Fair, G^2- and C^2-Continuous Circle Splines for the Interpolation of Sparse Data Points

Carlo H. Séquin, Kiha Lee, Jane Yen

ABSTRACT

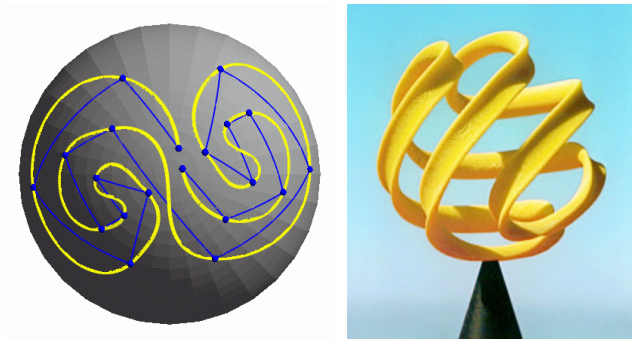
This paper presents a carefully chosen curve blending scheme between circles, which is based on angles, rather than point positions. The result is a class of circle-splines that robustly produce fair-looking G^2-continuous curves without any cusps or kinks, even through rather challenging, sparse sets of interpolation points. With a simple reparameterization the curves can also be made C^2-continuous. The same method is usable in the plane, on the sphere, and in 3D space.

Keywords

CAD, circle-splines, curves & surfaces, geometric modeling, sparse data interpolation.

1. INTRODUCTION, MOTIVATION

Fair, interpolatory spline curves are useful constructs for many application domains and design environments, ranging from the construction of ship hulls and aerodynamic profiles, through key-frame animation, to smooth camera motions for flying around objects of interests. For certain applications, such as aesthetic designs or camera paths, smooth, nicely rounded paths – free of cusps and abrupt hairpin turns – are more important than the property of affine invariance. In these situations, schemes based on circles can more easily satisfy such demands than polynomial splines. A few blending schemes have been developed that aim to accommodate circular arcs whenever possible. Biarcs generate segments of the overall curve with pairs of circular arcs connected with tangent continuity [2][15][9][16][8]. Other schemes use a gradual blending between corresponding points on two circular arcs; they can achieve smooth looking paths with G^1- or G^2-continuity, depending on the exact blending procedure used [18][6][14]. The most ambitious approaches involve global curve optimization, such as the Minimum Energy Curve (MEC) [4], or the Minimum Variation Curve (MVC) [7] [10], in which circles result as the zero-cost solution whenever permitted by the constraints.



(a) **(b)**

Figure 1: (a) A circle spline and its control polygon on a sphere;
(b) a sculpture model derived from a sweep along such circle spline.

The original motivation for the development of circle splines [11] was to provide smooth, pleasing curves embedded in the surface of a sphere, either for a special class of geometrical sculpture (Fig.1), or for the definition of a camera path that flies around a stationary object approximating a Grand Tour [1], looking inward to inspect that object from "all sides." However, the new robust solution presented in this paper gives equally good results in the plane and for space curves in three- or higher-dimensional Euclidean space. The primary goal still is to define aesthetically pleasing curves with just a few "fix-points." Artists often like to construct a well-rounded fair curve, with a "natural" look such as the shape of a stiff steel wire, confined in space at only a few points, but with the additional capability to adjust its length locally so as to give each loop an optimal bulge that leads to the smoothest possible transitions in curvature. Currently such shapes seem to be realizable only in the virtual world of a good CAD environment. Our goal for these curves was to achieve as much of the behavior of a MVC as possible, but with a strictly local support domain. The result is a new class of interpolating circle splines that not only achieve the desired goal on the sphere, but also improve the properties of the circle-blending schemes previously described in 2- or 3-dimensional Euclidean space.

2. BACKGROUND, PREVIOUS WORK

The simplest circle spline schemes look at four consecutive points P_{i-1}, P_i, P_{i+1}, P_{i+2} to calculate the curve segment between points P_i and P_{i+1}. These points alone determine the shape of that segment. Thus, these curves have tightly limited local support. For the special case where all but one interpolation point, P_i, lie on a straight line (Fig.2), only four curve segments will deviate from a perfect straight line. In the special case depicted in Figure 3, where all the interpolation points lie on two circular arcs, only the single transition segment will deviate from a perfect circular arc.

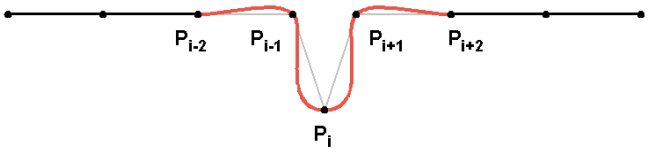


Figure 2: Influence zone of point P_i.

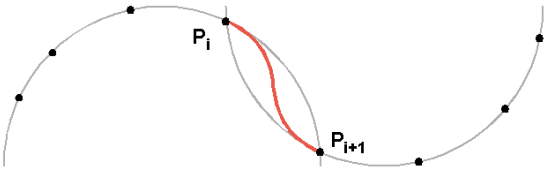


Figure 3: Zones of perfect circles.

In general, the curve segment between points P_i and P_{i+1} is formed by first fitting a circular arc, arc_i, through points P_{i-1}, P_i, P_{i+1}, in sequence, and a second arc, arc_{i+1}, through points P_i, P_{i+1}, and P_{i+2} (Fig.4). Then, in the region between P_i and P_{i+1} the two circular arcs are blended together by gradually shifting the weight from arc_i to arc_{i+1}, as the parametric curve point moves from P_i to P_{i+1}. The crucial question is, how exactly should one perform this blending operation?

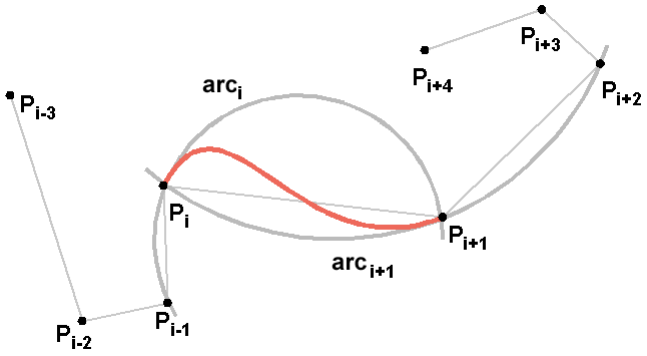


Figure 4: Generic blending approach using four consecutive data points.

Wenz [18] achieves this blending operation by performing a linear positional interpolation between corresponding points on the two base arcs that have the same parameter values. The two arcs, $arc_i(u)$ and $arc_{i+1}(u)$, are parameterized so that they are traced from P_i to P_{i+1} as the parameter u goes from 0 to 1. Using a simple linear blending scheme (Fig.5a), the new parametric curve point is then found as:

$$P(u) = (1-u) * arc_i(u) + u * arc_{i+1}(u). \quad (1)$$

Szilvasi-Nagi and Vendel [14] improved on that scheme, by replacing the linear interpolation function with a trigonometrically weighted blending function (Fig.5b):

$$P(u) = \cos^2(u\,\pi/2) * arc_i(u) + \sin^2(u\,\pi/2) * arc_{i+1}(u). \quad (2)$$

This has the effect that the blend curve clings more strongly to the base arcs near the end points P_i and P_{i+1}, and thereby picks up the curvature of the base arcs in addition to their tangents at P_i and P_{i+1}, respectively. This guarantees that the individually generated blend segments will join with G^2-continuity across all the interpolation points.

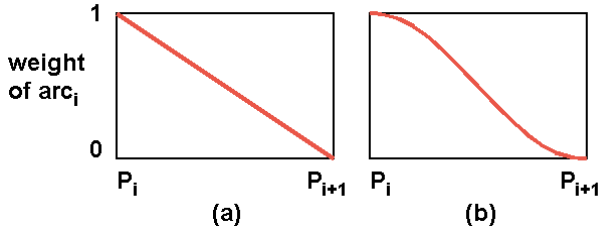


Figure 5: Blending functions: (a) linear and (b) trigonometric.

However, both these schemes can produce undesirable loops or even cusps, when the control polygon through the sequence of given data points shows large turning angles at some point (Fig.6). The main problem is that the straight parameter lines that connect equivalent points on the two base arcs, along which the positional interpolation is performed, may intersect one another, which in turn can lead to self-intersections of the blending segment itself.

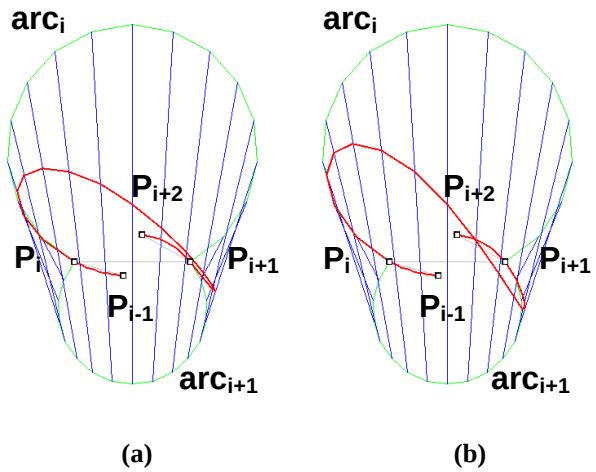


Figure 6: Interpolation of point positions on two circles, using (a) linear [18] and (b) trigonometric [14] blending.

In a 2001 SIGGRAPH sketch [11], we proposed to remedy that situation by using an angle-based interpolation scheme that could prevent excessive looping and cusps. This method was based on a subdivision approach. In each step it only constructed a new segment midpoint S_A between the interpolation points, P_i and P_{i+1}. The new segment midpoint S_A was found by constructing the halfway point on an intermediary circular arc through P_i and P_{i+1} that averaged the turning angles, angle($P_iS_iP_{i+1}$) and angle($P_iS_{i+1}P_{i+1}$), found at the midpoints S_i and S_{i+1} of the two base arcs (Fig.7). As Figure 7 shows, this scheme produced a new subdivision data point S_A that lies much closer to points P_i and P_{i+1} than the point S_P produced by positional interpolation between the two arc midpoints S_i and S_{i+1}. This makes it easier for the solid blend curve through S_A than for the dashed curve through S_P to reach the data point P_{i+1} with the proper tangent direction, without producing extraneous loops or cusps. However, with this subdivision scheme we were unable to make any formal claims about the degree of continuity of the curves it produced – even though visually the curves looked very pleasing. Indeed, it appears, that the achievable continuity might be G^1 (tangent continuity) at best. Another shortcoming of this subdivision approach arose from potential numerical instabilities when the scheme was extended to general 3D space curves, since the approach described required the explicit calculation of the sphere that could be fitted through the four control points P_{i-1}, P_i, P_{i+1}, P_{i+2}. As these points approached a nearly coplanar configuration, and the radius of the sphere thus became extremely large, the accuracy of the evaluation became questionable.

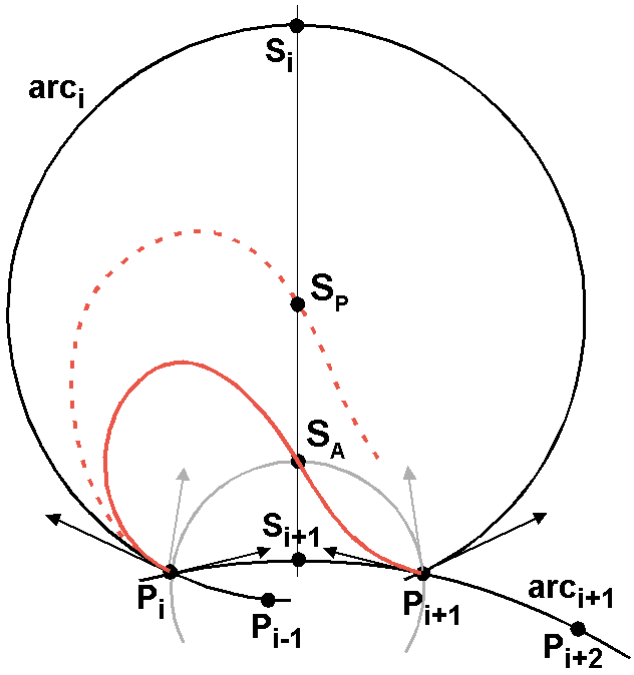


Figure 7: Comparison of positional (S_P), and angle-based (S_A) blending.

With this paper we present an approach that overcomes all these deficiencies. The key ideas of circle splines are retained: for each segment of the control polygon, a blend between two conceptual circular arcs is constructed. The main contribution of this paper is to show a simple and robust scheme that works in almost exactly the same way in 2D, in 3D, or on a sphere, so that the same algorithm can be used in all three cases. The scheme does not calculate the radius or the center of these arcs or spheres explicitly and thereby offers a smooth transition to linear segments and to planar blend curves. A second key point is to demonstrate that the angle-averaging technique is indeed crucial to the good behavior of these circle splines. Finally, the new approach allows proper analysis of the continuity properties of the new scheme. It is shown that strict G^2-continuity can be achieved for all points of the curve and that C^2-continuous curves can be obtained with a simple reparameterization.

3. CURVE CONSTRUCTION

We assume that the curve is defined by a sequence of interpolatory constraint points P_0, P_1,…P_i,…P_n (the "control polygon"). To form a circle spline, we form a blend between two circular arcs for every segment (P_i, P_{i+1}). Arc_i is defined to go through points P_{i-1}, P_i, P_{i+1} in sequence, and arc_{i+1} through points P_i, P_{i+1}, P_{i+2}. These two *base arcs* define the tangent vectors t_i and t_{i+1} and the curvatures of the composite curve at points P_i and P_{i+1}, respectively. Our approach guarantees that the blend curve picks up these end conditions at points P_i, P_{i+1}, that it is well-behaved in between (i.e., has no cusps and no self-intersections), and that it has finite curvature, as long as the control polygon does not have a joint with a turning angle of 180^o. If one of the two base arcs is undefined, because we are dealing with an end-segment of the control polygon where either P_{i-1} or P_{i+2} are missing, we use the remaining base arc directly without any blending.

Figure 8 shows the construction in the plane. The blend between arc_i (top) and arc_{i+1} (bottom) does not occur by simply interpolating circle point positions, as was the case in Figure 6. Instead, as the point $P(u)$ travels across an arc, $arc(u)$ from P_i to P_{i+1}, the arc morphs from arc_i (u =0) to arc_{i+1} (u =1). Any intermediate $arc(u)$ is defined by the two points P_i and P_{i+1} and

by its tangent $t(u)$ at P_i. The direction angle $\tau(u)$ of this tangent is used to parameterize the morphing process of these arcs. If we use a linear blend (Fig.8a), we will obtain G^1-continuity at the junctions between subsequent segments. To obtain G^2-continuity (Fig.8b), we use a trigonometric blend between the two extreme direction angles τ_i and τ_{i+1} given by the tangent vectors t_i and t_{i+1} of the base arcs, arc_i and arc_{i+1}:

$$\tau(u) = \tau_i \cos^2(u\,\pi/2) + \tau_{i+1} \sin^2(u\,\pi/2). \qquad (3)$$

The advantage of this angle-based morphing from one base arc to the other is that the parameter lines for constant u do not intersect; this is a key reason why cusps or self-intersections do not occur in these curve segments. Note that in the case of trigonometric blending (Fig.8b), the blend curve "hugs" the base arcs much longer, thus producing junctions with G^2-continuity.

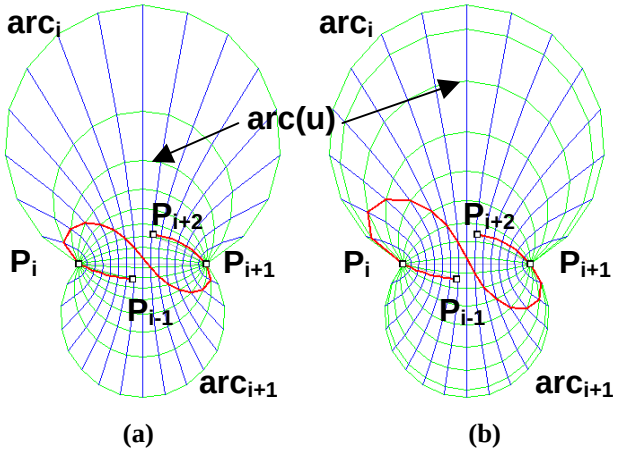


Figure 8: Family of arcs for forming the blend curves: (a) linearly, (b) trigonometrically interpolated circle splines.

3.1 Robust Calculations

To handle robustly the case of arcs of arbitrary large radii, including straight-line connections between P_i and P_{i+1}, we avoid calculations that involve the centers or radii of the circular arcs. When "fitting a circle through P_{i-1}, P_i, P_{i+1}" we only need to determine the tangent direction of arc_i at P_i and its relevant turning angle with respect to the control segment b between P_i and P_{i+1}. First we calculate several unit direction vectors for the control polygon:

$$\mathbf{a} = (\mathbf{P}_i - \mathbf{P}_{i-1}) / |\,\mathbf{P}_i - \mathbf{P}_{i-1}\,|\,; \quad \mathbf{b} = (\mathbf{P}_{i+1} - \mathbf{P}_i) / |\,\mathbf{P}_{i+1} - \mathbf{P}_i\,|\,; \quad \mathbf{c} = (\mathbf{P}_{i+1} - \mathbf{P}_{i-1}) / |\,\mathbf{P}_{i+1} - \mathbf{P}_{i-1}\,|\,;$$
$$\mathbf{d} = (\mathbf{P}_{i+2} - \mathbf{P}_{i+1}) / |\,\mathbf{P}_{i+2} - \mathbf{P}_{i+1}\,|\,; \quad \mathbf{e} = (\mathbf{P}_{i+2} - \mathbf{P}_i) / |\,\mathbf{P}_{i+2} - \mathbf{P}_i\,|. \qquad (4)$$

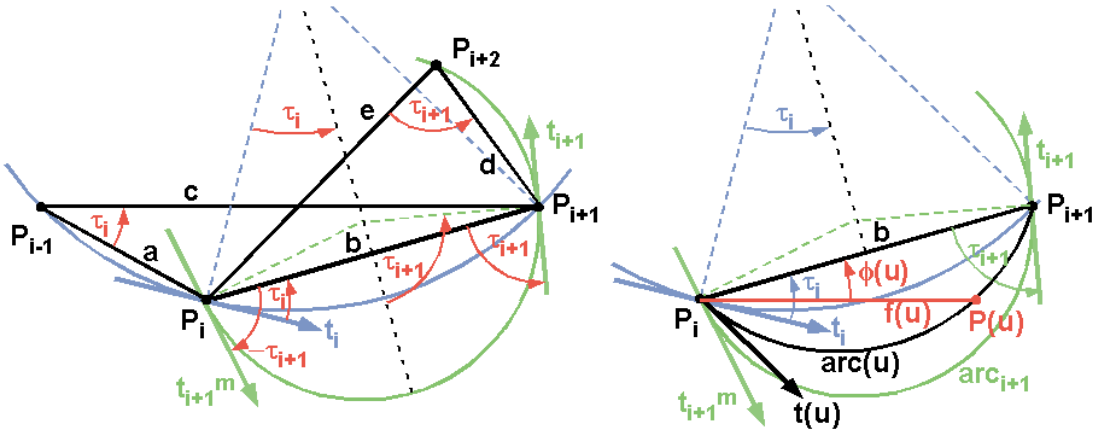
Then we calculate the relevant tangent angles at P_i and at P_{i+1}:

$$\tau_i = \text{angle}(P_i, P_{i-1}, P_{i+1}) = \arccos(\mathbf{a}\bullet\mathbf{c}); \qquad \tau_{i+1} = \text{angle}(P_{i+1}, P_{i+2}, P_i) = \arccos(\mathbf{e}\bullet\mathbf{d}); \qquad (5)$$

The actual tangent directions, t_i and t_{i+1}, that define the two arcs, arc_i and arc_{i+1}, can be found by rotating the direction vector **b** around the appropriate rotation axes by the amounts indicated in Eqn.(5):

$$\text{axis}_i = \mathbf{b}\times\mathbf{a}; \quad \text{axis}_{i+1} = \mathbf{b}\times\mathbf{d} \qquad (6)$$

The tangent angle $\tau_{i+1}^m = -\tau_{i+1}$, required to calculate the angle-interpolated arc(u) from its starting point at P_i, is obtained by mirroring the tangent t_{i+1} at the perpendicular bisector between P_i and P_{i+1}. In this way we can obtain all the elements needed for the computation of the intermediate arc, arc(u), without explicit reference to any circles (Fig.9a).



(a) **(b)**

Figure 9: Robust calculations of (a) the tangent angles at P_i and (b) the position of an intermediate blend point P(u).

In the same fashion, the point P(u) traveling on arc (u) is parametrically described as a distance from endpoint P_i:

$$f(u) = b \sin(u\,\tau(u)) / \sin(\tau(u)) \qquad (7)$$

and a deviation angle from line segment (P_i, P_{i+1}):

$$\phi(u) = (1-u)\,\tau(u), \qquad (8)$$

where b is the length $|P_i, P_{i+1}|$ of line segment (P_i, P_{i+1}). It should be pointed out that this constitutes a constant-velocity parameterization along the arc, and that the angle $\phi(u)$ linearly decreases from an initial value of $\tau(u)$ to a final value of zero (Fig.9b). For the case of a straight segment between P_i and P_{i+1}, implying an infinite radius, $\phi(u)$ is simply constant at value zero, and the distance function f(u) converges towards f(u) = u b. Note that this calculation is applicable, regardless whether we draw a circle spline in a given plane, on a given sphere, or freely in 3D space. Once the base tangent directions have been determined, the swiveling, interpolated tangent vectors determine the intermediate arcs, and for each one of them, the procedure above determines the point P(u) needed to form the blend curve.

3.2 Circle Splines in Space

If the data points happen to lie in an arbitrary configuration in 3- or higher-dimensional space, the above construction for the plane needs to be enhanced by an additional degree of freedom. For that purpose we introduce a Swivel-Plane that rotates around the line segment (P_i, P_{i+1}), from a position coplanar with P_{i-1}, P_i, P_{i+1}, to a position coplanar with P_i, P_{i+1}, P_{i+2}., as the parameter u increases from 0 to 1 (Fig.10). For any value of u, we conceptually construct that Swivel_Plane(u), and arc(u) embedded in it, and than a single curve point P(u) on that arc. To find that point P(u) it is sufficient to determine a tangent vector t(u) at point P_i, which implicitly defines the intermediate arc(u) . This tangent t(u) is found by a swivel operation in the plane spanned by t_i and by t_{i+1}^m, the mirrored tangent t_{i+1} drawn at P_i, rotating around the axis given by:

$$\text{axis}_{i\_t} = \mathbf{t}_i \times \mathbf{t}_{i+1}^m \qquad (9)$$

This swivel operation could simply be done at a linear rate, but with the aim for G^2-continuity, we use again the trigonometric blending function of Eqn.(3). With $t(u)$ defined, the point $P(u)$ is then found in the plane defined by $t(u)$, P_i and P_{i+1} using equations Eqn.(7) and Eqn.(8). In reality this means that we rotate the vector $f(u)$ that defines $P(u)$ around an axis:

$$\text{axis}_{i\_f} = \mathbf{t}(u) \times \mathbf{b} \tag{10}$$

by the amount $\phi(u)$, rather than around the z-axis as in the planar case.

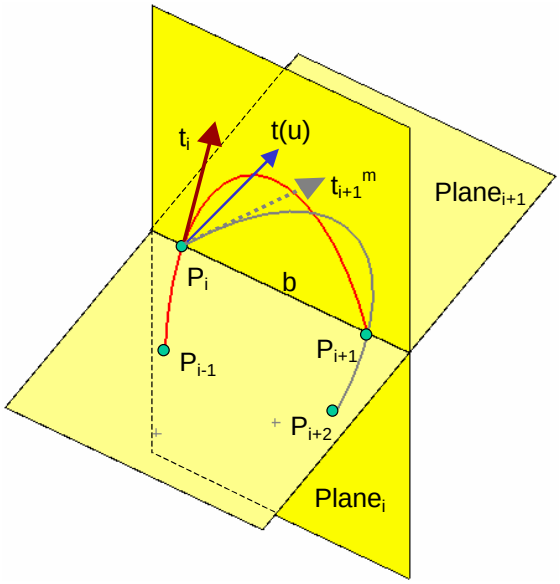


Figure 10: Planes and tangents in 3D space.

With this construction, each blended curve segment will individually lie on a sphere that passes through four consecutive points of the control polygon, and which has the plane spanned by t_i and by t_{i+1}^m as a tangent plane at point P_i. The new trigonometrically-weighted, angle-based blending will guarantee that adjoining curve segments adopt the same osculating circles at their junction point; this results in common tangent directions, osculating planes, and curvatures, i.e., G^2-continuity. Again, cusps and excessive loops in these segments are avoided; and this same scheme thus also produces very pleasing, smooth space curves. An example of a Figure-8-Knot defined by only 8 control points is shown in Figure 11.

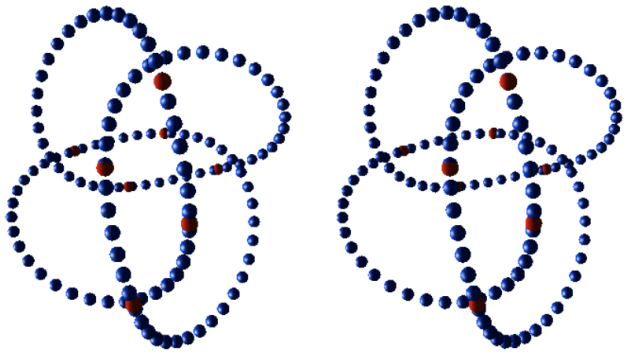


Figure 11: Circle-splines forming a Figure-8-Knot in 3-space (cross-eye stereo view).

In summary, here is a unified description of how to construct circle splines in (n-dimensional) Euclidean space. First, construct circles through three consecutive data points; these define the tangent directions at the middle points. Next, look at

the two base arcs that span the control segment P_i, P_{i+1}, and determine the unit tangents to both arcs at both points, P_i and P_{i+1}. The two tangents at each point define a plane in which the angle-based swivel operation will be carried out. For any value of the parameter u, the interpolated tangents at P_i and P_{i+1} will define an intermediary arc(u) from P_i to P_{i+1}, and the point with parameter value u on this arc will be point P(u) of the blended curve segment.

3.3 Circle Splines on the Sphere

The construction of a circle spline on a sphere automatically follows from the approach described above for 3D space. The only difference is that now all the data points lie on a single sphere. Since each blend curve segment lies on a sphere defined by four consecutive data point, the whole curve will now also lie on the given sphere. Figure 12 shows the construction of an arbitrary curve point P(u) on the intermediate arc(u) within Swivel_Plane(u).

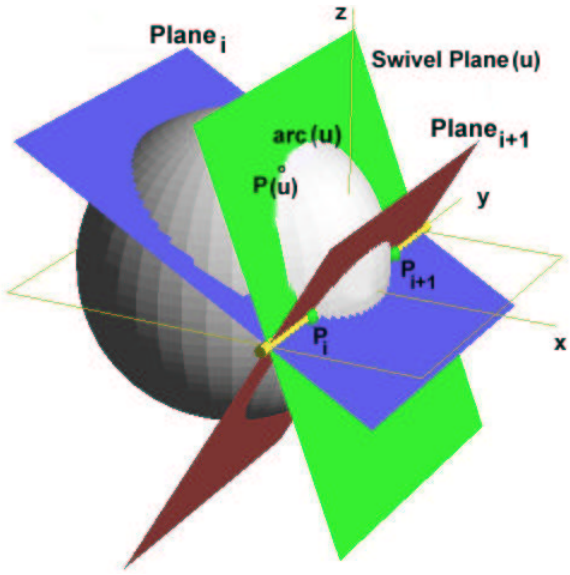


Figure 12: Construction of an intermediate curve point of a circle spline embedded in a sphere.

Thus, for one blend segment at a time, the construction is identical in 3D space and on a sphere. The implied spheres for subsequent blend segments share the same base arc through the common junction point. Also, using the approach discussed in Section 3.1, there is no need to calculate the parameters of these spheres explicitly, and thus the transition from a very large sphere to a plane is smooth and natural. Figure 13 shows examples of circle splines on spheres where the control polygons offer only very sparse data sets. The same local "spherical confinement" for consecutive blending segments also produces very pleasing looking splines for free-form 3D space curves in a rather robust manner (Fig.11).

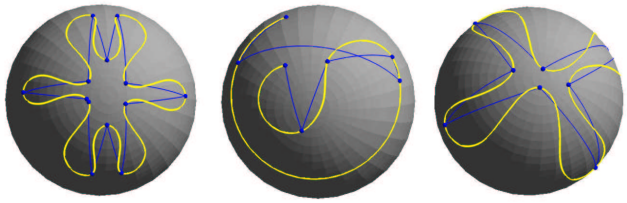


Figure 13: Examples of circle splines and their control polygons on spheres.

3.4 G^2-Continuity of Individual Blend Segments

Circle splines offer strict G^2-continuity. To prove this, we first analyze the smoothness of an individual blend curve, and then the continuity at the junctions between them.

A single blend curve is best analyzed in the context of the family of parameterized arcs (Fig.8). For the planar case, the motion of the point $P(u)$ has two components: a "tangential" component that carries the point along an arc from P_i towards P_{i+1}, and a "radial" component due to the change in the shape of the arc as its tangent angle $\tau(u)$ with (P_i, P_{i+1}) varies. Both these motion components are smooth and infinitely differentiable; thus C^2-continuity is guaranteed. To assert G^2-continuity, we have to show that the combination of these movements cannot produce any cusps, as may occur for polynomial curves when all component velocities simultaneously drop to zero. Both the above two motions are monotonic and without turn-around points, so the respective velocity components never go to zero. They also do not align anywhere, so that their individual velocities could cancel each other.

In the case of space curves, we also have to consider the "swivel" component (Fig.10) produced by the rotation of the arc plane around the axis (P_i, P_{i+1}). The tangent $t(u)$ can be found by rotating $t_i(u)$ by the amount of Eqn.(3) around axis Eqn.(9). Thus for u=0 the Swivel_Plane is identical to $Plane_i$ and for u=1 it is coplanar with $Plane_{i+1}$. This motion has a velocity component that drops to zero only on the rotation axis. However, since neither the tangential nor the radial velocity ever drop to zero, no cusps can form on this axis either.

An attempt was made to combine the expressions for the three individual component movements into a single explicit formula, which could then be differentiated to find the extremas of the velocities of $P(u)$. But this explicit formulation becomes too complicated to formulate a rigorous proof that under no combinations of parameters the length of the derivative vector ever assumes the value zero. A few minutes with the interactive graphics applet [12], wildly moving around the various control points, makes a much more convincing statement about the robust behavior of the interpolating curve segment.

3.5 G^2-Continuity of the Junctions

To analyze the continuity of the junction between two consecutive blend segments at P_i, we use the base arc through P_{i-1}, P_i, and P_{i+1} as a reference and study the rate at which the two blend curves on either side of point P_i deviate from this arc. Of course, we cannot expect to obtain C^1-continuity through this junction, unless we reparameterize the base arc segments on either side according to their relative arc lengths (see Section 3.6).

For the planar case, each blend segment is calculated in polar coordinates (f, ϕ) as a point $P(u)$ moving away from point P_i (Section 3.1). By converting to Cartesian coordinates and substituting $\tau(u)$ from Eqn.(3), we obtain a single continuous vector expression that explicitly describes the shape of the blend segment as a function of the parameter $u \in [0,1]$:

$$P(u) = [\, f(u) \cos\phi(u),\ f(u) \sin\phi(u) \,]. \qquad (11)$$

We differentiate this expression twice, and collect the non-zero terms for $u = 0$. The first derivative at point P_i evaluates to:

$$P'(0) = [\, b\, \tau_i \cos(\tau_i) / \sin(\tau_i),\ b\, \tau_i \,], \text{ with } b = |P_i - P_{i+1}|, \qquad (12)$$

which is the proper tangent velocity on arc_i through P_i. In order to calculate the curvature κ at point P_i, we evaluate the expression $\kappa = | P' \times P'' | / | P' |^3$ for $u = 0$. Using trigonometric blending (Eqn.(3)), the curvature κ becomes:

$$\kappa(0) = 2 \sin(\tau_i) / b, \qquad (13)$$

which is equal to the curvature of the arc_i through P_i. The same analysis applies at point P_{i+1}. More intuitively, the quadratic term in the trigonometric blending formula forces the deviation of curvature from the base arc near the junction to be at least linear, thus going to zero at the junction itself; therefore the curvatures of both segments match that of the base arc and thus are equal to each other.

For the case of space curves, there is an additional degree of freedom to be concerned about, but conceptually the situation is the same. Again, the deviation from the common base arc is governed by the swiveling tangent vector $t(u)$, but now the two deviations of the two segments may lie in different planes. However, since that deviation is a function of the same degree as in the planar case, it does not matter; the two curve segments will still meet at the junction with the curvature of the base arc.

With trigonometric blending, G^2-continuity can be maintained at all junctions, as long as we avoid degenerate control polygons with zero-length control segments. With a simple linear angular blending, on the other hand, not as many terms vanish, and some terms containing τ_{i+1} remain in the expression for the curvature at point P_i. Thus the curvatures of the two blend curves joining together at point P_i are not solely dependent on the curvature of the base arc through P_i, and curvature continuity can thus not be guaranteed.

3.6 Reparameterization for C^2-Continuity

In order to also obtain parametric continuity at the segment junctions, we need to reparameterize the velocities with which the individual segments are traversed. One possible adjustment would be to find an explicit arc-length parameterization, so that we can traverse all segments with uniform velocity; however, the necessary calculations are rather involved, and there is a simpler way. Assuming trigonometric blending, the blended curve segments pick up the behavior of the base arcs near the interpolation points to the second degree. Thus we may change the parameterization of all base arc segments so as to maintain a predefined fixed arc-length velocity on all base arc segments, valid on both sides of any interpolation point. To obtain C^1-continuity on the circle spline, we then need a parameterization for the blend segments that matches these end-velocities defined on the base arcs. For this we need at least a quadratic function to provide enough degrees of freedom to meet all the constraints. If we aim for C^2-continuity, we need a quartic function, so that we can also set the accelerations at the junctions to zero:

$$u(t) = a + b\,t + c\,t^2 + d\,t^3 + e\,t^4.$$

Now we use the following constraints:

$u(0) = 0$, implies $a=0$;

$u'(0) = v_i$, implies $b= v_i$;

$u''(0) = 0$, implies $c=0$;

$u(t_{max}) = 1$;

$u'(t_{max}) = v_{i+1}$;

$u''(t_{max}) = 0$;

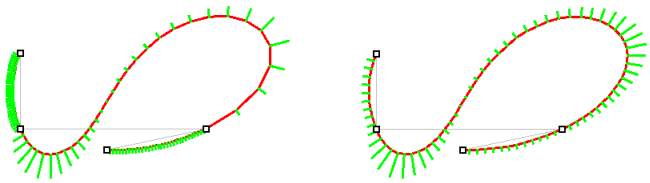
A few lines of arithmetic resolve these constraints to:

$$t_{max} = 2/(v_i + v_{i+1});$$

$$d = (v_{i+1} - v_i)\,(v_i + v_{i+1})^2 / 4;$$

$$e = (v_i - v_{i+1})\,(v_i + v_{i+1})^3 / 16;$$

By applying this reparameterization, we get C^2-continuity at the junctions and a reasonably regular spacing of the time tick marks along the whole curve, as shown in Figure 14b compared to the original spacing which employed a fixed number of tick marks on each segment (Fig.14a).

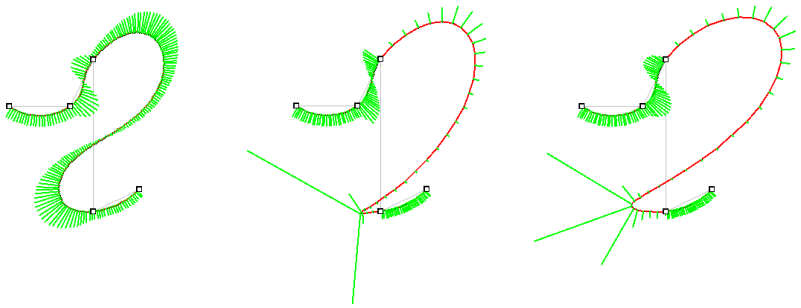


(a) **(b)**

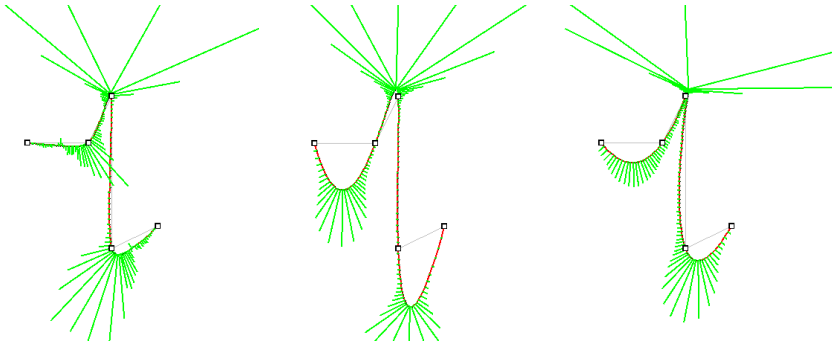
Figure 14: Comparison of (a) original parameterization with a fixed number of tick marks along every arc segment, and (b) C^2-continuous parameterization; also shown at every tick mark are "bristles" proportional in length to the curvature at that point.

4. RESULTS AND PROPERTIES OF CIRCLE SPLINES

Figure 15 shows a direct comparison of the new angle-based circle-spline (Fig.15a) with two previous circle blending schemes using either linear [18] (Fig.15b) or trigonometric interpolation [14] (Fig.15c) of point positions. Note that the latter two schemes look much "loopier" and that they may produce cusps for certain control polygons.



(a) **(b)** **(c)**



(d) **(e)** **(f)**

Figure 15: Comparison of (a) the circle spline, (b) linearly [18] and (c) trigonometrically [14] blended circular arcs, (d) a cubic subdivision scheme [19], and (e) uniform [17] and (f) non-uniform Waring-Lagrange interpolation [13].

Figure 15 further compares some polynomial interpolation schemes; (d) is a cubic subdivision scheme [19], and (e) and (f) use Waring-Lagrange interpolation [17],[13]. These schemes all exhibit rather sharp turns near some of the control points, accompanied with excessive values in curvature. The subdivision scheme also features some extraneous sign inversions in the middle of some segments. The basic Lagrange interpolation exhibits some excessive overshoots (Fig.15e), which can be mitigated somewhat by using non-uniform knot intervals, proportional in length to the lengths of the corresponding control polygon segments (Fig.15f). None of these schemes can compete with the smooth, curvy appearance (Fig.15a) of the angle-based circle spline. In the following section we will summarize some of the other properties of these new splines.

4.1 Linear and Circular Precision

If all given points are distinct and monotonically ordered along a circular arc or straight line, then the whole result curve will also fall onto this path – which corresponds to the behavior of an ideal Minimum-Variation Curve (MVC) [7]. Figure 3 shows a situation where all the control points lie on two circles. In this case, the deviation from these ideal shapes is strictly limited to the blended central segment (P_i, P_{i+1}).

4.2 Transformation Invariance

The circle spline construction is independent of the position, orientation, and uniform scaling of the coordinate system in which it is carried out. However, it is not invariant to non-uniform scaling or to general affine distortions of the coordinate system. This is a direct consequence of our circle-fitting step through three consecutive points of the control polygon, which counteracts any non-uniform scaling effects and gives equal weights to all dimensions in Euclidean space.

4.3 Symmetry Preservation

Circle splines preserve all symmetries exhibited by the original set of control points. The curve is not dependent on evaluation order – unlike some quaternion splines [5] – and thus also exhibits “front-to-back” symmetry.

4.4 Continuity

Circle splines offer strict G^2-continuity, and after suitable reparameterization even C^2-continuity, as shown in Sections 3.4-5 and 3.6, respectively.

4.5 Fairness

G^2-continuity is a major step towards providing fair-looking curve shapes, but the notion of fairness also entails a lack of gratuitous non-monotonicities, e.g., no unneeded inflection points. However, interpolating curves cannot possess the variation-diminishing property exhibited by many approximating splines. Consider a control polygon with a collinear sequence of points P_0, … P_i, a sharp bend at P_i, and another collinear sequence P_i, … P_n after that (Fig.16). If we want the curve to pass smoothly through the corner point P_i, it will clearly have to exhibit some overshoot with respect to the two collinear sequences on either side of it. The second segments on either side, however, will remain perfectly straight.

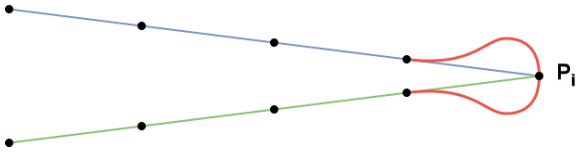


Figure 16: Non-variation-diminishing property of circle splines.

4.6 Stability

The globally optimized MEC and MVC have the undesirable property that they may run away to infinity during the global optimization process under certain constellations of data points. Special techniques have to be introduced to make them scale-invariant and thereby unconditionally stable [10]. In our context, dynamic run-away is not a problem, and each blended curve segment lies entirely between its two base arcs.

4.7 Discontinuous Behavior

Stable – and thus predictable – behavior also means that when a small change is made to the input constraints, i.e., to the given control points, then we also expect to see a small change in the resulting curve. This is generally true, unless the input modification implies a topological change. For instance, if point P_{i+2} is moved across the line segment (P_i, P_{i+1}), then the base arc, arc_{i+1}, through P_{i+1} will "flip" to the other side of the segment, and thus the resulting curve will show a discrete change. In the plane, the curve must "flip" from one side of the line segment (P_i, P_{i+1}) to the other side when the turning angle at one end exceeds 180° and thus changes sign. We must expect such discontinuous behavior, since we have ruled out the dimensional collapse associated with affine invariance. On a sphere, however, the curve segment may smoothly swing around the back of the sphere (Fig.17).

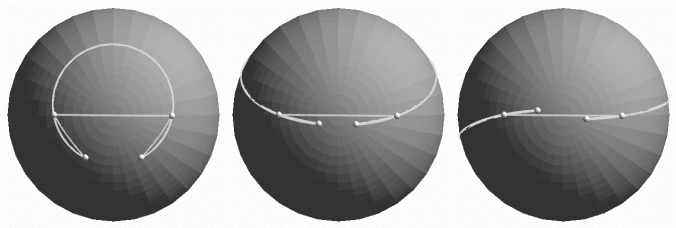
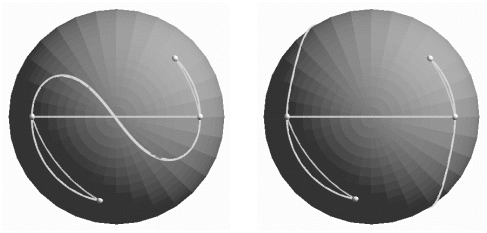


Figure 17: Smooth transition around the back of the sphere.

In other situations, some "flipping behavior" is unavoidable even on the sphere. If the tangent angles at P_i and P_{i+1} are increased in an anti-symmetrical manner, a wave with an inflection point appears, which then grows into a sweeping S-curve (Fig.18a). Eventually, as the two turning angles get closer to ±180°, a helical loop around the backside of the sphere is the

preferred solution (Fig.18b). The question thus arises: what is the exact point at which the curve should flip into this other mode? This is equivalent to asking: through which of the two possible sweep angles between τ_i and τ_{i+1} should we swivel the tangent vectors to produce the intermediate arcs for the blending process? A straightforward choice is to swivel them through the angle that is less than 180°. Figure 18 shows this approach. The flip to the backside occurs when the two planes containing the two base arcs are more then 180° apart.

Since we would like to see a smooth transition in behavior when going from a very large sphere to a plane, the size of the sphere should be taken into account; i.e., it does not seem appropriate to avoid a slight wiggle and an inflection point for the price of going around the backside of a very large sphere. The scheme illustrated in Figure 18 does this in an implicit manner. As the segment (P_i, P_{i+1}) spans an ever larger fraction of a great circle on the sphere, the condition of coplanarity for the four points, P_{i-1}, P_i, P_{i+1}, P_{i+2}, is reached for ever smaller turning angles of the control polygon at points P_i and P_{i+1}.



(a) **(b)**

Figure 18: A flip-point with four coplanar control points.

5. DISCUSSION

We have built an applet [12] that lets one explore in real time the behavior of the circle splines discussed in this paper. Interactively moving points around with a mouse and watching the effect on the resulting curve is a quick and effective way to compare different circle-blending schemes and to spot problems with any of them. Only a few minutes of playing with this applet are sufficient to confirm one of the key insights of this work: when blending circles, the **turning angles** are the preferred variables for carrying out the interpolation. In all the schemes that interpolate point **positions** between the two original base arcs, one can readily produce weird behavior, i.e., cusps, unnecessary inflection points, and unexpected flips of state, within seconds of moving a few control point positions. The angle-based schemes are less prone to such effects. This advantage was already visible in the angle-based subdivision scheme [11]; it did indeed yield nice and pleasing looking curves.

The swept (or iterated) variant of the circle spline described in this paper produces even better results. It is at least as pleasing as the subdivided circle spline, but gives more robust and more predictable results in extreme situations. Most importantly, with this approach we can now make quantitative claims about the continuity of these curves. In particular, we can guarantee strict G^2-continuity at all curve points, something not possible for traditional polynomial splines.

Curve constructions that guarantee G^1 or even G^2-continuity, do not necessarily guarantee a “fair” curve. There may be unnecessary undulations (wrinkles, wiggles) due to sign inversions of curvature, or uneven bends due to sign changes in the derivative of curvature. Often such artifacts result because the surface is represented with underlying primitives that are of a higher order than is needed to represent the basic smooth shape desired. Best results are obtained, if the fewest, simplest

primitives are used to form the desired shape, and if these primitives closely correspond to the desired goal shapes. If one prefers circular shapes wherever they are compatible with the constraints, then it is advantageous to use circles in the construction process itself. This reasoning motivated our study of circle blending schemes with the aim of obtaining a class of fair, interpolating curves.

One of our inspirations was the Minimum Variation Curve or MVC [7], since its superior fairness has been pointed out earlier [10]. This curve also produces circles wherever possible. Fitting circles through 3 consecutive data points destroys the invariance under general affine transformations, e.g., non-uniform scaling. We consider this a positive feature; it prevents dimensional collapse when all data points approach collinearity and thus avoids the generation of hairpin turns. The circle construction always considers the given constraints in a truly two-dimensional way that gives equal weights to both coordinate axes.

However, even the best circle splines fall short of the performance of the globally optimized MVC. A typical case is shown in Figure 19. Five data points define an S curve with an inflection point near the central control point. Since the central three points are collinear, a straight line segment is fitted at the center point, and the tangent direction at the central inflection point is chosen solely based on information from the nearest neighbor data points; clearly this is not optimal from a more global perspective.

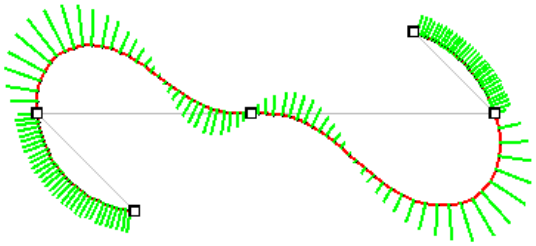


Figure 19: Extra inflection points due to localized support domain.

The extra undulations in Figure 19 are a direct consequence of the very limited local support domain. To find a better choice for the tangent direction at the midpoint, we would have to include information from the second-nearest neighbors as well. This may then require the use of a higher-order primitive for the blending operation. The Cornu spiral or clothoid [3] is a natural candidate. These curves have a monotonically changing curvature that varies linearly with arc length. If we must connect two points with different curvatures, then the clothoid is an MVC solution, since any non-uniform change in curvature would lead to a larger integral value when the square of the curvature change is summed. Thus, the clothoid is a natural primitive for constructing higher order approximations to an MVC.

Of course, this raises many new challenges: how difficult is it to find the best matching clothoid through four consecutive points? How robust is the determination of the solution? How do we then blend the overlapping clothoid segments together? How does this translate into 3D space? The first improvement we might want to make for the 3D domain is to obtain a natural match for helices – thus helices should be a primitive. However, to be compatible with the 2D domain, we would still need to employ clothoids. This in turn may then prompt us to also consider curves with linearly varying torsion. The investigation of such higher-order primitives is beyond the scope of this paper.

6. CONCLUSIONS

Circle splines provide a smooth transitional blend between two circular primitives. This shape transition can be seen as a gradual morph from one arc to the other as we move along the blend curve. Contrary to a simple positional interpolation between corresponding pairs of points on the two arcs, this morphing operation can be carried out in a way that does not produce undesirable artifacts such as loops or cusps. Circle splines using angle-based blending robustly produce fair-looking G^2-continuous curves even under extreme control point constellations, which can be enhanced to C^2-continuity with a simple reparameterization. They appear to be the best approximation to the desirable behavior of Minimum Variation Curves with the restriction of using a support domain of only four data points.

REFERENCES

[1] D. Asimov, “The Grand Tour: A Tool for Viewing Multidimensional Data.” SIAM Jour. of Scientific and Statistical Computing Vol 6, No1, pp 128-143 (1985).

[2] K. M. Bolton, “Biarc curves.” CAD Vol 7, No 2, pp 89-92 (1975).

[3] Gray, A. "Clothoids." §3.7 in Modern Differential Geometry of Curves and Surfaces with Mathematica, 2nd ed. Boca Raton, FL: CRC Press, pp. 64-66, 1997.

[4] B. K. P. Horn, “The curve of least energy.” ACM Trans. Math. Software Vol 9, No.4, pp 441-460 (1983).

[5] M. J. Kim, M.S. Kim, and S. Y. Shin: “A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives.” Proc. ACM SIGGRAPH '95, pp. 369-376 (1995).

[6] R. Liska, M. Shashkov, and B. Swartz. “Smoothly Blending Successive Circular Interpolants in Two and Three Dimensions.” Technical Report LA-UR 98-4969, LANL, Los Alamos, 1998.

[7] H. P. Moreton and C. H. Séquin: “Functional Optimization for Fair Surface Design.” Proc. ACM SIGGRAPH'92, pp 167-176 (1992).

[8] C. J. Ong, Y.S. Wong, H.T. Loh, and X.G. Hong: “An optimization approach for biarc curve-fitting of B-spline curves.” CAD Vol 28, No12, pp 951-959 (1996).

[9] J. Schonherr, “Smooth biarc curves.” CAD Vol 25, No 6, pp 365-370 (1993).

[10] C. H. Séquin, P. Y. Chang, and H. P. Moreton: “Scale-Invariant Functionals for Smooth Curves and Surfaces.” Proc. 1993 Dagstuhl Seminar on Geometric Modelling, Hans Hagen, ed., Computing Supplement 10, Springer, pp 303-321, (1995).

[11] C. H. Séquin and J. A. Yen: “Fair and Robust Curve Interpolation on the Sphere.” Sketches and Applications, p.182, SIGGRAPH '01, Los Angeles, Aug. 15, 2001.

[12] C. H. Séquin and K. Lee: “Circle Spline demonstration applets.” – http://www.cs.berkeley.edu/~sequin/CAD/CircleSplines.html 2004.

[13] Séroul, R. "Lagrange Interpolation." §10.9 in Programming for Mathematicians. Berlin: Springer-Verlag, pp. 269-273, 2000.

[14] M. Szilvasi-Nagy and T.P. Vendel: “Generating curves and swept surfaces by blended circles.” CAGD 17, pp 197-206, (2000).

[15] W. Wang and B. Joe, “Classification and properties of space biarcs.” SPIE Vol.1830: Curves and Surfaces in Computer Vision and Graphics III, Nov. 1992, Boston, pp184-195.

[16] W. Wang and B. Joe, “Orientation interpolation in quaternion space using spherical biarcs.” pp 24-32, Graphic Interface '93.

[17] E. Waring, “Problems Concerning Interpolations.” Philosophical Transactions of the Royal Society of London, vol. 69, pp. 59-67 (1779).

[18] H.-J. Wenz: “Interpolation of curve data blended by generalized circles.” CAGD 13, pp 673-680, (1996).

[19] D. Zorin, P. Schröder, and W. Sweldens, “Interpolating Subdivision for meshes with arbitrary topology.” Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp 189-192, (1996).