Curvature-controlled Curve Editing using Piecewise Clothoid Curves

Sven Havemann[a], Johannes Edelsbrunner[a], Philipp Wagner[a], Dieter Fellner[a,b]

[a]*Institut für ComputerGraphik & WissensVisualisierung (CGV), TU Graz, Austria*
[b]*TU Darmstadt & Fraunhofer IGD, Germany*

Abstract

Two-dimensional curves are conventionally designed using splines or Bézier curves. Although formally they are C^2 or higher, the variation of the curvature of (piecewise) polynomial curves is difficult to control; in some cases it is practically impossible to obtain the desired curvature. As an alternative we propose piecewise clothoid curves (PCCs). We show that from the design point of view they have many advantages: control points are interpolated, curvature extrema lie in the control points, and adding control points does not change the curve. We present a fast localized clothoid interpolation algorithm that can also be used for curvature smoothing, for curve fitting, for curvature blending, and even for directly editing the curvature. We give a physical interpretation of variational curvature minimization, from which we derive our scheme. Finally, we demonstrate the achievable quality with a range of examples.

1. Introduction

The creation of good-looking curves is a fundamental task in 2D (and 3D) shape design. Through a recent collaboration with the surfacing department of a car manufacturer we realized how difficult it is in practice to convert a given design curve into a high-quality parametric curve that is suitable for further processing in a high-end CAD system (CATIA). Observing the work of the surface engineers we developed three hypotheses about their requirements:

Hypothesis 1. *The control polygon alone must unambiguously, and in a predictable way, define the shape.*

Hypothesis 2. *The control polygon must be as sparse as possible (a) to offer control and (b) to avoid artifacts.*

Hypothesis 3. *Controlling curvature is essential; much time is spent on removing curvature artifacts.*

Curvature control is so important in high-quality ("*class-A*") shape design because curves are the basis for creating surfaces, and curvature artifacts lead to bad light reflections. State of the art in industrial design is the use of (piecewise) polynomial curves such as Bézier curves, B-splines, and their numerous variations. They are efficient to compute, their properties are well understood, and many algorithms exist for knot insertion, degree elevation, etc. The first hypothesis, however, precludes curve representations with invisible extra parameters such as NURBS, β-, τ-, or ν-splines; even varying the knot spacing is usually avoided. The use of any such non-graphical parameters has the crucial disadvantage that the curve shape cannot be judged only by looking at the control polygon; but this is exactly the expertise of surface engineers. This is why professional class-A surfacing software like *IcemSurf* still uses exclusively the oldest and most direct surface technique, tensor product Bézier surfaces of degree three up to nine.

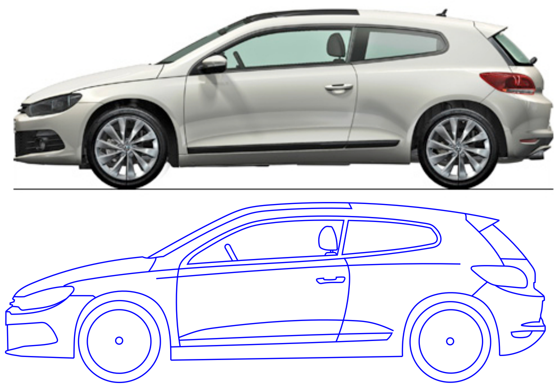
Figure 1: Design curves of a car captured with PCCs.

The curvature of spline curves is difficult to control. In some cases it is even impossible to obtain the desired curvature practically, i.e., using a sparse control polygon (see Section 8). The reason is that, informally speaking, spline control points exert an ’extra pull’ on the curve. A B-Spline with a regular n-gon as control polygon still deviates noticeably from a circle; without an extra weight parameter (rational splines) it is impossible to obtain a perfect circle. The real problem, however, is the lack of clear rules for the placement of control points. Surface engineers need years of experience to master the control point placement intuitively.

1.1. Piecewise clothoid curves as superior alternative

The work presented in this paper is the result of the quest for a curve representation that has no hidden parameters and offers exactly the degrees of freedom that designers need in order to control both the shape and the curvature of a 2D curve. We propose replacing splines by piecewise clothoid curves (PCCs). The heart of the framework is a fast algorithm to compute a PCC from a given (open or closed) sequence of input points. The problem can be stated formally as follows:

> Given 2D points $p_1, \ldots, p_n$, compute clothoid segments $c_1(t), \ldots, c_{n-1}(t)$ with arc lengths $d_1, \ldots, d_{n-1}$ so that $c_1(0) = p_1$, $c_i(0) = c_{i-1}(d_{i-1}) = p_i$ for $i > 1$, and $c_{n-1}(d_{n-1}) = p_n$.

There is no suitable closed-form representation of clothoid curves; they are defined via Fresnel integrals and computing them is impractical [1]. Therefore we use an iterative discrete scheme (see Section 3).

1.2. Contribution

Our contribution is a unified framework for curvature-controlled curve design with PCCs:

- Fast adaptive clothoid interpolation algorithm
- Curvature simplification by dynamic programming
- Clothoid blending by nonlinear subdivision
- Direct editing of the curvature plot
- Physical interpretation of curvature smoothing
- Different control modes for points: free, constrained tangents, and constrained curvature.

1.3. Benefit

Clothoid curves are well known for their aesthetic quality. However, they have never been widely used in shape design, maybe for efficiency reasons, but certainly also because of the lack of suitable design tools. We argue that with PCCs, designers can reach their goal much faster and with an excellent level of control, for example to meet max/min curvature constraints, to limit the curvature variation, and to obtain aesthetically pleasing results. PCCs are much more ’relaxed’ than splines; the infamous ’spikes’ with unbounded curvature are avoided. Tension can still be added to the curve intentionally by explicitly inserting short segments as it is demonstrated, e.g., in Figures 1 and 14.

In summary, we show that PCCs are superior to splines in practice: Any spline curve can be approximated by a PCC with a sparse control polygon, but the converse is not true (see Section 8).

2. Related Work

A variational approach for computing discrete approximations of piecewise clothoid curves was presented in a fairing context in [2]. Like in our algorithm, they refine a given polygon by inserting new points and moving them such that the curvature distribution becomes piecewise linear (direct approach). They also propose an indirect approach, iteratively alternating between curvature estimation at the control points and curvature interpolation at intermediate points. Our method is similar to their direct method, but we take the segment lengths into account. This assures fast convergence also for adjacent segments of greatly varying lengths. This is the key for adding fine details and ’tension’ to the clothoids. Clarifying about terminology, our *piecewise clothoid curve* (PCC) is called *discrete clothoid spline* (DCS) in [2]; but since splines are often associated with convolution or blending, we find it more appropriate to call a *clothoid spline* the result of the curvature refinement process presented in Section 5.

It was observed already in the 1970s that clothoids are useful for interpolating data points with specified tangent and curvature. As proposed in [3] they can be connected with *linear curvature elements* (linces) that are integrated to yield clothoids. The approach was extended by [4, 5] who developed blending patterns for bi-, tri-, quadrilinces to connect a pair of data points. They used a direct integration method while our method is global and works in a variational setting, interpolating all control points. While their method requires tangents and curvature in every control point, our method guarantees that every two points are connected only by a single clothoid. An advantage of their method is that modifications have only local impact, in contrast to our method where the impact is global; however, we can also achieve the same, since specifying constraints effectively decouples the parts of our curves.

An extension of the clothoid (Cornu spiral) to the generalized Cornu spiral (GCS) is presented in [6]. The curvature is not only linear but rational, so additional degrees of freedom are available; however, they are not visual, and so they are difficult to control by designers. A useful overview of clothoid (Euler spiral) techniques is given in the comprehensive thesis [7]. It includes mathematical foundations, application examples, and also historical background.

The approaches [8, 9] are concerned with fitting a clothoid curve to sketched input data. They use a dynamic programming approach in the curvature domain in order to identify portions of the input curve that can be approximated by a clothoid segment. The fitting algorithm presented in Section 7.3 borrows from their dynamic programming approach, but we use it to directly extract suitable PCC control points from the input curve. This simplifies the procedure since we do not have to fit so many clothoid segments; and our PCC does not deviate so much from the input curve since the control points are interpolated.

The method of [10] builds upon the principles of [8] for fitting clothoid curves to sketched input data. A large number of straight line, circle, and clothoid segments is tentatively fitted, and then represented as nodes of a graph from which the best candidate segments are identified using a flow algorithm. The segments found are then optimized in order to meet, thus obtaining a piecewise clothoid curve. This approach has fewer problems than [8] concerning the deviation from the input data, but the method is much more complex than ours; and it does not yield conveniently editable control points.

Another approach building upon [8] is [11]; it is concerned with the related topic of fitting French curves to sketched input data. Also concerned with fitting curves to sketched input data is [12]. They define a so-called *Elasticurve* which is roughly a smooth version of the input curve, represented as lines, parabolas, and arcs.

Like our paper, [13] is concerned with a clothoid curve representation that is interactively controllable. In contrast to our interpolated control points, they use a control polyline to which a clothoid curve is fitted. A generalization of clothoids to 3D was presented in [14]; their fitting algorithm produces 3D curves where not only curvature but also torsion varies linearly with arc length; however, they are not concerned with high-quality curve design as their main topic.

3. Computing Piecewise Clothoid Curves

A piecewise clothoid curve (PCC) consists of various clothoid segments, which may also comprise line segments and circle segments. The segments are joined together such that they are both tangent and curvature continuous in the joints (G^2 continuous).

3.1. Discrete PCC Curves

We use a method where a sequence of control points is either specified interactively by the user, or chosen appropriately from a given input curve; then our algorithm constructs clothoid segments joining these points. Clothoids are defined by Fresnel integrals, for which different approximations exist [1]. Instead of computing the parameters of the clothoid segments we use a variational approach generating a polyline with linear discrete curvature, thus approximating the clothoid segment. The approximation error can be made arbitrarily small by iterative refinement.

3.2. Iterative Algorithm

The iteration starts with the control polygon, i.e., the polyline through the control points. Then we repeat two alternating steps. First, a new point is inserted between every two points of the polyline; the sequence of points that are inserted between consecutive control points are called a *segment*. Second, the position of all inner segment points (i.e., except the original control points) is optimized. The new position of a point is computed on the perpendicular bisector of its neighbors in such a way that the curvature is the arithmetic mean of the curvatures of its neighbors.

Throughout this paper, the (discrete) curvature in a point p of a polyline is computed simply as the inverse radius of the circle through p and its two polyline neighbors. Consequently, five points are involved when optimizing C, namely its direct neighbours and their respective neighbours. The curvature information is transferred from one segment to the next over control points. The second consequence is that inner segment points quickly converge to equal spacing and to linear curvature distribution, thus obtaining a discrete clothoid.

3.3. Point positioning

Figure 2 illustrates the position computation. In order to insert (or update) point C with neighbours B and D, their respective neighbors A and E are considered. Without loss of generality we use the normalized configuration where B lies in $(-1, 0)^T$ and D in $(1, 0)^T$. We use the following notation:

- let X be one of the five control points, and X_l his left and X_r his right neighbor
- δ_X is the angle between $\overrightarrow{XX_l}$ and $\overrightarrow{XX_r}$
- κ_X is the discrete curvature in point X

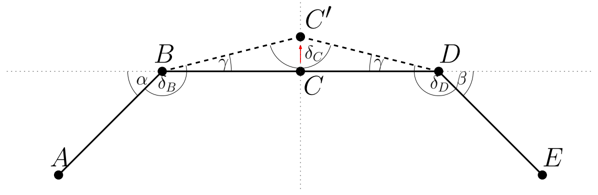

Figure 2: Optimization of point C. Angles α and β are given by the position of the points with respect to the dotted line g through B and D. C' is positioned on the perpendicular bisector of B and D such that δ_C is the arithmetic mean of δ_B and δ_D. γ is used for the calculation.

- g is the line through B and D
- α is the angle between g and $\overrightarrow{BA}$
- β is the angle between g and $\overrightarrow{DE}$
- γ is the angle between g and $\overrightarrow{BC}$

Two conditions must be fulfilled for C:

- C must lie on the perpendicular bisector between B and D.
- κ_C must be the arithmetic mean of κ_B and κ_D

As more and more points are inserted, the polyline gets refined and the angles between segments approach π, and angles α and β approach 0. In the following formulae we use some simplifications whose errors converge to 0 when α and β approach 0. From the conditions we obtain:

$$\kappa_C = \frac{1}{2}(\kappa_B + \kappa_D). \tag{1}$$

We approximate the discrete curvature of X by

$$\kappa_X = 2\frac{\pi - \delta_X}{|\overrightarrow{XX_l}| + |\overrightarrow{XX_r}|} \tag{2}$$

as proposed in [15]. Then we substitute

$$\begin{aligned} \delta_B &= \pi - \alpha + \gamma & (3)\\ \delta_C &= \pi - 2\gamma & (4)\\ \delta_D &= \pi - \beta + \gamma. & (5) \end{aligned}$$

Since $|\overrightarrow{BD}| = 2$ we approximate $|\overrightarrow{BC}|$ and $|\overrightarrow{CD}|$ with 1, because when α and β approach 0, γ also approaches 0. Solving the resulting formula for γ finally yields

$$\gamma = \frac{\beta(|\overrightarrow{BA}| + 1) + \alpha(|\overrightarrow{DE}| + 1)}{2|\overrightarrow{BA}||\overrightarrow{DE}| + 3(|\overrightarrow{BA}| + |\overrightarrow{DE}|) + 4}. \tag{6}$$

Point C is now inserted on the perpendicular bisector between B and D in distance $\tan\gamma$ from g.

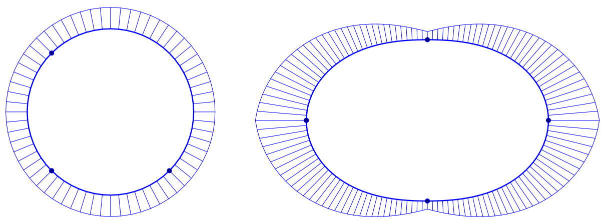
Figure 3: PCC configurations. The closed PCC through three control points always becomes a circle. Four control points in an 'elliptical' configuration do not lead to an ellipse, but to an ellipse-like ovoid with piecewise linear curvature.

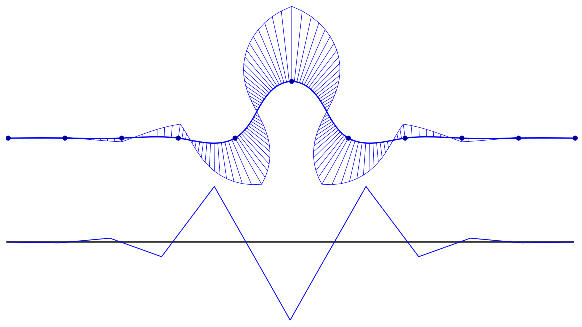
Figure 4: Damping of PCC curves. Top: The control points of the PCC are uniformly distributed, and the middle one is moved upwards by the same distance. Bottom: Curvature of the PCC.

When inserting a point next to an endpoint the problem is that a neighbor is missing on one side, i.e., A with α, or E with β. We need to specify an additional constraint to obtain a unique solution. We can enforce either a tangent or a curvature constraint, which are presented in Section 4. For endpoints the usual default is a curvature constraint that enforces zero curvature.

In the case of evenly spaced points this formula is identical to the discrete fairing approach proposed by [2]. Their *discrete clothoid spline* (DCS) is an evenly spaced polyline with the condition that the curvature at each point is the average of its neighbor curvatures:

$$\kappa_i = (\kappa_{i-1} + \kappa_{i+1})/2\,.$$

Thus, the curvature varies linearly, and the resulting curve must be a clothoid. Note that κ_i is the discrete curvature, i.e., the inverse radius of the circle through three points p_{i-1}, p_i, p_{i+1}. It can be computed using the inverse of the well-known triangle circumcircle formula $\kappa = 1/r = 4A/abc$ of a triangle with edge lengths a, b, c and (signed) area A.

3.4. Properties

PCCs have some nice properties for designers. The closed PCC through three points always is a circle (see

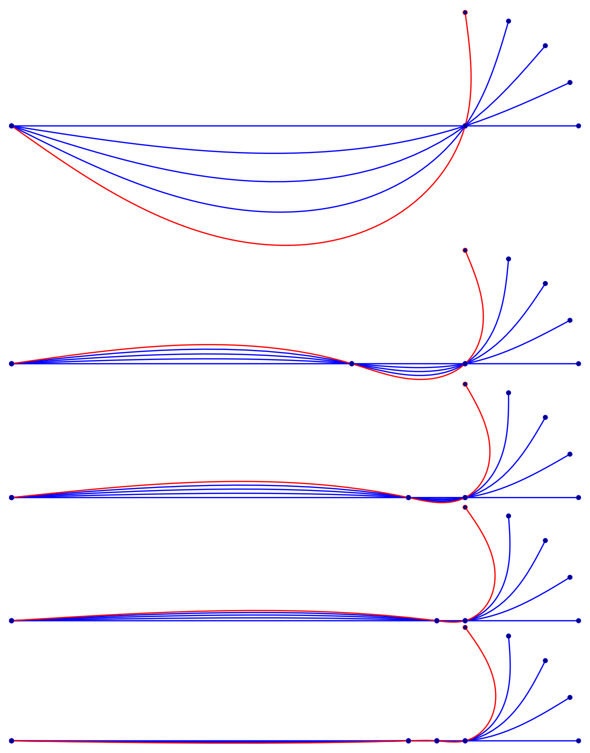

Figure 5: Damping effect of closely spaced control points.

Fig. 3); adding further control points is required only to define the deviation from a circle shape.

The position of a control point influences the PCC globally, meaning that every part of the PCC changes when moving a single control point. The influence of the control point, though, is heavily damped, as shown in Fig. 4. The damping effect can also be amplified by placing control points very closely together; moving a point on one side then has little effect on the curve on the other (Fig. 5). This is similar to a tangent constraint.

Inserting additional control points on a PCC neither changes the curve nor its curvature. This is extremely helpful in order to add fine detail, since a few close points can be inserted to 'nail down' some part of the curve by exploiting the damping effect. One consequence of the piecewise linear curvature behaviour is that curvature maxima *always* lie in the control points (in contrast to splines!). This makes PCCs much easier to understand and control by designers.

4. Constraints on tangents and curvature

To speed up the design process, tangent or curvature constraints can be defined for each control point:

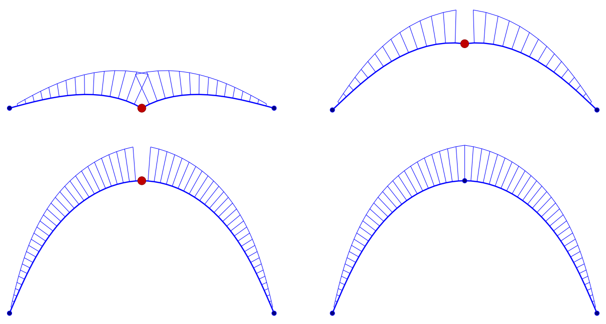

Figure 6: Trading shape against curvature. With three points in a row, a curvature constraint (marked in red) leads to a corner. It can be alleviated by moving the control point, changing shape until eventually the designer chooses to release the curvature constraint as it is sufficiently met. The curvature plot of all four curves is identical.

- A tangent constraint defines the tangent of an adjacent clothoid segment.
- A curvature constraint defines the curvature of an adjacent clothoid segment.

The constraints can be the same or different on both sides, e.g., different tangents produce a corner (G^0). It is also possible to define a tangent constraint on one side and a curvature constraint on the other, to maintain G^1. Note, however, that prescribing the same curvature on both sides will in general still lead to a corner (G^0). This mode is still valuable since the designer can move the control point to make the tangents gradually more collinear, to trade shape against curvature (see Fig. 6).

4.1. *Tangent constraints*

Formula (6) can be easily modified to account for a tangent constraint (c.f. Fig. 2): B is the point with the tangent constraint and neighbour A is discarded for the computation. Instead, angle α is set to the angle between the prescribed tangent and the line g through B and D. Length $|\overrightarrow{BA}|$ can be set to zero because for the computation we consider a point on the tangent instead of A, and the tangent condition has more influence the closer the point is to B.

4.2. *Curvature constraints*

Let again B be the point with the curvature constraint. δ_B and κ_B are not relevant now, but we can imagine a virtual segment $\overrightarrow{BA}$ and choose it in such a way that κ_B has the desired curvature value. If $|\overrightarrow{BA}| = 1$, then

$$\kappa_B \approx \frac{1}{\frac{1}{2}\sec\frac{\delta_B}{2}} = 2\cos\frac{\delta_B}{2} = 2\sin\left(\frac{\pi}{2} - \frac{\delta_B}{2}\right) \approx \pi - \delta_B \tag{7}$$

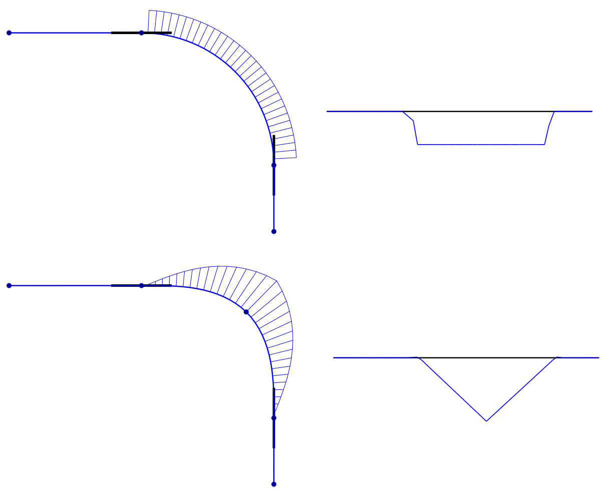

Figure 7: Restoring G^2 continuity. Top: A 90° configuration with straight sides and a rounded corner. The straightness of the sides is ensured with tangent constraints. The constraints permit continuous curvature, as can be seen in the curvature plot. Note: Because only the discrete curvature is measured, the spot with the curvature jump deviates from a vertical line. Bottom: An additional inserted point, moved to the right spot restores continuous curvature.

for small values of $\pi - \delta_B$. From this follows:

$$\gamma = \frac{2\beta + \kappa_B\,(|\overrightarrow{DE}| + 1)}{4\,|\overrightarrow{DE}| + 6} \tag{8}$$

At last, when we only have two endpoints and insert a point between them, on both sides a neighbor is missing. Taking $|\overrightarrow{DE}| = 1$ and $\kappa_D = \pi - \delta_D$ then leads to

$$\gamma = \frac{1}{4}(\kappa_B + \kappa_D)\ . \tag{9}$$

4.3. *Restoring G^2 continuity for collinear tangents*

A control point P with collinear tangents still achieves only G^1 continuity since in general, the curvatures do not match anymore. To restore G^2, another control point can be inserted on one side of P and moved such that the curvatures at P match on both sides; its position is not unique, so there is some design freedom.

Especially useful is the case of G^2 continuous rounded corners (see Fig. 7). First, control points are set in the positions for the start and end of the rounding. Then the tangents for the rounding are specified. At last, an additional control point is inserted in such a position that the joints at both control points simultaneously become G^2. The position of this point is unique and can, thus, be computed automatically, e.g., iteratively.

4.4. *Direct curvature control*

The properties presented so far suggest a very straightforward workflow for curve reconstruction with PCCs: Since control points are interpolated, and inserting a point does not change the curve, the designer can simply keep adding points along the desired contour. Curvature, however, is a very sensitive measure, and the resulting profile is likely to look like in Fig. 8 (top); obtaining a good curvature profile is a fiddling task. We now present a method where the designer can *directly edit the curvature profile* to obtain very rapidly the profile shown in the bottom.

The idea is to simply move the control point perpendicular to the line through its neighbours. Consider three points A, B, C and a desired target curvature κ_t for B. We use a simple linear estimate for the new position of B. Let κ_B be the discrete curvature for B, x the perpendicular projection of B to $\overrightarrow{AC}$, and d the distance between B and x. When x is positioned more towards the center of A and C, then the curvature does vary less when B is moved perpendicular to $\overrightarrow{AC}$. So we calculate a scalar factor λ for the relative position of x with $\lambda = 2\,\min(\,\|A - x\|, \|C - x\|\,)\ /\ \|C - A\|$.

The estimate for the needed offset distance is then $d_o = \lambda \cdot (\kappa_t - \kappa_B) \cdot \|C - A\|^2 \cdot c$ with the constant $c = \frac{1}{11}$ that was determined by experiment. Point B is moved by d_o along the perpendicular to line $\overrightarrow{AC}$. Then the whole PCC is recomputed, and the process is repeated until the desired accuracy is achieved, or further movement does not bring the curvature closer to the target curvature anymore, e.g., in case a high curvature was desired but the neighbours are too far apart.

5. Curvature blending and clothoid splines

5.1. *Eliminating curvature spikes*

The curvature of a PCC is piecewise linear but not bounded; there can be arbitrarily steep spikes. One possibility to get rid of a curvature spike in a control point P is to first insert two new control points close to P on either side, which changes neither the curve nor its curvature. Then removing P does change the curve, but only slightly; and the curvature-spike is cut off.

5.2. *Clothoid splines: Approaching the limit*

Our method for iterating this scheme is inspired by Chaikin's corner cutting method for obtaining quadratic B-Spline curves [16]. The method from the previous Section 5.1 can be repeated arbitrarily often.

For closed PCC curves, new control points are inserted on each clothoid segment at $\frac{1}{4}$ and $\frac{3}{4}$ of the arc

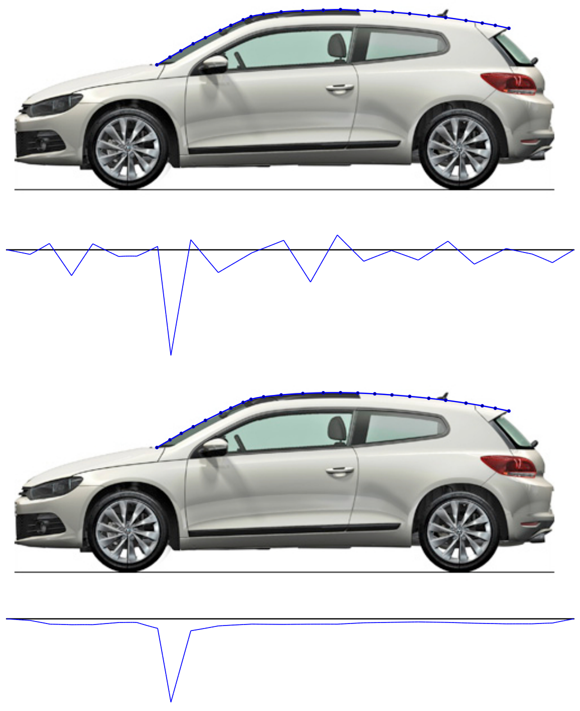

Figure 8: Manual curvature editing. Top: Control points of a PCC are placed in order to resemble a shape from a picture. Additional curvature plot is given. Bottom: A new PCC, obtained from manual editing of curvature values. Curvature plot is also given.

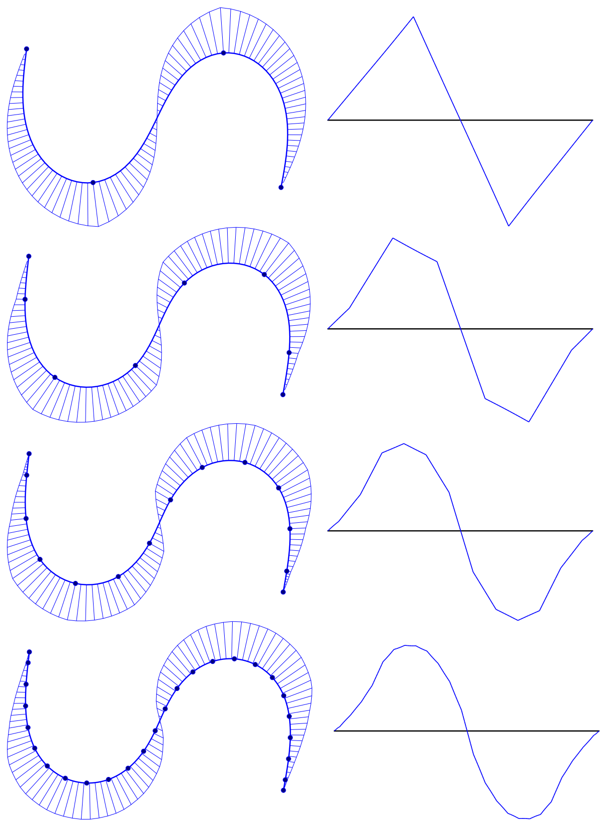

Figure 9: Construction of a clothoid spline. Repeated insertion of new points and removal of old points.

length. Then the old control points are removed and the PCC is recalculated. This leads to a PCC with twice the number of control points. This process is repeated for a certain number of iterations. The control points of the last step are taken as the points of a "*clothoid spline*".

For open curves, the process is nearly the same. However, the start point and the end point are never removed. In the first and in the last segment, only a single new control point is inserted at position $\frac{1}{2}$ in the first iteration; otherwise the refined points would accumulate at the start and at the end. Fig. 9 illustrates this construction of a clothoid spline with G^3 continuity.

6. A physical interpretation of clothoids

A vast number of different curve representations has been proposed in CAGD over the last 60 years. With any new representation the question must be answered how suitable it is for the targeted purpose. For high-end curve design, it should make designing aesthetic curves as simple as possible. Capturing the notion of aesthetics is difficult, however, which is witnessed by the lack of a common mathematical definition of *class A quality*.

We have approached the problem from a different side. The reader is encouraged to try a very simple physical experiment, namely designing a 'smooth' curve with fixed-length segments, e.g., *Kapla* bricks (Fig. 11). It will turn out sooner or later that instead of looking 'along' the curve, the obvious thing to do is focusing on the gap angles between the bricks, and to balance them; an uneven gap distribution is deemed ugly *by anybody*.

Since the gaps are small, attaching normal rods to the segments shows problems more clearly. Equalizing the gap angles can then be seen as equalizing the distances between the end points of consecutive rods. Physically, this could be accomplished by attaching springs on the end points; and for symmetry reasons, two such rods should be attached, one on eiter side of the segment.

This leads to a variational energy minimization problem: A *spring element* is made out of five points A, B, C, D, E connected by line segments with two normal rods that are connected by six springs with rest length zero (Fig. 12). The position of $C(t)$ is optimized on the perpendicular bisector between B and D, the other points are fixed. Finding t for which the spring

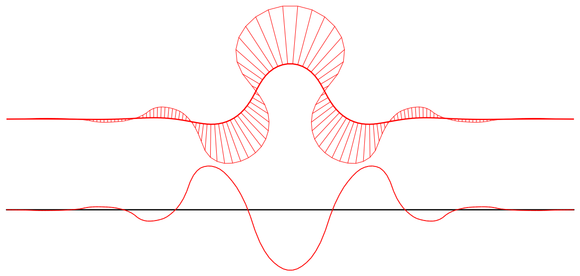

Figure 10: PCC spline. The control points are the same as in Fig. 4. There is hardly any difference between the curves, but a great difference between the curvatures: The PCC spline is G^3.

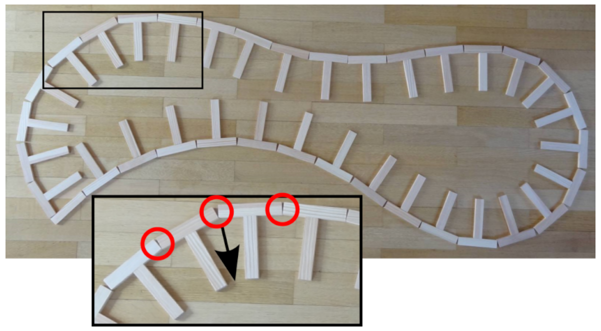

Figure 11: Design experiment: Aesthetic improvement of a curve. In this situation, virtually anybody will move the bricks down to equalize the gap angles. Normal rods show the angle distribution more clearly.

energy $E_s(t)$ is minimal is a convex univariate problem.

The spring energy is proportional to the square of the distance, but the derivative is not linear in t. However, since the energy is convex, the optimum is easily reached by a few Newton iterations. Mimicking the manual method, we turned this into a curve optimization scheme by keeping some of the input points fixed while optimizing the free points. The implementation revealed a surprise: It converged rapidly to a curve with a linear curvature profile between the fixed points – obviously a sequence of clothoids! So the spring elements provide a clear, intuitive physical interpretation of curve design with clothoids that can be easily grasped by artists and designers.

6.1. *Analytical solution for infinite rod lengths*

The length of the normal rods is a free parameter. We made the observation that the optimization yields better results with longer rods, which led us to examining the case of infinite rod length. As shown in the following, this yields the same formula as in Sec. 3.

The longer the rods, the smaller are the segments in comparison. So instead of infinite rod length, we consider rods of fixed (unit) length and zero length segments. The end points of the rods lie on the unit circle; let σ be the angle between two rods, then the force $f(\sigma)$ is the squared distance of their endpoints:

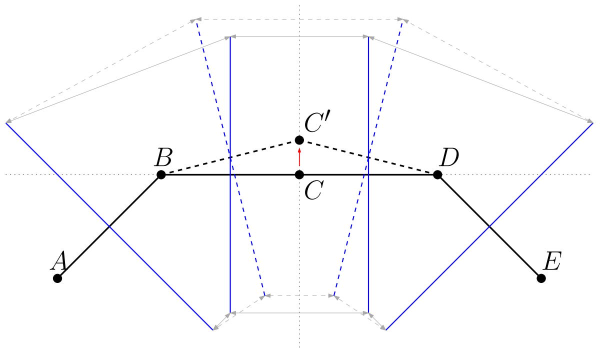

Figure 12: A spring element with fife points, four segments, eight normal rods and six springs connecting the end points of the rods. The position of the center point is spring-optimized along the perpendicular bisector of the line segment connecting its neighbours.

$$f(\sigma) = 2\,\mathrm{chord}^2\sigma = 8\sin^2\frac{\sigma}{2}. \tag{10}$$

For small values of σ the sine function is approximately the identity. This yields a simplified version f_s of the force:

$$f_s(\sigma) = 2\,\sigma^2. \tag{11}$$

Figs. 2 and 12 show that the angle between the 1st and the 2nd rod is $\alpha - \gamma$, between the 2nd and the 3rd it is 2γ, and between the 3rd and the 4th it is $\beta - \gamma$. For the spring element with points A, B, C, D, E we have a total (simplified) force f_t of

$$f_t = f_s(\alpha-\gamma) + f_s(2\gamma) + f_s(\beta-\gamma) \tag{12}$$

In order to find the value of γ with the minimal force in the spring elements we differentiate f_t, set it to zero, and solve for γ:

$$f_t' = -4\alpha + 24\gamma - 4\beta \tag{13}$$

$$f_t' = 0 \quad \Rightarrow \quad \gamma = \frac{1}{6}\alpha + \frac{1}{6}\beta \tag{14}$$

This is indeed the same formula as in Eq. (6) when $|\overrightarrow{BA}|$ and $|\overrightarrow{DE}|$ are equal to one. So with evenly spaced points, the two approaches yield the same results.

7. Applications

7.1. *Interactive shape design with PCCs*

An important property for a curve representation is how effective and accurate designers can work with them. The proposed PCCs have some nice properties for the design process:

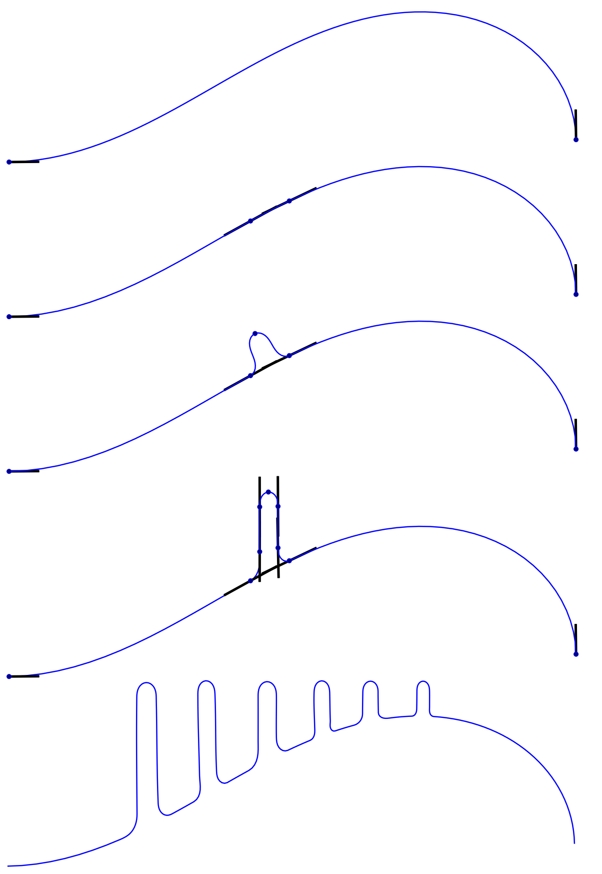

Figure 13: Inserting U-shapes on a large smooth curve segement without affecting the rest of the curve. Top-to-bottom: Original curve. Insertion of two control points with fixed tangents, then local modification on isolated part. The curvature of the seemably fixed-radius blend and in the tips can be controlled by editing the curvature plot.

- Reasonable approximations can be computed quite fast also for larger numbers of control points.
- Insertion of new control points neither changes the curve nor does it affect other control points.
- Redundant control points (with the same curvature slope on both sides) can be removed without affecting the curve.
- Tangents or curvature constraints can be specified for all control points.

7.2. *Curve design rules for PCCs*

We demonstrate the typical design flow with a particularly challenging example, the insertion of details on a larger curve with very smooth curvature (Fig. 13):

1. The designer draws a few spacious control points to define a shape with smoothly flowing curvature.

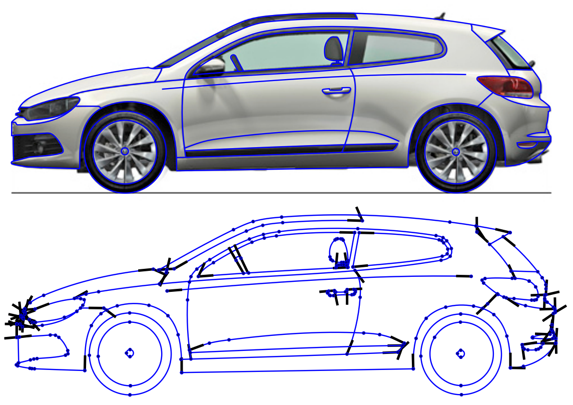

Figure 14: Shapes of a car captured with PCCs, c.f. Fig. 1. Zoom in to examine typical constraint configurations (vector graphics).

2. Detail zones are isolated by inserting pairs of control points with tangent constraints.
3. Editing the zones does not affect the rest of the curve, but reduces continuity to C^1 in these points.
4. Finally, G^2 continuity can be restored with the technique explained in Sec. 4.3 (c.f. Fig. 7).

We claim that PCCs are especially well suited for high-quality curve design. This is supported by empirical evidence since the characteristic curves of many existing high-quality shapes can be captured using only very few control points, as illustrated by Figs. 1 and 14. And even more important is that the control points can be obtained using a clear coarse-to-fine recipe.

7.3. *Computing a PCC for a sampled input curve*

Our second use case is fitting a PCC to a given densely sampled input curve (polyline). The automatic algorithm presented in this section uses dynamic programming to partition the curvature profile of the input curve in order to find ideal control point positions. The idea to use dynamic programming was borrowed from [8]; the algorithm chooses points from the input curve which are directly used as PCC control points. The PCC shall follow the input curve very closely and should have a sparse adaptive control polygon.

To penalize the number of control points, each PCC segment is associated with a cost; and to enforce the similarity of both curvature profiles, the deviation of the PCC curvature from the input curvature is penalized as well. Note that for demonstration purposes we do *not* even include an explicit cost for the geometric distance between the curves. In all of our examples it was sufficient that (a) the PCC control points are drawn from the

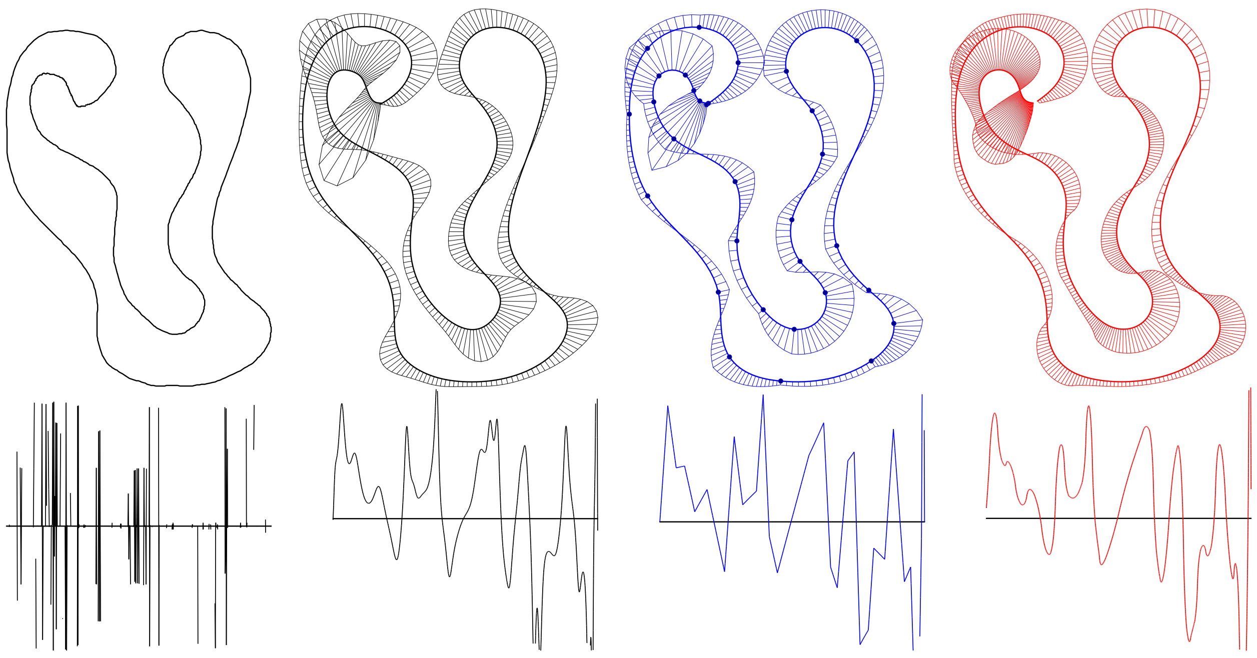

Figure 15: Fitting a PCC to input data. From left to right: 1) Original curve, 2) Smoothed curve, 3) PCC fitted to the smoothed curve using dynamic programming as described in Section 7.3. Big dots denote control points, small dots are points inserted by the PCC algorithm. 4) Clothoid spline with smooth curvature profile as described in Section 5.2. The input curve is taken from [10].

input polygon, and (b) the curvature profiles are very similar; note that 2D curves are uniquely defined by their curvature (up to rigid transformations).

In the following, let a and b denote the arclength parameters of two points $P(a)$, $P(b)$ of the input curve (a discrete polyline), and $P(a,b)$ is the curve segment between a and b. We define a cost matrix M where each entry $M(a,b)$ is the minimal cost of placing control points at a and b, and anywhere inside $P(a,b)$. It is defined recursively in the typical dynamic programming fashion:

$$M(a,b) = \min_{a<k<b} \{E_d(a,b) + \lambda E_i(a,b),\ M(a,k) + M(k,b)\} \tag{15}$$

The first term is the cost when no other control point is used between a and b, and the second term applies when splitting up the curve by inserting another control point at any possible parameter $a < k < b$ of a polyline point yields (recursively) even smaller cost. Concerning the first term, E_d penalizes the deviation of the PCC curvature from the input curvature (discrete integral of absolute differences). We define it as

$$E_d(a,b) = \sum_{c\,\in\,P(a,b)} f(c)\, \|\kappa(a) - \kappa(c) + l_s(c-a)\|^{e_d} \tag{16}$$

where $l_s = \frac{\kappa(b)-\kappa(a)}{b-a}$ is the slope of the curvature function in this interval and $f(c) = \frac{1}{2}(c_+ - c_-)$ is the local strip width with respect to the previous and next points c_-, c_+ of the polyline. E_i penalizes short segments:

$$E_i(a,b) = \left(\frac{d}{b-a}\right)^{e_i} \tag{17}$$

where d is the overall length of the polyline. Parameters λ, e_d, e_i can be tuned by the user in order to specify the different trade-offs.

The matrix M is computed in a bottom-up fashion, eventually yielding $M(0,d)$ as the cost of the best possible selection of polyline points as PCC control points to match the input curve. An example of the fitting process is shown in Fig. 15. The input curve is smoothed before the fitting starts in order to achieve better results.

8. Comparison with B-spline curves

As mentioned in the Introduction, a fair comparison from the design point of view can take into account only spline types without non-visual parameters (knots, weights). Comparing, for example, PCCs to uniform cubic B-splines, PCCs have the obvious advantages that control points are interpolated instead of approximated, that point insertion does not change the curve, and that circular arcs can be reproduced.

A more serious issue, though, is that despite the variation diminishing property and their C^2 continuity, the curvature even of higher-degree polynomial splines is unbounded and hard to control in practice. Uneven control point spacing is likely to result in unexpected curvature maxima, and even spikes. With PCCs, the curvature can clearly be (locally) maximal only in the control points, which is a clear and practical placement rule.

We claim that for all practical design tasks, PCCs are in fact ***superior*** to splines: While it is easy for PCCs to realize the 'extra pull' of spline curves, it is impossible for splines to reproduce the perfectly clean curvature profile of PCCs. For controlling curvature (class-A design), PCCs may even be seen as the *optimal* curve representation because any desired curvature profile can efficiently be approximated using linear segments.

To quantify these claims, Fig. 16 shows that a PCC with three times as many control points can nicely reproduce the curvature artifacts of a spline; and that by doubling the number of spline control points, the artefacts are only damped but their frequency is doubled as well. Note that curvature artifacts are unavoidable with splines, as shown by Augsdoerfer et al. [17] who treat the subject in considerable depth.

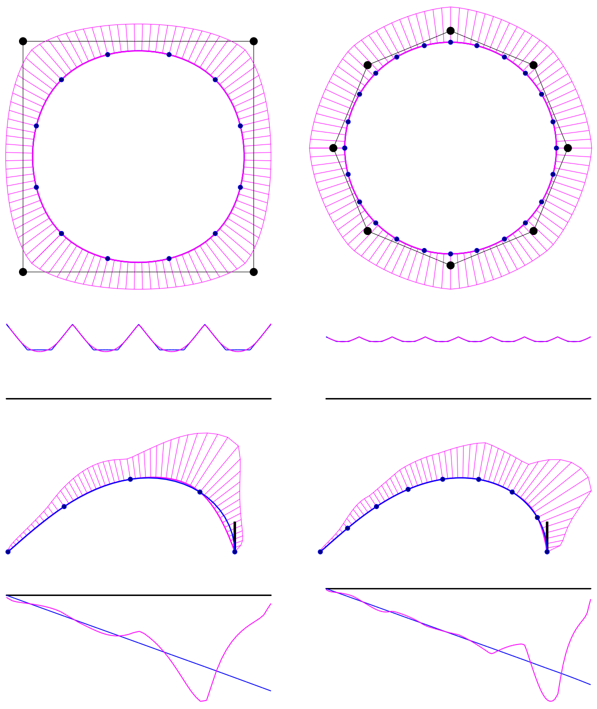
Figure 16: Curvature artifacts of spline curves. Top: A uniform cubic B-spline (magenta) with four control points can be approximated by a PCC with twelve control points - including the curvature artifacts (left). Completely avoiding the artifacts is not possible (right). Bottom: A single clothoid segment is approximated by a spline. Despite regular spacing the resulting curvature is simply inacceptable.

9. Conclusion and Future Work

We have presented in this paper a framework for the design of high-quality curves. To summarize, piecewise clothoid curves (PCCs) have the following key advantages over splines:

- Direct editing: The control polygon is interpolated, rather than just approximated.
- Insertion invariance: Inserting control points does not change the curve.
- Curvature extrema: The curvature is (locally) maximal or minimal only in the control points.
- Predictability: Control points are needed only in order to deviate from linear (circular) curvature.

Intuitive methods were presented for constraining the PCC to specific tangents or curvature values, and for automatically improving the piecewise linear curvature profile, eventually resulting in a 'clothoid spline' with smooth curvature (G^3). It was shown that a given input curve can be efficiently converted to a PCC, and controlling curvature is possible even to the point of directly editing the curvature profile.

For designers, this can be intuitively summarized as *"PCCs are class-A by default"*; a relaxed type of curve for which introducing tension requires specific effort.

An important area for future research is a practical method for designing high-quality space curves, where not only curvature but also torsion must be controlled; for a 3D generalization of clothoids, Ben-Haim et al.[14] propose varying also the torsion linearly. The main goal, however, will be to find also a surface representation that is acknowledged by practitioners as being 'class-A by default'.

Acknowledgements

The authors gratefully acknowledge the generous funding and support from the Joint PhD programme of TU Graz and NTU Singapore (Prof. Alexei Sourin). We would also like to thank Volkswagen AG, especially our collaborators Dr. Karin Spors and Dr. Gerd Podehl, for the great opportunity to gain so much insight into the practice of high-quality surfacing, and also for all the inspiring challenges - and for the VW Scirocco picture.

[1] M. A. Heald, Rational approximations for the fresnel integrals, Mathematics of Computation 44 (170) (1985) pp. 459–461.

[2] R. Schneider, L. Kobbelt, Discrete fairing of curves and surfaces based on linear curvature distribution, in: P.-J. Laurent, P. Sablonniere, L. L. Schumaker (Eds.), Curve and Surface Design, Saint-Malo 1999, Innovations in Applied Mathematics, Vanderbilt University Press, Saint-Malo, France, 2000, pp. 371–380.

[3] A. Nutbourne, P. McLellan, R. Kensit, Curvature profiles for plane curves, Computer-Aided Design 4 (4) (1972) 176–184.

[4] T. Pal, A. Nutbourne, Two-dimensional curve synthesis using linear curvature elements, Computer-Aided Design 9 (2) (1977) 121–134.

[5] A. Schechter, Synthesis of 2d curves by blending piecewise linear curvature profiles, Computer-aided design 10 (1) (1978) 8–18.

[6] J. M. Ali, R. Tookey, J. Ball, A. Ball, The generalised cornu spiral and its application to span generation, Journal of Computational and Applied Mathematics 102 (1) (1999) 37–47.

[7] R. L. Levien, C. Adviser-Sequin, From spiral to spline: Optimal techniques in interactive curve design, University of California at Berkeley, 2009.

[8] J. McCrae, K. Singh, Sketch-based interfaces and modeling (sbim): Sketching piecewise clothoid curves, Comput. Graph. 33 (4) (2009) 452–461.

[9] J. McCrae, K. Singh, Sketch-based path design, Canadian Information Processing Society, 2009.

[10] I. Baran, J. Lehtinen, J. Popovic, Sketching clothoid splines using shortest paths, Comput. Graph. Forum 29 (2) (2010) 655–664.

[11] J. McCrae, K. Singh, Neatening sketched strokes using piecewise french curves, in: Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling, SBIM ’11, ACM, New York, NY, USA, 2011, pp. 141–148.

[12] Y. Thiel, K. Singh, R. Balakrishnan, Elasticurves: exploiting stroke dynamics and inertia for the real-time neatening of sketched 2d curves, in: Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM, 2011, pp. 383–392.

[13] D. Walton, D. Meek, A controlled clothoid spline, Computers & Graphics 29 (3) (2005) 353–363.

[14] D. Ben-Haim, G. Harary, A. Tal, Piecewise 3d euler spirals, in: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling, SPM ’10, ACM, New York, NY, USA, 2010, pp. 201–206.

[15] A. G. Belyaev, A note on invariant three-point curvature approximations (singularity theory and differential equations), RIMS Kokyuroku 1111 (1999) 157–164.

[16] G. M. Chaikin, An algorithm for high-speed curve generation, Computer Graphics and Image Processing 3 (4) (1974) 346 – 349.

[17] U. Augsdoerfer, N. Dodgson, M. Sabin, Artifact analysis on b-splines, box-splines and other surfaces defined by quadrilateral polyhedra, Computer Aided Geometric Design 28 (3) (2011) 177 – 197.