Circular Spline Fitting Using an Evolution Process

Xinghua Song [a,b] , Martin Aigner [b] , Falai Chen [a]
and Bert Jüttler [b]

[a] *Department of Mathematics, University of Science and Technology of China, Hefei, Anhui, P.R.China*

[b] *Johannes Kepler University Linz, Institute of Applied Geometry, Austria*

Abstract

We propose a new method to approximate a given set of ordered data points by a spatial circular spline curve. At first an initial circular spline curve is generated by biarc interpolation. Then an evolution process based on a least-squares approximation is applied to the curve. During the evolution process, the circular spline curve converges dynamically to a stable shape. Our method does not need any tangent information. During the evolution process, the number of arcs is automatically adapted to the data such that the final curve contains as few arc arcs as possible. We prove that the evolution process is equivalent to a Gauss-Newton type method.

Keywords: Circular spline, biarc, organized points, spatial curve fitting

1 Introduction

A circular spline curve consists of circular arcs and line segments which are pieced together with G^1 continuity. This simple but powerful class of curves offers a number of remarkable advantages.

- The arc-length function of a segment of a circular spline curve can be evaluated in closed form. Moreover, also the inverse operation is explicitly available: one can easily find the point on the curve which possesses a given

Email addresses: xhsong@mail.ustc.edu.cn (Xinghua Song), martin.aigner@jku.at (Martin Aigner), chenfl@ustc.edu.cn (Falai Chen), bert.juettler@jku.at (Bert Jüttler).

arc-length distance to a given point, without any need for numerical approximation. Consequently, circular splines are especially useful for numerically controlled (NC) machining.

The first observation is also valid for the interesting class of Pythagorean hodograph curves, where the arc-length function is simply a polynomial, see [5]. The second observation, however, is not true for PH curves, since it requires the solution of a polynomial root finding problem.

- The offsets of arc-splines (i.e., constant radius pipe surfaces generated by them, see e.g. [3]) have simple closed-form parameterizations, since they consist of segments of tori and cylinders.
- The use of arc splines provides a simple and non-iterative method for closest point computation. More precisely, for a set of given points in space, the associated closest points on the curve can be computed by solving quadratic equations. For polynomial or rational spline curves, the same problem leads to non-linear optimization problems, which require iterative solution techniques.
- As observed by Wang and Joe [16], arc spline curves are very useful for sweep surface modeling, since they provide high-quality approximations of rotation-minimizing frames.
- Circular arcs are useful as geometric primitives for algorithms from computational geometry. They combine simplicity of elementary operations with a relatively high geometry approximation power, see [1].

Various computational methods for constructing circular spline curves in the plane and in three–dimensional space from given data (e.g., sequences of points, or a given smooth curve in another representation) have been described in the literature. Two classical references are a VTO report by Sabin [15] and a textbook by Nutbourne and Martin [13]. In particular, curves composed of biarcs (i.e., G^1 smooth curve segments consisting of two circular arcs) were used in a large number of algorithms for approximation or interpolation of given point (and possibly tangent) data.

The problem of approximating scattered points in the plane by circular splines has been discussed by Hoschek [6]. He presents an approximation algorithm which is based on a non-linear least-squares method.

Meek and Walton discussed arc splines in a number of publications. In [9], they propose a method that does not make use of a the least-squares approach. Instead, the discrete data are partitioned and then approximated by biarcs using standard algorithms. In a later paper [10], they partition a smooth planar curve and match the curve segments by biarcs. Since the curve and the biarcs are bounded by some bounding circular arcs within a given tolerance, the biarcs form an arc spline that approximates the smooth planar curve within the given tolerance. In another paper [11], they describe a method for generating planar osculating arc splines, which interpolate, match unit tangents, and

match curvatures at the interpolation points. In [12] they use arc splines to approximate the clothoid.

Yang and Du [18] use techniques from optimization theory to approximate planar digitized curves by arc splines. An arc spline is constructed such that it exhibits G^0 or G^1 continuity at each joint point and that its maximum approximation error is not bigger than a given tolerance.

Piegl and Tiller [14] describe an algorithm for data approximation with biarcs in the plane. They use a specific formulation of biarcs which is appropriate for parametric curves in Bézier or NURBS formulation and apply a base curve to obtain tangents and anchor points for the individual biarcs.

Recently, circular splines have been used for reconstructing pipe surfaces from unorganized measurement data by Bauer and Polthier [3]. A moving least-squares based technique is used to reconstruct the spine curve of a pipe surface from surface samples and approximate the spine curve by G^1 continuous circular arcs and line segments.

In the present paper, we present a novel method for approximating spatial point data by an arc spline curve. We interpret the intermediate solutions generate by a non-linear optimization method as instances of a continuous evolution process.

This approach to circular spline fitting is motivated by related results from the field of Computer Vision [4, 7]. Wang et al. [17] extended them by using curvature information and used them for curve and surface fitting with B-splines. In particular, they analyzed the relation to Gauss-Newton-type techniques. These papers describe certain geometrically motivated non-linear optimization techniques which generate a sequence of approximate solutions. Recently, Liu et al. [8] studied various extensions of the Gauss-Newton-type techniques described in [17] to the case of space curves.

In the case of planar curves, the corresponding continuous evolution process which corresponds to the non-linear optimization has been studied in [2], and it was also extended to a larger class of curves which can be described an arbitrary set of shape parameters. This has been made explicit for the class of Pythagorean-hodograph spline curves.

In the present paper makes the following contributions. First we derive an independent set of shape parameters, which uniquely describe a circular spline curve. Second, we formulate an evolution process (governed by a differential equation) for circular spline curves in three-dimensional space which drives an initial curve towards a limit shape, which approximates a given sequence of points. This extends the framework of [2] to the case of space curves. We also show how the discretized version of the evolution process is related to the

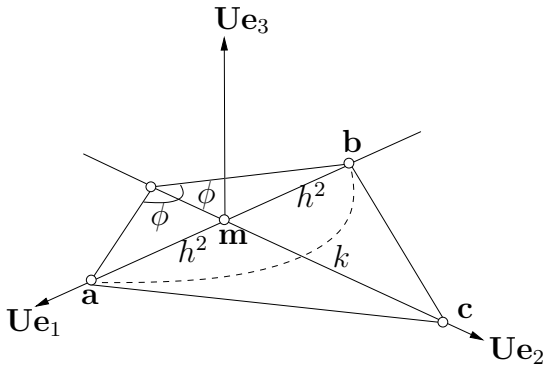


Fig. 1. Rational Bézier representation of a circular arc.

Gauss-Newton-type techniques described in [8]. Third, we demonstrate that the use of circular spline curve for curve fitting has the unique advantage of simple and explicit closest point computation. This makes them particularly useful for orthogonal distance regression, where one minimizes the shortest distances between the points and the approximating curve.

2 Shape parameters

We introduce a special representation of an arc spline curve. First we discuss single arcs, and then we use biarcs in order to extend this to the case of spline curves.

2.1 Single arcs

A circular arc in space has the rational Bézier representation

$$\mathbf{y}(u) = \frac{(1-u)^2\mathbf{a} + 2u(1-u)\omega\mathbf{c} + u^2\mathbf{b}}{(1-u)^2 + 2u(1-u)\omega + u^2}, \quad u \in [0,1], \tag{1}$$

with the control points $\mathbf{a}, \mathbf{b}, \mathbf{c}$, where $\mathbf{c}$ lies in the bisector plane of the line segment $\mathbf{ab}$. The weight ω satisfies $\omega = \cos\phi$, where

$$\phi = \frac{1}{2}(\pi - \angle(\mathbf{a}, \mathbf{c}, \mathbf{b})) \tag{2}$$

is half the sweep angle of the circular arc, see Fig. 1. The representation (1) has 10 free parameters. However, only 8 of these parameters can be chosen independently, as follows.

The vector

$$(m_x, m_y, m_z, h, k, \alpha, \beta, \gamma) \tag{3}$$

is said to be the *vector of shape parameters of the circular arc*. The first three parameters are the coordinates of the midpoint $\mathbf{m} = (m_x, m_y, m_z)$ of the line segment **ab**. The control points **a**, **b**, **c** and the weight w are computed from

$$\mathbf{a} = \mathbf{m} + h^2\mathbf{U}\mathbf{e}_1, \quad \mathbf{b} = \mathbf{m} - h^2\mathbf{U}\mathbf{e}_1, \quad \mathbf{c} = \mathbf{m} + k\mathbf{U}\mathbf{e}_2, \quad \omega = \frac{h^2}{\sqrt{h^4 + k^2}} \tag{4}$$

with the special orthogonal matrix

$$\mathbf{U}(\alpha, \beta, \gamma) = \begin{pmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{pmatrix} \begin{pmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

and the two unit vectors $\mathbf{e}_1 = (1, 0, 0)$ and $\mathbf{e}_2 = (0, 1, 0)$.

The vector $\mathbf{a} - \mathbf{m}$ is restricted to be a positive multiple of $\mathbf{U}\mathbf{e}_1$, by using the square of h in (4). We did not apply the same constraint to $\mathbf{c} - \mathbf{m}$, in order to allow the curve to change the orientation of its normal and binormal vector without having to perform a complete change of the angles controlling the matrix $\mathbf{U}$. Consequently, there are in general four vectors of shape parameters which describe the same circular arc.

If $k = 0$, then Eq. (4) describes the control points of a line segment, which is represented as a degree-elevated linear curve segment. One of the three angles becomes redundant in this situation, as a line segment is invariant under a rotation.

In Eq. (5), the special orthogonal matrix $\mathbf{U}$ is represented by zxz-Euler angles, i.e. by the composition of three rotations around the z, the x and the z–axis. If $\beta = 0$, then this representation has a singularity and other Euler angles should be used instead.

2.2 Biarcs connecting two circular arcs

Any two arcs $\mathbf{y}(t)$ and $\hat{\mathbf{y}}(t)$ with control points and weights $\mathbf{a}, \mathbf{b}, \mathbf{c}, \omega$ and $\hat{\mathbf{a}}, \hat{\mathbf{b}}, \hat{\mathbf{c}}, \hat{\omega}$, respectively, which we refer to as *primary arcs*, can be joined by one biarc (i.e. two arcs with G^1 continuity at the joint point) connecting $\mathbf{x}(1) = \mathbf{b}$ and $\hat{\mathbf{x}}(0) = \hat{\mathbf{a}}$, such that the overall curve is G^1 smooth, see Fig. 2. Note that the figure shows the planar case, while the method applies to spatial arcs as well.

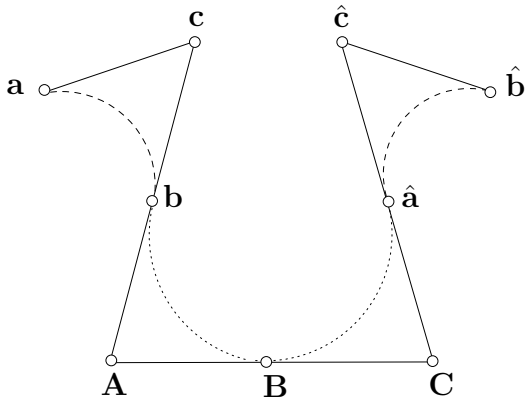


Fig. 2. Joining two primary arcs (dashed) by one biarc (dotted)

More precisely, there exists a one-parameter family of such biarcs. Referring to Fig. 2, the unknown points $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are sought for. Since $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are collinear, we get

$$\mathbf{A} = \mathbf{b} + \ell\,\mathbf{T}, \quad \mathbf{C} = \hat{\mathbf{a}} - \hat{\ell}\,\hat{\mathbf{T}}, \quad ||\mathbf{A} - \mathbf{C}||^2 = (\ell + \hat{\ell})^2 \tag{6}$$

where

$$\mathbf{T} = \frac{\mathbf{b} - \mathbf{c}}{||\mathbf{b} - \mathbf{c}||}, \quad \hat{\mathbf{T}} = \frac{\hat{\mathbf{c}} - \hat{\mathbf{a}}}{||\hat{\mathbf{c}} - \hat{\mathbf{a}}||}. \tag{7}$$

The first three equations imply

$$\mathbf{V} \cdot \mathbf{V} + 2\ell\mathbf{V} \cdot \mathbf{T} + 2\hat{\ell}\mathbf{V} \cdot \hat{\mathbf{T}} + 2\ell\hat{\ell}\,(\mathbf{T} \cdot \hat{\mathbf{T}} - 1) = 0 \tag{8}$$

where $\mathbf{V} = \mathbf{b} - \hat{\mathbf{a}}$. The only unknowns in this equation are ℓ and $\hat{\ell}$. Almost any value of ℓ uniquely determines a biarc. We call ℓ the *shape parameter of the biarc.*

Remark 1 Negative value of ℓ or $\hat{\ell}$ correspond to the case when the sweep angle of the first or second arc is bigger than π. Semicircles are excluded, since they require ℓ or $\hat{\ell}$ to take infinite values. In practice, if the number of segments is sufficiently small, the sweep angles are smaller than π, hence ℓ and $\hat{\ell}$ are both positive.

2.3 Circular splines

Given a sequence of $K + 1$ primary arcs $(\mathbf{y}_{3k})_{k=0,\ldots,K}$, every two consecutive arcs can be joined by one biarc. This leads to a circular spline curve which consists of $K + 1$ primary arcs and K biarcs. We represent it as a rational Bézier spline curve with the parameter domain $[0, 3K + 1]$ which is piecewise defined as

$$\mathbf{x}(t, \mathbf{s}) = \mathbf{y}_j(u_j, \mathbf{s}) \quad \text{for } t \in [j, j + 1], \quad j = 0, \ldots, 3K, \tag{9}$$

where $u_j = t - j$ and

$$\mathbf{y}_j(u_j, \mathbf{s}) = \frac{(1-u_j)^2 \mathbf{a}_j(\mathbf{s}) + 2u_j(1-u_j)\omega_j(\mathbf{s})\mathbf{c}_j(\mathbf{s}) + u_j^2 \mathbf{b}_j(\mathbf{s})}{(1-u_j)^2 + 2u_j(1-u_j)\omega_j(\mathbf{s}) + u_j^2}. \tag{10}$$

The global shape parameter vector $\mathbf{s} = (s_1, ..., s_m)$, where $m = 9K+1$, consists of the shape parameters of all arcs and all biarcs. Each primary arc contributes its 8 parameters of the form (3), while each biarc contributes one additional length ℓ. This vector of shape parameters uniquely determines the control points and the weights of the Bézier arcs.

3 Evolution-based fitting

We consider the following problem: Given a sequence of points $(\mathbf{p}_i)_{i=0,...,n}$, find a circular spline curve which approximates these points. In order to solve this problem, we generalize the method introduced in [2] to the case of space curves. The approximate solutions generated by an iterative solution algorithm for the non-linear fitting problem are seen as instances of a continuous evolution of an initial curve towards its final position.

3.1 Initial circular spline

We assume that the initial number $K + 1$ of primary arcs is specified by the user. In order to construct the initial spline curve, we consider the subset

$$\hat{\mathbf{p}}_j = \mathbf{p}_{\lceil i \cdot n/(6K+2) \rceil}, \quad j = 0, \ldots, 6K + 2. \tag{11}$$

The k–th primary arc $\mathbf{y}_{3k}$, where $k = 0, \ldots, K$, is now found as the unique arc connecting $\hat{\mathbf{p}}_{6k}$ and $\hat{\mathbf{p}}_{6k+2}$ via $\hat{\mathbf{p}}_{6k+1}$. Next, every pair of consecutive primary arcs is joined by one biarc as described in the last section, simply by setting $\ell = \hat{\ell}$ in Eq. (8). We obtain a circular spline $\mathbf{x}(t, \mathbf{s}_0)$ which is described by an initial vector $\mathbf{s}_0$ of shape parameters. The choice of the subset (11) of points guarantees that each arc corresponds to roughly the same number of points.

Remark 2 In order to obtain a closed circular spline curve, the last primary arc has to be identified with the first one. Consequently, a circular spline with K primary arcs has $3K$ segments.

Starting with the initial circular spline, we set up an evolution process which drives the curve towards the given data points, until they are approximated sufficiently well. More precisely, we assume that the shape parameters $\mathbf{s}$ depend on a time-like parameter τ, which gives us an evolving circular spline curve in space. The final curve is then defined by the shape parameters

$$\mathbf{s}_{\text{final}} = \lim_{\tau \to \infty} \mathbf{s}(\tau). \tag{12}$$

If we consider a fixed point $\mathbf{x}(t^*, \mathbf{s}(\tau))$ with parameter t^* on the curve, then it travels with the velocity

$$\vec{\mathbf{v}}(t^*, \mathbf{s}(\tau)) = \dot{\mathbf{x}}(t^*, \mathbf{s}(\tau)) = \sum_{j=1}^{m} \frac{\partial \mathbf{x}(t^*, \mathbf{s}(\tau))}{\partial s_j} \, \dot{s}_j(\tau) = [\nabla_{\mathbf{s}} \mathbf{x}(t^*, \mathbf{s}(\tau))] \; \dot{\mathbf{s}}(\tau), \tag{13}$$

where $\nabla_{\mathbf{s}} = (\frac{\partial}{\partial s_1}, \ldots, \frac{\partial}{\partial s_m})$ and the dot $\dot{}$ denotes the derivative with respect to the time variable τ. In order to keep the notation simple, we shall omit the time parameter τ from now on.

For each data point $\mathbf{p}_i$, we consider the associated closest point $\mathbf{x}(t_i, \mathbf{s})$ on the curve (or one of them, in case that several such points exist). The evolution of the curve will guided by the following principle: *The normal component of the velocity of a curve point* $\mathbf{x}(t_i, \mathbf{s})$ *shall be equal to the residual vector* $\mathbf{p}_i - \mathbf{x}(t_i, \mathbf{s})$.

In order to express this condition, we choose for each closest point $\mathbf{x}(t_i, \mathbf{s})$ two arbitrary unit vectors $\vec{\mathbf{n}}_i$ and $\vec{\mathbf{m}}_i$, which are mutually orthogonal and perpendicular to the tangent vector $\mathbf{x}'(t_i, \mathbf{s})$, where the prime $'$ indicates differentiation with respect to the curve parameter t. These two vectors form an orthonormal basis of the normal plane of the curve at $\mathbf{x}(t_i, \mathbf{s})$. The condition is then equivalent to the two equations

$$\begin{aligned} \vec{\mathbf{v}}(t_i, \mathbf{s}) \cdot \vec{\mathbf{n}}_i &= (\mathbf{p}_i - \mathbf{x}(t_i, \mathbf{s})) \cdot \vec{\mathbf{n}}_i \\ \vec{\mathbf{v}}(t_i, \mathbf{s}) \cdot \vec{\mathbf{m}}_i &= (\mathbf{p}_i - \mathbf{x}(t_i, \mathbf{s})) \cdot \vec{\mathbf{m}}_i, \end{aligned} \tag{14}$$

see Fig. 3.

In the case of an open curve, some data points $\mathbf{p}_i$ may have one of the two curve end points as their associated closest points, i.e., $t_i = 0$ or $t_i = 3K + 1$. If this is the case, then we consider the entire velocity vector and not only its normal component, by replacing the two equations (14) with the three equations

$$\vec{\mathbf{v}}(t_i, \mathbf{s}) = \mathbf{p}_i - \mathbf{x}(t_i, \mathbf{s}). \tag{15}$$

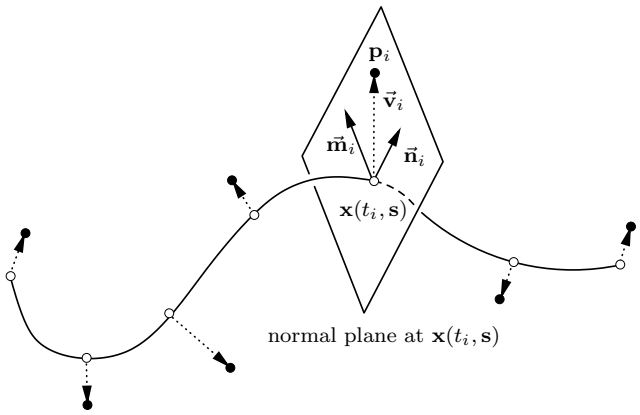


Fig. 3. Each data point $\mathbf{p}_i$ attracts the associated closest point $\mathbf{x}(t_i, \mathbf{s})$. This is expressed with the help of two auxiliary vectors $\vec{\mathbf{m}}_i$ and $\vec{\mathbf{n}}_i$

In order to keep the presentation simple, we exclude this case from now on.

In general, the number of data points exceeds the degrees of freedom of the curve which is to be fitted to these data. Hence, Eq. (14) cannot be fulfilled exactly for all data points. We use a least-squares approach to compute $\dot{\mathbf{s}}$ by minimizing

$$E = \sum_{i=1}^{n} \Big([(\vec{\mathbf{v}}(t_i, \mathbf{s}) - \mathbf{p}_i + \mathbf{x}(t_i, \mathbf{s})) \cdot \vec{\mathbf{n}}_i]^2 + [(\vec{\mathbf{v}}(t_i, \mathbf{s}) - \mathbf{p}_i + \mathbf{x}(t_i, \mathbf{s})) \cdot \vec{\mathbf{m}}_i]^2 \Big) + \lambda ||\dot{\mathbf{s}}||^2 \tag{16}$$

where $\lambda \ll 1$ is a non-negative weight and $||\dot{\mathbf{s}}||^2$ is a regularization term which ensures the existence of a unique minimizer of (16).

The value of E does not depend on the choice of the vectors $\vec{\mathbf{m}}_i$ and $\vec{\mathbf{n}}_i$. These two vectors form an orthogonal basis of the normal planes. Hence, the contribution of the velocity vector of each closest point is equal to the squared length of the projection into the normal plane.

Since the velocities $\vec{\mathbf{v}}(t_i, \mathbf{s})$ depend linearly on the time derivatives $\dot{\mathbf{s}}$ of the shape parameters, see Eq. (13), this is a quadratic optimization problem which can be solved easily. A short computation shows that the solution is found by solving the linear system

$$\mathbf{H}(\mathbf{s})\dot{\mathbf{s}} + \mathbf{r}(s) = \mathbf{0} \tag{17}$$

with

$$\mathbf{H}(\mathbf{s}) = 2 \sum_{i=1}^{n} \nabla_{\mathbf{s}} \mathbf{R}_i^\top (\vec{\mathbf{m}}_i \vec{\mathbf{m}}_i^\top + \vec{\mathbf{n}}_i \vec{\mathbf{n}}_i^\top) \nabla_{\mathbf{s}} \mathbf{R}_i + 2\lambda \mathbf{I} \tag{18}$$

and

$$\mathbf{r}(\mathbf{s}) = 2 \sum_{i=1}^{n} \mathbf{R}_i^\top \nabla_{\mathbf{s}} \mathbf{R}_i. \tag{19}$$

where we use the abbreviation

$$\mathbf{x}_i = \mathbf{x}(t_i, \mathbf{s}) \tag{20}$$

The *residual vector*

$$\mathbf{R}_i = \mathbf{p}_i - \mathbf{x}_i = \mathbf{p}_i - \mathbf{x}(t_i, \mathbf{s}) \tag{21}$$

lies in the normal plane of the curve at $\mathbf{x}_i$. Its gradient with respect to the shape parameters satisfies $\nabla_{\mathbf{s}}\mathbf{R}_i = -\nabla_{\mathbf{s}}\mathbf{x}_i$.

Note that the derivative $\nabla_{\mathbf{s}}\mathbf{x}_i$ of the closest point $\mathbf{x}_i$ with respect to the shape parameters does not take the dependency of t_i on these parameters into account; it is solely the derivative of $\mathbf{x}(t_i, \mathbf{s})$ with respect to its second argument.

The system (17) is equivalent to the differential equation $\dot{\mathbf{s}} = -(\mathbf{H}(\mathbf{s}) + 2\lambda\mathbf{I})^{-1}\mathbf{r}(\mathbf{s})$ for the unknowns $\mathbf{s}(\tau)$. The shape parameters are updated via

$$\mathbf{s}(\tau + \Delta\tau) = \mathbf{s}(\tau) + \dot{\mathbf{s}}(\tau)\Delta\tau \tag{22}$$

by using an explicit Euler step, where $\dot{\mathbf{s}}$ is found by solving the linear system (17). We choose the step size $\Delta\tau$ as

$$\Delta\tau = \min\{1,\ \{D/||\vec{\mathbf{v}}(t_i, \mathbf{s})||\}_{i=1,..,n}\ \} \tag{23}$$

where D is a user defined value. This shall ensure that the traveling distance of each point $\mathbf{x}_i$ of the curve, which is approximately equal to

$$||\vec{\mathbf{v}}(t_i, \mathbf{s})\,\Delta\tau|| \tag{24}$$

is constrained to be approximately less or equal than the constant D. This constant can be chosen, e.g., as 5% of the diameter of the bounding box.

3.3 Closest point computation

In each step of the evolution, we have to find the closest point $\mathbf{f}_i = \mathbf{x}(t_i, \mathbf{s})$ on the curve which is associated with every given point $\mathbf{p}_i$. More precisely, we have to find

$$t_i = \arg\min_{t\in[0,3K+2]} ||\mathbf{p}_i - \mathbf{x}(t, \mathbf{s})|| \tag{25}$$

The shortest distance from a data point to the curve is the minimum of the shortest distances to all arcs. First, consider a fixed circular arc $\mathbf{y}_j$, and let $\mathbf{m}_j$ be its center. The parameter t_i realizing the shortest distance can be computed as follows.

(1) We project $\mathbf{p}_i$ orthogonally into the plane which contains the arc $\mathbf{y}_j(t)$. The projected point is denoted with $\mathbf{q}_i$.
(2) If $\mathbf{q}_i$ lies inside the sweep angle of $\mathbf{y}_j(t)$, then the candidate values t_i of the global curve parameter $t = j + u_j$ are found by adding j to the root(s)

of the quadratic equation

$$(\mathbf{q}_i - \mathbf{m}_j) \cdot \mathbf{y}_j'(u_j, \mathbf{s}) = 0, \quad u_j \in [0,1], \tag{26}$$

where the prime $'$ denotes the derivative with respect to the local curve parameter u_j. Otherwise, the shortest distance is not realized by this circular arc (but see the next remark).

In the case of an open spline curve, the closest point of $\mathbf{p}_i$ can also be one of the two boundary points. Hence, the two end points have to be checked separately.

Remark 3 In order to keep the algorithm as simple as possible, we compute the closest point by first finding the closest points in all circular arcs, and then selecting the point with the minimum distance among them. One may improve the efficiency of the algorithm by using a suitable hierarchy of bounding volumes. Of course, this hierarchy has to be updated in each time step.

3.4 Adaptive refinement

The evolution drives the circular spline $\mathbf{x}(t, \mathbf{s})$ towards a stationary point of the evolution process (see next section for a theoretical analysis). However, if the number of arcs is too small, then the curve does not approximate the data points sufficiently well. In order to improve the quality of the fit, we apply a refinement operation to this curve, as follows:

1) Compute the error which is associated with every arc $\mathbf{y}_j(t, \mathbf{s})$ of the circular spline $\mathbf{x}(t, \mathbf{s})$,
$$\varepsilon_j = \sum_{i \in \mathcal{I}_j} ||\mathbf{p}_i - \mathbf{f}_i||$$
where $\mathcal{I}_j = \{i \mid t_i \in [j, j+1] :\}$. Let $\varepsilon > 0$ be the error threshold of error and $h_j = |\mathcal{I}_j|$ be the number of elements in the set $\mathcal{I}_j$.
2) If the average error $\dfrac{\varepsilon_j}{h_j} \leqslant \varepsilon$ for every circular arc $\mathbf{y}_j(t, \mathbf{s})$, or the number of iterations is larger than a given constant δ, then terminate the process; Otherwise, choose the three arcs with the largest error ε_j and subdivide each arc into two parts at the midpoint.
3) Since the number of arcs of the circular spline has increased by three, we re-arrange the circular arcs and choose the arcs with indices $3k$ as primary circular arcs. The remaining arcs form the biarcs which connect the primary circular arcs.
4) Apply the evolution process as described in the last section until the circular arcs converge to a stable curve (if the average error decreased less than a given threshold after one step iteration) and return to Step 1).

4 Relation to Gauss-Newton-type techniques

In the following we analyze the relation of the circular spline evolution to a Gauss-Newton-type method for orthogonal distance regression.

Proposition 4 *The Euler update for the shape parameters* $\mathbf{s}$ *for the curve evolution defined via* (22) *with* $\lambda = 0$ *is equivalent to one step of a Gauss-Newton method for the objective function*

$$\sum_{i=1}^{n} ||\mathbf{p}_i - \mathbf{x}(t_i, \mathbf{s})||^2 \rightarrow \min, \tag{27}$$

where the t_i *are parameter values associated with the closest points, in the sense that the Hessian is simplified by omitting all second order derivatives.*

Proof. We follow the discussion in [8] and compute the Gauss-Newton system directly from

$$F = \sum_{i=1}^{n} \|\mathbf{R}_i\|^2. \tag{28}$$

The gradient is found to be

$$\nabla_{\mathbf{s}} F = 2 \sum_{i=1}^{n} \mathbf{R}_i^\top \nabla_{\mathbf{s}} \mathbf{R}_i + \nabla_{\mathbf{s}} t_i \mathbf{R}_i^\top \mathbf{R}_i'] = 2 \sum_{i=1}^{n} \mathbf{R}_i^\top \nabla_{\mathbf{s}} \mathbf{R}_i \tag{29}$$

where we used that $\mathbf{R}_i^\top \mathbf{R}_i' = 0$, which is due to the fact that the parameter values t_i correspond to the closest points associated with the given points. Next we obtain the Hessian via

$$\nabla_{\mathbf{s}} (\nabla_{\mathbf{s}} F^\top) = 2 \sum_{i=1}^{n} (\nabla_{\mathbf{s}} \mathbf{R}_i^\top + \nabla_{\mathbf{s}} t^\top \mathbf{R}_i') \nabla_{\mathbf{s}} \mathbf{R}_i + (\nabla_{\mathbf{s}} (\nabla_{\mathbf{s}} \mathbf{R}_i^\top)) \circ \mathbf{R}_i + \nabla_{\mathbf{s}} t^\top \mathbf{R}_i \nabla_{\mathbf{s}} \mathbf{R}_i', \tag{30}$$

where

$$\left[\nabla_{\mathbf{s}} (\nabla_{\mathbf{s}} \mathbf{R}_i^\top) \circ \mathbf{R}_i\right]_{l,k} = \sum_{i=1}^{2} \left[\frac{\partial}{\partial s_l} \frac{\partial}{\partial s_k} [\mathbf{R}_i]_i \right] [\mathbf{R}_i]_i. \tag{31}$$

By omitting the second order derivatives ($\nabla_{\mathbf{s}} (\nabla_{\mathbf{s}} \mathbf{R}_i)$ and $\nabla_{\mathbf{s}} \mathbf{R}_i'$ we arrive at the simplified Hessian

$$\mathbf{H}_F^* = 2 \sum_{i=1}^{n} \nabla_{\mathbf{s}} \mathbf{R}_i^\top \nabla_{\mathbf{s}} \mathbf{R}_i + \nabla_{\mathbf{s}} t_i^\top \mathbf{R}_i' \nabla_{\mathbf{s}} \mathbf{R}_i \tag{32}$$

An expression for $\nabla_{\mathbf{s}} t_i$ is obtained by differentiating the identity $\mathbf{R}_i^\top \mathbf{R}_i' = 0$,

$$\mathbf{R}_i'^\top \nabla_{\mathbf{s}} \mathbf{R}_i + \nabla_{\mathbf{s}} t_i \mathbf{R}_i'^\top \mathbf{R}_i' + \nabla_{\mathbf{s}} t \mathbf{R}_i''^\top \mathbf{R}_i + \mathbf{R}_i^\top \nabla_{\mathbf{s}} \mathbf{R}_i' = 0, \tag{33}$$

which gives

$$\nabla_{\mathbf{s}} t_i = -\frac{\mathbf{R}_i^\top \nabla_{\mathbf{s}} \mathbf{R}_i' + \mathbf{R}_i'^\top \nabla_{\mathbf{s}} \mathbf{R}_i}{\mathbf{R}_i''^\top \mathbf{R}_i + \mathbf{R}_i'^\top \mathbf{R}_i'}. \tag{34}$$

Omitting again second order derivatives and substituting the result into H_F^* gives

$$\mathbf{H}_F^* = 2\sum_{i=1}^{n}[\nabla_{\mathbf{s}}\mathbf{R}_i^\top \nabla_{\mathbf{s}}\mathbf{R}_i - \frac{\nabla_{\mathbf{s}}\mathbf{R}_i^\top \mathbf{R}_i' \mathbf{R}_i'^\top \nabla_{\mathbf{s}}\mathbf{R}_i}{\mathbf{R}_i'^\top \mathbf{R}_i'}]. \tag{35}$$

Finally, we use the identity

$$\vec{\mathbf{n}}_i\vec{\mathbf{n}}_i^\top + \vec{\mathbf{m}}_i\vec{\mathbf{m}}_i^\top + \vec{\mathbf{t}}_i\vec{\mathbf{t}}_i^\top = \mathbf{E}, \tag{36}$$

where $\vec{\mathbf{t}}_i$ denotes the unit tangent vector at $\mathbf{x}_i$ and $\mathbf{E}$ is the 3×3 identity matrix. Together with $\frac{\mathbf{R}_i'}{\|\mathbf{R}_i'\|} = \pm\vec{\mathbf{t}}_i$ we get

$$\begin{aligned}\mathbf{H}_F^* =& 2\sum_{i=1}^{n}[\nabla_{\mathbf{s}}\mathbf{R}_i^\top \nabla_{\mathbf{s}}\mathbf{R}_i - \nabla_{\mathbf{s}}\mathbf{R}_i^\top(\mathbf{E} - \vec{\mathbf{n}}_i\vec{\mathbf{n}}_i^\top - \vec{\mathbf{m}}_i\vec{\mathbf{m}}_i^\top)\nabla_{\mathbf{s}}\mathbf{R}_i] &(37)\\ =& 2\sum_{i=1}^{n}\nabla_{\mathbf{s}}\mathbf{R}_i^\top(\vec{\mathbf{n}}_i\vec{\mathbf{n}}_i^\top + \vec{\mathbf{m}}_i\vec{\mathbf{m}}_i^\top)\nabla_{\mathbf{s}}\mathbf{R}_i. &(38)\end{aligned}$$

Finally we observe that the system

$$\mathbf{H}_F^*(\mathbf{s})\Delta\mathbf{s} + \nabla_{\mathbf{s}}F = \mathbf{0} \tag{39}$$

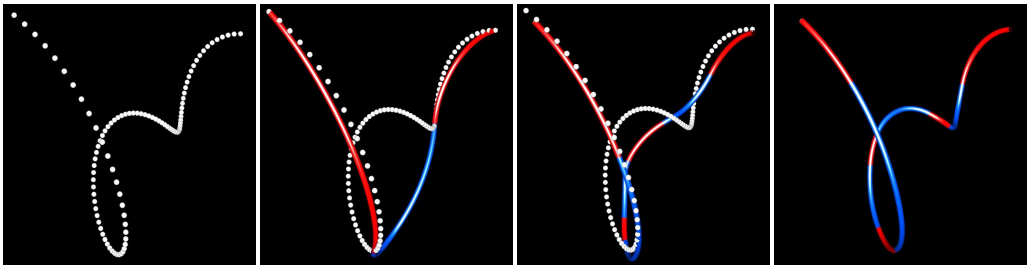
which is solved for computing the update vector $\Delta\mathbf{s}$, is equivalent to (17) with $\lambda = 0$ and $\Delta\mathbf{s} = \dot{\mathbf{s}}$. ■

This result has several important consequences.

- The minimization of (16) provides a direction of descent for the objective function (28). Hence, with a suitable stepsize control, the circular spline curve is driven towards a local minimum of this non-linear objective function.
- In the case of a zero-residual problem, where $\mathbf{R}_i = \mathbf{0}$ in the limit, the approximate Hessian $\mathbf{H}_F^*$ converges to the exact one, since all omitted terms contain $\mathbf{R}_i$ as a factor. Consequently, one obtains quadratic convergence in this case.
- For small residuals, the approximate Hessian is still a fairly accurate approximation of the exact one, which leads to good convergence properties.

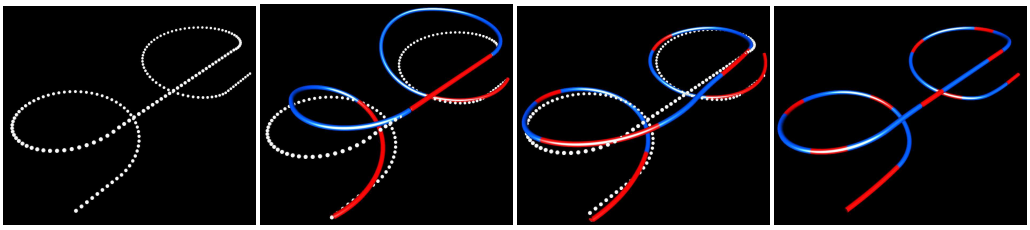
5 Examples

The first 5 figures in this section (Fig. 4-8) present five examples which demonstrate the performance of our algorithm. All computations were done on a PIV-1.73GHz PC with 1.0GB RAM. In the figures, the white points are the input data points. The blue and red curve segments are the biarcs and the primary circular arcs, respectively. The error threshold ε and maximum iteration number δ are specified by 0.001 and 100, respectively.



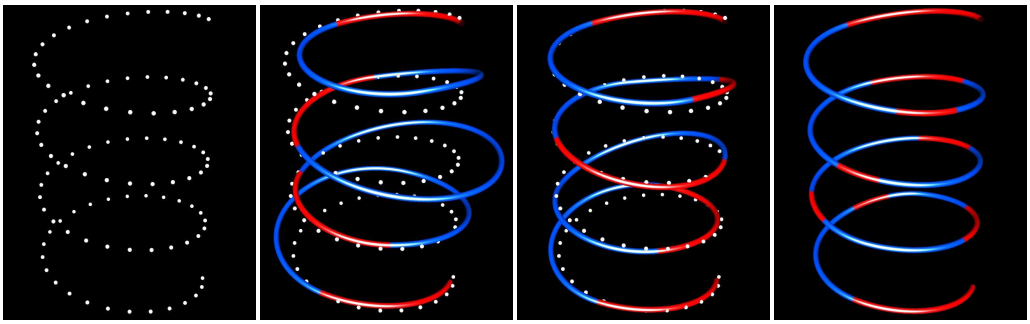
data points	initial curve	intermediate result	final curve
(100 points)	(4 arcs)	after 15 iterations	(13 arcs)

Fig. 4. Example 1 ("spline"): 100 data points were sampled from a cubic spline curve. The approximating circular spline was computed in 0.968 seconds.



data points	initial curve	intermediate result	final curve
(174 points)	(7 arcs)	after 15 iterations	(28 arcs)

Fig. 5. Example 2 ("glasses"): 174 points were sampled from a space curve which consists of several line segments and quadratic curve segments. The approximating circular spline was computed in 1.875 seconds.



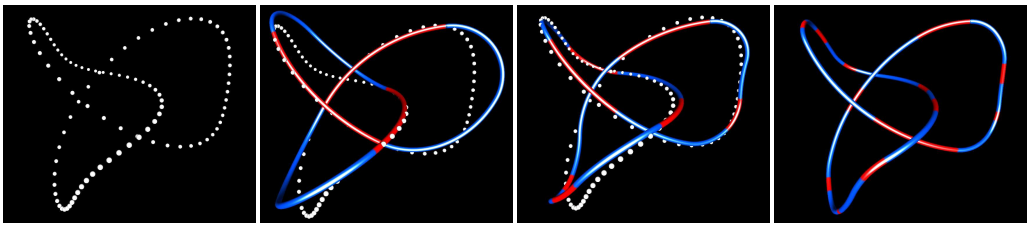
data points	initial curve	intermediate result	final curve
(100 points)	(10 arcs)	after 15 iterations	(25 arcs)

Fig. 6. Example 3 ("helix"): 100 points were sampled from a helix. The approximating circular spline was computed in 1.375 seconds.

The error reduction during the evolution is shown in Fig. 10, where the error is scaled with the diameter of the bounding box.

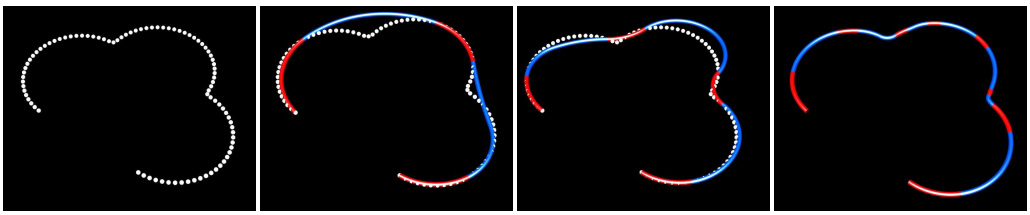
Finally, we discuss how the computational cost of the algorithm increases with the number of data points and with the number of arcs.

First, we sample the data points from the helix curve which is used in Example 6, and increase the number of data points from 100 to 10,000 with



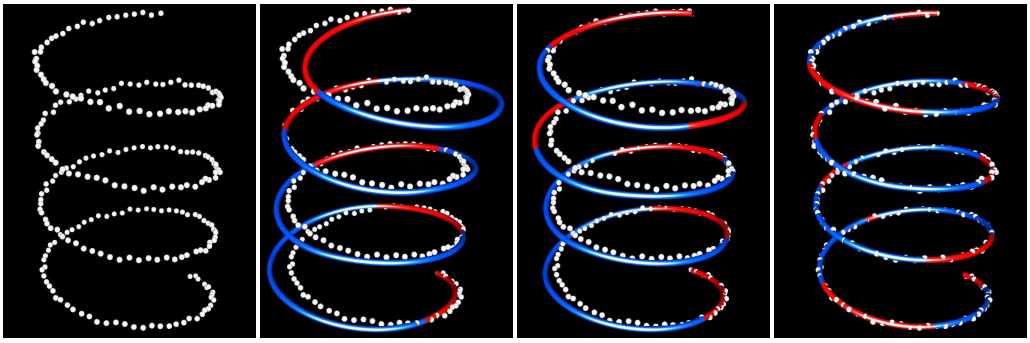
data points (150 points) | initial curve (9 arcs) | intermediate result after 15 iterations | final curve (36 arcs)

Fig. 7. Example 4 ("knot"): 150 points were sampled from a closed space curve. The approximating circular arc spline curve was computed in 1.657 seconds.



data points (90 points) | initial curve (7 arcs) | intermediate result after 15 iterations | final curve (28 arcs)

Fig. 8. Example 5 ("corner"): 90 data points were sampled from a curve with two sharp corners. The approximating circular spline was computed in 0.798 seconds.



data points (250 points) | initial curve (13 arcs) | intermediate result after 15 iterations | final curve (28 arcs)

Fig. 9. Example 6 ("noisy helix"): 100 points were sampled from a helix and artificial noise was added. The approximating circular spline was computed in 4.078 seconds.

roughly the same number of arcs. Given the same control threshold of error, the computation time depends approximately linearly on the number of data, see Table 1. This is due to the fact that the computational effort is dominated by the closest point computation, whose effort grows linearly with the number of points, provided that the number of arcs remains constant.

Second, we study the effect of an increasing number of arcs, by sampling data from a helix with an increasing number of turns, sampled with constant density. The number of arcs grows linearly with the number of data points. The approximation process becomes slower for a larger number of points, since

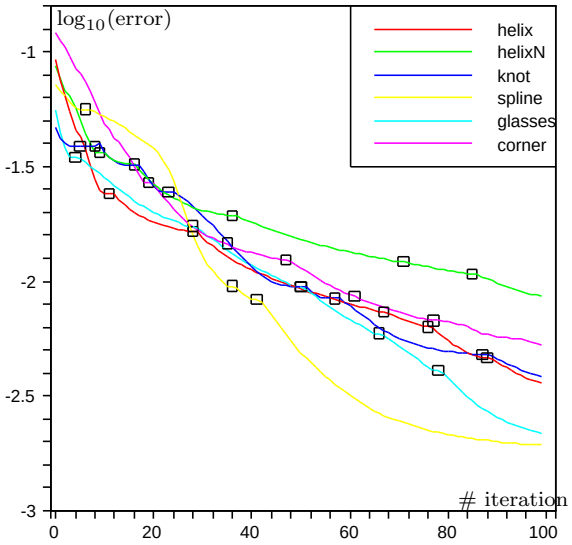


Fig. 10. Reduction of the average error per point during the evolution for the 5 examples in Fig. 4-8. The small boxes indicate the refinement events.

Table 1
Computation times in the helix example for increasing number of points but constant number of arcs

# points	100	1,000	10,000
time (sec)	1.4	17.7	142.0

Table 2
Computation times in the helix example for increasing number of points and increasing number of arcs.

# points	100	200	300	400	500	600	700	800	900	1000
# arcs	25	49	76	100	124	151	175	199	226	250
time (sec)	1.4	3.3	6.7	15.1	20.7	28.4	39.1	50.3	58.9	74.6

the closest point computation needs more time, see Table 2. The computation time grows superlinear with the number of points.

This is again due to the fact that the computational effort is dominated by the closest point computation. In the current implementation, in order to compute the closest point associated with a given one, we test all arcs whether or not they contain the closest point. This should be accelerated by using a suitable hierarchy of bounding volumes, which helps to identify a smaller number of candidate arcs for each point. This, however, has not yet been implemented, and the computation is also relatively fast without it.

6 Conclusion

We proposed an evolution method for approximating a given set of organized space points by a circular spline curve. During the evolution process, the circular spline curve converges dynamically to a stable limit shape. We proved the evolution process is in fact a Gauss-Newton type method. The technique of constructing a closed circular spline curve has also been discussed.

References

[1] O. Aichholzer, F. Aurenhammer, T. Hackl, B. Jüttler, M. Oberneder, and Z. Šír. Computational and structural advantages of circular boundary representation. In F. Dehne, J.-R. Sack, and N. Zeh, editors, *WADS*, volume 4619 of *Lecture Notes in Computer Science*, pages 374–385. Springer, 2007.

[2] M. Aigner, Z. Šír, and B. Jüttler. Evolution-based least-squares fitting using Pythagorean hodograph spline curves. *Comput. Aided Geom. Design*, 24(6):310–322, 2007.

[3] U. Bauer and K. Polthier. Parametric reconstruction of bent tube surfaces. In *CyberWorld 2007 Conference Proceedings, Workshop New Advances in Shape Analysis and Geometric Modeling, IEEE 2007*, 2007. to appear.

[4] A. Blake and M. Isard. *Active Contours*. Springer, New York, 1998.

[5] R. T. Farouki. *Pythagorean-hodograph curves: algebra and geometry inseparable*, volume 1 of *Geometry and Computing*. Springer, Berlin, 2008.

[6] J. Hoschek. Circular splines. *Comput.-Aided Des.*, 24:611–618, 1992.

[7] M. Kass, A. Witkin, and Terzopoulos. D. Snakes: active contour models. *Int. J. Comput. Vis.*, 24:321–331, 1987.

[8] Y. Liu and W. Wang. A revisit to least squares orthogonal distance fitting of parametric curves and surfaces. In F. Chen and B. Jüttler, editors, *Advanves in Geometric Modelling and Processing*, volume 4975 of *Lecture Notes in Computer Science*, pages 384–397. Springer, 2008.

[9] D. S. Meek and D. J. Walton. Approximation of discrete data by G^1 arc splines. *Comput.-Aided Des.*, 24:301–306, 1992.

[10] D. S. Meek and D. J. Walton. Approximating smooth planar curves by arc splines. *J. Comput. Appl. Math.*, 59(2):221–231, 1995.

[11] D. S. Meek and D. J. Walton. Planar osculating arc splines. *Comput. Aided Geom. Design*, 13(7):653–671, 1996.

[12] D. S. Meek and D. J. Walton. An arc spline approximation to a clothoid. *J. Comput. Appl. Math.*, 170(1):59–77, 2004.

[13] A. W. Nutbourne and R. R. Martin. *Differential geometry applied to curve and surface design, Vol. 1.* Ellis Horwood, Chichester, 1988.

[14] L. A. Piegl and W. Tiller. Data approximation using biarcs. *Eng. Comput.*, 18:59–65, 2002.

[15] M. A. Sabin. The use of circular arcs to form curves interpolated through empirical data points. Technical Report VTO/MS/164, British Aircraft Corporation, 1976.

[16] W. Wang and B. Joe. Robust computation of the rotation minimizing frame for sweep surface modeling. *Comput.-Aided Des.*, 29(5):379–391, 1997.

[17] W. Wang, H. Pottmann, and Y. Liu. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM Trans. Graph.*, 25(2):214–238, 2006.

[18] S. N. Yang and W. C. Du. Numerical methods for approximating digitized curves by piecewise circular arcs. *J. Comput. Appl. Math.*, 66(1-2):557–569, 1996.