www.cad-journal.net
Computer-Aided Design AND APPLICATIONS™

Visualization of the Curvature Monotonicity Regions of Polynomial Curves and its Application to Curve Design

Norimasa Yoshida[1], Seiya Sakurai[2], Hikaru Yasuda[3], Taisei Inoue[4], Takafumi Saito[5]

[1]Nihon University, yoshida.norimasa@nihon-u.ac.jp
[2]Nihon University, cise20009@g.nihon-u.ac.jp
[3]Nihon University, cihi22003@g.nihon-u.ac.jp
[4]Nihon University, inoue.taisei@nihon-u.ac.jp
[5]Tokyo University of Agriculture and Technology, txsaito@cc.tuat.ac.jp

Corresponding author: Norimasa Yoshida, yoshida.norimasa@nihon-uac.jp

Abstract. Freeform curves, such as Bézier curves or B-spline curves are widely used in many applications. Although freeform curves have many nice properties, controlling the curvature variation by manually moving a control point is not easy. For polynomial Bézier and B-spline curves, this paper proposes a real-time method to visualize the region of a control point where the curvature becomes monotonically varying. By representing the numerator of the curvature derivative in Bernstein form, the proposed method checks the curvature monotonicity of a specific control point for every pixel on the screen using a GPU. With this approach, a user can determine where to move the control point to achieve monotonically varying curvature. Additionally, two applications of the proposed method are presented.

Keywords: curvature monotonicity region, Bézier curves, B-spline curves, curve design
DOI: https://doi.org/10.14733/cadaps.2024.75-87

1 INTRODUCTION

Freeform curves, such as Bézier curves or B-splines curves, are widely used in many applications, such as Illustration software and CAD systems. Although these curves have many desirable properties, controlling the curvature variation by manually moving a control point is not easy.

Sapidis et al. clarified the theoretical region of a control point where the curvature of quadratic Bézier curves becomes monotonically varying [13]. However, for Bézier curves of degree 3 or higher, the curvature monotonicity region, i.e., the region of a control point where the curvature becomes monotonically varying, has not been visualized before. By visualizing the curvature monotonicity region in real time, a user can determine where to move the control point to achieve monotonically varying curvature.

Yan et al. proposed κ-curves [18], which are interpolating quadratic Bézier curves that have local maxima of curvature only at interpolating points. Miura et al. extended the method to cubic curves, providing additional

control by α. However, in both approaches, the curve shape cannot be modified locally without introducing additional curvature extrema. A method that can locally control the curve shape without such side effects would be desirable.

In this study, we present a real-time method for visualizing the curvature monotonicity regions of a specific control point for polynomial curves. By using this method, a user can determine where to move a specific control point to achieve monotonically varying curvature. We also demonstrate two applications of this approach.

2 RELATED WORK

To design aesthetically pleasing objects, the use of fair curves is essential. According to [3], fair curves are defined as curves whose curvature plots have relatively few regions of monotonically varying curvature. For polynomial quadratic Bézier curves, Sapidis and Frey [13] presented the necessary and sufficient condition for the curvature to be monotonically varying. Frey and Field [5] demonstrated the conditions for rational quadratic curves. For cubic polynomial Bézier curves, Dietz et al. [2] proposed a method of generating curves with monotonically varying curvature using precomputed tables. Wang et al. used the sufficient monotone curvature conditions to generate curves with monotonically varying curvature.

Over the past two decades, various works related to class A Bézier curves [4] and log-aesthetic curve [6, 9, 19] have emerged. Planar class A Bézier curves are curves with monotonically varying curvature, where the control points are generated by repeatedly applying the 2×2 matrix $\mathbf{M}$ to the first edge of the control polygon. If $\mathbf{M}$ is a scaling and a rotation and satisfies a specific condition, the curvature becomes monotonically varying, and the curves become Mineur's typical curves [8]. Romani et al. [12] showed the necessary and sufficient conditions for the curves to be class A if $\mathbf{M}$ has two real eigenvalues. In [14], 3D typical curves are investigated.

Log-aesthetic curves [19] are curves with linear logarithmic curvature graphs. The linearity of logarithmic curvature graphs constrain the curvature monotonically varying. Approximation methods of log-aesthetic curves in terms of freeform curves have been proposed in [20, 7]. Recently, Tsuchie et al. [15] proposed a high quality approximation method of log-aesthetic curves using the fourth order derivative based on the equation presented in [21]. The method approximates log-aesthetic curves in terms of rational Bézier or B-spline curves.

κ-curves [18] are quadratic Bézier curves that interpolate points and have local maxima of curvature only at the interpolating points. Yan et al. [17] extended this method for reproducing circles using rational quadratic curves. Miura et al. [10] further extended the method to cubic curves so that the method have additional control by α. In contrast, smooth interpolating curves with local control have been proposed in [22, 1], but both methods depend on non-polynomial parametric curves.

In summary, there is no method to visualize the region of curvature monotonicity for polynomial curves with a degree higher than 2. This means that users are often unaware of where to adjust the control point to achieve a monotonically varying curvature. By providing real-time visualization of the curvature monotonicity region, users can identify the region of the control point where the curvature becomes monotonically varying. We present two applications of our approach: a curve design tool, similar to the one in Adobe Illustrator, with the visualization of the curvature monotonicity regions, and an application that locally modifies the shape of the curves generated by κ-curves with G^1 continuity.

3 CHECKING THE CURVATURE MONOTONICITY

Let $\mathbf{P}(t)$ be a planar Bézier curve of degree n given by

$$\mathbf{P}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{P}_i, \tag{1}$$

where $B_i^n(t)$ are Bernstein polynomials and $\mathbf{P}_i$ are control points. Let κ and s be the curvature and the arc length, respectively. Curvature monotonicity can be checked if $\frac{d\kappa}{ds}$ does not change sign for $t \in [0,1]$. For planar curves, $\frac{d\kappa}{ds}$ can be computed by the following equation [3, 16]:

$$\frac{d\kappa}{ds} = \frac{\det(\dot{\mathbf{P}}, \dddot{\mathbf{P}})\dot{\mathbf{P}} \cdot \dot{\mathbf{P}} - 3\,\det(\dot{\mathbf{P}}, \ddot{\mathbf{P}})\dot{\mathbf{P}} \cdot \ddot{\mathbf{P}}}{|\dot{\mathbf{P}}|^6}, \tag{2}$$

where $\dot{\mathbf{P}} = \frac{d\mathbf{P}}{dt}$, $\ddot{\mathbf{P}} = \frac{d^2\mathbf{P}}{dt^2}$, and $\dddot{\mathbf{P}} = \frac{d^3\mathbf{P}}{dt^3}$. Assuming the curve is regular, the denominator of Eq. (2) is always positive. Therefore, checking whether $\frac{d\kappa}{ds}$ changes sign reduces to checking whether the numerator of $\frac{d\kappa}{ds}$ changes sign.

Let the numerator of Eq. (2) be $\lambda(t)$:

$$\lambda(t) = \det(\dot{\mathbf{P}}, \dddot{\mathbf{P}})\dot{\mathbf{P}} \cdot \dot{\mathbf{P}} - 3\,\det(\dot{\mathbf{P}}, \ddot{\mathbf{P}})\dot{\mathbf{P}} \cdot \ddot{\mathbf{P}}. \tag{3}$$

For a polynomial curve of degree n, the degree of $\lambda(t)$ is $4n-7$ [16]. Since $\lambda(t)$ is a polynomial, it can be represented in Bernstein basis as

$$\lambda_B(t) = \sum_{i=0}^{4n-7} B_i^{4n-7}(t)\xi_i. \tag{4}$$

where $B_i^{4n-7}(t)$ is the i-th Bernstein basis of degree $4n-7$ and ξ_i are the corresponding coefficients. In [16], for cubic Bézier curves, the sufficient condition of $\xi_i \geq 0 (i = 0, \ldots, 5)$ or $\xi_i \leq 0 (i = 0, \ldots, 5)$ is used to guarantee the monotonicity of curvature. In this paper, we check the curvature monotonicity more strictly.

Fig 1 (a) shows a cubic Bézier curve with monotonically varying curvature and its corresponding $\lambda(t)$. Since ξ_i are negative, the curve is judged to be monotonically decreasing. Fig 1 (b) shows a cubic Bézier curve with non-monotonically varying curvature. Since $\xi_0 \cdot \xi_5 < 0$, the curve is immediately judged to be non-monotonically varying. If ξ_i do not satisfy either of the two conditions, the curve is recursively subdivided until the first condition is satisfied for all the subdivided segments, or the second condition is satisfied in one of the subdivided segments. We use Algorithm 1 for checking the curvature monotonicity. Note that Algorithm 1 does not take into account the sign change of the curvature, which indicates the presence of an inflection point. We will shortly describe a method to detect the existence of an inflection point.

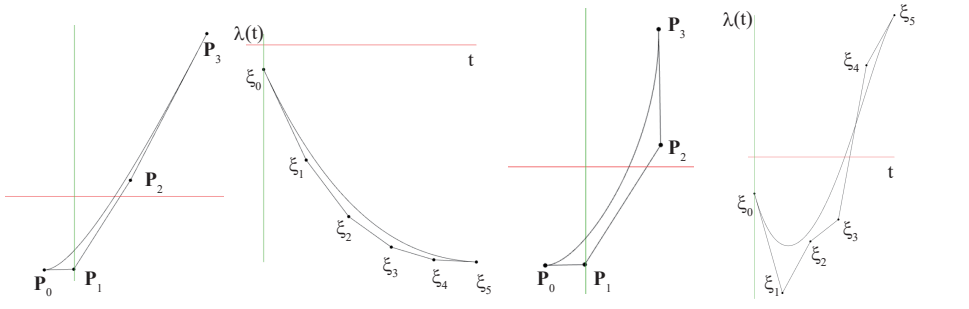


Figure 1: Cubic Bézier curves and their $\lambda(t)$.

Algorithm 1: Curvature monotonicity

(1) If $\xi_i \geq 0 (i = 0, \ldots, 4n-7)$, the (subdivided) curve segment is judged to be monotonically increasing.

(2) If $\xi_i \leq 0 (i = 0, \ldots, 4n-7)$, the (subdivided) curve segment is judged to be monotonically decreasing.

(3) If $\xi_0 \cdot \xi_{4n-7} < 0$, the (subdivided) curve segment is judged to be not monotonically varying.

(4) Recursively subdivide the curve at $t = 0.5$, until the condition (1) or (2) is satisfied for all subdivided curve segments. If the condition (1) and (2) are simultaneously satisfied, the condition (3) is satisfied, or the recursion reaches the user-specified depth, the curvature is judged to be not monotonically varying.

For a specific control point, the curvature monotonicity is checked for all the case where the control point is placed on every pixel on the screen, and the corresponding pixel is colored depending on whether the curvature is monotonically decreasing or increasing. In the fragment shader, we obtain the screen coordinate of the pixel being processed. Using the obtained coordinates, we replace the coordinate of the specified control point, and compute ξ_i in Eq. (4). We then check the curvature monotonicity using Algorithm 1. If the curvature is monotonically decreasing or increasing, we paint the corresponding pixel blue or red, respectively. Since this process is performed for all the pixels in parallel using a GPU, the curvature monotonicity region can be visualized efficiently.

There are two ways to implement Algorithm 1: depth-first search and bread-first search. Let $maxDepth$ denote the maximum depth specified by the user. In depth-first search, a stack with the array size proportional to $maxDepth$ is used. In breadth-first search, a queue with the array size proportional to $2^{(maxDepth-1)}$ is required. Since the size of memory in the fragment shader is limited, we employ the depth-first search in Algorithm 1. For small values of $maxDepth$, breadth-first search can be used, but the strict curvature monotonicity check is not possible.

Algorithm 1 does not check the existence of an inflection point. Therefore, a curve segment may include an inflection point, which may be undesirable in certain practical situations. To avoid this issue, it is necessary to check for the existence of an inflection point by representing the numerator of the curvature function

$$\kappa = \frac{\det(\dot{\mathbf{P}}, \ddot{\mathbf{P}})}{|\dot{\mathbf{P}}|^3} \tag{5}$$

in terms of Bernstein form and checking for the existence of 0 in the parameter range $t \in [0, 1]$. The numerator of the curvature function of a Bézier curve of degree n can be represented by a Bernstein polynomial of degree $2n - 4$:

$$\eta(t) = \det(\dot{\mathbf{P}}, \ddot{\mathbf{P}}) = \sum_{i=0}^{2n-4} B_i^{2n-4}(t)\nu_i. \tag{6}$$

where ν_i are the coefficients of the Bernstein polynomial. Algorithm 2 is used to check for the existence of an inflection point. In our implementation, we determine that a curve has an inflection point if the recursion reaches the user-specified depth, as we aim to avoid having inflection points on the curve.

Algorithm 2: Existence of an inflection point

(1) If $\nu_0 \cdot \nu_{2n-4} \leq 0$, the curve has an inflection point.

(2) If $\nu_i < 0 (i = 0, \ldots, 2n - 4)$ or $\nu_i > 0 (i = 0, \ldots, 2n - 4)$, the (subdivided) curve segment is judged to be without an inflection point.

(3) Recursively subdivide the curve at $t = 0.5$ until one of the following conditions is satisfied: the condition (1) is satisfied in one of the subdivided curve segments, the condition (2) is satisfied for all the subdivided curve segments, or the recursion reaches the user-specified depth.

We have two types of curvature monotonicity regions:

- CMRWI: CMRWI refers to a region where the curvature varies monotonically, but an inflection point may be present.

- CMR: CMR refers to a region where the curvature of a curve varies monotonically without an inflection point.

If a control point is placed within its CMRWI, the curvature varies monotonically, and the curve may contain an inflection point. If a control point is placed within its CMR, the curve does not have an inflection point. To visualize a CMR, we first check for the existence of an inflection point and then verify the curvature monotonicity. If the existence of an inflection point is confirmed, the curvature is considered to be non-monotonically varying.

To visualize the CMRWI of a specific control of a Bézier curve, we compute ξ_i in Eq. (4) and check for the curvature monotonicity using Algorithm 1 for all cases where the specific control point is placed at every pixel in the screen window. Visualization of CMR is performed similarly, except that we initially check for the existence of an inflection point. For efficiency, we perform the computation using a GPU. For CMRWIs, we compute ξ_i and check for the curvature monotonicity using Algorithm 1 in the fragment shader. For CMRs, we also compute ν_i. Depending on whether the curvature is monotonically increasing, monotonically decreasing, or not monotonically varying, we paint the corresponding pixel with user-specified colors. In this research, we paint regions with monotonically increasing curvature using red and regions with monotonically decreasing curvature using blue.

4 VISUALIZATION OF THE CURVATURE MONOTONICITY REGIONS

4.1 Bézier Curves

Figure 2 illustrates the CMRWIs for $\mathbf{P}_0$ and the corresponding curvature plots with varying positions of $\mathbf{P}_0$. If $\mathbf{P}_0$ lies within the blue region, the curvature monotonically decreases, as depicted in Figure 2(a). If $\mathbf{P}_0$ lies within the red region, the curvature monotonically increases, as shown in Figure 2(c). Note that the curve has an inflection point. Figures 2(b) and (d) depict the cases where $\mathbf{P}_0$ is outside these regions, indicating that the curvature does not monotonically vary.

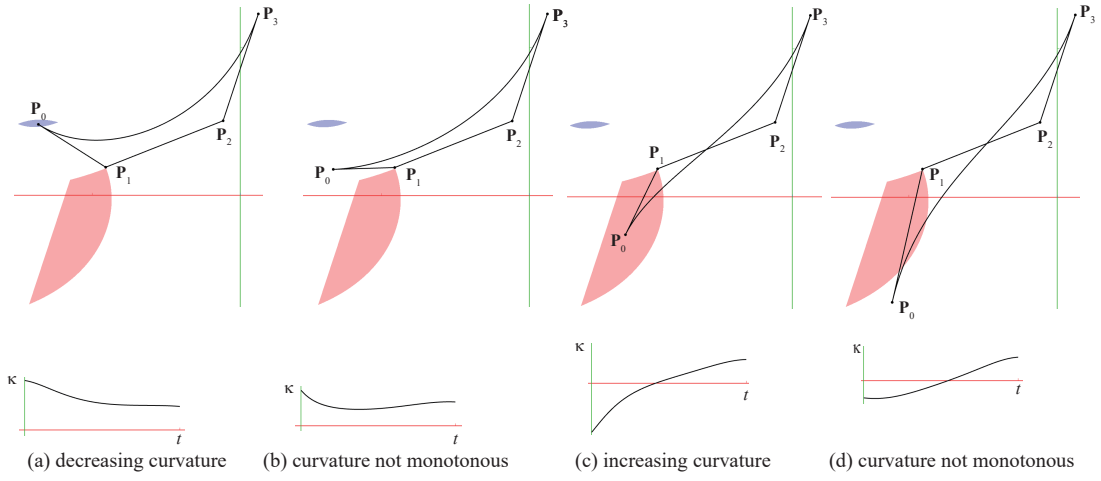
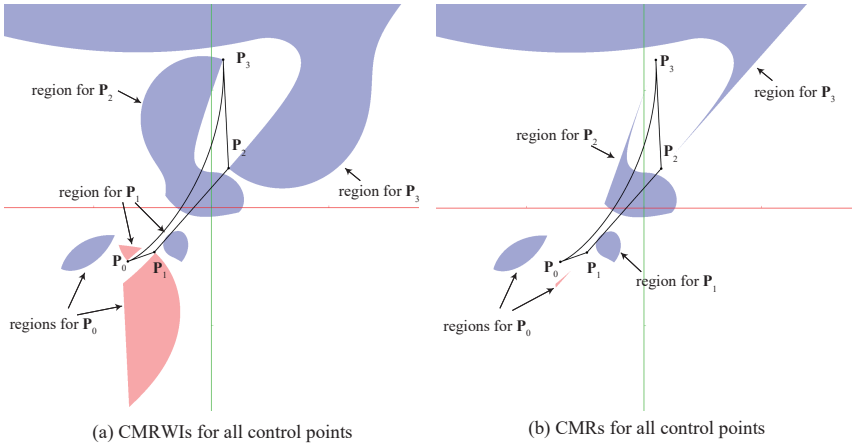


Figure 2: CMRWIs for $\mathbf{P}_0$ and the curve plots.

Fig. 3 shows CMRWIs and CMRIs of all control points for a cubic Bézier curve. To visualize the regions for all control points, the curvature monotonicity is repeatedly checked for each control point. For the overlapping regions where the CMRWIs (or CMRs) of different control points overlap, the color values are subtracted from white by the number of times they overlap. Therefore, a darker region means that the curvature monotonicity

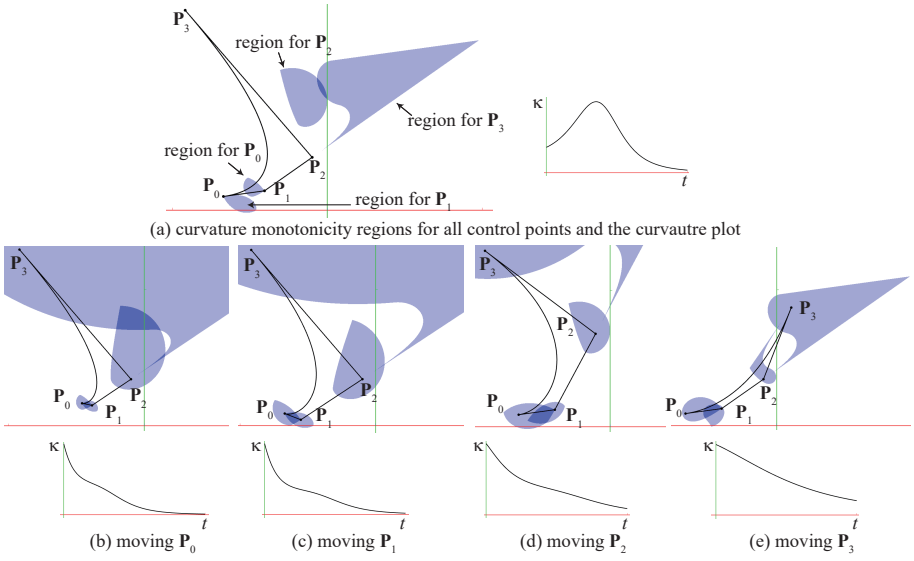
regions overlap. Note that there is no overlapping region in Fig. 3, whereas there are overlapping regions in Fig. 4. To see which region corresponds to which control point, we can interactively move one of the control point, and the region that does not change its shape is the region for the control point. We can also select to visualize the region for a specific control point from the user interface.



(a) CMRWIs for all control points (b) CMRs for all control points

Figure 3: CMRWIs and CMRs for a cubic Bézier curve.

Figure 4 shows the CMRs for all control points and the corresponding curvature plots. Figure 4(a) shows a cubic Bézier curve with the CMRs and its curvature plot. The curvature is not monotonically varying. Figure 4(b), (c), (d), or (e) shows an example of moving each control point within its curvature monotonicity region so that the curvature becomes monotonically varying.



(a) curvature monotonicity regions for all control points and the curvautre plot

(b) moving $\mathbf{P}_0$ (c) moving $\mathbf{P}_1$ (d) moving $\mathbf{P}_2$ (e) moving $\mathbf{P}_3$

Figure 4: CMRs for all control points and the curvature plots. In (b), (c), (d), (e), each control point is moved within its CMR so that the curvature becomes monotonically varying.

Figure 5(a) and (b) show a cubic Bézier curve and its curvature plot. Although the CMRs are set to visualize for all the control points, no regions are shown. This is because the curvature cannot be modified to be monotonically varying by moving only one of its control points at every pixel on the screen. For cubic polynomial curves, the regions can be immediately shown by moving one of their control points. Fig. 5(c) shows an example of moving $\mathbf{P}_1$ so that the curvature monotonicity region of a control point is visualized. Fig. 5(d) shows a curve moving $\mathbf{P}_2$ within its corresponding blue region to make the curvature monotonically varying. Fig. 5(e) shows its curvature plot.

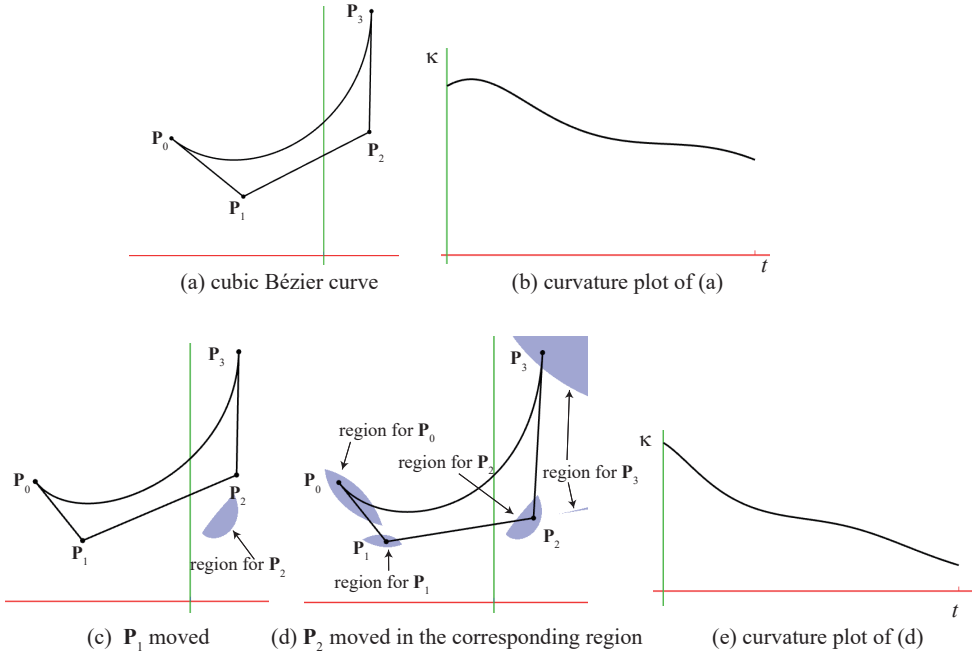


Figure 5: A cubic Bézier curve with no CMRs and an example of moving control points to make the curvature monotonically varying.

Fig. 6(a) and (b) show CMRWIs and CMRs for a quintic Bézier curve. Fig. 6(c) shows an example of moving $\mathbf{P}_5$ within its CMR so that the curvature becomes monotonically decreasing. Fig. 6(d) shows its curvature plot. In general, it is much harder for quintic or higher degree Bézier curves to place control points such that CMRWIs or CMRs are shown. One approach is to find a cubic Bézier curve with monotonically varying curvature and degree elevate it. Another approach is to place control points close to typical curves [8].

The curvature is monotonically varying if condition (1) or (2) in Algorithm 1 is satisfied in the curve segment or every subdivided curve segment. On the other hand, if (3) in Algorithm 1 is satisfied in at least one (subdivided) segment, then the curvature is not monotonically varying. Fig. 7 shows the CMRWIs and non-CMRWIs (the region where the curvature is not monotonically varying) with different depths for each control point. The red or blue regions become darker as the required depth in Algorithm 1 increases for confirming curvature monotonicity. The gray region also becomes darker as more depth is required to confirm that the curvature is not monotonically varying. The lightest blue or red regions correspond to the sufficient condition in [16].

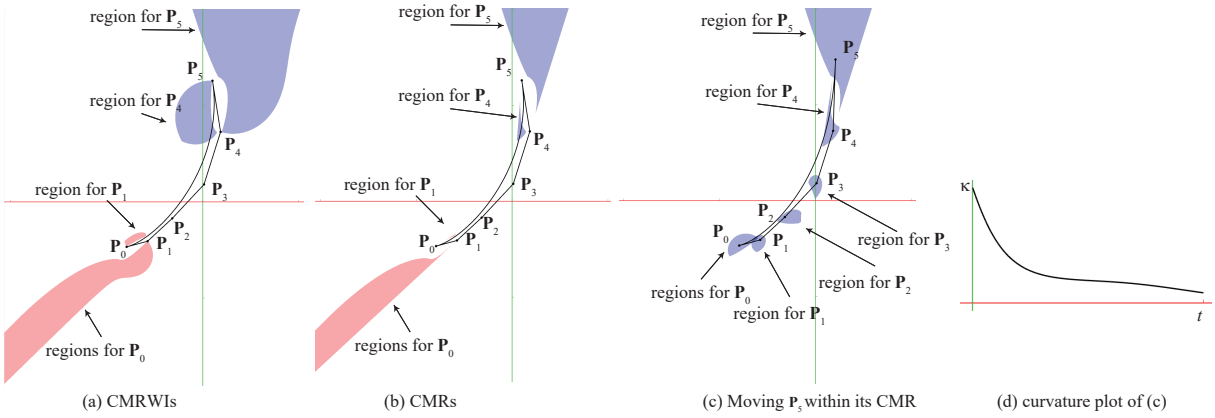


Figure 6: CMRWIs and CMRs for a quintic Bézier curve.

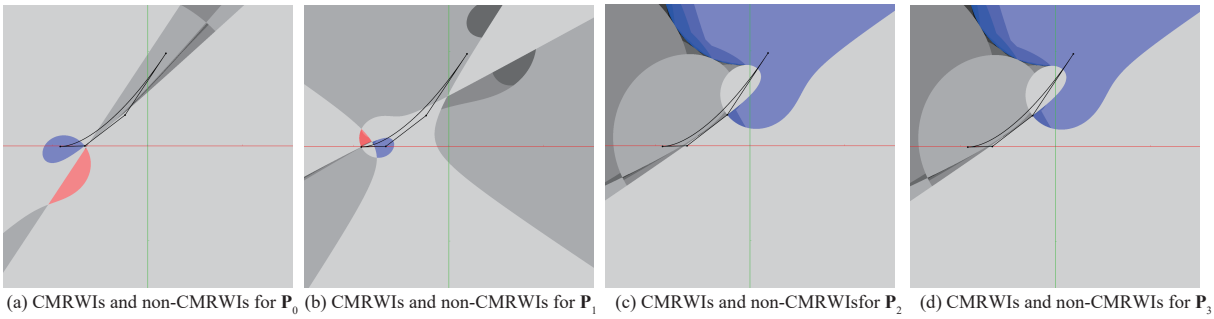


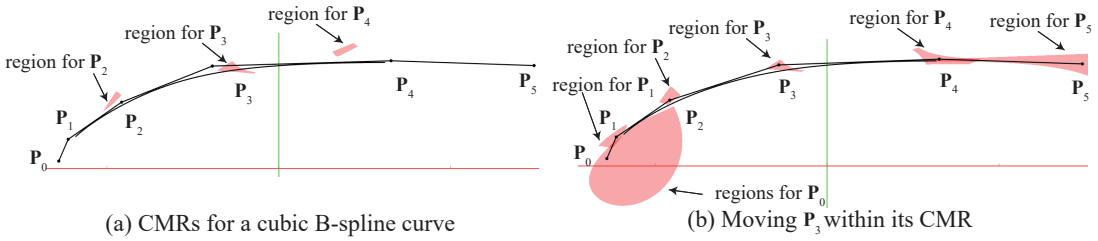
Figure 7: CMRWIs and non-CMRWIs for a cubic Bézier curve with different depth. The darker the red or blue region is, more depth is required in Algorithm 1 to confirm the curvature monotonicity. The darker the gray region is, more depth is required to confirm that the curvature is not monotonically varying.

4.2 B-spline Curves

Visualization of CMRs or CMRWIs can also be applied to B-spline curves. In this section, we show examples of polynomial cubic B-spline curves. By using B-spline curves, users can generate a curve composed of multiple segments with monotonically varying curvature. Additionally, users can generate a curvature extremum that satisfies C^2 continuity.

To visualize CMRs or CMRWIs of B-spline curves, the Bézier control points are computed, and the curvature monotonicity is checked for all the segments in the fragment shader. If the curvature is either monotonically increasing or decreasing for all the segments, the corresponding pixel is painted with red or blue, respectively. This setting is useful to ensure that the curvature varies monotonically throughout the entire curve.

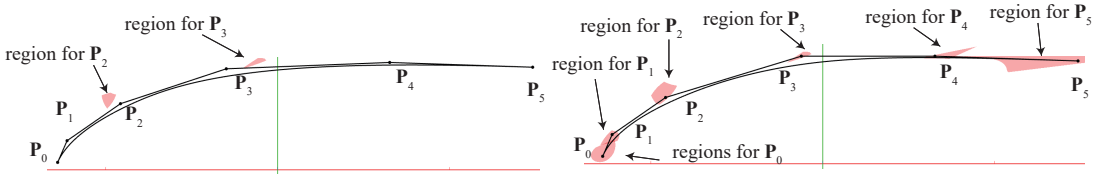
Fig. 8(a) shows CMRs for a cubic B-spline curve with knots $[0, 1, 2, 3, 4, 5, 6, 7]$ (uniform knots) and 6 control points. Note that knots are specified in polar form [11]. CMRs are shown only for $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$. This means that no matter where $\mathbf{P}_0$, $\mathbf{P}_4$, or $\mathbf{P}_5$ is moved, the curvature will never be monotonically varying. If $\mathbf{P}_1$, $\mathbf{P}_2$, or $\mathbf{P}_3$ is moved within its corresponding CMR, the curvature becomes monotonically increasing. Fig. 8(b) shows an example of moving $\mathbf{P}_3$ within its CMR. Note that the control points are placed in a clockwise direction, so the curvature is negative. Therefore, the CMRs are shown in red, indicating that the curvature will be monotonically increasing if a control point is placed within its corresponding CMR.



(a) CMRs for a cubic B-spline curve (b) Moving $\mathbf{P}_3$ within its CMR

Figure 8: CMRs for a cubic B-spline with knots $[0, 1, 2, 3, 4, 5, 6, 7]$.

Fig. 9(a) shows CMRs for a cubic B-spline curve with knots $[0, 0, 0, 1, 2, 3, 3, 3]$ and 6 control points. Fig. 9(b) shows an example of moving $\mathbf{P}_3$ to ensure that the curvature increases monotonically for the entire curve.

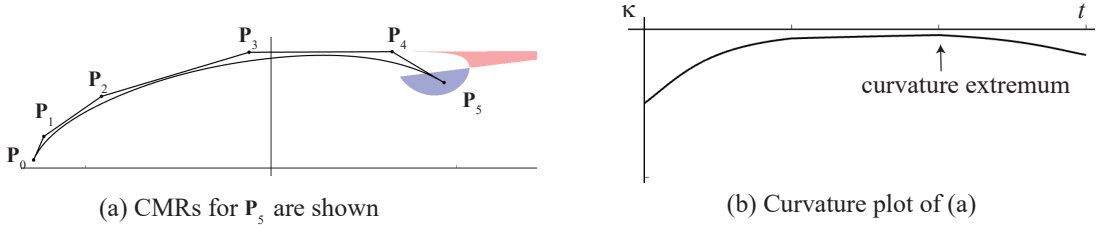


(a) CMRs for a cubic B-spline curve (b) Moving $\mathbf{P}_3$ within its CMR

Figure 9: CMRs for a cubic B-spline with knots $[0, 0, 0, 1, 2, 3, 3, 3]$.

Suppose we desire a curvature extremum at the connection point between k-th segment and $(k+1)$-th segment. Then in the fragment shader, the curvature monotonicity is checked separately for the first k segments and the remaining segments. If we want a curvature maxima, then the first k segment should be monotonically increasing and the remaining segments should be monotonically decreasing. This setting is useful for intentionally creating a curvature extremum at the connection point of curve segments. Note that a control point may affect the two segments separated by the curvature extremum. In our current implementation, the corresponding pixel is painted with blue if the curvature of the latter segment is monotonically decreasing, or red if the curvature is monotonically increasing.

Fig. 10(a) shows an example of creating a curvature extremum between the 2nd and 3rd curve segment. In Fig. 10(a), $\mathbf{P}_5$ in Fig. 9(b) is moved and CMRs for $\mathbf{P}_5$ are shown. To visualize the CMRs of $\mathbf{P}_5$, the curvature monotonicity is checked separately for the first two segments and the third segment. In this example, the curvature is monotonically increasing for the first two segments and monotonically decreasing for the third segment. If $\mathbf{P}_5$ is positioned in the red region, the curvature becomes monotonically increasing for the entire curve, as shown in Fig. 9(b).



(a) CMRs for $\mathbf{P}_5$ are shown (b) Curvature plot of (a)

Figure 10: An example of creating a curvature extremum.

5 APPLICATIONS

5.1 Application 1: Curve Design - Visualizing the Regions of Curvature Monotonicity

We have developed a curve design tool, similar to the one in Adobe Illustrator and have added the visualization of the CMRWIs. Furthermore, our current tool can be easily adjusted to visualize CMRs instead. By visualizing CMRWIs or CMRs, we can determine where to move a control point to achieve the curvature monotonicity for a particular segment of the curve.

Fig. 11 illustrates an example of visualization. The dark red region close to $\mathbf{Q}_3$ indicates that the curvature of the cubic Bézier curve defined by $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$ and $\mathbf{Q}_4$ becomes monotonically varying if $\mathbf{Q}_3$ lies within the region. Similarly, the cyan region close to $\mathbf{Q}_5$ indicates the same for the curve defined by $\mathbf{Q}_4$, $\mathbf{Q}_5$, $\mathbf{Q}_6$, $\mathbf{Q}_7$. When $\mathbf{Q}_4$ is moved, both $\mathbf{Q}_3$ and $\mathbf{Q}_5$ are moved accordingly to maintain G^1 continuity between the two Bézier curve segments. If $\mathbf{Q}_4$ is placed within the purple region, then both the curvature of the curve defined by $\mathbf{Q}_1$, $\mathbf{Q}_2$, $\mathbf{Q}_3$ and $\mathbf{Q}_4$ and the curvature of the curve defined by $\mathbf{Q}_4$, $\mathbf{Q}_5$, $\mathbf{Q}_6$ and $\mathbf{Q}_7$ become monotonically varying.

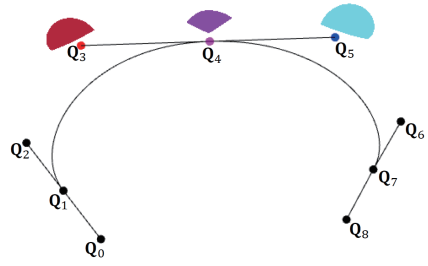


Figure 11: Visualization of monotone curvature regions in a curve design tool. Two Bézier curve segments with G^1 continuity.

Fig. 12(a) shows an apple designed without visualizing the curvature monotonicity region. The basis apple shape, except for the core and the leaf, is composed of 8 cubic Bézier curve segments with G^1 continuity. Fig. 12(b) to (f) demonstrate the process of adjusting the curve shape to ensure that the curvature becomes monotonically varying within user-specified curve segments. Fig. 12(g) shows the final design.

5.2 Application 2: Modifying the Shape of the Curve Generated by κ-curves

Visualization of CMRWIs (and CMRs) can also be applied to modify the curve shape generated by κ-curves. κ-curves [18] are interpolating quadratic Bézier curves and have local maxima of curvature only at the interpolating points.

We begin by subdividing the curves generated by κ-curves at the curvature maxima using the de Casteljau's algorithm. As quadratic Bézier curves are employed in κ-curves, determining the parameter value at the points of curvature maxima is straightforward. Let $\mathbf{P}_i = [x_i \ y_i]^{\mathrm{T}} (i = 0, 1, 2)$ represent the control points. Then the parameter value t at the point of maximum curvature is computed as the point at $\lambda(t) = 0$:

$$t = \frac{(\mathbf{P}_0 - \mathbf{P}_1) \cdot (\mathbf{P}_0 - 2\mathbf{P}_1 + \mathbf{P}_2)}{|\mathbf{P}_0 - 2\mathbf{P}_1 + \mathbf{P}_2|^2} \tag{7}$$

Next, we degree elevate the curves to cubic Bézier curves. As shown in Fig. 13(f), we can modify $\mathbf{P}_{k-1}$ and $\mathbf{P}_{k+1}$ within the blue or red regions without introducing another curvature extremum. If $\mathbf{P}_{k-1}$, $\mathbf{P}_k$, $\mathbf{P}_{k+1}$ lie on a straight line, G^1 continuity is guaranteed. Note that almost G^2 continuity is guaranteed in κ-curves.

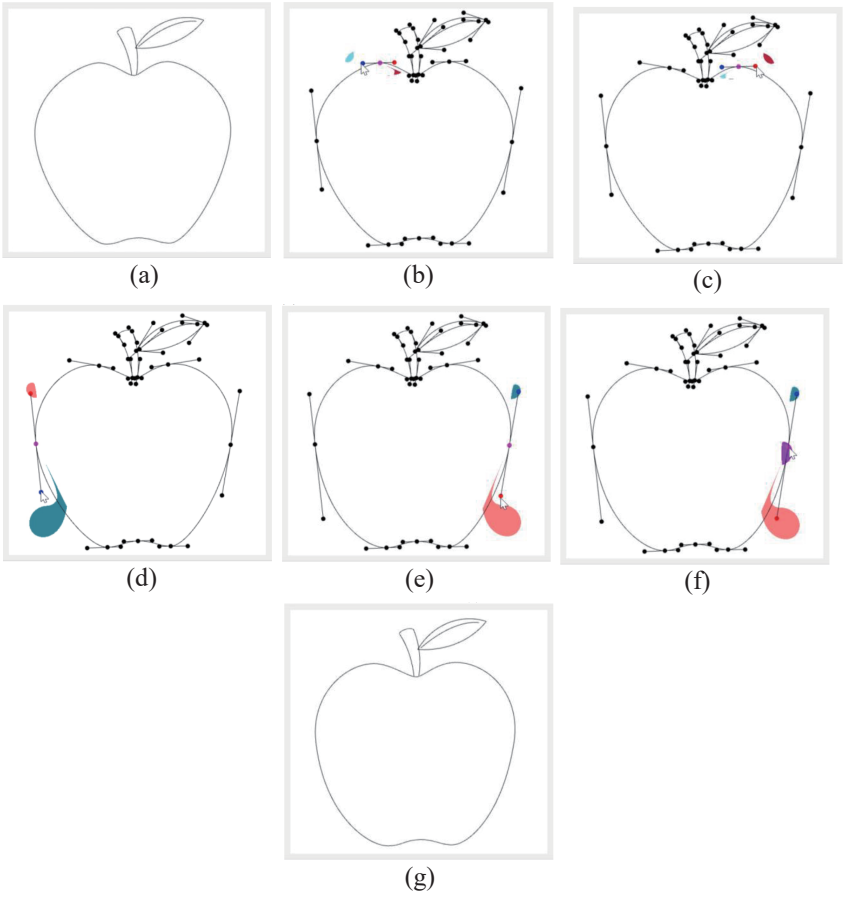


Figure 12: Design of an apple shape. (a) is the original shape created without visualizing the curvature monotonicity regions. (b) to (f) are the process of modifying the shape so that the curvature becomes monotonically varying within a user-specified curve segment. (g) is the shape of the final design.

Although our approach may reduce the continuity to G^1 where we modify the curve shape, it may not be a problem in many illustration applications.

Fig. 13(a) shows a shape created by κ-curves. Fig. 13(b) shows an instance of modifying the curve shape using our approach. Fig. 13(c) to (f) depict the intermediate process. Our method enables local modification of curve shape without introducing another curvature extremum.

6 CONCLUSIONS

In this work, we presented a real-time method to visualize the curvature monotonicity regions (CMRWIs) and the curvature monotonicity regions without an inflection point (CMRs) of polynomial Bézier or B-spline curves using a GPU. By visualizing these regions, users can determine the control point region where the curvature is monotonically varying. We showed that our approach can be immediately applied to a curve design tool, similar to the one in Adobe Illustrator. Futhremore, we demonstrated that the curve shape generated by κ-curves can be locally modified without introducing another curvature extremum, while maintaining G^1 continuity.

Currently, we are working on applying this idea to planar rational curves and 3D polynomial and rational

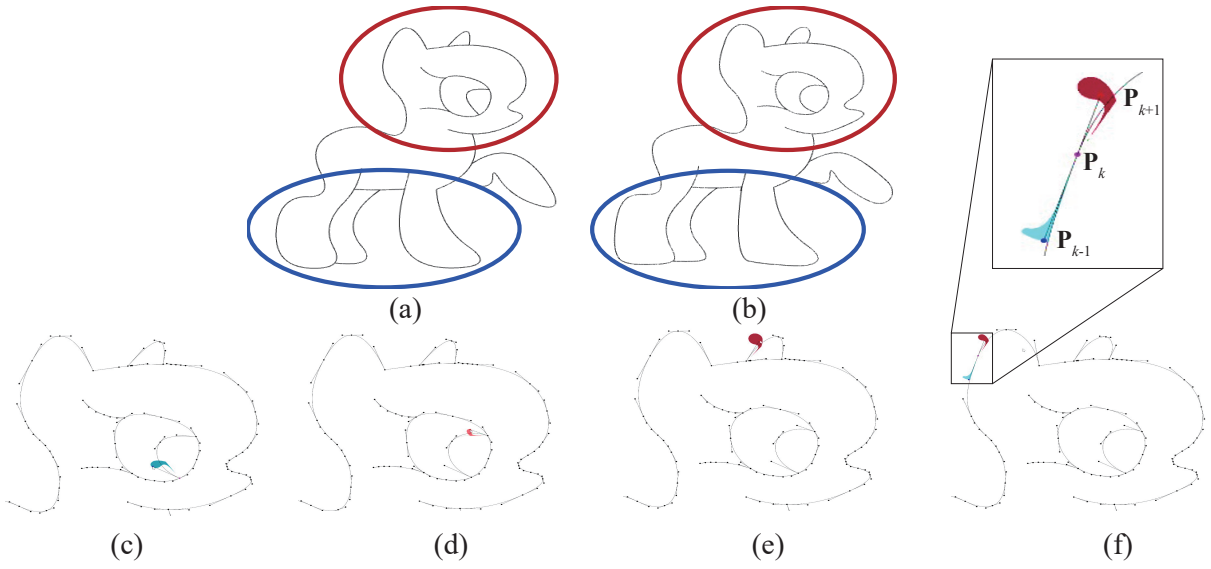


Figure 13: Locally modifying the shape (a) generated by κ-curves to (b) using the proposed approach without introducing another curvature extremum. (c) to (f) are intermediate processes.

curves. In higher degree polynomial curves (degree 9 or higher), visualizations of CMRWIs becomes significantly slower, possibly due to the large fragment shader file size and computational cost. A concise and efficient representation of the numerator of $\frac{d\kappa}{ds}$ in Bernstein form, particularly for rational curves, is desired. Our future work also includes theoretically clarifying how the shape of the curvature monotonicity region changes depending on the position of a control point.

Norimasa Yoshida, http://orcid.org/0000-0001-8889-0949
Seiya Sakurai, http://orcid.org/0000-0003-1171-0318
Hikaru Yasuda, http://orcid.org/0000-0002-3054-2623
Taisei Inoue, http://orcid.org/0000-0002-2215-6589
Takafumi Saito, http://orcid.org/0000-0001-5831-596X

REFERENCES

[1] Binninger, A.; Sorkine-Hornung, O.: Smooth interpolating curves with local control and monotone alternating curvature. Computer Graphics Forum, 41(5), 25–38, 2022. http://doi.org/10.1111/cgf.14600.

[2] Dietz, A.D.; Piper, B.: Interpolation with cubic spirals. Computer Aided Geometric Design, 21(2), 165–180, 2015. http://doi.org/10.1016/j.cagd.2003.09.002.

[3] Farin, G.: Curves and Surfaces for Computer Aided Geometric Design, 5th ed. Morgan Kafumann, 2001.

[4] Farin, G.: Class A Bézier curves. Computer Aided Geometric Design, 23(7), 573–581, 2006. http://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), 457–483, 2000. http://doi.org/10.1016/S0167-8396(00)00011-X.

[6] Harada, T.; Yoshimoto, F.; Moriyama, M.: An aesthetic curve in the field of industrial design. In IEEE Symposium on Visual Languages, 38–47, 1999. http://doi.org/10.1109/VL.1999.795873.

[7] Lu, L.; Xiang, X.: Quintic polynomial approximation of log-aesthetic curves by curvature deviation. Journal of Computational and Applied Mathematics, 296, 389–396, 2016. http://doi.org/10.1016/j.cam.2015.10.002.

[8] Mineur, Y.; Lichah, T.; Castelain, J.M.; Giaume, H.: A shape controled fitting method for bézier curves. Computer Aided Geometric Design, 15(9), 879–891, 1998. http://doi.org/10.1016/S0167-8396(98)00025-9.

[9] Miura, K.T.: A general equation of aesthetic curves and its self-affinity. Computer Aided Design & Applications, 3(1–4), 4570–464, 2006. http://doi.org/10.1080/16864360.2006.10738484.

[10] Miura, K.T.; Gobithaasan, R.U.; Salvi, P.; Wang, D.; Sekine, T.; Usuki, S.; Inoguchi, J.; Kajiwara, K.: $\epsilon\kappa$-curves: controlled local curvature extrema. The Visual Computer, 38, 2723–2738, 2022. http://doi.org/10.1007/s00371-021-02149-8.

[11] Ramshaw, L.: Blossoms are polar forms. Computer Aided Geometric Design, 6, 323–328, 1989. http://doi.org/10.1016/0167-8396(89)90032-0.

[12] Romani, L.; Viscardi, A.: Planar class A Bézier curves: The case of real eigenvalues. Computer Aided Geometric Design, 89, 2021. http://doi.org/10.1016/j.cagd.2021.102021.

[13] Sapidis, N.S.; Frey, W.H.: Controlling the curvature of a quadratic Bézier curve. Computer Aided Geometric Design, 9, 85–91, 1992. http://doi.org/10.1016/0167-8396(92)90008-D.

[14] Tong, W.; Chen, M.: A sufficient condition for 3D typical curves. Computer Aided Geometric Design, 87, 2021. http://doi.org/10.1016/j.cagd.2021.101991.

[15] Tsuchie, S.; Yoshida, N.: High-quality approximation of log-aesthetic curves based on the fourth order derivative. Journal of Computational Design And Engineering, 9(6), 2439–2451, 2022. http://doi.org/10.1093/jcde/qwac117.

[16] 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), 515–527, 2004. http://doi.org/10.1016/j.cagd.2004.04.001.

[17] Yan, Z.; Schiller, S.; Schaefer, S.: Circle reproduction with interpolatory curves at local maximal curvature points. Computer Aided Geometric Design, 72, 98–110, 2019. http://doi.org/10.1016/j.cagd.2019.06.002.

[18] Yan, Z.; Schiller, S.; Wilensky, G.; Carr, N.; Schaefer, S.: Interpolation at local maximum curvature. ACM Transaction on Graphic, 36(4), 1–7, 2017. http://doi.org/10.1145/3072959.3073692.

[19] Yoshida, N.; Saito, T.: Interactive aesthetic curve segments. The Visual Computer (Proc. of Pacific Graphics), 22(9–11), 896–905, 2006. http://doi.org/10.1007/s00371-006-0076-5.

[20] Yoshida, N.; Saito, T.: Quasi-aesthetic curves in rational cubic Bézier forms. Computer-Aided Design & Applications, 4(1–4), 477–486, 2007. http://doi.org/10.1080/16864360.2007.10738567.

[21] Yoshida, N.; Saito, T.: Shape information of curves and its visualization using two-tone pseudo coloring. Computer-Aided Design & Applications, 2023, accepted.

[22] Yuksel, C.: A class of c^2 interpolating splines. ACM Transactions on Graphics, 39(5), 1–14, 2020. http://doi.org/10.1145/3400301.