Storage-efficient reconstruction framework for planar contours

Hiroyuki Gotoa and Yoichi Shimakawab

aDepartment of Industrial and System Engineering, Hosei University, Tokyo, Japan; bDepartment of Computer Science and Technology, Salesian Polytechnique, Tokyo, Japan

ABSTRACT
A storage-efficient reconstruction framework for cartographic planar contours is developed. With a smaller number of control points, we aim to calculate the area and perimeter as well as to reconstruct a smooth curve. The input data forms an oriented contour, each control point of which consists of three values: the Cartesian coordinates \((x, y)\) and tangent angle \(\theta\). Two types of interpolation methods are developed, one of which is based on an arc spline while the other one is on a cubic Hermite spline. The arc spline-based method reconstructs a \(G^1\) continuous curve, with which the exact area and perimeter can be calculated. The benefit of using the Hermite spline-based method is that it can achieve \(G^2\) continuity on most control points and can obtain the exact area, whereas the resulting perimeter is approximate. In a numerical experiment for analytically defined curves, more accurate computation of the area and perimeter was achieved with a smaller number of control points. In another experiment using a digital elevation model data, the reconstructed contours were smoother than those by a conventional method.

1. Introduction
A storage-efficient framework for handling cartographic planar contours is developed in this article. With a smaller number of control points for a target contour, we aim to calculate the area and perimeter of the contour as well as to draw a smooth curve. The framework can also be positioned as a contour reconstruction method.

Contour reconstruction is an important task in various engineering fields such as structural analysis, image processing, geographical analysis, and so on. A contour is a curve along which a function associated with a physical quantity has an equal value. In the usage of geography, examples of the quantity include elevation, depth, atmospheric pressure, temperature, air moisture, and so on. Although we hereafter focus on applications to cartographic systems, the same framework can also be applied to other fields.

In general, a contour is reconstructed through two interpolation procedures: we first find control points on a target curve followed by connecting the points with a curve. In most practical cases, the control points are not given directly and estimated using simulated or observed values on discrete points. The physical value of an arbitrary point is calculated by giving an interpolation function; in handling a planar contour, we set \(z = f(x, y)\) and consider a contour specified by \(z = c\), where \(c\) is a target elevation. The most simple and commonly used function is a piecewise linear function for given elevations on a rectangular or triangular mesh, in which the resulting contour becomes a polygon and all control points can be calculated without an iterative calculation.

There are other popular functions such as bilinear and bicubic splines (Li, Zhu, and Gold 2004) to obtain a smooth curve. The resulting shape by bilinear spline is piecewise hyperbola. A disadvantage of this model is that the reconstructed contour breaks on the edges of a mesh, which is called an edge effect (Schneider 2005); the curve is \(G^0\) continuous but not \(G^1\) continuous. On the other hand, bicubic splines can reconstruct \(G^1\) continuous curves but needs an iterative calculation to find control points.

The second concern in contour reconstruction is how to connect control points with a certain curve, which requires another interpolation scheme. The simplest method is to adopt a polygonal approximation, in which the ordered control points are connected with straight segments. In this method, however, the resulting contour tends to be too angular and occasionally unnatural, which becomes notable if the contour is magnified or the observation density is low.

To overcome this flaw, there would mainly be two approaches. The first one is to increase the number of control points, which can be achieved by subdividing a mesh or by using a continuous interpolation function.
For example, McCullagh (1981) handled rectangular and triangular meshes, Dyn, Levin, and Liu (1992) proposed a method by generating polyhedrons in three-dimensional space, and Gousie and Franklin (2005) adopted a spline function whose coefficients are determined using the thin-plate technique (Briggs 1974). Chai, Miyoshi, and Nakamae (1998) constructed a method to generate intermediate points by solving a partial derivative equation derived from an analogy of electric field. Furthermore, Pradhan and Ghose (2013) proposed a creative method to make a dense mesh by mimicking a spider of weaving a web.

This type of approach, however, requires recalculation whenever the target resolution is changed. Hence, a compromise approach is to calculate control points by constructing a dense mesh followed by eliminating intermediate control points that do not significantly affect the shape. This procedure is occasionally called contour compression or decimation, of which a pioneering method is called the Ramer-Douglas-Peucker (RDP) algorithm (Ramer 1972; Douglas and Peucker 1973). There are also other methods derived from the RDP algorithm (Ukasha, Saffor, and Hassan 2008; Ukasha 2011).

The second approach is to interpolate the ordered control points with a smooth curve, also known as contour smoothing. With respect to smoothing curves or surfaces for given control points, it has been a popular topic in the field of computer graphics. For example, Lawonn et al. (2014) developed a smoothing method based on a reduction of geodesic curvature with an application example to surgical planning; Taubin (1995) introduced a method based on Gaussian smoothing that does not cause a shrinkage. The common situation in these researches is that the observed or simulated data are abundant but they include noises or errors.

By contrast, in cartographic systems, which are the main target of this article, the density of observed data is occasionally low due to technical and budget limitations. Thus, it is of use to develop a method for reconstructing a smooth curve with sparsely observed data. Adopting a spline-based framework can be popular and practical to fulfill this requirement. Li and Meek (2005) developed a method to achieve G2 continuity for arc splines by inserting relaxed curves. Thibault and Gold (2000) developed a method called skeleton retraction by which minor ridges or valleys can be reconstructed.

In addition to visual smoothness, this article puts an emphasis on calculating the area and length of reconstructed curves with high accuracy. The aim is to apply the method to civil engineering and disaster prevention; for example the resulting area would be useful in predicting the region damaged by a flood, and length in estimating necessary resource or required time between two points or for a cycle.

In view of these needs, the authors have developed contour detection algorithms called the inner skin algorithm (Goto and Shimakawa 2016, 2013). A feature of the algorithm is that each control point consists of three elements: the Cartesian coordinates (x,y) and tangent angle θ. The reconstructed curve forms an oriented contour, the area and perimeter of which can be calculated using Green’s theorem (Stroud and Booth 2005).

Goto and Shimakawa (2013) introduced a framework based on a first-order approximation, while Goto and Shimakawa (2016) managed to reduce the number of control points by a piecewise arc approximation. An oriented contour has another benefit by which the contour which is a part of a mountain or a basin can be judged; along with a visualization technique, a hill and punchbowl can be distinguished.

These works, however, do not pay attention to smoothing curves, and the reconstructed curves are G0 continuous. Therefore, in this article, we develop a framework for reconstructing smooth contours based on arc and cubic Hermite splines. As introduced above, there are many existing researches on smoothing curves, but to the authors’ best knowledge there is no research focusing on calculating geodetic quantities such as area and length. The arc spline-based method achieves G1 continuity and exact area and perimeter can be calculated by subdividing each portion into two arcs. By contrast, the cubic Hermite spline-based method can reconstruct G2 continuous curves and achieve G2 continuity on some points. The exact area can be computed without subdividing each portion, but the exact perimeter is approximate. Since an accurate computation of the area and length is achieved with a small number of control points, the proposed framework is storage-efficient.

In summary, the focuses of this article are threefold:

- Reconstruction: draw a smooth contour;
- Computation of area: calculate the area of a closed contour;
- Computation of perimeter: calculate the length of a contour.

2. Storage format and intended use

The commonly used storage format for handling planar contours is the ordered set of two-dimensional Cartesian coordinates in the following manner:

\[(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\]  \(1\)

where n is the number of control points. To handle a closed contour, it is convenient if we set \((x_n, y_n) = (x_1, y_1)\), wherein the number of independent control points is \(n-1\).

If we aim to draw a smooth contour or calculate feature quantities accurately, the control points are usually
If the contour is closed, we set $O$ of which is on point $x_i$.

Let $x_i$ be the target closed contour and the origin, the specific value of which is: $D_i$.

The perimeter or the length of a part of $\partial D$ can be calculated by accumulating the length elements of portion $S_i - S_{i+1}$ along $\partial D$. We denote the length of this portion by $L_i$. This length clearly differs between the interpolation methods.

For reconstruction, we consider drawing a smooth contour, the primary essence of which is to determine the coordinates of the intermediate control points on portion $x_i - x_{i+1}$ along $\partial D$. To achieve scaling to any resolution, we develop a method for which the number of intermediate points can be varied arbitrarily.

3. Arc spline

In previous study by Goto and Shimakawa (2016), a method to partition $\partial D$ based on roundness is proposed. The interpolation is performed by piecewise arc approximation, by which the exact area and length can be calculated. We can determine a unique arc by two control points $x_i, x_{i+1}$ and the tangent angle $\theta_i$.

3.1. Calculation and reconstruction of a $G^0$ continuous arc

Let the center and radius of a target arc be denoted by $c_i$ and $\rho_i$, respectively. If we put $s_i = x_{i+1} - x_i$ and $s_j = |s_i|$, in view of Figure 2, $2\rho_i \sin \theta_i = s_j$ follows. The area element $S_i$ is calculated by subtracting the area of triangle $c_i - x_i - x_{i+1}$ from sector $c_i - x_i - x_{i+1}$, which results in

$$S_i = \frac{\rho_i^2 (2\theta_i) - \rho_i^2 \sin 2\theta_i}{2} = \frac{s_j^2 (2\theta_i - \sin 2\theta_i)}{8 \sin^2 \theta_i} \quad (4)$$

The length $L_i$ is calculated by

$$L_i = \rho_i (2\theta_i) = s_i \theta_i / \sin \theta_i \quad (5)$$

If $|\theta| = 0$, we use the following approximations because the above formulae become indeterminate:

$$S_i = \frac{s_j^2 \theta_i (\theta_i / 6 + O(\theta_i^3))}{1 + \theta_i^2 + O(\theta_i^4)} \quad (6)$$

which are obtained by performing the Taylor expansion around $\theta = 0$.

We next draw a smooth arc by interpolation. In Figure 2, let $y, c$, and $\varphi$ be an arbitrary interpolation point, tangent vector on endpoint $x_i$, and angle from vector $y - x_i$ to $m_i$ in clockwise order, respectively. Angle $\varphi$ follows $0 \leq \varphi \leq \theta_i$.

![Figure 1](image1.png)

**Figure 1.** Relation between the difference vector and tangent angle.

![Figure 2](image2.png)

**Figure 2.** Two diametric points and center of the interpolated arc.
If we write $|m_i| = m_i$, then $m/m_i$ represents the unit tangent vector, and can be represented by the following two forms:

$$\frac{m_i}{m_i} = R(-\varphi)\left(\frac{y - x_i}{\rho_i \sin \varphi}\right) = R(-\theta)\left(\frac{x_{i+1} - x_i}{\rho_i \sin \varphi}\right)$$

where $R(\theta)$ represents the rotation matrix for $\theta$ in counter-clockwise. We then obtain $y$ as follows:

$$y = x_i + (\sin \varphi / \sin \theta)R(\varphi - \theta)s_i$$

Using this relation, the portion can be subdivided into any portions. One natural approach is to divide portion $x_i - x_{i+1}$ into $N$ pieces, for which the length of each piece becomes smaller than or equal to a given target length $s_M$. This length is given by considering the rotation of an output device. If we aim to determine $N$ strictly, the following relation should be used: $s_i(\theta_i / \sin \theta_i) / N \leq s_M$, which results in $N = [s_i / s_M(\theta_i / \sin \theta_i)]$. In practice, however, it is simple and would be sufficient to determine this using

$$N = \left\lceil \frac{s_i}{s_M} \right\rceil$$

which we shall also use in other interpolation methods.

### 3.2. Assurance of G¹ continuity on the endpoints

A benefit of using an arc spline is that the feature quantities such as area and length can be computed without subdividing portions. However, the contour is not G¹ continuous on the endpoints if we use the method in the previous subsection. We hence consider assuring G¹ continuity by subdividing each portion into two arcs.

Let $m_{i+1}$ and $\varphi_i$ be the tangent vector on $x_{i+1}$ and angle from the difference vector $s_i = x_{i+1} - x_i$ to $m_{i+1}$ in counter clockwise order, respectively. With the help of Figure 3, the unit vector $m_{i+1}/m_i$ on point $x_{i+1}$ holds the following relations:

$$m_{i+1}/m_i = R(-\varphi)(s_i/s_i) = R(-\theta)(s_i/s_i)$$

The relations between the second and third terms are expressed as

$$\frac{1}{s_i} \left[ s_i \right] \begin{bmatrix} \cos \varphi_i \\ \sin \varphi_i \end{bmatrix} = R(-\theta)\left(\frac{s_{i+1}}{s_{i+1}}\right)$$

The values of $\cos \varphi_i$ and $\sin \varphi_i$ are then calculated in accordance with

$$\left[ \begin{array}{c} \cos \varphi_i \\ \sin \varphi_i \end{array} \right] = \frac{1}{s_i} \left[ s_i \right] \left(s_i^{-1}R(-\theta)\right)\left(\frac{s_{i+1}}{s_{i+1}}\right)$$

The angle $\varphi_i$ is obtained by calculating the arc cosine of $\sin \varphi_i$.

If $\varphi_i < 0$, we assume that the portion is a C-shaped curve and there is not an inflection point. By contrast, if $\varphi_i > 0$, we assume that there is one inflection point on the portion for which an S-shaped curve is formed. We herein consider the two cases separately.

### 3.3. Subdivision of a C-shaped curve

To achieve G¹ continuity on the endpoints, we subdivide the portion into two arcs. Here we consider the case for $\varphi_i < 0$, where the portion is a C-shaped curve. Figure 4 depicts a case of $\varphi_i > 0$ with the relations of the two arcs and its pertinent indices, where we wrote $\varphi_i = -\varphi_i$. Let $\rho_1, \rho_2$ and $y$ be the radii of the arcs and intersection of the arcs, respectively. We determine $\rho_1$ and $\rho_2$, for which the tangents on $y$ become both parallel to $s_i$. Then the following relations hold considering the geometric properties:

$$\rho_1 \sin \theta + \rho_2 \sin \varphi = s_i, \quad \rho_1 \cos \theta - \rho_2 \cos \varphi = \rho_1 - \rho_2$$

which can be written by

$$\begin{bmatrix} \sin \theta & \sin \varphi \\ \cos \theta & 1 - \cos \varphi \end{bmatrix} \begin{bmatrix} \rho_1 \\ \rho_2 \end{bmatrix} = \begin{bmatrix} s_i \\ 0 \end{bmatrix}$$

Figure 3. Two tangent vectors and three angles relevant to three control points.

Figure 4. Subdivision of a C-shaped curve into two arcs.
Then the radii of the two arcs are obtained as

\[
\begin{align*}
\left( \frac{\rho_1}{\rho_2} \right) &= \frac{s_i}{\sin\theta + \sin\varphi - \sin(\theta + \varphi)} \left( \frac{1 - \cos\varphi}{1 - \cos\theta} \right) \\
&= \frac{s_i}{2\sin\frac{\theta + \varphi}{2}} \left( \frac{\sin\left(\frac{\varphi}{2}\right)}{\sin\left(\frac{\theta}{2}\right)} \right) \\
\end{align*}
\]

and the intersection \(y\) is calculated as follows:

\[
\begin{align*}
y &= x_i + 2\rho_1 \sin\left(\frac{\theta}{2}\right) \cdot R\left(\frac{\theta}{2}\right) \cdot \left( \frac{s_i}{s_i} \right) \\
&= x_i + \sin\left(\frac{\varphi}{2}\right) / \sin\left(\frac{\theta}{2}\right) \cdot R\left(\frac{\theta}{2}\right) s_i \\
&= x_i + \sin\left(\frac{-\varphi}{2}\right) / \sin\left(\frac{\theta - \varphi}{2}\right) \cdot R\left(\frac{\theta}{2}\right) s_i
\end{align*}
\]

where we replaced \(\theta\) and \(\varphi\) with \(\theta_i\) and \(-\varphi_i\), respectively, on the last line.

To simplify this expression, we introduce the following two variables \(\eta\) and \(\mu\):

\[
(\theta_i - \varphi_i)/2 = \eta, \theta_i/(\theta_i - \varphi_i) = \mu
\]

We then obtain \(-\varphi_i/2 = (1 - \mu)\eta\), and Equation (12) is rewritten as follows:

\[
y = x_i + \sin[(1 - \mu)\eta]/\sin\eta \cdot R((-\mu)\eta)s_i
\]

(14)

If \(\theta = \theta \rightarrow 0\) and \(\varphi = -\varphi \rightarrow 0\) hold, \(\eta \rightarrow 0\) follows and, sin \([(1 - \mu)\eta]/\sin \eta \) becomes indeterminate. We thus derive an approximate form of Equation (14) by Taylor expansion around \(\eta = 0\). The specific resulting form is

\[
y = x_i + [(1 - \mu) + (\mu(1 - \mu)(1 - 2\mu))/3\eta^2 + O(\eta^2)] \cdot s_i + [\mu(1 - \mu)\eta + O(\eta^2)]s_i
\]

(15)

With respect to the tangent angles, we first recall Figure 2, in which the tangent angle \(\theta_i\) on point \(x_i\) is the half of the central angle of the approximated arc. Then in view of Figure 4 again, the tangent angle on point \(x_i\) is \(\theta_i/2 = \theta_i/2\) and the angle on point \(y\) is \(\varphi_i/2 = -\varphi_i/2\) in clockwise order.

If \(\theta_i < 0\) and \(\varphi_i > 0\), the positional relation of the portion \(x_i \sim y \sim x_{i+1}\) is axisymmetric in Figure 4. If we put \(\theta = -\theta_i\) and \(\varphi = \varphi_i\), then the geometric relation becomes the same as Equation (10) and the radii are expressed by Equation (11).

On the other hand, the representation of intersection \(y\) is different from Equation (12), which results in

\[
y = x_i + 2\rho_1 \sin\left(\frac{\theta}{2}\right) \cdot R\left(\frac{\theta}{2}\right) \cdot \left( \frac{s_i}{s_i} \right) \\
= x_i + \frac{\sin\varphi/2}{\sin[(\theta + \varphi)/2]} \cdot R\left(\frac{\theta}{2}\right) s_i
\]

By substituting \(\theta\) and \(\varphi\) with \(-\theta_i\) and \(\varphi_i\), respectively, we eventually obtain the same results as Equations (12) and (14).

With respect to the tangent angles on points \(x_i\) and \(y\), the former angle is \(\theta_i/2 = -\theta_i/2\) in counter clockwise order, whereas the latter is \(\varphi_i/2 = \varphi_i/2\), which become the same angles in clockwise order as for \(\theta_i > 0\) and \(\varphi_i < 0\).

In summary for the above two cases, if \(\theta_i \cdot \varphi_i < 0\) holds, we assume that there is not an inflection point on the portion and subdivide it into two arcs. The intermediate point \(y\) is calculated using Equations (13) and (14), and the tangent angles on points \(x_i\) and \(y\) are

\[
\theta_i/2 \text{ and } -\varphi_i/2
\]

(16)

in clockwise order.

### 3.4. Subdivision of an S-shaped curve

The case for \(\theta_i \cdot \varphi_i > 0\) is now examined. This case has an inflection point on the portion which we subdivide into two arcs. Letting be the inflection point, we adjust the radii of the arcs on which the tangents are shared.

Figure 5 illustrates the case of \(\theta_i > 0\) and \(\varphi_i > 0\), where we expressed \(\theta = \theta_i\) and \(\varphi = \varphi_i\), simply. There is a degree of freedom in angle \(\delta\). Considering the geometric properties to vertical and horizontal directions, the following relations hold:

\[
\rho_1(\cos \theta - \cos \varphi) + \rho_2(\sin \varphi + \sin \delta) = s_i \\
\rho_1(\cos \theta - \cos \varphi) + \rho_2(\sin \varphi - \sin \delta) = 0
\]

which are expressed as

\[
\begin{pmatrix}
\sin \frac{\theta + \delta}{2} \\
-\sin \frac{\theta - \delta}{2}
\end{pmatrix}
\begin{pmatrix}
\rho_1 \\
\rho_2
\end{pmatrix}
= \begin{pmatrix}
s_i/2 \\
0
\end{pmatrix}
\]

Then the radii \(\rho_1\) and \(\rho_2\) are calculated according to

\[
\begin{pmatrix}
\rho_1 \\
\rho_2
\end{pmatrix}
= \frac{s_i}{2\sin \frac{\theta + \delta}{2}} \begin{pmatrix}
-\sin \frac{\theta - \delta}{2} \\
\sin \frac{\theta + \delta}{2}
\end{pmatrix}
\]

---

**Figure 5.** Partitioning of an S-shaped curve into two arcs.
The coordinate of the inflection point \( y \) is then calculated as follows:

\[
y = x_i + \rho_i (\sin \theta + \sin \delta)(s_i / s_i)
\]

\[
+ \rho_i (\cos \theta - \cos \delta) \cdot R(x_i / 2)(s_i / s_i)
\]

\[
= x_i + \frac{\sin(\varphi - \delta')/2}{\sin(\varphi - \theta)/2} \cdot \cos \theta - \cos \delta \cdot s_i + \sin \theta - \sin \delta \cdot \bar{s}_i
\]  \hspace{1cm} (18)

With respect to angle \( \delta \), if \( \theta < \varphi \) holds, point \( y \) must be on the counter-clockwise side with reference to segment \( x_i \sim \bar{x}_{i+1} \) as shown in Figure 5. Considering the central angles of the two arcs, \( \delta \) must hold \( \theta < \delta < \varphi \), but there is a degree of freedom as long as \( \delta \) holds this relation. One candidate is to take a certain average of \( \theta \) and \( \varphi \). From the results of several inspections, we propose to use the harmonic average given below:

\[
\delta = 2 \cdot \varphi \theta / (\varphi + \theta)
\]  \hspace{1cm} (19)

the reason for which is mentioned below.

To simplify Equation (18), we introduce the following two variables \( \mu \) and \( \eta \):

\[
(\theta - \varphi)/2 = \eta, \theta / (\theta + \varphi) = \mu
\]  \hspace{1cm} (20)

Then \( \varphi - \delta)/2 \) and \( \delta - \theta)/2 \) in Equation (18) are represented as \( (\varphi - \delta)/2 = -(1 - \mu)\eta \) and \( (\theta - \delta)/2 = \mu \eta \), and Equation (18) is simplified to

\[
y = x_i + \sin[(1 - \mu)\eta]/ \sin \eta \cdot [\cos \mu \eta \cdot \bar{s}_i + \sin \mu \eta \cdot \bar{s}_i]
\]  \hspace{1cm} (21)

Since this form is exactly the same as Equation (14), we can use the approximation form Equation (15) if \( \eta \to 0 \) i.e. \( \varphi \to 0 \).

If \( \theta \to 0 \), then \( \eta \to -\varphi/2 \) and \( \mu \to 0 \) follow in both Equations (13) and (20) but the signs of \( \mu \) are the opposite of each other. Thus, the positions of \( y \) for Eqs. (14) and (21) are symmetric across segment \( x_i \sim \bar{x}_{i+1} \). This property is desirable for smaller tangent angles, and this is the reason for using the harmonic average.

In considering the tangent angles, we recall again that the tangent angle on a point is the half of the central angle. Then in view of Figure 5, the tangent angle on point \( x_i \) is

\[
(\theta + \delta)/2 = (\theta - \delta)/2 \cdot \theta = \theta - \mu \cdot \eta
\]  \hspace{1cm} (22)

in clockwise order and the angle on point \( y \) is

\[
-(\theta + \delta)/2 = (\varphi - \delta)/2 - \varphi = -\varphi - (1 - \mu)\eta
\]  \hspace{1cm} (23)

If \( \theta > \varphi \) i.e. \( \theta_i - \varphi_i > 0 \), the same geometric properties as Equations (17) and (18) hold, but point \( y \) is on the clockwise side of segment \( x_i \sim \bar{x}_{i+1} \) and \( \delta \) must hold \( \varphi < \delta < \theta \). If we set \( \delta \) to the same as Equation (19), we obtain the same result as Equation (21).

In the case of \( \theta_i < 0 \) and \( \varphi_i < 0 \), the positional relations of the relevant points and vectors are laterally inverted to those depicted in Figure 5. If we put \( \theta = -\theta \) and \( \varphi = -\varphi \), the same geometric properties as Equation (17) hold, and the inflection point \( y \) is then calculated as

\[
y = x_i + \frac{\varphi - \delta' - \varphi - \theta}{2} \cdot \sin \frac{\varphi - \theta}{2} \cdot \cos \frac{\theta - \delta'}{2} \cdot \bar{s}_i
\]  \hspace{1cm} (24)

where we appended the prime symbol to \( \delta \) to distinguish from that of Equation (18), noting the sign of the coefficients of \( \bar{s}_i \) is the opposite.

If \( \theta < \varphi \) i.e. \( \theta_i - \varphi_i > 0 \) holds, point \( y \) is on the clockwise side with reference to segment \( x_i \sim \bar{x}_{i+1} \) and \( \delta \) must hold \( \theta < \delta < \varphi \). We thus set \( \delta' \) to

\[
\delta' = 2 \cdot \frac{\varphi}{\varphi + \theta} = -2 \cdot \frac{\varphi \varphi}{\varphi + \theta} = -\delta
\]  \hspace{1cm} (25)

By substituting \( \delta' = -\delta, \theta = -\theta_i \) and \( \varphi = -\varphi_i \) into Equation (24), we obtain the following simplified expression

\[
y = x_i + \frac{\sin(-\varphi + \delta)/2}{\sin(-\varphi + \theta)/2} \cdot \cos \frac{-\theta + \delta}{2} \cdot s_i - \sin \frac{-\theta + \delta}{2} \cdot \bar{s}_i
\]

which results in the same as Equation (18). We can thus use Equation (21) to calculate \( y \).

Regarding the tangent angles, the angles on points \( x_i \) and \( y \) in counter clockwise order are \( (\theta + \delta)/2 = -(\theta + \delta)/2, -(\varphi + \delta)/2 = (\varphi + \delta)/2 \), which are eventually the same as Equations (22) and (23).

If \( \theta > \varphi \) i.e. \( \theta_i - \varphi_i < 0 \), point \( y \) is on the counter clockwise side of segment \( x_i \sim \bar{x}_{i+1} \), and we set \( \delta' \) to the same as Equation (25). We can thus use Equation (21) to calculate \( y \), and Equations (22) and (23) for the tangent angles.

In summary for the above four cases, if \( \theta_i \cdot \varphi_i > 0 \) holds, we assume there is one inflection point on the portion. The coordinate of the intermediate point \( y \) is calculated according to Equations (20) and (21), and the tangent angles on points \( x_i \) and \( y \) are Equations (22) and (23), respectively.

### 3.5. Overall flow

The overall flow of the arc spline interpolation is confirmed. Table 1 presents a pseudo code of the proposed algorithm, the input of which is the set of Cartesian coordinates and tangent angle \( (x_i, y_i, \theta_i)(i = 1, \cdots, n) \). Algorithm 1 is designed to subdivide each portion into two arcs to make all endpoints \( G^3 \) continuous. The generic calls “Calc.” are the abbreviation of “Calculate.” Function 1 calculates the area and length elements for a single arc, followed subdividing the arc into multiple broken line segments. The area and length element can be calculated irrespective of the number of divisions \( N \).
Cubic Hermite spline

Cubic Hermite spline, also referred to as Cubic Hermite interpolation, is widely used in computer graphics to draw a smooth curve for given control points. The curve is determined uniquely by giving two endpoints \( x_i \) and \( x_{i+1} \) and tangent vectors \( m_i \) and \( m_{i+1} \) on the points.

### 4.1. Interpolant of a portion

An arbitrary point on the curve between portion \( x_i \sim x_{i+1} \), denoted by \( p_r \), is represented in the following manner:

\[
p(t) = h_3(t)x_i + h_2(t)x_{i+1} + h_1(t)m_i + h_0(t)m_{i+1}
\]

(26)

where \( t \) is a continuous \([0, 1]\) parameter. Functions \( h_i(t) \) \((i = 1, 2, 3, 4)\) are cubic functions of \( t \) referred to as basis function, the coefficients of which are determined by giving the following boundary conditions:

\[
p(0) = x_i, \quad p(1) = x_{i+1}, \quad p'(0) = m_i, \quad p'(1) = m_{i+1}
\]

(27)

For later convenience, we derive a matrix representation of Equation (26) here. First, the following symbols are introduced:

\[
X = \begin{bmatrix} x_i & x_{i+1} & m_i & m_{i+1} \end{bmatrix}, \quad u(t) = \begin{bmatrix} 1 \ t \ t^2 \ t^3 \end{bmatrix}^T, \quad v(t) = \begin{bmatrix} 0 \ 2t \ 3t^2 \ 4t^3 \end{bmatrix}^T
\]

\[
du(t)/dt = \begin{bmatrix} 0 \ 2 \ 6t \ 12t^2 \end{bmatrix}^T \equiv w(t),
\]

\[
\begin{bmatrix} h_3(t) & h_2(t) & h_1(t) & h_0(t) \end{bmatrix}^T = Hu(t)
\]

and the suffix \( i \) of \( p_i(t) \) will be dropped when no confusion is likely to arise. Then, \( p(t) \) and its derivative for \( t \) can be represented as follows:

\[
p(t) = XHu(t), \quad p'(t) = XHv(t)
\]

To determine the coefficients of \( H \), we set \( t = 0 \) and \( t = 1 \) to take into account the conditions in Equation (27):

\[
p(0) = XHu(0) = x_i = X[1 \ 0 \ 0 \ 0]^T,
\]

\[
p(1) = XHu(1) = x_{i+1} = X[0 \ 1 \ 0 \ 0]^T,
\]

\[
p'(0) = XHv(0) = m_i = X[0 \ 0 \ 1 \ 0]^T,
\]

\[
p'(1) = XHv(1) = m_{i+1} = X[0 \ 0 \ 0 \ 1]^T.
\]

By putting,

\[
U = [u(0) \ u(1) \ v(0) \ v(1)] = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 3 \end{bmatrix}
\]

the above conditions are summarized as \( XHU = XI = X \). Since this is followed for arbitrary \( X \), matrix \( H \) must satisfy

\[
H = U^{-1} = \begin{bmatrix} 1 & 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}
\]

Accordingly, the interpolant \( p(t) \) is then represented as follows:

\[
p(t) = XHu(t) = \begin{bmatrix} 1 \ 0 & -3 & 2 \\ 0 & 0 & 3 & -2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix}
\]
4.2. Calculation of the area and length

We consider a method to calculate the area element $S_i$ surrounded by straight segment $x_i \sim x_{i+1}$ and its interpolated curve. Noting the displacement of $p(t)$ for an infinitesimal time $dt$ is $p'(t)$, $S_i$ is calculated in the following manner in reference to Equation (3):

$$S_i = \int \left( \frac{1}{2} \right) p(t) \times p'(t) dt$$  \hspace{1cm} (28)

For simplicity, we translate $x_i$ to the origin and set $x_i = 0$ and $x_{i+1} = s_i$. Then, $p(t)$ and its first and second derivatives are simplified into

$$[p(t) \ p'(t) \ p''(t)] = XH[u(t) \ v(t) \ w(t)]$$

where we simply wrote $u_i = [Hu(t)]$ and $v_i = [Hv(t)]$. After performing integrations of $(u_j v_j - u_j v_j)$ on the right-hand side, we obtain

$$\int_0^1 (u_j v_j - u_j v_j) dt = \int_0^1 (u_j v_j - u_j v_j) dt = -\frac{1}{5},$$

Then, the area element $S_i$ can be calculated by

$$S_i = \frac{s_i \times m_i - m_i \times m_{i+1} - m_{i+1} \times s_i}{10}$$

$$= \frac{s_i \times (m_{i+1} - m_i)}{60} + \frac{m_{i+1} \times m_i}{60}$$  \hspace{1cm} (30)

The integrand in Equation (28) is then represented by

$$\left[ p(t) \times p'(t) \right] = \frac{3t^2 - 2t^3}{t - 2t^2 + t^3} \left[ XH[u(t) \ v(t) \ w(t)] \right]$$

where $\lambda_i = \frac{1}{6}(m_{i+1} - m_i)$.

4.3. Determination of tangent vectors

In performing a cubic Hermite interpolation, it is necessary to determine the two tangent vectors $m_i$ and $m_{i+1}$ in Equation (26). If the original data is given by Cartesian coordinates such as Equation (1), the tangent vectors must be determined only with $x_i(1 \leq i \leq n)$. Several methods have been proposed thus far, a simple and commonly used one of which is referred to as the cardinal spline (Chui 1988). We provide a tangent vector on $x_i$ by

$$m_i = (1 - c) \frac{x_{i+1} - x_i}{2}$$  \hspace{1cm} (32)

where $c(0 \leq c \leq 1)$ is referred to as the shape or tension parameter. If $c = 0$, this method is called the Catmull-Rom spline (Catmull and Rom 1974). On the other hand, the focus of this research is to use the storage format proposed in Equation (2). This indicates that the direction of the tangent is fixed on every control point, whereas there is a degree of freedom in the length.

Here we express the tangent vector $m_i$ with two elements as follows:

$$m_i = m_i e_i, \ e_i = R(-\theta_i)s_i/s_i$$  \hspace{1cm} (33)

where $m_i$ and $e_i$ represent the length and unit tangent vector of $m_i$, respectively. Once $s_i$ and $\theta_i$ are given, $e_i$ is determined uniquely while there is a degree of freedom in determining $m_i(> 0)$. One straightforward variant is to take the length of $m_i$ in Equation (32) in the following manner:

$$m_i = (1 - c) \frac{|x_{i+1} - x_{i-1}|}{2}$$  \hspace{1cm} (34)

If $c$ is close to 1, the tension is high, $m_i$ is small, and the portion $x_i \sim x_{i+1}$ is approximately straight. By contrast, the portion becomes roundish if $c$ is close to 0. The difference between using only Equation (32) and using Equations (33) and (34) lies in the direction on point $x_i$, a demonstrative example of which will be presented in Section 5.

4.4. Smoothing the reconstructed curve

A benefit of using the cubic Hermite spline is that $G^1$ continuity is satisfied without subdividing a portion. On the other hand, there is a drawback in calculating the length element $L_i$ because Equation (31) is an approximate value and not an exact one.
To exploit another advantage, we consider smoothing the reconstructed curve for which $G^2$ continuity is partially achieved. Let $p_i(t)$ and $k_i(t) (0 \leq i \leq 1)$ be the interpolant of the parametric curve and signed curvature of the $i$th portion, respectively. If $k_i(1) = k_{i+1}(0)$ holds, the curve is $G^2$ continuous on point $x_{i+1}$. This can be achieved by adjusting the lengths of the two tangent vectors $m_i$ and $m_{i+1}$.

On a point determined by parameter $t$, the signed curvature $k_i(t)$ is calculated based on

$$k_i(t) = \frac{p_i'(t) \times p_i''(t)}{|p_i'(t)|^3}\tag{35}$$

where the sign is positive if the oriented curve moves in the counter-clockwise direction. Recalling Equation (29), the first and second derivatives of $p_i(t)$ on the two end-points are calculated as follows:

$$p_i'(0) = m_i = m_i e_i, \quad p_i'(1) = m_{i+1} = m_{i+1} e_{i+1}\tag{36}$$

where we denoted $m_i = m_i e_i$. By substituting these into Equation (35), we obtain the following expressions on the two end-points:

$$k_i(0) = \left[6(e_i \times s_i) - 2(e_i \times e_{i+1}) m_{i+1}\right] / m_i^2, \quad k_i(1) = \left[6(s_i \times e_{i+1}) - 2(e_i \times e_{i+1}) m_i\right] / m_i^2\tag{36}$$

These two equations can be reduced to a quartic equation for $m_i$ by eliminating $m_{i+1}$. The condition for the existence of positive solutions $m_i$ and $m_{i+1}$ depends on the relations between $e_i \times s_i, s_i \times e_{i+1}, e_i \times e_{i+1}, k_i(0)$, and $k_i(1)$, which is quite hard to discuss for general properties. We thus adopt a policy in which we try to smooth the curve only if there may be positive solutions $m_i$ and $m_{i+1}$ that satisfy Equation (36).

### 4.5. Smoothing Algorithm

An algorithm for smoothing the spline curve is developed. We initially set the length of tangent vector for each control point $x_i (0 \leq i \leq n)$ in accordance with

$$m_i = \left(1 - c\right) \frac{|x_{i+1} - x_i|}{2}\tag{37}$$

where $c(0 \leq c \leq 1)$ is the tension parameter. The tangent vector $m_i$ is then calculated with Equation (33).

Focusing on portions $x_{i-1} \sim x_i \sim x_{i+1}$ once the three tangent vectors $m_{i-1}, m_i$, and $m_{i+1}$ on the control points are given, four curvatures $k_{i-1}(0), k_{i-1}(1), k_i(0)$ and $k_i(1)$ are calculated using Equation (36). We then set a target curvature on $x_i$, which we shall denote by $k_i(1)$, by the average curvatures of the two adjacent portions as follows:

$$k_i = [k_{i-1}(1) + k_i(0)]/2\tag{38}$$

Figure 6 will help in comprehending this concept, and this setting is performed for all $(2 \leq i \leq n - 1)$. If the contour is closed, i.e. $x_i = x_n$, follows, we set $k_i = k_i(0)$ whereas $k_i = k_i(1)$ if the contour is open.

After the target curvatures are determined, the lengths of the tangent vectors are adjusted to fit the curvatures to the target values. We update $m_i$ and $m_{i+1}$ based on $m + \delta m, m = (m_i m_{i+1})^T$ where the difference $\delta m$ is determined by solving the simultaneous equations:

$$\delta k = J \delta m \tag{39}$$

where

$$\delta k = \begin{pmatrix} k_i(0) - k_i(1) \\ k_{i+1}(0) - k_{i+1}(1) \end{pmatrix}, \quad J = -2 \begin{pmatrix} (e_i \times e_{i+1}) / m_i^2 & (e_i \times e_{i+1}) / m_{i+1}^2 \\ (e_i \times e_{i+1}) / m_{i+1}^2 & (e_i \times e_{i+1}) / m_i^2 \end{pmatrix}\tag{40}$$

This is derived by considering tiny variations of $k_i(0)$ and $k_i(1)$ in Equation (36). We repeat updating $m$ until $|\delta m| < \epsilon_i$ is satisfied or the number of iterations reaches a certain count, where $\epsilon_i$ is a small positive constant.

Should one of the following conditions hold during the update, we discontinue updating and restore the initial values:

- $m_i \leq 0$ or $m_{i+1} \leq 0$
- $\text{cond}(J) > 1/\epsilon_i$
- $k_i(0) \cdot k_i \geq 0$ and $k_i(0) \cdot (e_i \times s_i) < 0$;
- $k_i(1) \cdot k_i(1) \geq 0$ and $k_i(1) \cdot (s_i \times e_{i+1}) < 0$;
- $k_i(0) \cdot k_i(1) < 0, k_i(0) \cdot k_i(0)(e_i \times s_i) < 0$ and $k_i(1) \cdot (s_i \times e_{i+1}) < 0$

where $\text{cond}(J)$ and $\epsilon_i$ are the condition number of matrix and a small positive constant. The first criterion is to avoid orienting the tangent vector in the opposite direction, while the second one is the condition for the existence of a solution of Equation (39). The third to fifth criteria arise from the property of cubic functions; a cubic function cannot have more than one inflection point. We thus exclude cases in which there must be at least two inflection points between two adjacent control points as shown in Figures 7 and 8. Figure 7 illustrates four cases with equal sign curvatures on the two endpoints. The upper-left case is an example with positive curvatures on points $x_i$ and $x_{i+1}$, in which $e_i$ is directed counter-clockwise with reference to $s_i$. Figure 8 presents two cases with opposite signed curvatures, in which both
must be specified in addition to the same input as the arc spline interpolation. Algorithm 2 first calculates the current and target curvatures for each control point, followed by adjustment of the lengths of tangent vectors on the two endpoints. Function 2 interpolates a single portion with the tangent vectors on two endpoints specified. The function first calculates the area element, which can be obtained regardless of the number of divisions $N$. By contrast, the resulting length element is calculated by accumulating the lengths of the divided segments.

5. Numerical experiments

Two types of numerical experiments are performed. The first experiment compares the shapes of interpolated curves, while the second one inspects the relation between computation accuracy and storage efficiency. We use a PC equipped with an Intel® Core™ i5-5250U 1.60 GHz running Windows 7 64-bit. The programs are coded for MATLAB R2015b.

5.1. Comparison of the shapes

A simple closed contour featured by four control points is handled. In accordance with the storage format in Equation (2), we give the following data-set:

$$(0, 0, \theta), (3, 10, \theta), (8, 15, \theta), (0, 20, \theta), (0, 0, \theta),$$

$e_i$ and $e_{i+1}$ are directed to the same direction in reference to $s_i$.

If one of the above cases arises during the update of $m$, we give up accomplishing $G^2$ continuity on the endpoints and restore the initial values of $m_i$ and $m_{i+1}$ determined by Equation (37).

4.6. Overall flow

The overall algorithm of the cubic Hermite spline interpolation is summarized. Table 2 presents a pseudo code for the proposed framework. The tension parameter $c$

Table 2. Pseudo code for the cubic Hermite spline interpolation.

| Algorithm 2: Contour smoothing |
|--------------------------------|
| **Input:** $(x_i, y_i, \theta_i)$ ($i = 1 \ldots n$), $c$ \hspace{1cm} (Eq. 2, 32) |
| 01 Loop $i$ for control points |
| 02 Calc. initial length $m_i$ \hspace{1cm} (Eq. 34) |
| 03 Calc. curvatures $\kappa_i(0), \kappa_i(1)$ \hspace{1cm} (Eqs. 33, 36) |
| 04 Calc. target curvatures $k_i$ \hspace{1cm} (Eq. 38) |
| 05 Loop $i$ for control points |
| 06 Loop for convergence |
| 07 Calc. Jacobi matrix $J$ \hspace{1cm} (Eq. 40) |
| 08 If $J$ is ill-conditioned, restore $\kappa_i$ & Exit Loop \hspace{1cm} (Eq. 41) |
| 09 Calc. $\Delta m$ & Update $m$ \hspace{1cm} (Eq. 39) |
| 10 Calc. curvatures $\kappa_i(0), \kappa_i(1)$ \hspace{1cm} (Eqs. 33, 36) |
| 11 If $\kappa_i(0)$ or $\kappa_i(1)$ is invalid, restore $\kappa_i$ & Exit Loop \hspace{1cm} (Eq. 41) |
| 12 If $|\Delta m| < e_0$, Exit Loop |

| Function 2: Cubic Hermite spline interpolation |
|-----------------------------------------------|
| **Input:** $(x_i, y_i, \theta_i), (x_{i+1}, y_{i+1}, \theta_{i+1}), m_i, m_{i+1}, S_M$ \hspace{1cm} (Eqs. 30, 31) |
| 01 Calc. $S_0 + S_1$ \hspace{1cm} (Eqs. 30, 31) |
| 02 Calc. $N$ \hspace{1cm} (Eq. 8) |
| 03 Loop $t$ for divided segments ($0 \leq t \leq 1$) |
| 04 Calc. intermediate point $p(t)$ \hspace{1cm} (Eq. 29) |
| 05 Calc. $\Delta L_f$ & accumulate \hspace{1cm} (Eq. 31) |

Figure 9 depicts the results of reconstruction by four methods: (a) polygonal approximation, (b) cardinal spline, (c, d) arc spline, and (e, f) cubic Hermite spline.
Figure 9. Comparison of the reconstructed contours by four methods.

The middle and bottom figures on the left side depict the case for $\theta = -\pi/20$, whereas for $\theta = \pi/10$ in the right figures. The thick arrows originate from each control point to represent the tangent vectors. The arrows in cases (b), (c), and (d) are scaled to twofold while cases (e) and (f) to the quarter of $m$. In case (a), every pair of two adjacent control points is connected with a straight segment. Case (b) shows the result of the cardinal spline with tension parameter $c = 0.1$, and the direction of the tangent vectors are determined using Equation (32) which is irrespective of $\theta$. This yields that the direction of the arrows are different from those for cases (c)–(f).

In cases (c) and (d), four intermediate points are added between two adjacent control points by subdivision. In cases (e) and (f), the smoothing algorithm explained in Section 4.5, is applied in which a tension parameter $c = 0.1$ is used for an initial length.

In cases (c) and (e), all tangent vectors are directed to the inner side of the contour, while directed to the outer side in cases (d) and (f). The difference of the shapes between cases (c) and (d) is more remarkable than that of between cases (e) and (f); case (c) has a “slim” shape but case (d) is “plump.” The notable difference between cases (e) and (f) lies in the lengths of the tangent vectors. In cases where there is a steep curve and there must be an inflection point between two control points, the tangent vectors would be expanded to achieve $G^2$ continuity.

As the above results imply, various shapes of contours can be reconstructed by appending a tangent angle to a usual Cartesian coordinate.

### 5.2. Computation accuracy and storage efficiency

The relationship between the efficiency of the storage format and computation accuracy is inspected. We examine the following two closed curves:

- Ellipse: $(a^2 + b^2)(x^2 + y^2) + 2(b^2 - a^2)xy - 2a^2b^2 = 0$,  
- Cassini oval: $(x^2 + y^2)^2 - 4a^2xy + (a^4 - b^4) = 0$.

The areas of the ellipse and Cassini oval are $\pi ab$ and $2b^2E(k)$, respectively, where $k = (a/b)^2$ and $E(k)$ is the complete elliptic integral of the second kind. The parameters of the curves are $4aE(\sqrt{1-k})$ and $(\pi b/2)/\sqrt{1+k} \cdot F[1/4,1/2,1;4k/(1+k)]$, where $F[a,b;\gamma;z]$ is the Gauss hypergeometric function.

We now set $(a, b) = (2, 1/2)$ for the ellipse and $(1, 1/0.999)$ for the Cassini oval, the profiles of which are depicted in Figure 10. We encode the curves in the form of Equation (2) using a piecewise arc algorithm (Goto and Shimakawa 2016) with varying roundness parameters $\mu = 0.2, 0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001$. In calculating the perimeter of the Hermite spline, we set the number of divisions in Equation (31) to $N = ||p(1) - p(0)||/\gamma_M, \gamma_M = 10^{-3}$. For comparison, the following two existing methods are performed for varying resolutions:

- Marching Square (MS) (Lopes and Brodlie 2003; Maple 2003), with varying lattice intervals $l = \{5 \cdot 10^{-2}, 2 \cdot 10^{-2}, 1 \cdot 10^{-2}, 5 \cdot 10^{-3}, 2 \cdot 10^{-3}, 1 \cdot 10^{-4}, 5 \cdot 10^{-4}, 2 \cdot 10^{-4}\}$

- RDP algorithm (Ramer 1972; Douglas and Peucker 1973), with latticeinterval$=2 \cdot 10^{-4}$ and varyingtolerances $d = \{10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$

MS is a contour detection algorithm for given elevations on a regular rectangular lattice; a contour is formed by connecting edges obtained for each patch.
number of control points increases 10-fold, the error level is decreased approximately $10^{-4}$-fold. The computation errors for the perimeter and area are thus estimated by $O\left(\frac{1}{n^4}\right)$. By contrast, the slopes of the MS and MS + RDP are approximately the same; the error is decreased approximately $10^{-2}$-fold if the number of control points is increased 10-fold. Thus, the computation errors are estimated by $O\left(\frac{1}{n^2}\right)$. These orders are the same for both the perimeter and area.

Furthermore, we measure the computation times. The time for each trial is measured using "tic" and "toc" functions, and the average time of 100 executions is adopted. Figure 13 presents the computation times for interpolating the curves including the calculation of the perimeter and area. The horizontal axis represents the number of control points $n$, while the vertical one gives the computation time in seconds. Each time approximately increases in proportion to $n$, which can be presumed in view of Tables 1 and 2.

The four values on the upper right represent the total number of interpolated points $N$ including the original control points. For a fixed $n$, though $N$ for the arc spline is two- to threefold compared with the Hermite spline, the arc spline is more lightweight than the cubic Hermite spline. This would be because the arc spline does not need a convergence calculation for adjusting the lengths of tangent vectors.

For a fixed error level, if a certain profile is located to the left of another one, it follows that the method is more efficient in storage than the other one. Moreover, if the negative slope of a profile is steeper than another one, the method is more beneficial in terms of space complexity than the other one.

In view of Figures 11 and 12, the arc splines significantly excel the other three methods in both storage efficiency and space complexity. With respect to the cubic Hermite spline, the errors for the Cassini oval are close to the MS + RDP method, while those for the ellipse are beneficial to some extent. The MS + RDP method excels the MS in most cases, which would because the control points are generated from the MS with the smallest lattice interval. In addition, the errors of the MS + RDP converge on a level close to the MS in $n > 10^4$.

In terms of space complexity, the arc spline yields the best performance among the four methods. If the RDP algorithm is a contour compression technique by which the number of control points can be reduced; for two fixed control points, it is based on a straight line that passes the points. If the distances between intermediate points and the line are all under a given tolerance $d$, we remove the intermediate points. Since the RDP is designed to compress open curves, we split a closed contour at a medium control point and divide it into two series.

Figures 11 and 12 compare the computation accuracies for four reconstructing methods: arc spline, cubic Hermite spline, MS, and the combination of MS and RDP (MS + RDP). The horizontal and vertical axes represent the logarithm of the computation error and the number of control points $n$, respectively. Figure 11 represents the errors for the Cassini oval, while Figure 12 presents those for the ellipse. The upper figures represent the errors for the perimeter, while the lower ones for the area. In the MS + RDP method, we first calculate intersections with MS for a fine lattice $l = 2 \cdot 10^{-4}$, followed by a reduction of the number of control points using RDP with varying tolerance $d$.

For a fixed error level, if a certain profile is located to the left of another one, it follows that the method is more efficient in storage than the other one. Moreover, if the negative slope of a profile is steeper than another one, the method is more beneficial in terms of space complexity than the other one.

In view of Figures 11 and 12, the arc splines significantly excel the other three methods in both storage efficiency and space complexity. With respect to the cubic Hermite spline, the errors for the Cassini oval are close to the MS + RDP method, while those for the ellipse are beneficial to some extent. The MS + RDP method excels the MS in most cases, which would because the control points are generated from the MS with the smallest lattice interval. In addition, the errors of the MS + RDP converge on a level close to the MS in $n > 10^4$.

In terms of space complexity, the arc spline yields the best performance among the four methods. If the number of control points increases 10-fold, the error level is decreased approximately $10^{-4}$-fold. The computation errors for the perimeter and area are thus estimated by $O(1/n^4)$. By contrast, the slopes of the MS and MS + RDP are approximately the same; the error is decreased approximately $10^{-2}$-fold if the number of control points is increased 10-fold. Thus, the computation errors are estimated by $O(1/n^2)$. These orders are the same for both the perimeter and area.

Furthermore, we measure the computation times. The time for each trial is measured using "tic" and "toc" functions, and the average time of 100 executions is adopted. Figure 13 presents the computation times for interpolating the curves including the calculation of the perimeter and area. The horizontal axis represents the number of control points $n$, while the vertical one gives the computation time in seconds. Each time approximately increases in proportion to $n$, which can be presumed in view of Tables 1 and 2.

In view of Figures 11 and 12, the arc splines significantly excel the other three methods in both storage efficiency and space complexity. With respect to the cubic Hermite spline, the errors for the Cassini oval are close to the MS + RDP method, while those for the ellipse are beneficial to some extent. The MS + RDP method excels the MS in most cases, which would because the control points are generated from the MS with the smallest lattice interval. In addition, the errors of the MS + RDP converge on a level close to the MS in $n > 10^4$.

In terms of space complexity, the arc spline yields the best performance among the four methods. If the number of control points increases 10-fold, the error level is decreased approximately $10^{-4}$-fold. The computation errors for the perimeter and area are thus estimated by $O(1/n^4)$. By contrast, the slopes of the MS and MS + RDP are approximately the same; the error is decreased approximately $10^{-2}$-fold if the number of control points is increased 10-fold. Thus, the computation errors are estimated by $O(1/n^2)$. These orders are the same for both the perimeter and area.

Furthermore, we measure the computation times. The time for each trial is measured using "tic" and "toc" functions, and the average time of 100 executions is adopted. Figure 13 presents the computation times for interpolating the curves including the calculation of the perimeter and area. The horizontal axis represents the number of control points $n$, while the vertical one gives the computation time in seconds. Each time approximately increases in proportion to $n$, which can be presumed in view of Tables 1 and 2.

In view of Figures 11 and 12, the arc splines significantly excel the other three methods in both storage efficiency and space complexity. With respect to the cubic Hermite spline, the errors for the Cassini oval are close to the MS + RDP method, while those for the ellipse are beneficial to some extent. The MS + RDP method excels the MS in most cases, which would because the control points are generated from the MS with the smallest lattice interval. In addition, the errors of the MS + RDP converge on a level close to the MS in $n > 10^4$.

In terms of space complexity, the arc spline yields the best performance among the four methods. If the number of control points increases 10-fold, the error level is decreased approximately $10^{-4}$-fold. The computation errors for the perimeter and area are thus estimated by $O(1/n^4)$. By contrast, the slopes of the MS and MS + RDP are approximately the same; the error is decreased approximately $10^{-2}$-fold if the number of control points is increased 10-fold. Thus, the computation errors are estimated by $O(1/n^2)$. These orders are the same for both the perimeter and area.

Furthermore, we measure the computation times. The time for each trial is measured using "tic" and "toc" functions, and the average time of 100 executions is adopted. Figure 13 presents the computation times for interpolating the curves including the calculation of the perimeter and area. The horizontal axis represents the number of control points $n$, while the vertical one gives the computation time in seconds. Each time approximately increases in proportion to $n$, which can be presumed in view of Tables 1 and 2.

The four values on the upper right represent the total number of interpolated points $N$ including the original control points. For a fixed $n$, though $N$ for the arc spline is two- to threefold compared with the Hermite spline, the arc spline is more lightweight than the cubic Hermite spline. This would be because the arc spline does not need a convergence calculation for adjusting the lengths of tangent vectors.

Figure 10. Two target curves for measuring the computation accuracy.

Figure 11. Computation errors of the perimeter (upper) and area (lower) of a Cassini oval.
rectangular lattice are recorded, and adopt a bicubic spline function (Li, Zhu, and Gold 2004) for an elevation function.

As a study area, we choose an isolated island called Yakushima, located in the southwest part of Japan, an overview of which is shown in Figure 14. There is the highest peak called Mt. Miyanoura, which is close to the center of the island. The peak is 1936(m) in altitude and located on 30°20′9″N, 130°30′17″E. We use a DEM data, the lattice interval of which is 2.25 × 1.5 (m) in longitude and latitude directions. This corresponds to approximately 60.1 × 46.2 (m) around the peak. We detect and encode contours using a piecewise arc algorithm (Goto and Shimakawa 2016).

Now the existing and proposed reconstruction methods are compared by investigating a peninsula on the northern part and a small valley located in northeast. These targets are pointed to by black arrows in Figure 14. Figure 15 presents the reconstructed contours around the peninsula, while Figure 16 reveals those around the valley. Both figures include results by the three methods; the left figure is by the MS, middle one by the arc spline with $s_M = 5$ (m) in Equation (8), and right one by the cubic Hermite spline with the same $s_M$.

The outermost thick contours represent coastlines $z = 0$ and the internal ones are those for every five meters. We extracted coastlines by assuming the elevation of the sea level is $z = -1$m, the reproducibility of which is thus not necessarily good. As long as the shapes of the reconstructed curves are compared, there seem few differences between the arc spline- and cubic Hermite spline-based methods. Contrary to the experiments in the previous section, which spline is used may not notably affect the visual shapes.

In view of Figure 15, the contours reconstructed by the MS are angular, and the contours break only on the edges of patches. Since a linear interpolant is used in the MS, the contour interval within a single edge is constant. Moreover, the peak of a local hill appears only on an intersection of four adjacent patches. These features are unprobable in real terrains and a flaw compared

---

**6. Application to geographic information systems**

Practical application examples to geographic information systems are presented. We use a digital elevation model (DEM) data in which the elevations on a regular
Figure 16 depicts the reproduced contours around the valley in the northeast. In this zone, a river flows from the lower left toward the upper right, which is not depicted in the figures. In the proposed methods, several closed chain lines are observed. These indicate that there is a concave inside the contour. Since there are double chain contours in the lower left part, the depths of the concaves are at least 5 m. This is a benefit of using oriented contours, and such features cannot be identified by the conventional methods.

The proposed methods would be useful for both personal and engineering purposes. In contrast to using contour image data, the proposed approach can detect contours for arbitrary elevations, as well as reconstruct curves for an arbitrary resolution. Since the input data is vector-based, the storage space can be reduced in mapping applications. In the use of civil engineering, the $G^1$ and $G^2$ continuities are an important feature in designing a walkway or cycle road. For another application, the volume of a pit or hill can be estimated more accurately, which is achieved by accumulating the areas of with the proposed methods. Another disadvantage is that ambiguity arises in connecting points on edges where the elevations of two opposite corners of a patch are greater or smaller than those of the remaining two elevations. We examine a hill on the upper right as an example. Focusing on the patch in the left result pointed to by a dashed arrow, there seems a saddle which is not observed in the middle and right results. This would be an adverse effect of the ambiguity. According to a real topographical map, no saddle exists around there and there is only a single hill.

The salient benefit of the proposed methods is that detailed computations and analyses are achievable even if only a coarse data-set is available. For example, because of the good reproducibility of local convexities or concavities, small local peaks or pits can be located more precisely. Ridges and valleys would also be detected more easily. If we aim to find a route to climb up to the center hill from a coast, the path indicated by the dashed arrow in the right figure can be a candidate. It is not easy to find this from the left figure.

Figure 16. Reconstructed contours of a northeastern valley (left: marching square, middle: arc spline, right: cubic Hermite spline).
closed contours to vertical direction. Such usage is useful for example in predicting the area of water immersion caused by torrential rain or a landslide dam.

7. Conclusions

This research has developed a storage-efficient reconstruction framework for vector-based planar contours. The storage format is Cartesian coordinates with tangent angles. Two types of interpolation methods were developed: one is based on the arc spline, while the other one is on the cubic Hermite spline. In a numerical experiment for analytically defined curves, the arc spline-based method achieved better performance in computation accuracy than the cubic Hermite-based method. The shapes of reconstructed curves by the two methods would be notably different where there is a steep curve. By contrast, in another experiment using DEM data, there were little differences in the visual shapes. Which spline should be adopted depends on its objective; a benefit of using the arc spline is that an exact perimeter can be calculated, while with the cubic Hermite spline $G^2$ continuity is achieved in most control points. Although this research has examined using overland DEM data, the same framework is also applicable to undersea data. The calculation of a black-water-region and estimation of total reserves are probable applications.

Notes on contributors

Hiroyuki Goto is a professor in the Department of Industrial & System Engineering, Hosei University, Japan. He received his BS and MS degrees from The University of Tokyo in 1995 and 1997, respectively. He received his DE degree from Tokyo Metropolitan Institute of Technology in 2004. His research interests include operations research and high-performance computing.

Yoichi Shimakawa is a professor and director of the Department of Computer Science and Technology, Salesian Polytechnique, Japan. He received his BS and MS degrees from Chuo University in 1990 and 1996, respectively. In 1998, he joined the staff as a research assistant on the research project “Integrated Geographic Information Systems” at Chuo University. He received his DE degree from Chuo University. He received a paper award from the Operations Research Society of Japan in 2002.

References

Briggs, I. C. 1974. “Machine Contouring using Minimum Curvature.” Geophysics 39 (1): 39–48.
Catmull, E., and R. Rom. 1974. “A Class of Local Interpolating Splines.” In Computer Aided Geometric Design, edited by R. E. Barnhill and R. F. Riesenfeld, 317–326. New York: Academic Press.
Chai, J., T. Miyoshi, and E. Nakamae. 1998. Contour Interpolation and Surface Reconstruction of Smooth Terrain Models. North Caroline: IEEE Visualization Research Triangle Park.
Chui, C. K. 1988. Multivariate Splines. Philadelphia, PA: SIAM. doi:10.1137/1.9781611970173.
Douglas, D. H., and T. K. Peucker. 1973. “Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature.” Cartographica: The International Journal for Geographic Information and Geovisualization 10 (2): 112–122.
Dyn, N., D. Levin, and D. Liu. 1992. “Interpolatory Convexity Preserving Subdivision Schemes for Curves and Surfaces.” Computer-Aided Design 24 (4): 211–216.
Goto, H., and Y. Shimakawa. 2013. “Lightweight Calculation of the Area of a Closed Contour by an Inner Skin Algorithm.” Journal of Computations and Modelling 3 (1): 57–77.
Goto, H., and Y. Shimakawa. 2016. “Storage-Efficient Method for Generating Contours Focusing on Roundness.” International Journal of Geographical Information Science 30 (2): 200–220.
Gousie, M. B., and W. R. Franklin. 2005. “Augmenting Grid-Based Contours to Improve Thin-Plate DEM Generation.” Photogrammetric Engineering & Remote Sensing 71 (1): 69–79.
Lawonn, K., R. Gasteiger, C. Rössl, and B. Preim. 2014. “Adaptive and Robust Curve Smoothing on Surface Meshes.” Computers & Graphics 40: 22–35.
Li, Z., and D. S. Meek. 2005. “Smoothing an Arc Spline.” Computers & Graphics 29 (4): 576–587.
Li, Z., C. Zhu, and C. Gold. 2004. Digital Terrain Modeling. Boca Raton: CRC Press.
Lopes, A., and K. Brodlie. 2003. “Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing.” IEEE Transactions on Visualization and Computer Graphics 9 (1): 16–29.
Maple, C. 2003. “Geometric design and space planning using the marching squares and marching cube algorithms.” Paper presented at the 2003 International Conference on Geometric Modeling and Graphics, July 16–18.
McCullagh, M. J. 1981. “Creation of Smooth Contours over Irregularly Distributed Data Using Local Surface Patches.” Geographical Analysis 13 (1): 51–63.
Pradhan, M. P., and M. K. Ghose. 2013. “Improvised Interpolation of Contour Lines Using Spider Weaving Approach.” International Journal on Computer Science and Engineering 5 (9): 782–796.
Ramer, U. 1972. “An Iterative Procedure for the Polygonal Approximation of Plane Curves.” Computer Graphics and Image Processing 1 (3): 244–256.
Schneider, B. 2005. “Extraction of Hierarchical Surface Networks from Bilinear Surface Patches.” Geographical Analysis 37 (2): 244–263.
Stroud, K. A., and D. Booth. 2005. Vector Analysis. New York: Industrial Press.
Taubin, G. 1995. “Curve and Surface Smoothing without Shrinkage.” Paper presented at the Fifth International Conference on Computer Vision, Cambridge, Massachusetts, USA.
Thibault, D., and C. M. Gold. 2000. “Terrain Reconstruction from Contours by Skeleton Construction.” Geoinformatica 4 (4): 349–373.
Ukasha, A. A. 2011. “A New Method for Contour Approximation Using Basic Ramer Idea.” World Academy of Science, Engineering and Technology 5 (1): 168–173.
Ukasha, A. A., E. Saffor, and M. A. S. Hassan. 2008. “Contour Compression Using Trapezoid Method.” Paper presented at the 23rd International Symposium on Computer and Information Sciences. Istanbul: IEEE.