation

We compose each curve out of piecewise quadratic Hermite and cubic Hermite splines. Each spline is connected at a control point by the same local velocity. This leads to a $\mathcal{C}^1$-smooth curve. We define the tangent/velocity at control point A_i as $T_i = c(A_{i+1} - A_{i-1})$. The parameter c defines the curvature or the influence of that velocity. In our experiments, we set c to 0.5.

A cubic Hermite spline requires velocities at both end points, a quadratic Hermite spline only at one end point. Therefore, we use the quadratic form for the first and the last part of the curve and the cubic form for all middle parts.

A. *Quadratic Hermite Spline*

Given the points X_0 and X_1 and the tangent T_0, find a quadratic spline $P(t)$ so that the following holds:

$$P(0) = X_0, P(1) = X_1, P'(0) = T_0 \tag{5}$$

Solving this system of equation leads to the following formula:

$$P(t) = (t^2 - 2t + 1)X_0 + (-t^2 + 2t)X_1 + (t^2 - t)T_0 \tag{6}$$

B. *Cubic Hermite Spline*

Given the points X_0 and X_1 and the tangents T_0 and T_1, find a cubic spline $P(t)$ so that the following holds:

$$P(0) = X_0, P(1) = X_1, P'(0) = T_0, P'(1) = T_0 \tag{7}$$

This equation is solved similar to the quadratic case, leading to:

$$\begin{aligned} P(t) &= (2t^3 - 3t^2 + 1)X_0 + (t^3 - 2t^2 + t)T_0 \\ &+ (-2t^3 + 3t^2)X_1 + (t^3 - t^2)T_1 \end{aligned} \tag{8}$$

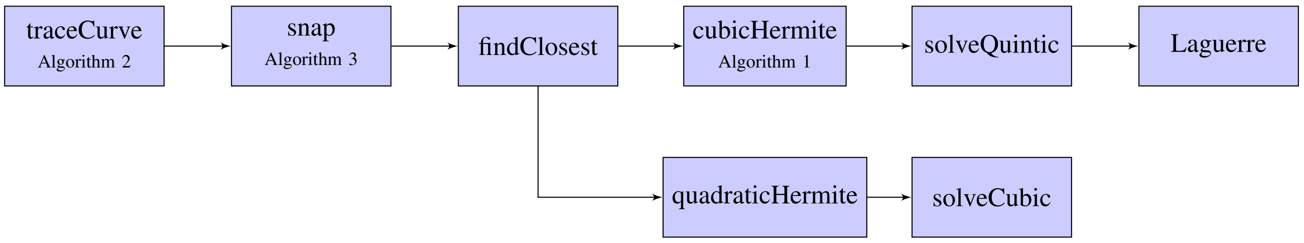


Figure 1. Block diagram for the major functions involved in tracing the average curve

V. Average Curve Computation

A. Distance functions

To satisfy the distance constraints (1) and (2) in the objective function, we need to first define the distance functions for a point and a curve. Note that since our curve representation is composed of multiple splines, we are actually interested in distance between a point and a hermite (quadratic or cubic). If $P(t)$ represents a point on the spline, parameterized with time t, we are interested in finding t such that the tangent at that point $P'(t)$ and the vector from that point to the input point Q are perpendicular:

$$P'(t) \cdot (Q - P(t)) = 0 \tag{9}$$

If $P(t)$ is a quadratic polynomial in t, then the perpendicularity constraint is a cubic in t and the points where the constraint are satisfied can be solved in closed form. Otherwise, if $P(t)$ is a cubic polynomial in t, then the perpendicularity constraint is a quintic in t and its roots need to determined with a local optimizer.

Once all the roots of the perpendicularity constraint is found, the ones with imaginary coefficients and the ones with real values that are out of the [0,1] bound need to be eliminated. Moreover, to choose the closest perpendicular point, an iterative comparison has to be executed as shown in Algorithm 1 for a quintic function. Note that the coefficients for the polynomial are obtained through Matlab's symbolic toolbox.

Algorithm 1: closestPointOnCubicHermite()

```
Input: $\{P_0, T_0, P_1, T_1\}$: control points, $Q$: input point
Result: $t$: time in $[0,1]$ for the closest point
1  $[a,b,c,d,e,f]$ ←generateDistanceCoeffs($P_0, T_0, P_1, T_1, Q$);
2  $res$ ←solveQuintic($a,b,c,d,e,f$);
3  $\{minDist, t^*\} \leftarrow \{\infty, -1\}$;
4  foreach $r \in res$ do
5      if $|r.im| > 0$ then continue;
6      if $r.re < 0$ or $r.re > 1$ then continue;
7      $P_t \leftarrow \tau(P_0, T_0, P_1, T_1, r.re)$;
8      if $|P_t - Q| < minDist$ then
9          $\{minDist, t^*\} \leftarrow \{|P_t - Q|, r.re\}$;
10 return $t^*$;
```

B. Optimization methods

Here, we would like to make a brief mention of the Java Commons Math library which provides a plethora of optimizer and share our experience for future users.

1) Newton-Raphson Method: The Newton-Raphson method is one of the few methods that does not need an initial bracketing (limiting the solution space) and is also conceptually simple: just follow the derivative at the initial sample until it hits a zero-crossing and repeat until convergence. However, unfortunately, the library implementation does not always return the closest solution to the sample and thus, hard to use to determine all the roots of the quintic function.

2) Laguerre's Method: The Laguerre's method is dependent on two initial bounds which have different signs when the quintic is evaluated. The positive aspect of the implementation of the method is that it is guaranteed to find a solution if one exists within the bounds and if not, throw an exception. Thus, we discretize the time domain and search for the roots systematically.

C. Curve tracing

The block diagram that outlines the main functions in our curve tracing algorithm is presented in Figure 1. The main idea is to start with the intersection of the two curves as the first sample and continuously project out in the "medial-line" of the projected points' tangents iteratively. The traceCurve function in Algorithm 2 demonstrates the steps involved.

Algorithm 2: traceCurve()

```
Input: $\{\mathbf{A}, \mathbf{B}\}$: curves
Result: $\mathbf{C}$: average curve
1  $\mathbf{C}$.add($\mathbf{A}[0], 0$);
2  $\{current, t_a, t_b, t'_a, t'_b\} \leftarrow \{\mathbf{C}[0], 0, 0, 0, 0\}$;
3  while true do
4      $\vec{T_A} \leftarrow \vec{n}(\mathbf{A}.\tau(min(1, t_a + \epsilon))$ - $\mathbf{A}.\tau(min(1, t_a)))$;
5      $\vec{T_B} \leftarrow \vec{n}(\mathbf{B}.\tau(min(1, t_b + \epsilon))$ - $\mathbf{B}.\tau(min(1, t_b)))$;
6      $current \overset{+}{\leftarrow}$ STEP_SIZE $\vec{n}(\vec{T_A} + \vec{T_B})$;
7      $\{t'_a, t'_b, r, P_c\} \leftarrow$ snap($\mathbf{A}, \mathbf{B}, current$);
8      if $max(t_a, t_b) < 1$ or $t'_a < t_a$ or $t'_b < t_b$ then break;
9      $\{t_a, t_b\} \leftarrow \{t'_a, t'_b\}$;
10     $\mathbf{C}$.add($P_c$);
11 return $\mathbf{C}$;
```

For a sample *current*, the algorithm first finds the tangents at the projected point $\vec{T}_A$ and $\vec{T}_B$, and *projects out* the current sample by their median. However, since this point is not guaranteed to represent the distance constraints (1) and (2), nor since that sphere would be minimal, we need to incrementally move it with the *snap* function.

The pseudo-code for the *snap* function is presented in Algorithm 3. At each iteration, two main changes are made on the initial sample. First, the point is moved towards its closest projections as shown in the lines 3-7. Then, the point is moved towards the planes defined by the projections and the sphere center to minimize the radius of the sphere (in lines 8-10).

Algorithm 3: snap()

```
Input: $\{\mathbf{A}, \mathbf{B}\}$: curves, $Q$ : input point
Result: $\{t_a, t_b\}$: projections, $\{P_c, r\}$: sphere center/radius
1  $P_c \leftarrow Q$;
2  while $i$++ < MAX_ITERS do
3      $\{P_A, \vec{T}_A, r_A, t_A\}$ ←closestPoint($\mathbf{A}, P_c$);
4      $\{P_B, \vec{T}_B, r_B, t_B\}$ ←closestPoint($\mathbf{B}, P_c$);
5      $error \leftarrow |P_A - P_c| - |P_B - P_c|$;
6      if $error < \epsilon$ then break;
7      $P_c \leftarrow P_c - error\ \vec{n}(\vec{T}_A - \vec{T}_B)$;
8      $\{d_A, \vec{V}_A\}$ ←planeDist($P_c, P_A, P_B, \mathbf{A}.\tau(t_A + \epsilon)$);
9      $\{d_B, \vec{V}_B\}$ ←planeDist($P_c, P_B, P_A, \mathbf{B}.\tau(t_B + \epsilon)$);
10     $P_c \leftarrow P_c -$ PLANE_PROJ_COEFF $(d_A\vec{V}_A + d_B\vec{V}_B)$;
11 return $\{c_A.t, c_B.t, P_c, c_A.r\}$;
```

D. Average Distance (geodesic) Sampling

One challenge with the output of the naive tracing algorithm is that the samples on the curve C can have arbitrary relative distances. This can be an issue when additional visualization tools are built upon this curve such as tubes and boardwalk patterns. One approach is to predefine the number of points desired and sample the curve C based on the total arc length and the chosen resolution.

Algorithm 4 outlines this approach where the relevant C samples for the new sample are determined with a preliminary search (lines 6-12) and then, an intermediary point is computed with linear interpolation. Next, the interpolated point is pushed back on to the curve C to satisfy the object constraints 1-3 using the snap function in lines 14-15.

The effect of the average distance sampling in comparison to the original average curve C can be observed in Figure 3 between items (a) and (b). As expected, the samples along the geodesic curve is much more refined and can sample more thoroughly the "incompatible" regions.

Algorithm 4: geodesicTrace()

```
Input: $\{\mathbf{A}, \mathbf{B}\}$: curves, $N$ : number of points
Result: $\mathbf{G}$: geodesic average curve
1  $\mathbf{C} \leftarrow$ trace($\mathbf{A}, \mathbf{B}$);
2  $length \leftarrow \mathbf{A}$.arcLength()+$\mathbf{B}$.arcLength();
3  $\{\delta l, i, \mathbf{G}\} \leftarrow \{length/N, 0, \{\mathbf{C}[0]\}\}$;
4  while true do
5      $\{g, \underline{i}, \underline{d}, \bar{i}, \bar{d}\} \leftarrow \{\mathbf{G}.\text{last}, i, 0, -1, 0\}$;
6      while $i < \mathbf{C}$.length do
7          $d_A \leftarrow \mathbf{A}$.arcLength($proj(g, \mathbf{A})$, $proj(C[i], \mathbf{A})$);
8          $d_B \leftarrow \mathbf{B}$.arcLength($proj(g, \mathbf{B})$, $proj(C[i], \mathbf{B})$);
9          if $d_A + d_B < \delta l$ then $\{\underline{i}, \underline{d}\} \leftarrow \{i, d_A + d_B\}$;
10         else
11             $\{\underline{i}, \underline{d}\} \leftarrow \{i, d_A + d_B\}$;
12             break;
13     if $i \geq \mathbf{C}$.length then break;
14     $P_c$ ←LERP($\mathbf{C}[\underline{i}], \mathbf{C}[\bar{i}], (\delta l - \underline{d})/(\bar{d} - \underline{d})$);
15     $\mathbf{G}$.add(snap($\mathbf{A}, \mathbf{B}, P_c$));
16 return $\mathbf{G}$;
```

E. Detect incompatible control curves

An important aspect of the average curve formulations provided in the first section is that the there may not always be a solution to the three fundamental constraints of the average curve. In such cases, the average curve may not (1) exist, (2) be smooth, or (3) be close to the medial axis, and thus, it is important to inform the user when such cases may arrive as they manipulate the control points.

One method of doing so is to monitor the arclength between the consecutive points in the average curve (geodesic) trace. If the arclength is significantly larger than the expected step size (1.5 times more in our implementation), then a warning can be given by tracing the corresponding curve sections in red. Figure 2 below demonstrates examples of such extreme cases and the incompatible sections of the input control curves that would lead to a branching in the medial axis.

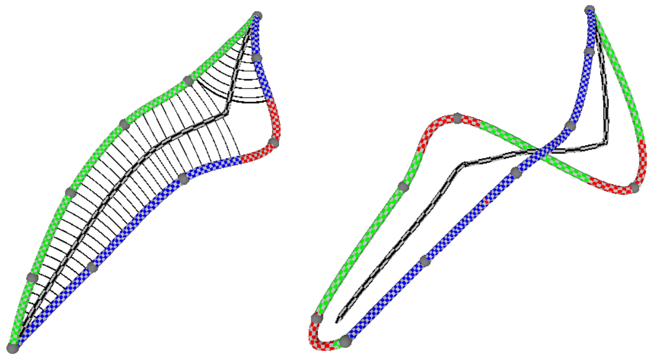


Figure 2. Extreme cases and non-compatibility detection

VI. Visualization and Editing

We provide a framework for editing the control curve and visualizing the curve average. More specific, we support the following features:

- Moving the control points of the two input curve
- Displaying the curve average with or without geodesic sampling
- Displaying the closest projections
- Displaying the circular arc and animating the curve average along it
- Showing the inflation tube, the envelope of the smallest spheres

We now describe the single features in more detail.

A. Editing and displaying the control curve

We display the control points as gray spheres and the two control curves in green and blue. The user can click the control points and change their positions by dragging them over the screen.

B. Displaying the average curve

The curve average is displayed by connecting the sample points using piecewise straight tubes. The faces of the tubes are rendered in alternating black and gray so you can see where the trace points are. The user has now the option to toggle geodesic sampling on or off. The effects are shown in Fig.3ab. As you can see, without geodesic sampling, the distances between the samples of the curve average are almost equispaced, but the closest projections on the control curve, indicated by the circular arcs, vary greatly in the distances. With geodesic sampling turned on, the curve average is sampled in that way that the sum of the distances between two consecutive closest projects on the control curves is constant. However, this leads to large variation of the distances of the curve average samples.

C. Closest projections

To visualize the closest projections, i.e. the points on the control curve that are closest to the average curve at this point, we draw straight lines between the sample of the curve average and its closest projections. These can be seen in Fig.3c. Note that the lines to both control curves are of the same length and that they stand orthogonal to the control curve at the intersections. These are the properties that define the closest projections and the curve average.

D. Circular arc

Instead of displaying the closest projections as straight lines, we could also draw them using circular arcs. In Fig.4, you can see parts of the two input curves in green and blue and the curve average in black in the middle. The current trace point is P and the closest projections of P on the control points are A and B.
We can now construct a unique circle in $\mathbf{R}^3$ with center C and radius r that goes through A and B, the tangents at these points are AP and BP and lies in the convex hull of A,B and P.

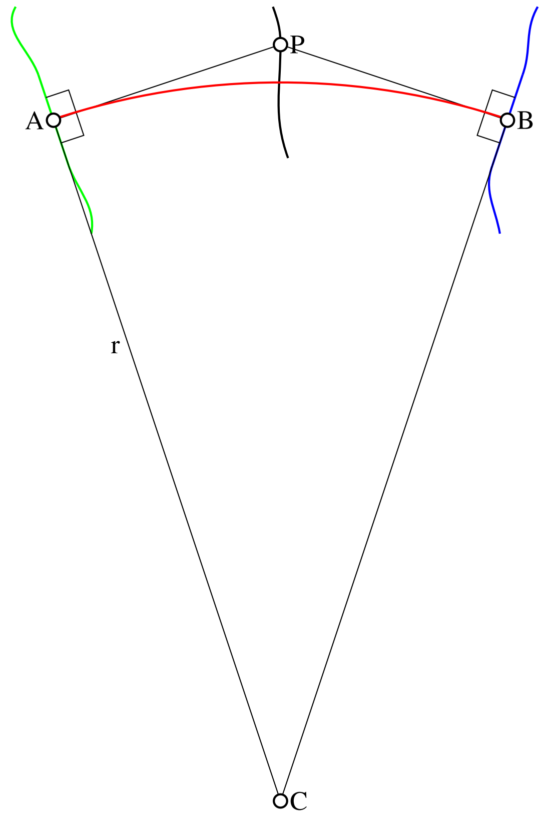


Figure 4. circular arc

First, assume A,B,P are not collinear. Then ABP defines a plane with the normal vector $N = \underline{PA} \times \underline{PB}$. The circular arc lies in that plane. The vectors $U = N \times \underline{PA}$ and $V = N \times \underline{PB}$ are normals of the circle at these points. Therefore, the rays $A + \alpha U$ and $B + \beta V$ intersect exactly at the circle center C. This system is overdetermined but we can solve for α in the following way:

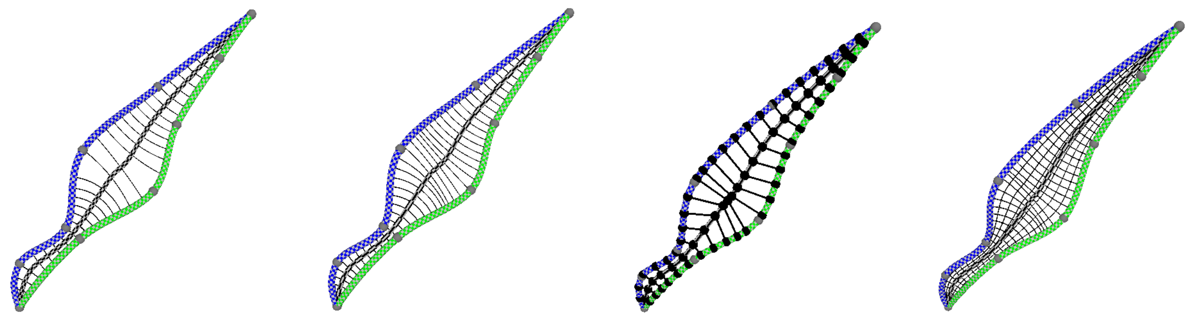
$$\begin{aligned} A + \alpha U &= B + \beta V \\ \alpha U &= B - A + \beta V \\ (\alpha U) \times V &= (AB + \beta V) \times V \\ \alpha (U \times V) &= (AB) \times V \end{aligned} \tag{10}$$

And now we have $\alpha = \frac{((AB) \times V).x}{(U \times V).x}$. We can pick any coordinate, all have to be the same.

This then leads to $C = A + \alpha U$, $r = |AC|$ and N as described above.

If the points A,B,P are collinear, indicated by $N = 0$ (plus epsilon), then the circular arc degenerates to a linear interpolation (lerp) from A to B through P.

We draw the circular arc by approximating it using piecewise straight tubes. The samples are obtained by rotating the



a) no geodesic sampling b) with geodesic sampling c) the closest projections d) net of circular arcs

Figure 3. Geodesic sampling, closest projections and circular arcs

vector CA around N until it becomes CB. This is done the same way as in the Swirl-project:

$$\begin{aligned} CA^{\circ}(\theta;N) &:= C + U^{\circ}(\theta;N) \\ &= C + \cos\theta U - \sin\theta(N \times U) \\ \text{with } V &:= CA\angle\underline{N} = (W \bullet \underline{N})\underline{N} \\ \text{and } U &= CA - V \\ \theta \text{ ranges from } 0 &\text{ to } CA^{\wedge}CB = \cos^{-1}(\underline{CA} \bullet \underline{CB}) \end{aligned} \tag{11}$$

Furthermore, if we sample the circular arcs at some angles and combine the samples with the same angle, we obtain a net, as shown in Fig.3d. If both control curves lie in a plane, this net also lies in the same plane.

We further provide the ability to animate the average curve between the two control curves. In that animation, each sample of the average curve follows the circular arc between the closest projections.

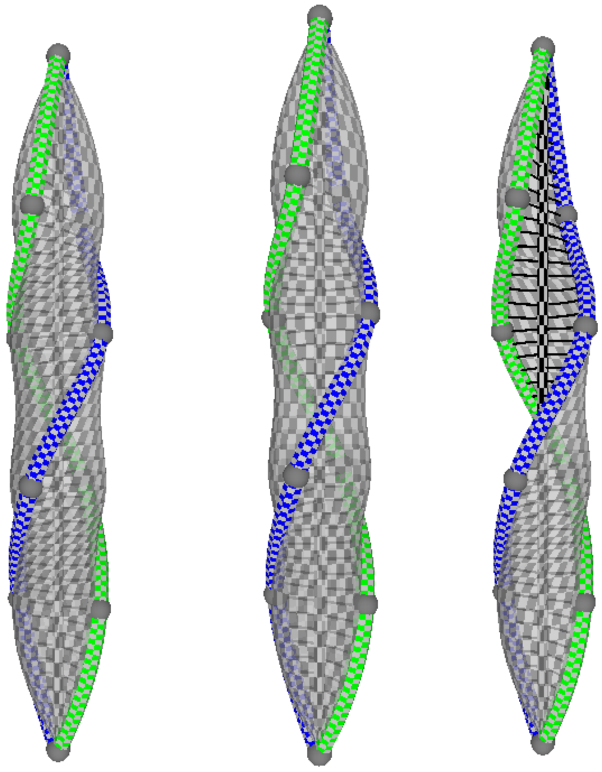
E. *Inflation tube*

As a last graphic feature, we draw the inflation tube. The inflation tube is the envelope of the minimal balls along the curve average. It can be seen as the union of all balls with the center on the curve average that exactly touches the control curve and contains no other point in the interior.

1) Drawing the inflation tube: To draw the inflation tube, we get the following input from the tracing algorithm: the samples of the average curve P_i, $i \in \{0, ...n-1\}$, the closest projections on the two control curves A_i and B_i and the radii or distances to the closest projections r_i. We further define the tangent T_i at the average curve sample i as $T_i = \frac{1}{2}(P_{i+1} - P_{i-1})$.

The first way one might think of drawing it, is to circles around the average curve samples P_i in the plane orthogonal to the tangents T_i with radii r_i and connect them to tubes. However, this approach will lead to wrong results since the closest projections to the control curves are not orthogonal to the average curve tangent. Therefore, the distance from the closest projections to the average curve is not equal to the radius of the circle when it should touch the control curves.

Create a new section? We present a different approach now. Instead of drawing circles orthogonal to the tangents, we directly take the vectors P_iA_i and rotate them around T_i. This technique is visualized in Fig.6. We rotate P_iA_i as described in



a) no parallel transport b) with parallel transport c) half tube

Figure 5. the inflation tube

Eq.11 the whole 360°. At 180°, the vector becomes P_iB_i. We then sample the circles at constant angles and connect samples with the same angle, leading to the tube as shown in Fig.5a.

2) Parallel transport: When the control curves rotate around the average curve, the inflation tube follows this rotation, as seen in Fig.5a. However, often Fig.5b is required. Here, the rotation of the inflation tube stays constant although the control curves rotate around it.

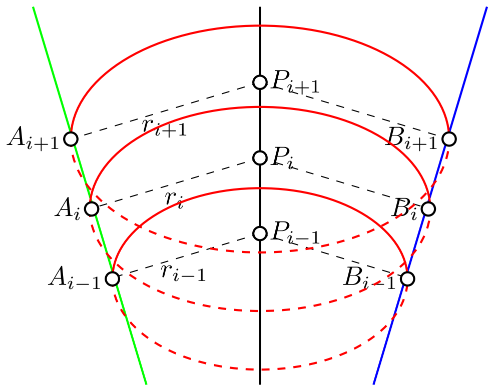


Figure 6. inflation tube

The technique to achieve this result is called Parallel Transport. The only thing we have to do is to modify the start angle α_i. Instead of sampling each circle (see previous section) starting with $0°$, we start it at an angle that minimizes the rotation to the previous circle.

The task is now the following: given the previous tangent $\underline{T_{i-1}}$, rotated vector $P_{i-1}A_{i-1}$ and start angle α_{i-1}, and the current tangent $\underline{T_i}$ and rotated vector P_iA_i, find the current start angle α_i so that the twist is minimized. In other words, we have to find α_i so that $P_iA_i^\circ(\alpha_i;T_i)$ lies in the plane spanned by T_{i-1} and $X := P_{i-1}A_{i-1}^\circ(\alpha_{i-1};T_{i-1})$:

$$\begin{array}{rcl}
\underline{X} & := & \underline{P_{i-1}A_{i-1}^\circ(\alpha_{i-1};T_{i-1})} \\
\underline{N'} & := & \underline{T_{i-1}} \times \underline{X} \\
\underline{PA'} & := & \underline{P_iA_i} \\
x & := & \underline{PA'} \bullet \underline{N'} \\
y & := & \underline{PA'} \bullet \underline{X} \\
PA'' & := & xN' + yX \\
\alpha_i & := & ((x<0)?(-1):1) \cdot \cos^{-1}(\underline{X} \bullet \underline{PA''})
\end{array} \qquad (12)$$

$\underline{N'}$ is the normal of the reference plane, then we project P_iA_i in that plane and obtain PA''. The final angle α_i is then the angle between the reference vector X and the projected vector PA''. Since this angle does not include the direction of the twist, we have to invert the angle if $x < 0$.

3) half tube: Instead of drawing the whole inflation tube without parallel transport, we could also draw only half of it. This leads to the interesting effect of a waterslide as seen in Fig.5c.

VII. Results

In conclusion, we are able to trace the average curve of two piecewise hermite interpolated curves in real-time, whereat the user defines the control points by dragging them around. We presented ideas on how to detect if the two control curves are compatible or not. Furthermore, we showed different ways on how to display the average curve, using line segments to the closest projections, circular arcs or the inflation tube.

VIII. Future work

This work can easily be extended to any kind of input curve and any number of control points. Also more freedom for the user to specify the input curve, like editing the velocities, might be helpful for specific applications.

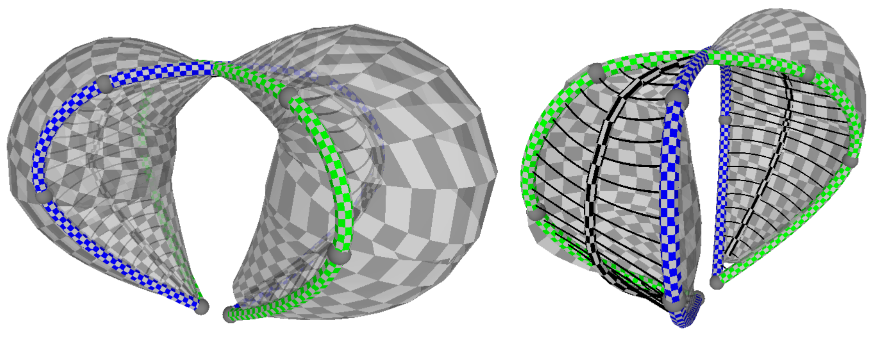


Figure 7. Example 1: a bow

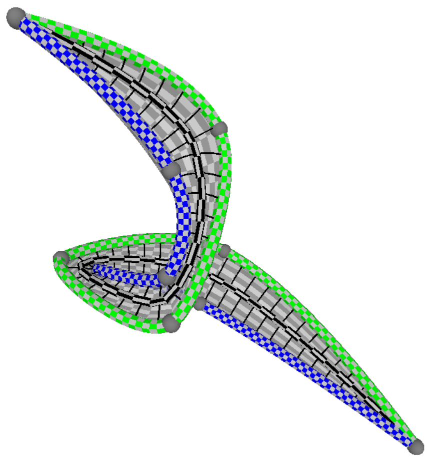


Figure 8. Example 2: a waterslide