Planar Curves based on Explicit Bézier Curvature Functions

Norimasa Yoshida[1] and Takafumi Saito[2]

[1]Nihon University, norimasa@acm.org
[2]Tokyo University of Agriculture and Technology, txsaito@cc.tuat.ac.jp

Corresponding author: Norimasa Yoshida, norimasa@acm.org

Abstract. For controlling the curvature variation with given G^1 or G^2 Hermite interpolation conditions, we propose intrinsically defined planar curves based on explicit Bézier curvature functions. In the proposed curve, the curvature variation is specified by an explicit Bézier curve. To perform G^1 or G^2 Hermite interpolation, some of control curvatures of the explicit Bézier curve are modified to fit the given conditions. We have implemented the method in C++ and confirmed that curves can be generated in fully interactively. We clarify how the viable regions for G^2 Hermite interpolation changes depending on the degree of explicit polynomial Bézier curves. Applications of the proposed curves include the design of aesthetic curves for aesthetic shape desing as well as 2D illustrations.

Keywords: Aesthetic Curves, Curvature Functions, Explicit Bézier Curves.
DOI: https://doi.org/10.14733/cadaps.2020.77-87

1 INTRODUCTION

In the design of highly aesthetic surfaces, such as the exterior of automobiles, designers design surfaces by evaluating the reflected images such as reflection lines. To design aesthetic surfaces, it is also important to use aesthetic curves whose curvature varies monotonically within a specified region since the curvature variation dominates the distortion of reflected images. Recently, Yan et al. have pointed out that curvature behavior is also important for illustrations [24].

For controlling curvature variation as much as possible with given G^1 or G^2 Hermite interpolation conditions, we propose intrinsically defined planar curves based on explicit Bézier curvature functions. We represent the curvature function in terms of explicit polynomial or rational Bézier curves and the curve is generated by computing integrals. Our approach is most closely related to [21], but our approach is different and is more efficient than [21] because we show that arc length can be determined by G^1 or G^2 Hermite interpolation conditions. Moreover, we use explicit Bézier curvature functions and clarify some characteristics including experimental viable G^2 Hermite regions. Tangential angle parameterization curves [16, 23], where the curvature radius function is represented by cubic polynomials, is also related. No numerical integration is required in their approach, but inflection points cannot be represented. In our approach, although numerical integration of an explicit polynomial Bézier curvature function is required once to generate a curve, we confirmed that the curves can be generated fully interactively.

Log-aesthetic curves [9,10,25] are high quality curves whose curvature functions are relatively simple functions with monotonically varying curvature. However, when performing G^2 Hermite interpolation, they cannot match a wide variety of G^2 Hermite interpolation conditions. This work can be considered as a generalization of the curvature function of log-aesthetic curves in terms of explicit polynomial or rational Bézier curves.

The contributions of this work is the following:

(1) We present a framework for generating a curve segment based on an explicit polynomial or rational Bézier curvature function with given G^1 or G^2 Hermite interpolation conditions.
To our knowledge, no other approaches have used explicit Bézier curvature functions. As shown in Eqn. (7), the integration to compute the tangential angle can be simply computed using the average of control curvatures.

(2) For given G^1 Hermite interpolation conditions, our method can generate curves with various curvature variations.
As shown in Fig. 2 (c), (d) and (e)-(g), various kinds of curvature variation can be generated for the same G^1 Hermite interpolation conditions by modifying control curvatures that are not constrained. It is also possible to generate curves for the same G^2 Hermite interpolation conditions, if we use explicit Bézier curvature functions of degree higher than 3.

(3) We have implemented our curves in C++ and confirmed that curves can be generated in real time.
Curve segments can be generated fully interactively for explicit polynomial and rational Bézier curvature functions, if the degree is not too high.

(4) For explicit polynomial Bézier curvature functions and given G^1 Hermite interpolation conditions, we visualize the regions of curvatures at two endpoints where a curve segment with monotonically varying curvature (without an inflection point) can be generated.
To our knowledge, no other papers have visualized such regions for intrinsically defined curves based on curvature functions. This kind of visualization is important to see if the proposed curves can match a wide variety of G^2 Hermite interpolation condition.

This paper is organized as follows. Section 2 describes the related work for controlling curvature. In section 3, our proposed curves based on explicit polynomial or rational Bézier curvature functions are described. Section 4 describes G^1 or G^2 Hermite interpolation methods. Section 5 describes the methods for finding an inflection point and checking if the curvature is monotonically varying. Section 6 shows the examples of generated curves with G^1 and G^2 Hermite interpolation conditions. In Section 7, the proposed curves are compared with curves based on curvature radius functions [16, 23]. Finally, conclusions and future work are presented in Section 8.

2 RELATED WORK

This section reviews works on controlling curvature of curves that can be divided into works (a) on freeform curves and (b) on curves generated from curvature or curvature radius.

2.1 Controlling Curvature of Freeform Curves

Freeform curves, such as Bézier curves and NURBS curves, are widely used in many applications including CAD system. In freeform curves, we are able to know the curvature variation after the curve shape is completely determined by computing the derivatives. Sapidis and Frey presented a necessary and sufficient condition for monotonically varying curvature of quadratic polynomial Bézier curves [17]. Frey and Field shows the condition for curvature monotonicity of quadratic rational Bézier curves [5]. Dietz and Piper used precomputed tables to generate curves with monotonically varying curvature for cubic polynomial Bézier curves [2] and for cubic rational Bézier curves [3]. Wang et al. showed the sufficient condition for polynomial Bézier curves of degree n to be monotonically varying [21]. Farin proposed class A Bézier curves where both curvature and torsion are monotonically varying [4]. Yoshida et al. proposed a method for interactively controlling planar class A Bézier curves [26] and 3D class A Bézier curves [27]. They showed that typical class A Bézier curves get closer to a logarithmic spiral when the degree gets higher. Yan et al. proposed κ-Curves where local maximum curvature is always placed at control points[24]. They used quadratic Bézier curves for interpolation, but it is not trivial to extend their method for cubic or higher degree Bézier curves that include inflection points. κ-Curves are not G^2-continuous because they are used for illustration. In freeform curves, the curvature is computed by the norm of the cross product of first and second derivatives divided by the cubed norm of the first derivative. The difficulty of controlling curvature of freeform curves is due to the complexity of computing the curvature.

2.2 Curves based on Curvature

Nutbourne et al. [11] and Pal et al. [12, 13] proposed methods for generating curves by specifying curvature plots. In these works, the curvature is restricted to be (piecewise) linear with respect to arc length. Thus, the generated curves are composed of Clothoid segments. Ali et al. proposed a generalized Cornu spiral [1], which is a generalization of the Clothoid and logarithmic spirals. Our proposed curves can also be considered as a generalization of this work in terms of explicit rational Bézier curvature functions. Watanabe et al. proposed a method to generate planar curves based on cubic Bézier curves [21]. This work is closely related to our work, but our work is different in that we use explicit Bézier curves for curvature functions and the arc length is not used for an optimization parameter.

Log-aesthetic curves [9,10,25] are curves whose logarithmic curvature graphs are straight lines. Because of the constraint of simple curvature functions [28] of log-aesthetic curves, it is not possible to use a single log-aesthetic curve for wide a variety of G^2 Hermite interpolation conditions. For a given G^1 Hermite interpolation condition, we have changed the value of α to see how the curvatures at two endpoints change and confirmed that the curvatures change only slightly. This means that a single log-aesthetic curve segment is not good for G^2 Hermite interpolation. Miura et al. have used more than one log-aesthetic curve segment for G^2 Hermite interpolation [11]. However, it is not clear if the proposed curves can match a wide variety of G^2 Hermite interpolation conditions. There are several generalizations of log-aesthetic curves, such as generalized log-aesthetic curves by Gobithaasan et al. [6] and quadratic log-aesthetic curves by Yoshida et al. [30]. It is not clear if these curves can match a wide variety of G^2 Hermite interpolation conditions.

Wu et al. [23] have proposed curves whose radius of curvature is represented by a polynomial in terms of tangential angle. They showed that the integration for computing curve points is in closed form (no numerical integration is necessary) and G^1 Hermite interpolation can be simply performed by solving linear equations. Saito et al. [16] have also worked on the same curve and they provide more detail analysis of viable G^2 Hermite regions and non-polynomial radius curvature functions by using the principle of superposition. Although the curve points can be simply computed and G^1 Hermite interpolation is performed by solving a linear system, the curves cannot include an inflection point and a cusp may arise in a segment. In the proposed curves, inflection points can be representable.

3 CURVES BASED ON EXPLICIT BÉZIER CURVATURE FUNCTIONS

Let s_t be the arc length of a curve segment. Let n be the degree of an explicit Bézier curve and $\kappa_i (i = 0,1,\cdots,n)$ be the *control curvatures*. An explicit polynomial Bézier curvature function $\kappa(s)$ in terms of arc length s is defined by

$$\kappa(s) = s_t\, K(\tau) \;\; (s \in [0, s_t]), \quad K(\tau) = \sum_{i=0}^{n} B_i^n(\tau)\; \kappa_i \;\; (\tau \in [0,1]), \tag{1}$$

where $B_i^n(\tau)$ is the Bernstein polynomial and $\tau = \frac{s}{s_t}$. An explicit rational Bézier curvature function $\kappa_R(s)$ is given by

$$\kappa_R(s) = s_t\, K_R(\tau)\ \ (s \in [0, s_t]),\ \ \mathrm{K}_R(\tau) = \frac{\sum_{i=0}^{n} B_i^n(\tau)\ w_i \kappa_i}{\sum_{i=0}^{n} B_i^n(\tau)\ w_i}\ \ (\tau \in [0,1]). \tag{2}$$

where $w_i (i = 0,1,\cdots,n)$ are weights. We use $\kappa_G(s)$ to mean either $\kappa(s)$ or $\kappa_R(s)$. We also use $\mathrm{K}_G(\tau)$ to mean $K(\tau)$ or $\mathrm{K}_R(\tau)$. Arc length s and tangential angle θ are related by the following equation

$$\mathrm{d}\theta = \kappa(s)\mathrm{d}s. \tag{3}$$

Integrating both sides of Eqn. (3), we get

$$\theta\ (s) = \int_0^s \kappa_G(t)\mathrm{d}t. \tag{4}$$

As will be shown in the next section, the integration of Eqn. (4) can be computed in closed form if $\kappa_G(s)$ is an explicit polynomial Bézier curvature function. The curve position $\mathbf{P}(s)$ in the standard form, where $\mathbf{P}(s)$ is at the origin if $s = 0$ and its tangent vector is $[1\quad 0]^{\mathrm{T}}$, is computed by

$$\mathbf{P}(s) = \begin{bmatrix} \int_0^s \cos\theta\ (s)\mathrm{d}t \\ \int_0^s \sin\theta\ (s)\mathrm{d}t \end{bmatrix}. \tag{5}$$

The curve in general position can be obtained by performing an appropriate similarity transformation to the curve generated by Eqn. (5).

4 G^1 OR G^2 HERMITE INTERPOLATION METHOD

4.1 Computing the Arc Length

To draw a curve segment, we need to know the arc length of the curve segment. We show that the arc length of a curve segment can be computed from the tangential angle given in G^1 Hermite interpolation conditions. For explicit polynomial Bézier curvature functions, the arc length can be simply computed without using numerical integration. For explicit rational Bézier curvature functions, numerical integration is necessary.

Tangential angle θ and arc length s are related by Eqn. (4). Let θ_d and s_t be the change of tangential angle and the arc length of a curve segment, respectively. Putting Eqn. (1) into Eqn. (4) and replacing $\theta\ (s)$ and s with θ_d and s_t respectively, we get

$$\theta_d = \int_0^{s_t} \kappa(t)\mathrm{d}t \tag{6}$$

$$= s_t \int_0^1 K(t)\mathrm{d}t \tag{7}$$

$$= s_t \sum_{i=0}^{n} k_i/(n+1)\,. \tag{8}$$

Note that the modification from Eqn. (7) to Eqn. (8) is based on the characteristics of explicit polynomial Bézier curves [19]. Thus, once all the control curvatures and the change of tangential angle are known, arc length s_t is computed by

$$s_t = \frac{\theta_d(n+1)}{\sum_{i=0}^{n} k_i}. \tag{9}$$

The arc length s_t is simply θ_d divided by the average of all the control curvatures.

For explicit rational Bézier curvature functions, the arc length s_t is computed by

$$s_t = \frac{\theta_d}{\int_0^1 K(\tau)\mathrm{d}\tau}. \tag{10}$$

To compute the denominator of the right-hand side of Eqn. (10), numerical integration is necessary because $K(\tau)$ is a rational function.

4.2 G^1 or G^2 Interpolation Method

We consider G^1 or G^2 Interpolation in the standard form. In G^1 Hermite Interpolation in the standard form, we are given two points $\mathbf{P}_s, \mathbf{P}_e$ and their tangents $\mathbf{t}_s, \mathbf{t}_e$. θ_d is the change of tangential angle that is the angle formed by $\mathbf{t}_s$ and $\mathbf{t}_e$. In the standard form, $\mathbf{P}_s$ is at the origin, $\mathbf{t}_s$ is directed toward the positive x-axis, and the y coordinate of $\mathbf{P}_e$ is greater than 0. See Fig. 1. In this section, without of loss of generality, we assume that $\mathbf{P}_s, \mathbf{P}_e$, $\mathbf{t}_s$ and $\mathbf{t}_e$ are given in the standard form. By an appropriate translation and a rotation, any $\mathbf{P}_s, \mathbf{P}_e$, $\mathbf{t}_s$ and $\mathbf{t}_e$ can be transformed into the standard form. If the y coordinate of $\mathbf{P}_e$ is smaller than 0, we perform a mirroring operation. If we are given all the control curvatures κ_i, the arc length of the curve segment can be computed by Eqn. (8) for polynomial curvature functions or by Eqn. (10) for rational curvature functions.

Since Eq. (5) is in standard form, the positional constraint $\mathbf{P}(0) = \mathbf{P}_s$ and the tangential constraint $\left.\frac{\mathrm{d}\mathbf{P}(s)}{\mathrm{d}s}\right|_{s=0} = \mathbf{t}_s$ at the start point $s = 0$ are automatically satisfied. The endpoint tangential condition $\left.\frac{\mathrm{d}\mathbf{P}(s)}{\mathrm{d}s}\right|_{s=s_t} = \mathbf{t}_e$ is satisfied by using arc length s_t computed using Eqn. (8) or (10). The remaining condition for satisfying G^1 Hermite interpolation condition is $\mathbf{P}(s_t)$ to be equal to $\mathbf{P}_e$. This condition is satisfied by an optimization using two of κ_i, typically κ_0 and κ_n, as optimization parameters. Other control curvatures are either user-specified or interpolated using κ_0 and κ_n. If we use a linear interpolation to compute $\kappa_1, \cdots, \kappa_{n-1}$, the generated curve will be the Clothoid curve.

In G^2 Hermite interpolation, curvature κ_s, κ_e of start and end points are specified in addition to G^1 Hermite interpolation conditions. G^2 Hermite interpolation is performed in a similar manner by setting $\kappa_0 = \kappa_s$ and $\kappa_n = \kappa_e$. κ_1 and κ_{n-1} are typically used as optimization parameters and other control curvatures are either user-specified or interpolated using κ_1 and κ_{n-1}. Note that any G^2 (and G^1) Hermite interpolation condition can be converted to the standard form shown in Fig. 1 by an appropriate similarly transformation. If the control points are uniformly scaled by a factor σ, the curvature at both endpoints must be scaled by a factor $\frac{1}{\sigma}$.

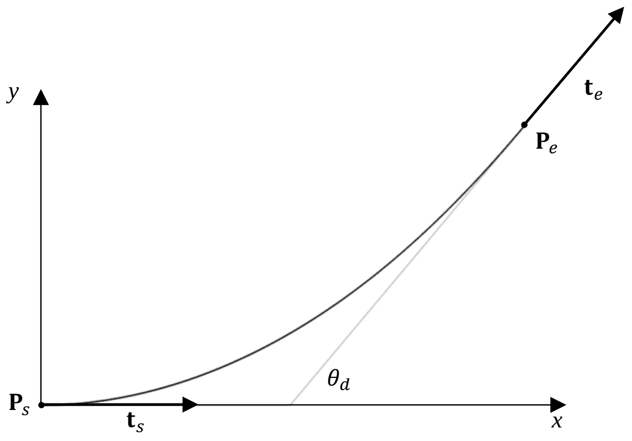


Figure 1: G^1 Hermite Interpolation in the standard form.

5 INFLECTION POINTS AND CURVATURE MONOTONICITY

The curves generated by the proposed method may include an inflection point and the curvature may not to be monotonically varying. The existence of an inflection point can be checked by applying Bézier clipping [18] to $\kappa_G(\tau)$ within $\tau \in [0,1]$ for the general case. If the degree of $\kappa(\tau)$ is low, we can directly compute τ such that $\kappa(\tau) = 0$. For rational Bézier curvature functions with positive weights, we only need to check if the numerator becomes zero within $\tau \in [0,1]$. If $\kappa_G(\tau)$ becomes 0 within $\tau \in [0,1]$, the curve includes an inflection point within the curve segment.

The curvature of the curve is monotonically varying if the first derivative of $\kappa_G(\tau)$ does not change its sign within $\tau \in [0,1]$. The monotonicity of curvature can be similarly checked by applying Bézier clipping to see if there is a sign change within $\tau \in [0,1]$. For low degree $\kappa(\tau)$, we can directly compute τ such that $\kappa'(\tau) = 0$. For explicit rational Bézier curvature functions with positive weights of degree n, we need check if the first derivative becomes zero within $\tau \in [0,1]$. Suppose that the rational function $\kappa_R(\tau) = f(\tau)/g(\tau)$. The first derivative is

$$\kappa_R'(\tau) = \frac{f'(\tau)g(\tau) - f(\tau)g'(\tau)}{g(\tau)}. \tag{11}$$

Although the degree of the numerator seems to be $2n - 1$, it is known that the actual degree is $2n - 2$ [15, 20]. Therefore, for an explicit rational Bézier curvature function of degree n, the monotonicity of curvature can be checked by computing the zeros of a polynomial of degree $2n - 2$, such as using Bézier clipping. If there is a sign change, the curvature of the curves is not monotonically varying.

In the case where an inflection point is not desirable, a user can move control points and/or control curvatures. Similarly, if a user wants the curvature to be monotonically varying and if the curve is not, the user can move control points and/or control curvatures so that the curvature to be monotonically varying. In the case of using a rational curvature function, the user can also change the weights of the rational function.

6 RESULTS

This section shows the results of generated curves and their characteristics. We have implemented our method in C++ and confirmed that curve segments can be generated in real time using an Intel Core i7 2.2 GHz computer. In the figures, a red circle on a curve segment represents an inflection point whereas a blue circle represents a point of curvature extrema. Curves are shown with their curvature combs.

Fig. 2 shows various planar curves based on explicit polynomial Bézier curvature functions. For G^1 Hermite interpolation shown in Fig. 2 (a)-(g), κ_0 and κ_n, where n is the degree, are used as optimization parameters. In the example of Fig. 2 (h) where G^2 Hermite interpolation is performed, κ_1 and κ_{n-1} are used as optimization parameters.

Fig. 2 (a) is an example of a linear curvature function. The generated curve is the Clothoid curve segment. Fig. 2 (b) is an example where the curve has an inflection point. Fig. 2 (c) and (d) are examples of using cubic Bézier curvature functions. The constraint of $\kappa_0 = \kappa_1 = \kappa_2$ is used in (c), whereas in (d) the constraint of $\kappa_1 = \kappa_2 = \kappa_3$ is used. Fig. 2 (e) and (f) are similar examples but quintic Bézier curvature functions are used. The constraint of $\kappa_0 = \kappa_1 = \kappa_2 = \kappa_3 = \kappa_4$ is used in (e), whereas in (f) the constraint of $\kappa_1 = \kappa_2 = \kappa_3 = \kappa_4 = \kappa_5$ is used. Fig. 2 (g) is an example of using cubic Bézier curvature function where the constraint of $\kappa_0 = \kappa_1, \kappa_2 = \kappa_3$ is given. Thus in (g), $\frac{d\kappa}{ds} = 0$ at both endpoints. Harada et al. [9] have pointed out that in many aesthetically pleasing connection between curve segments, the first derivative of curvature become 0. Thus, the example of (g) is important for aesthetically pleasing connection between segments. Note that in Fig. 2 (a), (c-g), the same G^1 Hermite interpolation conditions are used. Various kinds of curvature variation can be generated for the same G^1 Hermite interpolation conditions. Fig. 2 (h) shows an example of G^2 Hermite interpolation using cubic curvature function.

For G^2 Hermite interpolation, there is no guarantee that the curvature of generated curve is monotonically varying as shown in Fig. 3 (a). As shown in Fig. 3 (b), by appropriately modifying weights of explicit rational cubic Bézier curvature function, we can generate a curve with monotonically varying curvature with the same G^2 Hermite interpolation conditions. Thus by using rational functions, the generated curves can match a greater variety of G^2 Hermite interpolation conditions than using polynomial functions if the degree is the same.

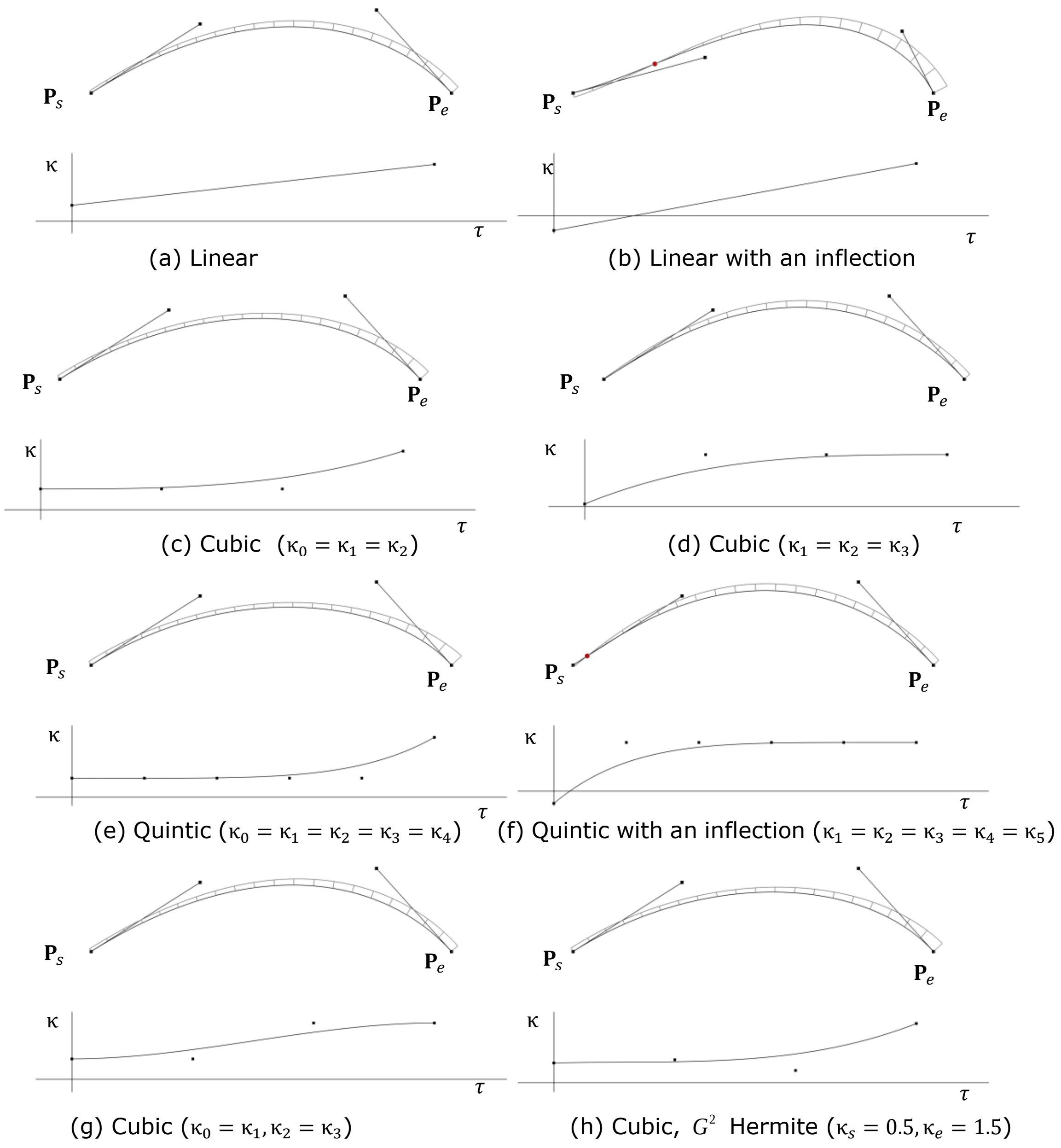
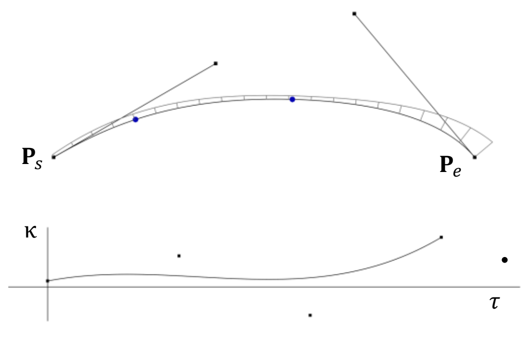
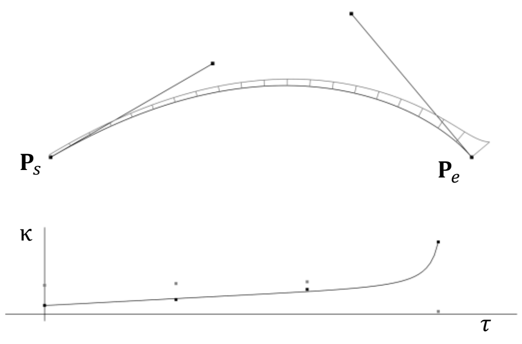


Figure 2: Generated curves based on explicit Bézier curvature functions.



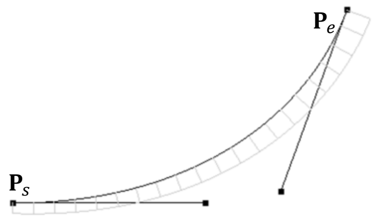
(a) G^2 Hermite using polynomial Bézier



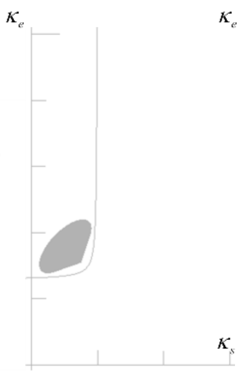
(b) G^2 Hermite using rational Bézier

Figure 3: G^2 Hermite Interpolation using polynomial and rational Bézier curves ($\kappa_s = 0.5, \kappa_e = 2.0$).

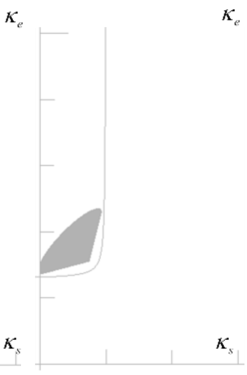
For curves based on explicit polynomial Bézier curvature functions of degree 3, 5 and 10, Fig. 4 (b), (c), (d) show experimentally generated G^2 Hermite interpolation regions of κ_s, κ_e where curves with monotonically varying curvature are to be generated for the given G^1 Hermite interpolation conditions shown in Fig. 4 (a). κ_1, κ_{n-1} are used as optimization parameters and $\kappa_{2,} \cdots, \kappa_{n-2}$ are linearly interpolated using κ_1 and κ_{n-1}. The hyperbolas shown in (b), (c), (d) show the theoretically viable regions where curves with monotonically varying curvature exist [1]. As the degree of the explicit Bézier curvature function increases, the viable region also gets larger.



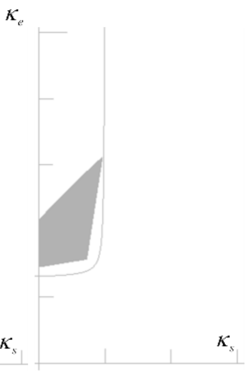
(a) G^1 Hermite interpolation conditions



(b) $n = 3$



(c) $n = 5$



(d) $n = 10$

Figure 4: G^2 Hermite region for curves based on explicit polynomial Bézier curvature functions.

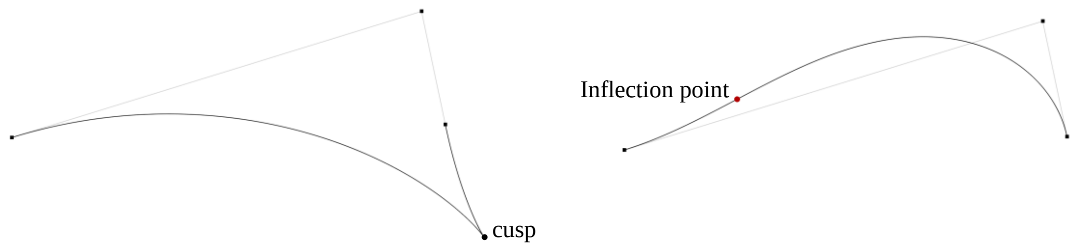
7 COMPARISON WITH CURVES BASED ON CURVATURE RADIUS FUNCTION

This section compares the proposed curves with curves based on curvature radius functions [16, 23]. We call curves based on curvature radius functions TAP (Tangential Angle Parameterization) curves. The proposed curves and TAP curves have a similar property in that the curvature profile can be controllable under G^1 or G^2 Hermite interpolation conditions. In TAP curves, no numerical integration is necessary and G^1 or G^2 Hermite interpolation can be performed by solving linear equations. Although a cusp, which is a point of zero curvature radius, can be represented, an inflection point cannot. In the proposed curves, numerical integration is required to compute the points on a curve and an optimization of two parameters is required for G^1 or G^2 Hermite interpolation. However, as we have confirmed in our implementation, it is fast enough for interactive design of curve segments. In TAP curves, a cusp may arise in G^1 Hermite interpolation, whereas an inflection point may arise in the proposed curves which is more preferable in most cases. See Fig. 5.

Both TAP curves and the proposed curves are related to log-aesthetic curves. Let α be the shape parameter of log-aesthetic curves and Λ be a constant. See [25] for the details of α and Λ. The curvature of log-aesthetic curves is represented by

$$\kappa = \begin{cases} e^{-\Lambda s} & \text{if } \alpha = 0 \\ (\Lambda \alpha s + 1)^{-\frac{1}{\alpha}} & \text{otherwise} \end{cases}. \tag{12}$$

TAP curves can exactly represent log-aesthetic curves with $\alpha = 2, \frac{3}{2}, \frac{4}{3}, \ldots$ [16]. Log-aesthetic curves of $\alpha = -1, -\frac{1}{2}, -\frac{1}{3}, \ldots$ can be exactly represented by the proposed curves, since Eqn. (12) becomes polynomial for such α. Yoshida et al. have used Taylor expansion of the equation of log-aesthetic curves to approximate log-aesthetic curves in terms of polynomial Bézier curves [29]. Another way to approximate log-aesthetic curves is to use Taylor expansion of Eqn. (12) for approximating the curvature function in terms of explicit polynomial Bézier curves. How to perform G^1 Hermite interpolation for such an approximation is an area for future research.



(a) TAP curve segment (b) Proposed curve segment

Figure 5: A TAP curve segment a proposed curve segment for the same G^1 Hermite interpolation conditions.

8 CONCLUSIONS

This paper proposed planar curves based on explicit polynomial or rational Bézier curvature functions and a method for G^1 and G^2 Hermite interpolation. We have implemented the method and confirmed that curves segments can be generated in real time. In addition to examples of generated curves, we visualized the regions of curvatures at two endpoints where a curve segment with monotonically varying curvature can be generated for given G^1 Hermite interpolation conditions.

There are several directions for future work. We first would like to extend our approach by using explicit B-spline curves so that the curves can match a wider variety of G^2 Hermite interpolation conditions. We would also like to generate curves that can cover all the possible G^2 Hermite regions shown in Fig. 4 (b)-(d). We are also planning to extend the idea to 3D curves by specifying curvature and torsion plots in terms of explicit Bézier curves. Since there is no concept of tangential angle in 3D, how to extend our idea to 3D curves in an efficient way is a problem for future work.

9 ORCID

Norimasa Yoshida, http://orcid.org/0000-0001-8889-0949
Takafumi Saito, http://orcid.org/0000-0001-5831-596X

ACKNOWLEDGEMENTS

The authors are grateful for anonymous reviewers for their helpful comments on ways of improving the paper. This works was partially supported by the funding provided by JSPS KAKENHI grant (No. 17K00172 and 16H02824).

REFERENCES

[1] Ali J. M.; Tookey, R. M.; Ball, J. V.; Ball, A. A.: The generalized Cornu spiral and its application to span generation, Journal of Computation and Applied Mathematics, 102(1), 1999, 37-47. https://doi.org/10.1016/S0377-0427(98)00207-6
[2] Dietz, D. A.; Piper, B.: Interpolation with cubic spirals, Computer Aided Geometric Design, 21(2), 2004, 165-180. https://doi.org/10.1016/j.cagd.2003.09.002
[3] Dietz, D. A.; Pieper, B.; Sebe, E.: Rational cubic spirals, Computer Aided Design, 40(1), 2008, 3-12. https://doi.org/10.1016/j.cad.2007.05.001
[4] Farin, G.; Class A Bézier curves, Computer Aided Geometric Design, 23(7), 2003, 573-581. https://doi.org/10.1016/j.cagd.2006.03.004
[5] Frey, W. H.; Field, D. A.: Designing Bézier conic segments with monotone curvature, Computer Aided Geometric Design, 17(6), 2000, 457-483. https://doi.org/10.1016/S0167-8396(00)00011-X
[6] Gobithaasan, R. U.; Yee, L. P.; Miura K. T.: Shape analysis of generalized log-aesthetic curves, International Journal of Mathematical Analysis, 7(33-36), 1751-1759, 2013.
[7] Guggenheimer, H. W.: Differential Geometry, Dover, 1997.
[8] Harada, T.; Yoshimoto, F.; Moriyama, M.: An aesthetic curve in the field of industrial design, In Proceedings of IEEE Symposium on Visual Languages, 1999, 38–47. https://doi.org/10.1109/VL.1999.795873
[9] Harada, T.; Kono, M.: Reconstruction of rough sketch by connecting visual languages aesthetically, Journal of the Science of Design, 55(5), 2008, 55-64 (in Japanese). https://doi.org/10.11247/jssdj.55.5_55
[10] Miura, K.T.: A general equation of aesthetic Curves and its self-affinity, Computer-Aided Design & Applications, 3(1-4), 2006, 457-464. https://doi.org/10.1080/16864360.2006.10738484
[11] Miura, K. T.; Shibuya, D.; Gobithaasan, R. U.; Usuki, S.: Designing log-aesthetic splines with G^2 continuity, Compute-Aided Design and Applications, 2013, 10(6). https://doi.org/10.3722/cadaps.2013.1021-1032
[12] Nutbourne, A. W.; McLellan, P. M.; Kensit, R. M. L.: Curvature profiles for plane curves, Computer Aided Design, 4(4), 1972, 176–184. https://doi.org/10.1016/0010-4485(72)90072-3
[13] Pal, T. K.; Nutbourne, A. W.; Two-dimensional curve synthesis using linear curvature elements, Computer Aided Design, 9(2), 1977, 121–134. https://doi.org/10.1016/0010-4485(77)90044-6
[14] Pal, T. K.; Intrinsic spline curve with local control, Computer Aided Design, 10(1), 1978, 19–29. https://doi.org/10.1016/0010-4485(78)90005-2
[15] Saito, T.; Wang, G.-J.; Sederberg, T. W.: Hodographs and normal of rational curves and surfaces, Computer Aided Geometric Design, 12(4), 1995, 417-430. https://doi.org/10.1016/0167-8396(94)00023-L
[16] Saito, T.; Yoshida, N.: Proposal of Tangential Angle Parameterization Curves, Mathematical Methods for Curves and Surfaces, 2016.
[17] Sapidis, N. S.; Frey, W. H.: Controlling the curvature of quadratic Bézier curve, Computer Aided Geometric Design, 9(2), 1992, 85-91. https://doi.org/10.1016/0167-8396(92)90008-D
[18] Sederberg, T. W.; Nishita, T.: Curve intersection using Bézier clipping, Computer-Aided Design, 22(9), 1990, 538-549. https://doi.org/10.1016/0010-4485(90)90039-F
[19] Sederberg, T. W.: Computer Aided Geometric Design, Course Note, 2012. http://hdl.lib.byu.edu/1877/2822

[20] Sederberg, T. W.; Wang, X.: Rational hodographs, Computer Aided Geometric Design, 4(4), 1987, 333–335. https://doi.org/10.1016/0167-8396(87)90008-2

[21] Watanabe, Y.; Saito, T.; Kuroda, M.: A method for generation of curves from specified curvature profile (in Japanese), in: Graphics and CAD Work-shop of Information Processing Society of Japan, 1997, 7-12.

[22] Wang, Y.; Zhao B.; Zhang L.; Xu J.; Wang, K.; Wang S.; Designing fair curves using monotone curvature pieces, Computer Aided Geometric Design, 21(5), 2004, 515-527. https://doi.org/10.1016/j.cagd.2004.04.001

[23] Wu, W.; Yang, X.: Geometric Hermite interpolation by a family of intrinsically defined planar curves. Computer Aided Design, 77, 2016, 86-97. https://doi.org/10.1016/j.cad.2016.04.001

[24] Yan, Z.; Schiller, S.; Wilensky, G.; Carr, N.; Schaefer, S.: κ-curves: interpolation at local maximum curvature, ACM Transactions on Graphics, 36(4), 2017, 129:1-129:7. https://doi.org/10.1145/3072959.3073692

[25] Yoshida, N.; Saito. T.: Interactive aesthetic curve segments, Visual Computer, 22(9–11), 2006, 896–905. https://doi.org/10.1007/s00371-006-0076-5

[26] Yoshida, N.; Hiraiwa, T.; Saito, T.; Interactive control of planar class A Bézier curves using logarithmic curvature graphs, Compute-Aided Design and Applications, 5(1-4), 2008, 121-130. http://dx.doi.org/10.3722/cadaps.2008.121-130

[27] Yoshida, N.; Fukuda, R.; Saito, T.: Interactive generation of 3D class A Bezier curve segments, Computer-Aided Design and Application, 7(2), 2010, 163-172. http://dx.doi.org/10.3722/cadaps.2010.163-172

[28] Yoshida, N.; Fukuda, R.; Saito, T.: T. Saito, Logarithmic curvature and torsion graphs, in Mathematical Methods for Curves and Surfaces 2008 edited by Daehlen et al., LNCS 5862, Springer, 2010, 434-443. https://doi.org/10.1007/978-3-642-11620-9_28

[29] Yoshida, N., Fukuda, R., Saito, T. Saito, T.: Quasi-log-aesthetic curves in polynomial Bézier form, Computer-Aided Design and Applications, 10(6), 2013, 983-993. https://doi.org/10.3722/cadaps.2013.983-993

[30] Yoshida, N.; Saito, T.: Quadratic log-aesthetic curves, Computer-Aided Design and Applications, 14(2), 2017, 219-226. https://doi.org/10.1080/16864360.2016.1223434