FLORIDA INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

TECHNICAL REPORT: CS-2013-01

Minimizing N-Points Interpolation Curvature, Heuristics for Solutions Using Arcs and Lines

Rahul Vishen, Florida Institute of Technology
Marius C. Silaghi, Florida Institute of Technology

Knowing a set of points on a curve, the interpolation problem is to hypothesize the location of the intermediary ones. A large set of interpolation techniques are known. We address the problem of generating a path with minimal maximum curvature, passing through N ordered points and joining the two end-points at predefined directions. This is related to R-geodesics, which have been used to generate paths with minimum average curvature between two given points that have to be joined at predefined directions and curvature.

For example, when interpolating GPS points to reconstruct a vehicle's trajectory, we may know that the centripetal acceleration is upper bounded due to physical constraints, hence adding constraints on the trajectory curvature. Among two interpolations with the same maximum curvature, we prefer the one with shorter trajectory.

We compare experimentally several interpolations techniques, and propose heuristics to generate paths based on concatenated arc and line segments (also known as R-geodesics) inferred based on tuples of three consecutive points. Benchmarks with over 1000 simulated and real scenarios show that this algorithm is 73% percent better then the next candidate method we propose and which is based on bi-arcs with hill-climbing. A remaining open question is whether a global optima can be achieved and proven.

1. INTRODUCTION

Interpolation is a very common problem occurring in various fields ranging from graphics design and art to learning models in artificial intelligence, and path planning in robotics. Interpolation is the problem of fitting a curve to a set X of N given data points, described by a parametric function $\mathcal{C}_X(t)$. The points may be defined in an n-dimensional space, as common in learning models for artificial intelligence, but we mainly focus on the 2-dimensional cases occurring in graphics and vehicle movements.

While common interpolation techniques focus on optimizing length, or on guaranteeing smoothness, we want to guarantee an upper bound on the trajectory curvature while traversing all the N points. We propose and analyze variations of methods generating such interpolations by concatenating path segments built from two or three points at a time, and compare them with more straightforward adaptations of general interpolation algorithms.

The guarantee of an upper bound on curvature is particularly important when the curve represents the recorded or planned trajectory of a vehicle or robot. Most vehicles are subject to a limit on the maximum supported acceleration. For example, we know that human beings found in such vehicles can only sustain an acceleration of some $4.0g$ [Voshell 2004]. Robots can suffer damage if subjected to accelerations above their designed limits. Additional practical limits on the changes in this acceleration are often handled separately using Euler Spirals (clothoids) [Makino 1988; Yao and Joneja 2007].

We first analyze interpolated trajectories obtained using cubic Bézier curves, splines, and bi-arcs:

— The Bézier interpolation is generated by creating two control points between each pair of original data points. The two control points next to each data point are chosen on the line passing through the data point and parallel to the segment joining the two neighboring data points.
— As an example of spline interpolation we use the variation described in [Yu et al. 2004], which was proposed in the past for the reconstruction of vehicle trajectories from sparse GPS locations.

— A C^1 *continuous curve* is a curve whose first derivative is continuous [Bartels et al. 1987]. A Bi-arc is a C^1 continuous piecewise curve between two end-points with predefined direction tangents, that is composed of two consecutive arcs. We implement the two versions of bi-arc algorithms described in [Rossignac and Requicha 1987] and [Koc et al. 2000a].

Second, we design an interpolation technique tuned to return a C^1 class curve with a small maximal curvature wherever the curvature is defined. Ideally, we want to approach:

$$\underset{\mathcal{C}_X}{\texttt{argmin}} \max_t curvature\left(\mathcal{C}_X(t)\right)$$

The algorithm constructs a curve by concatenating a set of line and arc segments that pass through the given data points while maintaining continuity and smoothness of first order (first derivative of the curve is continuous). An initial and a final direction can be specified. Unlike the related technique used in Dubins curves [Dubins 1957] where directions at each point are predefined, we have to compute the directions at the intermediary points. An arc segment is associated with each data point and each two consecutive arc segments are joined via a common tangent segment. Each arc or line segment can be of zero length.

The proposed interpolation techniques proceed sequentially from the first to the last data point (hierarchically iterating a greedy algorithm). At each point, the arc segment is selected first by considering the previous arc segment and the next data point. A set of constraints on the arc segment and its supporting circle are defined for various possible configurations (relative positions of the previous arc and next data point/direction).

The variations studied for the proposed algorithms are evaluated both: using benchmarks of GPS point sequences recorded from a vehicle, as well as on benchmarks of randomly generated points. For randomly generated points, each point is generated either independently of the previous position, or relative to the previous point. The benchmarks used in the evaluation as well as the implementation of best proposed version of the new interpolation software is made available at [Vishen and Silaghi 2013].

In the next section we describe some existing interpolation techniques. Next we introduce the concepts involved in the newly proposed methods based on arc and line segments. Subsequent sections detail the constraints used for selecting arc segments for 2-dimensional. We conclude with experimental evaluations.

2. BACKGROUND

In the absence of a tightly constrained model, curves are fitted to data points by techniques such as: linear interpolation [Evans and Kim 1998; Blu et al. 2004], Bézier interpolation [Shao and Zhou 1996; Piegl 1987; Forrest 1972], splines [Unser et al. 1991; Schumaker 1983] and Dubbin curves [Dubins 1957]. We adopt as metrics of interest the maximum curvature and path length [Dubins 1957].

Linear interpolation (see Figure 1) is the shortest curve traversing each data point. The main drawback from our perspective is that obtained paths can have infinite curvature at connection points, being unrealistic for applications such as vehicle trajectory modeling.

Bézier interpolation was developed at Renault for designing smooth surfaces for automobiles. While they are easy to manipulate by designers using control points, their curvature is more difficult to limit, specially when interpolating automatically a large number of data points. Points on a Bézier interpolation are constructed by weighting the involved control points with variable weights (see Figure 2). The control points give the initial and final direction of the curve. A variation weights corresponding points

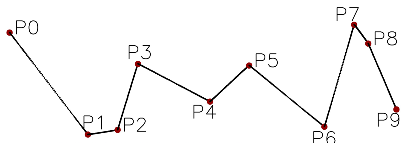


Fig. 1. Linear Interpolation

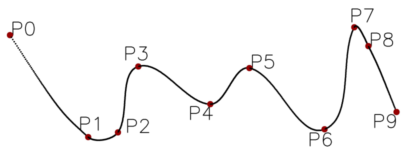


Fig. 2. Bézier curve based interpolation.

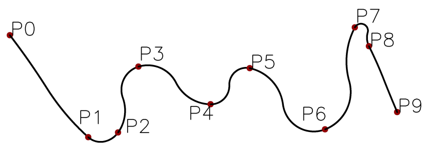


Fig. 3. Bi-arc based interpolation.

on a set of circles [Sequin et al. 2005]. The circles give initial and final curvatures but cannot control intermediary curvature.

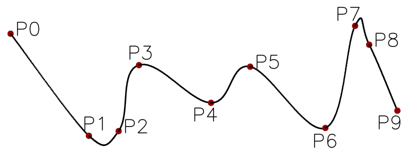


Fig. 4. Splines based interpolation.

Spline interpolation is used to fit a polynomial (or a different function) to pass through a set of points (Figure 4). Some of them can put bounds on the second derivatives, avoiding the need of adjustments based on clothoids [McCrae and Singh 2009; Bianco and Piazzi 2001; Meek and Walton 1992; Shin and Singh 1990].

Kalman filters have been commonly used to interpolate vehicle trajectory under assumptions of linearity. These assumptions are stronger than our assumptions for the problem. The Kalman filters are easily used to estimate points at given instants, but do not yield C^1 continuous curves, presenting discontinuities when new information is integrated.

A problem that is related to interpolation is the design of a path joining two given points at specified direction and curvature, and having a bounded average curvature. This problem appears when joining two highways, and is studied in [Dubins 1957; Boissonnat et al. 1992; Rossignac and Requicha 1987]. Such a path of minimal length is called an R-geodesic. Several R-geodesics can be concatenated into a single continuous path to interpolate N points with given direction and curvature at each point, and the result is called a Dubins curve [LaValle 2006]. Such paths do not have a curvature at each point, and the techniques minimize the *average curvature* [Dubins 1957] computed over the points where curvature is defined. Dubins curves have been used befor to plan paths for robotic vehicles [Barraquand and Latombe 1989; Mirtich and Canny 1992; Laumond et al. 1994; Svestka and Overmars 1995]. To use Dubins curves for the interpolation of N points, first one has to find the optimal directions at intermediary points. Bi-arcs are a common member of the family of Dubbin's curves where interpolation is done by concatenating only arc segments, being less general but easier to create [Rossignac and Requicha 1987] (Figure 3). Bi-arcs have been used for data approximation [Piegl and Tiller 2002] and interpolation [Lee et al. 2007; Koc et al. 2000b; Schönherr 1993]

3. NOTATIONS

Here we provide a summary of the notations introduced in more detail in the subsequent sections. The reader can return here for reference.

— $\langle x_k, y_k\rangle$**:** A point P_k at coordinates x_k and y_k.
— $||UV||$**:** The Euclidean distance between point U and point V.
— $\mathcal{K}_i$**:** The maximum curvature of a path segment $[P_iP_{i+1}]$.
— $\mathcal{P}$**:** A parabola.
— $\vec{j}$**:** A direction is represented using a vector, e.g., $\vec{j}$.
— $|P_iP_j|$**:** A line segment between points P_i and P_j.
— $|\overline{P_iP_j}|$**:** A directed line segment for $|P_iP_j|$ starting at P_i and ending at P_j.
— $\widehat{AC}$**:** The directed arc from A to C.
— C_i**,** O_i**,** $\langle x_i^C, y_i^C\rangle$**,** R_i**:** A center of a circle C_i is denoted by $O_i = \langle x_i^C, y_i^C\rangle$, and its radius by R_i.
— O_i^k**:** The k^{th} candidate for O_i.
— d_i**:** A circle C_i has an associated direction.
— $t_{C_j}^{C_i}$**,** T_j^s**,** T_i^e**:** By $t_{C_j}^{C_i}$ we denote the unique common tangent segment of circles C_i and C_j, that leaves from circle C_i with the direction d_i in point T_i^e and joins the circle C_j with direction d_j at point T_j^s.
— $t_{C_j}^{P_i}$**:** A tangent from point P_i to circle C_j, joining the circle with its direction d_j.
— $t_{P_j}^{C_i}$**,** T_i**:** By $t_{P_j}^{C_i}$ we denote a tangent from circle C_i to point P_j, leaving the circle with its direction d_i at point T_i.
— θ_i^s**,**θ_i^e **:** We denote by θ_i^s the angle on C_i where T_i^s is found, and with θ_i^e the angle on C_i where we find T_i^e.
S_i the support line of the tangent segment $t_{C_{i-1}}^{P_i}$ from point P_i to circle C_{i-1},
— α_i**:** The angle between the support line S_i and the segment $|P_iO_i|$.
iff if and only if.

4. INTERPOLATION USING ARCS AND TANGENTS

In this section we introduce the main concepts and techniques we use to interpolate N points with bounded maximal curvature.

4.1. Formal problem definition

We are given a set of $N+1$ points: $P_0, ..., P_N$. Each point $P_i = \langle x_i, y_i \rangle$. The problem is to generate a C^1 continuous trajectory passing through each of these points in order and having a upper bound $\mathcal{K}_i$ on the curvature at each point P_i. The curvature on each trajectory segment $[P_i P_{i+1}]$ has to be upper bounded by $max(\mathcal{K}_i, \mathcal{K}_{i+1})$. The trajectory has to start in P_0 according to a given direction vector $\vec{j}$, and has to reach P_N along a given direction vector $\vec{o}$.

4.2. Concepts

In the solution proposed here, each given point $P_i, i \in \{0, ..., N\}$ is associated with a circle C_i of center O_i of coordinates $\langle x_i^C, y_i^C \rangle$. Each point P_i belongs to the circle C_i. The final trajectory consists of a sequence of tangents to these circles together

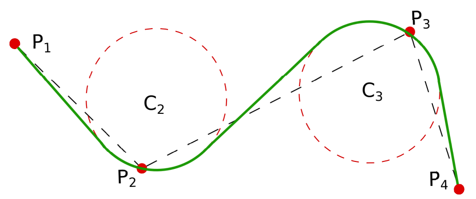


Fig. 5. Basic idea.

with the arcs connecting the points of tangency (see Figure 5). Each arc i contains the corresponding point P_i, and is connected to the next arc using a common tangent. We achieve a trajectory of curvature $\mathcal{K}_i$ from P_i to P_{i+1} iff circle C_i and C_{i+1} satisfy the condition:

$$\min(R_i, R_{i+1}) \geq \frac{1}{\mathcal{K}_i}.$$

We associate each circle C_i with a direction.

Definition 4.1 (*Directed Circle*). A directed circle is a circle associated with a direction $d_i \in \{1, -1\}$ where (1) stands for *clockwise*, and (-1) stands for *counterclockwise*.

We also associate line segments $|AB|$ with a direction obtaining a directed segment $|\overline{AB}|$ (a vector with the point of application in A) [Larson et al. 2012]. We say of a directed line segment $|\overline{AB}|$ that it leaves a circle iff it is tangent to that circle at A. We say that it joins a circle iff it is tangent to that circle at B.

A directed line segment has, with reference to a directed circle, a direction defined as follows. A directed line segment $|\overline{AB}|$ *joins or leaves a directed circle* C_i with direction 1 iff the center of C_i is on the right-hand side of $|\overline{AB}|$. A directed line segment $|\overline{AB}|$ joins or leaves a directed circle C_i with direction -1 iff the center of C_i is on the left-hand side of $|\overline{AB}|$. Tangent line segments are considered to be directed:

Definition 4.2 (*Directed Tangent*). By $t_{C_j}^{C_i}$ we denote the unique common tangent segment of circles C_i and C_j, that leaves from circle C_i with the direction d_i in point T_i^e and joins the circle C_j with direction d_j at point T_j^s. By $t_{C_j}^{P_i}$ we denote a tangent from point P_i to circle C_j, joining the circle with its direction d_j. By $t^{C_i}_{P_j}$ we denote a tangent from circle C_i to point P_j, leaving the circle with its direction d_i at point T_i.

LEMMA 4.3. *For any given pair of distinct directed circles there exists at most one directed common tangent leaving from the first circle and joining the second circle.*

PROOF. Of the 8 possible directed tangents, only 2 are compatible with each combination of possible directions for the two circles. Of the 2 compatible tangents, only one starts from the first circle. □

Arcs on circles. Denoting with $step()$ the common step function, the angle where P_i is positioned on C_i (between $-\frac{\pi}{2}$ and $\frac{3\pi}{2}$) is in Equation 1.

$$\theta_i = arctg\left(\frac{y_i - y_i^C}{x_i - x_i^C}\right) + \pi * step(x_i^C - x_i). \quad (1)$$

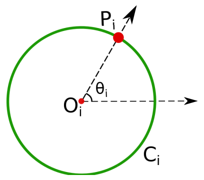


Fig. 6. Computing θ_i.

We denote by θ_i^s the angle on C_i where T_i^s is found, and with θ_i^e the angle on C_i where we find T_i^e. Angles are measured counter-clockwise with respect to the abscissa axis (See Figure 6).

Definition 4.4. If A, B, and C are three points on a directed circle C_i, we say that $B \in \widehat{AC}$ iff B is between A and C when traveling on the circle in its direction from A to C.

Remark 4.5 (*Self Intersecting Arc*). A trajectory does not traverse a circle completely only if P_i is on the arc between the point where the trajectory joins C_i, T_i^s, and the point where it leaves C_i, T_i^e i.e., $P_i \in \widehat{T_i^s T_i^e}$. Computationally, $P_i \in \widehat{T_i^s T_i^e}$ holds iff:

$$\overline{d}_i * \theta_i \in [\overline{d}_i * \theta_i^s, \overline{d}_i * \theta_i^e] \vee \left(\overline{d}_i * \theta_i^s \geq \overline{d}_i * \theta_i^e \wedge \overline{d}_i * \theta_i \notin (\overline{d}_i * \theta_i^e, \overline{d}_i * \theta_i^s)\right),$$

where $\overline{d}_i$ stands for $-d_i$.

4.3. General Idea

In order to build a trajectory that passes through the points $P_0, ..., P_N$ and has a curvature upper bounded by $\mathcal{K}_i$ in P_i and by $max(\mathcal{K}_i, \mathcal{K}_{i+1})$ on each segment P_iP_{i+1} one can draw a set of $N+1$ circles, C_0,...,C_N, each C_i having radius $R_i \geq \frac{1}{\mathcal{K}_i}$ such that $P_i \in C_i$. This is unlike previous approaches to building R-geodesics [Dubins 1957; Rossignac and Requicha 1987] where each segment is taken separately, and original data points end up being the connection points of circles.

While in the rest of the paper we discuss heuristics for selecting circles C_i, first we present the equations for the components of the paths, namely tangent segments and points of tangency, once the circles are decided. We start with the simpler case of identical radii (occurring when the bound on the curvature is constant throughout the interpolation), followed by the general case.

Identical Radius. Each ordered sequence two distinct directed circles C_{i-1} and C_i of identical radius $\mathcal{R}$ and identical direction d are joined by exactly one common directed tangent of equation:

$$\frac{y - y^C_{i-1} - \delta_i}{\Delta_{y_i}} = \frac{x - x^C_{i-1}}{\Delta_{x_i}}$$

where

$$\Delta_{y_i} = y^C_i - y^C_{i-1}$$

$$\Delta_{x_i} = x^C_i - x^C_{i-1}$$

$$\Delta_i = \sqrt{\Delta^2_{y_i} + \Delta^2_{x_i}}$$

$$\delta_i = d * \mathcal{R} * \frac{\Delta_i}{\Delta_{x_i}}$$

The corresponding points of tangency are:

$$\begin{aligned} T^e_{i-1} &= \langle x^C_{i-1} + \mathcal{R} * \frac{\Delta_{y_i}}{\Delta_i}, y^C_{i-1} + \mathcal{R} * \frac{\Delta_{x_i}}{\Delta_i}\rangle \\ T^s_i &= \langle x^C_i + \mathcal{R} * \frac{\Delta_{y_i}}{\Delta_i}, y^C_i + \mathcal{R} * \frac{\Delta_{x_i}}{\Delta_i}\rangle \end{aligned}$$

Different Radii. For circles of different radii and directions, the points of tangency are [Krasilnikov 2010]:

$$\begin{aligned} T^e_{i-1} &= O_{i-1} + R_{i-1} * \vec{n} \\ T^s_i &= O_i + d_i * d_{i-1} * R_i * \vec{n} \end{aligned}$$

where $\vec{n}$ is the solution of:

$$\overrightarrow{O_{i-1}O_i} * \vec{n} = R_{i-1} - d_i * d_{i-1} * R_i \tag{2}$$

Out of the two possible solutions of Equation 2, we select one such that:

$$sign(\overrightarrow{O_{i-1}O_i} \times \vec{n}) = d_{i-1} \tag{3}$$

where $\times$ is the cross-product of the vectors.

Assembling the trajectory from circles. The final trajectory is built according to the steps in Algorithm 1.

```
procedure TrajectoryFromCircles($P_0$,...,$P_N$,$C_0$,...,$C_N$) do
    start with $P_0$;
    forever do
        stop when you are in $P_N$;
        when you are in $P_i$ go on $C_i$ to $T^e_i$;
        when you are in $T^e_i$ go on $t^{C_i}_{C_{i+1}}$ to $T^s_{i+1}$;
        when you are in $T^s_i$ go on $C_i$ to $P_i$;
```

ALGORITHM 1: Algorithm to follow a path given N+1 points on N+1 circles

Remark 4.6 (*Directed endpoints*). The circles C_0 and C_N have to be selected such that they are tangent to the direction vector $\vec{\jmath}$at P_0, and to $\vec{o}$ at P_N, respectively.

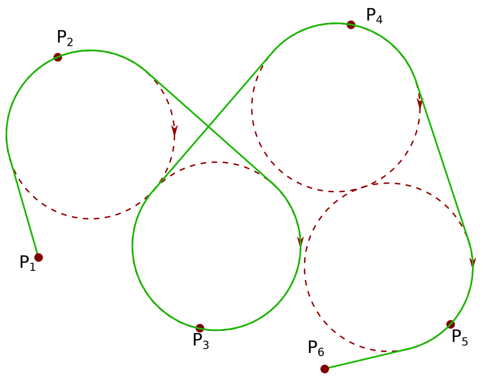


Fig. 7. Same direction circles, with random selection of centers.

Free endpoints. In case there is no constraint to start or end the trajectory according to vectors $\vec{j}$or $\vec{o}$, only points $P_i, i \in \{1, ..., N-1\}$ are associated with circles C_i. This can be modeled by setting circles C_0 and C_N to have radius 0. The trajectory contains a tangent from point 0 to circle C_1 and from circle C_{N-1} to point P_N.

5. HEURISTICS FOR ASSOCIATING CIRCLES TO POINTS

We introduce the studied heuristics sequentially, starting from the simpler ones.

5.1. Circles of same direction

First we cover increasingly sophisticated heuristics for selecting circles where these circles have identical direction. The notation is kept general from the beginning, (e.g. the direction of the circle C_i is still denoted d_i) since the equations will be referred in subsequent sections.

Centers at constant translation. A first simple heuristic to select the circles is to select each center O_i as a translation of P_i along the negative direction of the ordinate axis. Therefore, C_i can be defined as the clockwise[1] directed circle:

$$(x - x_i^C)^2 + (y - y_i^C)^2 = R_i^2$$

where $x_i^C = x_i$ and $y_i^C = y_i - R_i$.

Circles tangent at the given points. An observation is that the trajectories obtained using the previous heuristic self-intersect in many cases (as per Remark 4.5). Figure 8 illustrates an example of such self-intersections and of the opportunities to reduce the obtained path length. One of the reasons for which this happens is that frequently the point P_i is relatively far from T_i^s on C_i. The next discussed strategy is to select the circles such that $P_i = T_i^s$. This happens if the circles are selected to be tangent in P_i at $t_{P_i}^{C_{i-1}}$.

C_1 is built to be tangent to P_0P_1 at P_1. The equation of P_0P_1 is:

$$(y - y_0)(x_1 - x_0) = (x - x_0)(y_1 - y_0). \tag{4}$$

[1]If the translation is along the negative direction of the ordinate axis then the direction of the circle would be selected as counter-clockwise.

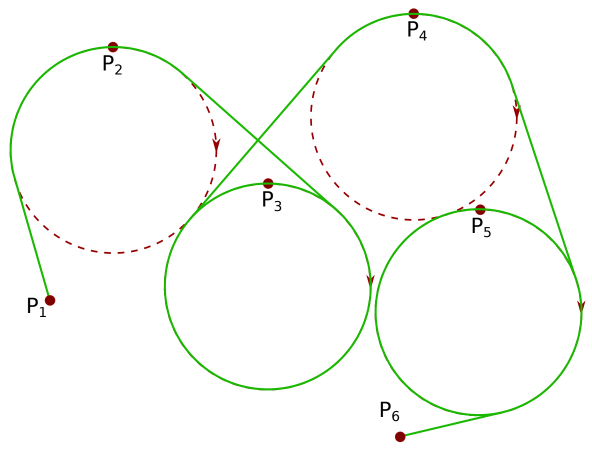


Fig. 8. Same direction circles, with centers under the points.

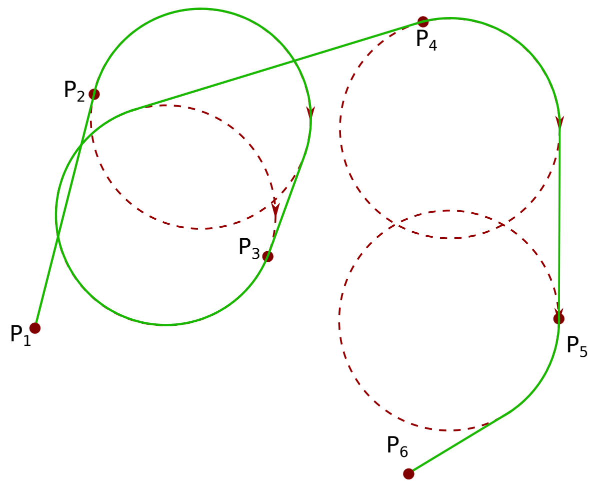


Fig. 9. Same direction circles tangent at the given point.

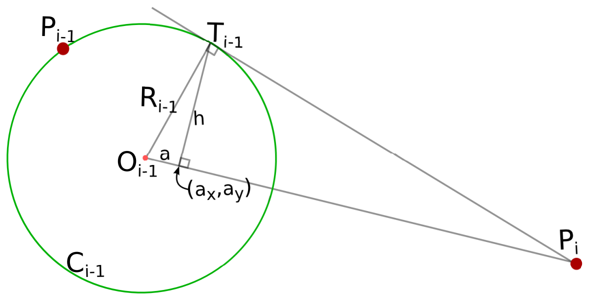


Fig. 10. Computing T_{i-1}.

The equation of $t_{P_i}^{C_{i-1}}$ (using Figure 10) is:

$$a = \frac{R_{i-1}^2}{||P_i O_{i-1}||}$$

$$h = \sqrt{R_{i-1}^2 - a^2}$$

$$a_x = x_{i-1}^C + \frac{a}{||P_i O_{i-1}||} * (x_i - x_{i-1}^C)$$

$$a_y = y_{i-1}^C + \frac{a}{||P_i O_{i-1}||} * (y_i - y_{i-1}^C).$$

$$T_{i-1} = \langle a_x - \frac{d_i * h * (y_i - y_{i-1}^C)}{||P_i O_{i-1}||}, a_y + \frac{d_i * h * (x_i - x_{i-1}^C)}{||P_i O_{i-1}||} \rangle$$

$$(y - T_{i-1}(y))(x_i - T_{i-1}(x)) = (x - T_{i-1}(x))(y_i - T_{i-1}(y)) \tag{5}$$

The perpendicular on $t_{P_i}^{C_{i-1}}$ in P_i is:

$$(x - x_i) * \frac{x_i - T_{i-1}(x)}{-||T_{i-1} P_i||} = (y - y_i) * \frac{y_i - T_{i-1}(y)}{||T_{i-1} P_i||} \tag{6}$$

Therefore $d_i = d_{i-1}$ and:

$$O_i = \langle x_i + d_{i-1} * R_{i-1} * \frac{y_i - T_{i-1}(y)}{||T_{i-1} P_i||}, y_i - d_{i-1} * R_{i-1} * \frac{x_i - T_{i-1}(x)}{||T_{i-1} P_i||} \rangle$$

5.2. Circles of different directions

In the next improved heuristic we select d_i based on the position of the center O_{i-1} and the next point P_{i+1} with respect to $t_{P_i}^{C_{i-1}}$. If the center O_{i-1} and point P_{i+1} lie on the same side of the tangent $t_{P_i}^{C_{i-1}}$, then $d_i = d_{i-1}$, otherwise $d_i = -d_{i-1}$. In the case when the they are both on the line supporting $t_{P_i}^{C_{i-1}}$ then d_i is set to a default value.

The function $sign(U, A, B)$ returns 1 if U is a point on the left hand side of the vector $\overrightarrow{AB}$, and -1 if it is on its right. If U is a point on the support of the vector $\overrightarrow{AB}$ then the function returns a default value (1 or -1). If:

$$side(U, A, B) = sign((y_u - y_A)(x_B - x_A) - (x_U - x_A)(y_B - y_A)). \tag{7}$$

$$d_i = side(O_{i-1}, T_{i-1}, P_i) * side(P_{i+1}, T_{i-1}, P_i) * d_{i-1} \tag{8}$$

then

$$O_i = \langle x_i + d_i * R_{i-1} * \frac{y_i - T_{i-1}(y)}{||T_{i-1} P_i||}, y_i - d_i * R_{i-1} * \frac{x_i - T_{i-1}(x)}{||T_{i-1} P_i||} \rangle \tag{9}$$

5.3. Reducing the number of loops

Next we introduce heuristics for selecting circles. These heuristics aim to reduce interpolation length at a given maximum curvature. Some of the evaluated heuristics are designed to avoid selecting circles that require arc segments larger than 2π (as per Remark 4.5).

Remark 5.1. We do not guarantee that we completely avoiding loops on any trajectory segment between two given points, but our procedure reduces their occurrences to the case where consecutive points are very close to each other (relative to the inverse of the curvature), as well as for special starting or terminating directions. This is illustrated in the experimental section.

The described heuristic is greedy. Circles are defined sequentially starting with circle C_0.

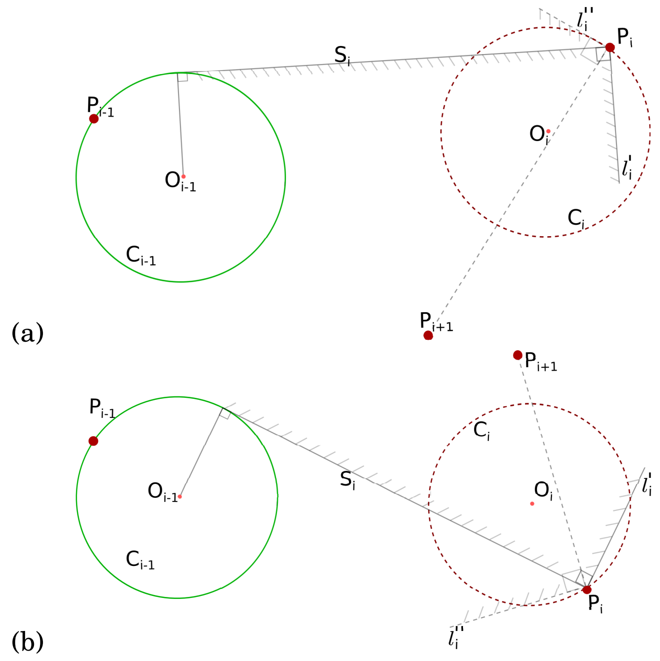


Fig. 11. Selecting the side of the center: (a) O_{i-1} and point P_{i+1} on the same side of S_i, (b) O_{i-1} and point P_{i+1} on the opposite sides of S_i.

CONDITION 1. *At each point P_i, the circle is built to respect the following conditions (Figure 11):*

i) the circle circumference contains point P_i.
ii) the circle's center is interior to the angle formed by:
  a) the support line S_i of the tangent $t_{C_{i-1}}^{P_i}$ from point P_i to circle C_{i-1} (Equation 5), or of the segment $|P_0P_1|$ for circle C_1, (Equation 4).
  b) the perpendicular l_i' on S_i in the point P_i, found in the half-plane bounded by S_i and containing P_{i+1} (see Equation 6 and Section 5.2). In case $P_{i+1} \in S_i$ then either half-plane may be selected, but the half-plane not containing P_{i+2} is preferred (e.g., replacing P_{i+1} with P_{i+2} in Equation 9 and 8).
iii) the circle's center is also interior to the angle formed by:
  a) the semi-line l_i' at condition ii.b)
  b) the perpendicular l_i'' in P_i on P_iP_{i+1} and located in the half-plane bounded by S_i and containing P_{i+1}.

iv) d_i *is 1 if the center of* C_i *(or* P_{i+1}*) is to the right of directed segment* $t_{P_i}^{C_{i-1}}$*, and -1 otherwise (see Equation 7 and 8).*

We have two cases, namely where the current circle has the same direction as the previous one ($d_i = d_{i-1}$ shown in Figure 11.a), or when the new circle has opposite direction ($d_i \neq d_{i-1}$ as in Figure 11.b). When $P_{i+1} \in S_i$, one of the other two cases is picked randomly.

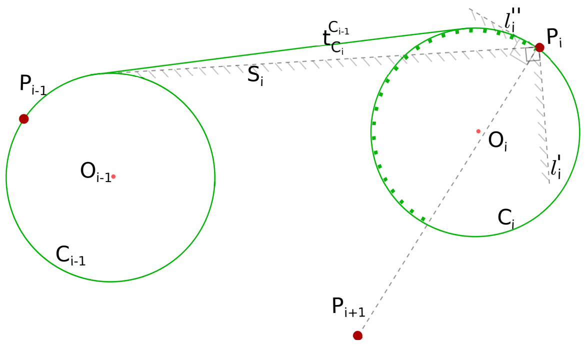


Fig. 12. Conditions for Same Direction Circle

Case I: Same direction circles. When the circles C_{i-1} and C_i are on the same side of S_i, we first attempt to ensure that T_{i-1}^e is in the first half-circle after P_{i-1} while T_{i-1}^s is in the first half-circle prior P_{i-1}. Whenever these preferences can be accomplished, then $T_{i-1}^e \notin \widehat{T_{i-1}^s P_{i-1}}$. We therefore give preference to selections of C_i such that it is intersected by the common tangent $t_{C_i}^{C_{i-1}}$ in its half-circle prior to P_i (see Figure 12).

CONDITION 2. *The center* O_i *is preferably selected on the interior bisector of the angle formed by* S_i *and* $|\overline{P_i P_{i+1}}|$.

LEMMA 5.2. *If circles are always selected according to Condition2, then for any intermediary point* P_{i-1}*,* P_i *will be on the side of* $|\overline{P_{i-1} O_{i-1}}|$ *for which*

$$side(P_i, P_{i-1}, O_{i-1}) = d_{i-1}$$

.

PROOF. Based on Conditions 1 and 2, as seen in Figure 12, the condition is satisfied for both possible cases by selection O_i such that P_{i+1} is on the right hand side of vector $P_i O_i$. Also, P_{i+1} will be on the same side of S_i as O_i, and therefore on the same side of the tangent in P_i (the angle $\angle O_i P_i P_{i+1} \leq 90$). □

LEMMA 5.3. *The fact that the point of tangency with* C_i *is in the half-circle prior to* P_i *is ensured by the paragraph (ii.b) in Condition 1.*

PROOF. See Figure 11. □

Let us denote with t_i the tangent to C_i in point P_i (i.e., t_{i-1} is the tangent to C_{i-1} in point P_{i-1} as in Figure 13). In order to ensure that T_{i-1}^e is in the half-circle after P_{i-1} (in the direction of the trajectory), the circle C_i should not intersect t_{i-1}. This condition is written as:

$$R_i \leq ||O_i t_{i-1}|| \qquad (10)$$

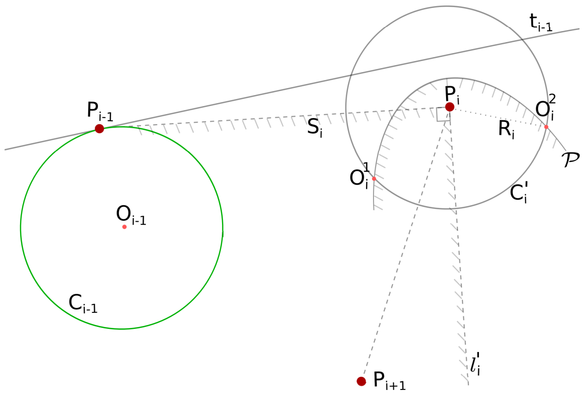


Fig. 13. Parabola Constraint

where by $||O_i t_{i-1}||$ we denote the distance from point O_i to line t_{i-1}. Equation 10 states that the center O_i of the circle C_i has to be found inside the parabola $\mathcal{P}$ defined by the t_{i-1} as directrix and P_i as its focus (Figure 13).

Let C'_i be a circle of center P_i and radius R_i. In case circle C'_i intersects parabola $\mathcal{P}$, let O_i^1 and O_i^2 be the intersections of $\mathcal{P}$ with the circle C'_i. The center O_i can be anywhere on C'_i between O_i^1 and O_i^2. We will select any of the two points, O_i^1 and O_i^2, that satisfy the Condition 1(ii).

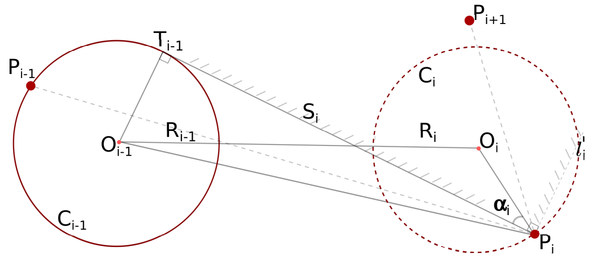


Fig. 14. Constraints for opposite direction circles.

Case II: Opposite direction circles. Now let us consider the case where circles C_{i-1} and C_i have opposite directions, as in Figure 14. We have to find constraints on the location of the circle center O_i such that C_i does not intersect or contain C_{i-1}. This constraint is needed in order for the two circles to have a common directed tangent in the context of Condition 1. We will express the constraint in terms of angle α_i between the support line S_i and the segment $|P_iO_i|$. The triangle $\Delta O_{i-1}P_iT_{i-1}$ is right angle (see Figure 14), therefore:

$$||O_{i-1}P_i|| = \sqrt{||P_iT_{i-1}||^2 + R_{i-1}^2}$$

In triangle $\Delta O_{i-1}O_iP_i$, in order for C_i to not intersect C_{i-1} we require that $R_{i-1} + R_i \leq ||O_{i-1}O_i||$. Based on cosine theorem:

$$(R_{i-1} + R_i)^2 \leq ||O_{i-1}O_i||^2 = ||P_iO_{i-1}||^2 + R_i^2 - 2||P_iO_{i-1}||R_i \cos \angle O_iP_iO_{i-1}$$

Therefore:

$$R_i \leq \frac{||P_iT_{i-1}||^2}{2R_{i-1} + 2||P_iO_{i-1}|| \cos \angle O_iP_iO_{i-1}}$$

and

$$\cos \angle O_iP_iO_{i-1} \leq \frac{||P_iO_{i-1}||^2 + R_i^2 - (R_{i-1} + R_i)^2}{2||P_iO_{i-1}||R_i}$$

$$\cos \angle O_iP_iO_{i-1} \leq \frac{||P_iT_{i-1}||^2 - 2R_{i-1}R_i}{2||P_iO_{i-1}||R_i}$$

The center O_i has to be in the semi-plane bounded by S_i not containing O_{i-1}. If we denote with b_i the value of $\max(-1, \min(1, \cos \angle O_iP_iO_{i-1}))$, we conclude that the center O_i has to be on the ray (half-line) bounded by P_i and forming with S_i an angle:

$$\alpha'_i \geq \arccos b_i - arctg \frac{R_{i-1}}{||P_iT_{i-1}||} \tag{11}$$

To satisfy Condition 1(iii), the center O_i has to be on the ray (half-line) bounded by P_i and forming with S_i an angle

$$\alpha''_i \geq \angle T_{i-1}P_iP_{i+1} - \frac{\pi}{2} \tag{12}$$

From Equations 11 and 12, the center O_i has to be on the ray (half-line) bounded by P_i and forming with S_i an angle:

$$\alpha_i \geq \max(\alpha'_i, \alpha''_i)$$

In our experiments we evaluate the cases $\alpha_i = \max(\alpha'_i, \alpha''_i)$ as well as the case where α_i is selected as the bisector of $\angle T_{i-1}P_iP_{i+1}$. The center O_i is given by:

$$\langle T_{i-1}(x) + \frac{\delta x * k_i + d_i * \delta y * v_i)}{||T_{i-1}P_i||}, T_{i-1}(y) + \frac{\delta y * k_i - d_i * \delta x * v_i)}{||T_{i-1}P_i||} \rangle$$

where $\delta x = x_i - T_{i-1}(x)$, $\delta y = y_i - T_{i-1}(y)$, $v_i = \sqrt{u_i^2 - k_i^2}$, $k = \frac{||T_{i-1}P_i|| + u_i^2 - R_i^2}{2||T_{i-1}P_i||}$, and

$$u_i = \sqrt{R_i^2 + ||T_{i-1}P_i||^2 - 2 * R_i * ||T_{i-1}P_i|| * \cos \alpha_i}.$$

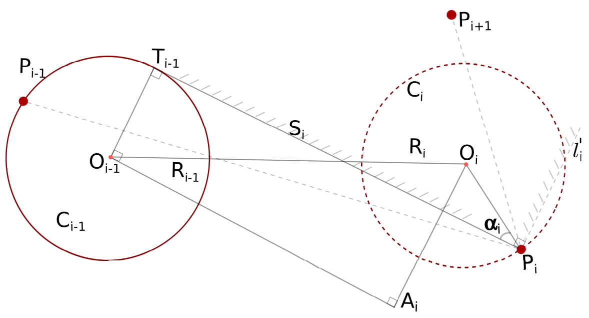


Fig. 15. Computing R_i from α_i.

Curvature optimization. When searching for a circle C_i of minimum curvature that does not intersect or contain C_{i-1} and P_{i+1}, we use the following expression for deriving R_i given some value for α_i. In Figure 15 we build a parallel through O_{i-1} to S_i and let A_i be the base of the perpendicular from O_i on it. $O_{i-1}A_i||S_i$ and $O_iA_i \perp S_i$. In the right triangle $\Delta O_iO_{i-1}A_i$:

$$(R_i + R_{i-1})^2 \leq ||O_iO_{i-1}||^2 = ||O_iA_i||^2 + ||A_iO_i||^2$$

i.e.:

$$(R_i + R_{i-1})^2 \leq (||P_iT_{i-1}|| - R_i \cos\alpha_i)^2 + (R_{i-1} + R_i \sin\alpha_i)^2$$

And the condition is:

$$R_i \leq \frac{||P_iT_{i-1}||^2}{2(R_{i-1} + ||P_iT_{i-1}|| \cos\alpha_i - R_{i-1}\sin\alpha_i)}$$

6. ADAPTATION OF COMMON TECHNIQUES

In the following subsections we specify our adaptations of several commonly used interpolation techniques.

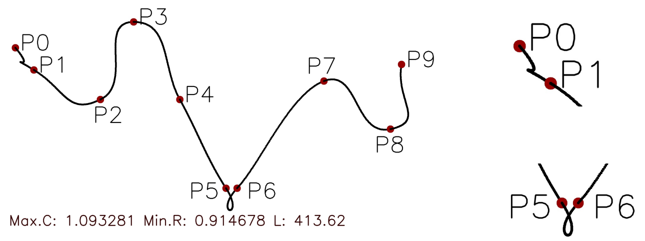


Fig. 16. Weaknesses with splines

6.1. Splines

The spline algorithm we adapt for our problem is described in [Yu et al. 2004]. The techniques is used to reconstruct the trajectory of a moving object using sparse data points(position and velocity) recorded at certain frequency. The technique is intended for representing a trajectory with high accuracy while storing relatively small amount of data points. The velocity of the moving object at given points is an input parameter. Our inputs in this report do not contain velocity and a modification was applied (tuning a constant parameter in its place). An example is shown in Figure 19. Weaknesses, in terms of what can go wrong (tight loops of high curvature that are difficult to avoid), are illustrated in Figure 16. We also specify a direction for this velocity. The direction vector for each point P_i, except for P_0 and P_{N-1}, is parallel to the direction vector $\overrightarrow{P_{i-1}P_{i+1}}$. For P_0 the direction vector is parallel to $\overrightarrow{P_0P_1}$, and for P_{N-1} the direction vector is parallel to $\overrightarrow{P_{N-2}P_{N-1}}$.

6.2. Bézier

We use cubic Bézier curves for our experiments. A cubic Bézier curve between two given end-points is a curve connecting them and designed to be smooth. Its directions in the end-points are given by two control points. For our experiments with the Bézier interpolation we define a heuristic to generate the control points for each pair of data points. For each point P_i, except P_0 and P_{N-1}, two points P_A^i and P_B^i(See Figure 17) are defined on a line parallel to $|P_{i-1}P_{i+1}|$ passing through P_i, where $||P_iP_A^i||$ and $||P_iP_B^i||$ is some fraction, f, of $||P_{i-1}P_i||$ and $||P_iP_{i+1}||$ respectively. Points P_B^i and P_A^{i+1} are then used as control points for generating Béziercurve from P_i to P_{i+1}. For P_0 we define P_B^0=P_A^1, and P_A^0 is not defined. Similarly for P_{N-1} we define P_A^{N-1} to be the same as P_B^{N-2}, and P_B^{N-1} is not defined.

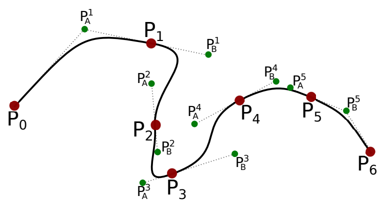


Fig. 17. An example of Cubic Bézier based interpolation.

6.3. Bi-arcs

A version of Bi-arcs are described in [Rossignac and Requicha 1987]. The Bi-arcs is a techniques to connect two points P_1 and P_2 with defined direction tangents using two arcs. The joining point P_{12} of the two arcs is on a common tangent making the curve between the points continuous. The approach described in [Rossignac and Requicha 1987] uses the ratio, ρ, between $||P_1P_{12}||$ and $||P_{12}P_2||$ as a parameter to generate a bi-arc between the points P_1 and P_2. In order to minimize the maximum curvature between the two points we want to minimize the curvature difference of the two arcs. To achieve this we generate bi-arcs using two different approaches. The first approach searches for an optimal value of ρ for which the curvature difference is the minimum possible. The second approach uses an analytical technique proposed in [Koc et al. 2000a] to achieve the minimum curvature difference between the two arcs. The second approach is significantly inexpensive(computationally) in comparison to the first approach. However, as we shall see in the next section, our experiments suggest that the two approaches yield different solutions.

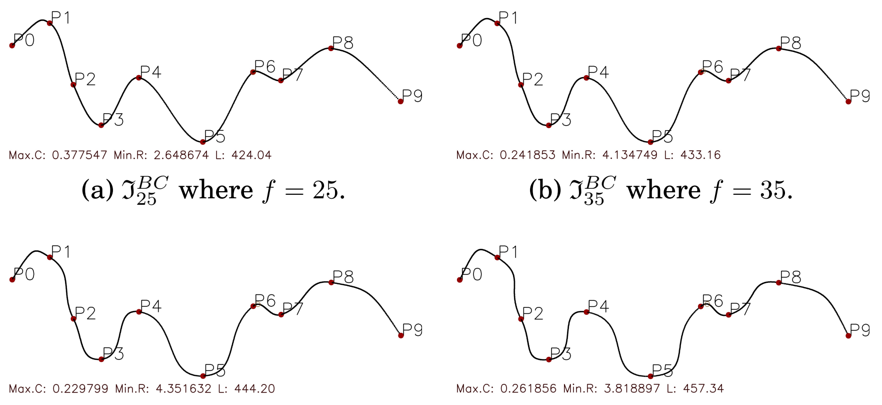
The direction vector for each point P_i, except for P_0 and P_{N-1}, is set parallel to the direction vector $\overrightarrow{P_{i-1}P_{i+1}}$. For P_0 the direction vector is parallel to $\overrightarrow{P_0P_1}$, and for P_{N-1} the direction vector is parallel to $\overrightarrow{P_{N-2}P_{N-1}}$. Using the points and their corresponding direction vectors an initial bi-arc solution is generated and then we perform a hill-climbing search for a local optima while updating the direction vectors by $\pm\Delta$ radians ($\pm\Delta$ can be reduced/halved on convergence up to a minimum value). An optima is reached when the global maximum curvature cannot be minimized. Each search iteration attempts to minimize the curvature difference between two consecutive bi-arcs segments of a bi-arc interpolation solution.

7. EXPERIMENTS

We conduct experiments to compare our solutions building on general interpolation techniques: Bézier, Splines, Bi-arcs (see Section 6) with the new specialized algorithms based on R-geodesics (Section 4). We use two types of benchmarks:

Type A) Random data points: These experiments are setup to use N random points, generated sequentially where each point has a random ordinate and follows the previous point at a positive random displacement along the abscissa axis on an unbounded Cartesian plane. P_i(y) is uniform within the range of ± 50 from P_{i-1}(y),

and P_i(x) is uniform within the range of 50 from P_{i-1}(x). For all our 1200 benchmarks of this type: $N = 10$, P_0(x)=0 and P_0(y)=50.
Type B) GPS points: GPS readings (latitude and longitude) are recorded while driving a vehicle. These recorded points are then used as input data points. The used recorded data is based on approximately 1000 GPS points.



(a) $\mathfrak{I}_{25}^{BC}$ where $f = 25$. (b) $\mathfrak{I}_{35}^{BC}$ where $f = 35$.

(c) $\mathfrak{I}_{45}^{BC}$ where $f = 45$. (c) $\mathfrak{I}_{55}^{BC}$ where $f = 55$.

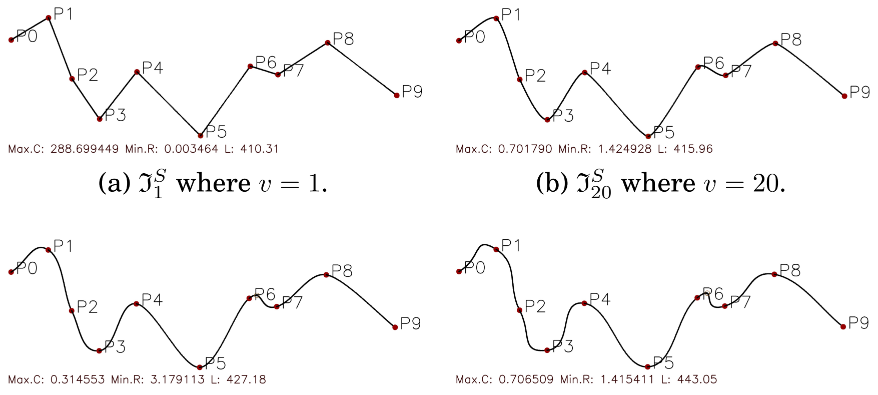
Fig. 18. Bézier curve based interpolation using different values for f.

Table I.

600 Scenarios	Maximum curvature		Curve length	
Parameter	Avg.	Relative Diff.(%)	Avg.	Relative Diff.(%)
f=15	1.79	126.59	436.10	-3.86
f=25	0.93	17.64	443.93	-2.13
f=35	0.79	0.00	453.60	0.00
f=45	0.96	21.40	465.07	2.53
f=55	2.24	184.06	478.42	5.47

Relative comparison of solutions using $\mathfrak{I}_{f}^{BC}$ with different values of f.

Bézier($\mathfrak{I}_{f}^{BC}$):. The adaptation of the Bézier interpolation algorithm, as described in Section 6.2, requires defining two control points between each pair of input data points. Previously we specified how to define the control points in terms of the parameter f. Most experiments presented in this paper employ $f = 35$, a value that we show to perform well on Type A benchmark scenarios. Table I presents the statistics of results obtained using 600 different examples of Type A benchmarks with the values of f shown in the first column. The second column is the average of the maximum curvature for all examples, and the third column shows the average relative difference in comparison to results obtained using $f = 35$.



(a) $\mathfrak{I}_1^S$ where $v = 1$. **(b) $\mathfrak{I}_{20}^S$ where $v = 20$.**

(c) $\mathfrak{I}_{40}^S$ where $v = 40$. **(d) $\mathfrak{I}_{60}^S$ where $v = 60$.**

Fig. 19. $\mathfrak{I}_v^S$ interpolation solutions using different values for v.

Table II.

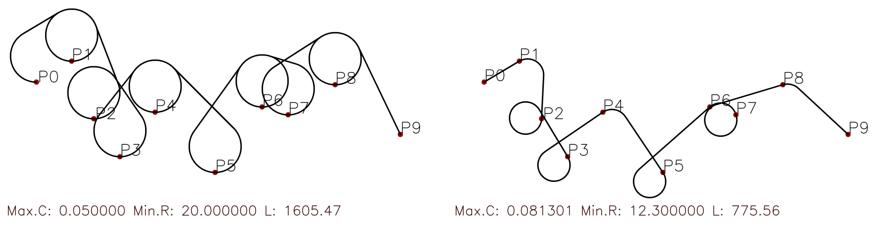
600 Scenarios	Maximum curvature		Curve length	
Parameter	Avg.	Relative Diff.(%)	Avg.	Relative Diff.(%)
v=5	12.86	650.38	428.59	-0.73
v=10	3.37	96.83	429.95	-0.42
v=15	1.71	0.00	431.76	0.00
v=20	2.16	26.21	433.93	0.50
v=25	70.96	4040.80	436.45	1.09

Relative comparison of solutions using $\mathfrak{I}_v^S$ with different values of v.

Splines($\mathfrak{I}_v^S$):. For the version of Splines algorithm in Section 6.1 we experiment with five different values for v (see Table II, Figure 19). Figure 19 shows the $\mathfrak{I}_v^S$ interpolation of an example set of data points. It can be observed that at $v = 1$, the interpolation is closer to a linear interpolation. As the value of v is increased the maximum curvature of the resulting curve may increase. This relationship is strongly dependent on the input data points. For the scenarios generated in our experiments, $\mathfrak{I}_v^S$ reaches its best performance at $v = 15$ (see Table II).

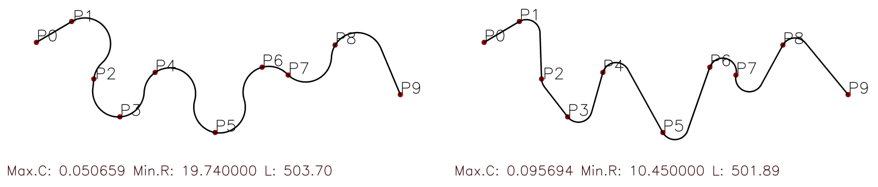
Tangent-Arc:. We evaluate three families of heuristics for the techniques based on R-geodesics (discussed in Section 5):

a) $\mathfrak{I}_a^{AT}$: Circle center O_i is placed at constant translation from point P_i and all circles have the same direction (see Figure 20(a)).
b) $\mathfrak{I}_b^{AT}$: Circle center O_i is placed on a perpendicular on the tangent from C_{i-1} to the point P_i and all circles have the same direction (see Figure 20(b)).
c) Circle center O_i is placed as described in Section 5.3. For this heuristics we evaluate four alternatives ways of choosing the parameter α_i:
   1) $\mathfrak{I}_{c1}^{AT}$: Set α_i to the low limit of its range (see Figure 21(a)).
   2) $\mathfrak{I}_{c2}^{AT}$: Set α_i to the upper limit of ts range (see Figure 21(b)).
   3) $\mathfrak{I}_{c3}^{AT}$: Set α_i to the bisector of the angle describing the space of its possible locations (see Figure 22(a)).
   4) $\mathfrak{I}_{c4}^{AT}$: Initially set α_i as per the previous case $\mathfrak{I}_{c3}^{AT}$ and then follow the gradient searching for a local optima in its neighborhood (see Figure 22(b)).


P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
Max.C: 0.050000 Min.R: 20.000000 L: 1605.47
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
Max.C: 0.081301 Min.R: 12.300000 L: 775.56

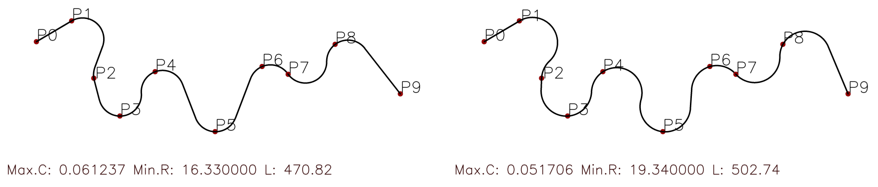
(a) $\mathfrak{I}_a^{AT}$ (b) $\mathfrak{I}_b^{AT}$

Fig. 20. Tangent-Arc based interpolation.


P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
Max.C: 0.050659 Min.R: 19.740000 L: 503.70
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
Max.C: 0.095694 Min.R: 10.450000 L: 501.89

(a) $\mathfrak{I}_{c1}^{AT}$ (b) $\mathfrak{I}_{c2}^{AT}$

Fig. 21. α_i set to: (a) lower limit, (b) upper limit


P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
Max.C: 0.061237 Min.R: 16.330000 L: 470.82
P0
P1
P2
P3
P4
P5
P6
P7
P8
P9
Max.C: 0.051706 Min.R: 19.340000 L: 502.74

(a) $\mathfrak{I}_{c3}^{AT}$ (b) $\mathfrak{I}_{c4}^{AT}$

Fig. 22. α_i set to: (a) angle bisector, (b) local optima close to bisector

Table III.

600 Scenarios	Maximum curvature		Curve length	
Algorithm	Avg.	Relative Diff.(%)	Avg.	Relative Diff.(%)
$\mathfrak{I}_a^{AT}$	0.0517	-21.14	1667.55	214.12
$\mathfrak{I}_b^{AT}$	0.0688	5.03	1118.45	110.68
$\mathfrak{I}_{c1}^{AT}$	0.0672	2.53	531.17	0.06
$\mathfrak{I}_{c2}^{AT}$	0.0949	44.89	581.44	9.53
$\mathfrak{I}_{c3}^{AT}$	0.0763	16.42	518.45	-2.34
$\mathfrak{I}_{c4}^{AT}$	0.0655	0.00	530.87	0.00

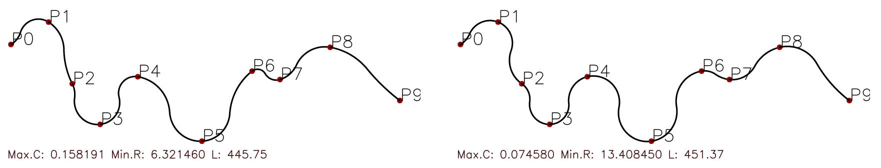
The comparison of Tangent-Arc algorithms relative to $\mathfrak{I}_{c4}^{AT}$

An interesting property of the algorithm obtained in the case $\mathfrak{I}_a^{AT}$ is that there is no lower bound on the maximum curvature of the solution. To comparatively evaluate the length it can offer for its results, in our experiments we set the upper bound on the maximum curvature to the smallest value obtained from of the other tangent-arc algorithms. This allows $\mathfrak{I}_a^{AT}$ to be the best performing technique with respect to maximum curvature. However, there is a trade off between the total length of the curve and the maximum curvature (see Table III). Its curve length is orders of magnitude worse. All the other tangent-arc algorithms perform an iterative search for the maximum curvature curve that passes through each data point. The search stops when the maximum curvature cannot be improved without ignoring one or more data points. Table III shows statistical analysis of the tangent-arc algorithms. The second column in the table shows the maximum curvature averaged over 600 test cases. The forth column shows the average curve length. The table also shows average relative difference in maximum curvature and total length when taking $\Im_{c4}^{AT}$ as reference ($\Im_{c4}^{AT}$ being observed an outstanding method). After $\Im_{a}^{AT}$, $\Im_{c4}^{AT}$ has the best performance in terms of maximum curvature, among considered candidates. However, it is 2.34% below $\Im_{c3}^{AT}$ with respect to curve length. Since we give higher priority to the maximum curvature, we conclude that $\Im_{c4}^{AT}$ is the best among the tangent-arc algorithms without self-intersecting loops. Further we compare $\Im_{c4}^{AT}$ against candidates based on general interpolation techniques. Table IV shows this comparison. The third and fifth columns show the relative performance of each algorithm as compared to $\Im_{c4}^{AT}$, using 600 different Type A test cases. Based on the maximum curvature, $\Im_{c4}^{AT}$ outperforms all the other algorithms. $\Im_{35}^{BC}$ comes in second with 968.8% relative deterioration on maximum curvature with respect to $\Im_{c4}^{AT}$.

Table IV.

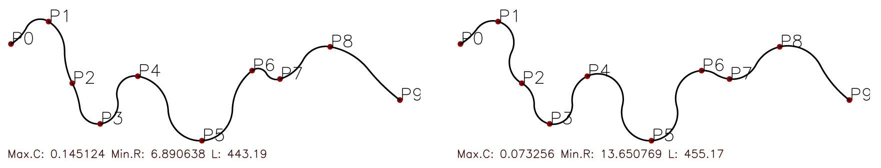
600 Scenarios	Maximum curvature		Curve length	
Algorithm	Avg.	Avg. Relative Diff.(%)	Avg.	Avg. Relative Diff.(%)
$\Im_{15}^{S}$	1.7192	2524.21	447.89	-15.63
$\Im_{35}^{BC}$	0.7002	968.80	469.22	-11.61
$\Im_{c4}^{AT}$	0.0655	0.00	530.87	0.00

Comparison of general interpolation algorithms relative to$\Im_{c4}^{AT}$



(a) Initial interpolation. (b) Local optima after 6 iterations.

Fig. 23. $\Im_{\rho}^{BA}$ interpolation result



(a) Initial interpolation. (b) Local optima after 9 iterations.

Fig. 24. $\Im_{a}^{BA}$ interpolation result

Table V.

	Maximum curvature	Curve length
Algorithm	Avg	Avg
$\Im_a^{BA}$	0.08	458.5
$\Im_\rho^{BA}$	0.10	458.8

Comparison of $\Im_a^{BA}$ and $\Im_\rho^{BA}$ based on maximum curvature and total length using 20 different scenarios.

Table VI.

Comparison based on computation time(seconds).	
Algorithms	Avg.
$\Im_{c4}^{AT}$[TT]	0.00121441
$\Im_a^{BA}$[TT]	0.00135743
$\Im_\rho^{BA}$[TT]	69.75591109
$\Im_\rho^{BA}$[FRT]	69.75522075

Comparison of $\Im_\rho^{BA}$ and $\Im_a^{BA}$ based on total computation time using 20 different scenarios. The comparison is with respect to $\Im_{c1}^{AT}$. TT: total time to find solution, FRT: find ρ ratio time, total time spent in finding value for ρ

Bi-arcs($\Im^{BA}$):. As described in Section 6.3 we experiment with two different approaches to create a bi-arc curve between two given points. The first approach, $\Im_\rho^{BA}$, requires finding a value for parameter ρ which minimizes the curvature difference between the two arcs. This is achieved by performing an iterative search for a ratio between 0 and 1000 while increasing the allowed curvature difference. The algorithm starts with the curvature difference of 1 and linearly increases the difference while searching a value for ρ until a bi-arc is found. Figure 23 is an example of interpolation using this approach. The second approach, $\Im_a^{BA}$, uses the analytical technique described in [Koc et al. 2000a] (see Figure 24). We perform Hill-climbing on solutions from both approaches in search for a local optima. As specified in Section 6.3, for each iteration the direction vectors are updated by Hill-climbing until convergence with a resolution of $\Delta = 0.1$ radians.

Figures 23(a) and 24(a) show the results from both approaches for a given set of data points, after the first step of Hill-climbing. Figures 23(b) and 24(b) show results after convergence. For the given example, the two approaches yield close results with respect to maximum curvature. However, in Table V we present statistics that show $\Im_a^{BA}$ performs better in comparison to $\Im_\rho^{BA}$. While the two techniques performed similar on the total length, the maximum curvature result for $\Im_a^{BA}$ was 20% better compared to $\Im_\rho^{BA}$. This table was built with only 20 test cases of Type A benchmarks, since algorithm $\Im_\rho^{BA}$ is particularly slow (see Table VI) and the presented results to not warrant further investigations for it.

Combined Tangent-Arc and Bi-arcs ($\Im_a^{BA}$+$\Im_{c4}^{AT}$):. This algorithm is similar to the $\Im_a^{BA}$ technique described in Section 6.3 (see Figure 26), except that we initialize the direction vectors for the data points with their value as generated using $\Im_{c4}^{AT}$. Note that, due to their limitation to using only arc segments, the Bi-arcs are not powerful enough to model the result of $\Im_{c4}^{AT}$ as it is illustrated in Figure 25, getting poorer results. Table VIII presents statistical data comparing the results of $\Im_{c4}^{AT}$ and the combined $\Im_a^{BA}$+$\Im_{c4}^{AT}$. Based on the 1200 test cases, $\Im_{c4}^{AT}$ is 8.22% better than $\Im_a^{BA}$+$\Im_{c4}^{AT}$ with respect to curvature. However, an interesting observation here is that when initialized with $\Im_{c4}^{AT}$, the resulting maximum curvature of $\Im_a^{BA}$ is improved by 65%.

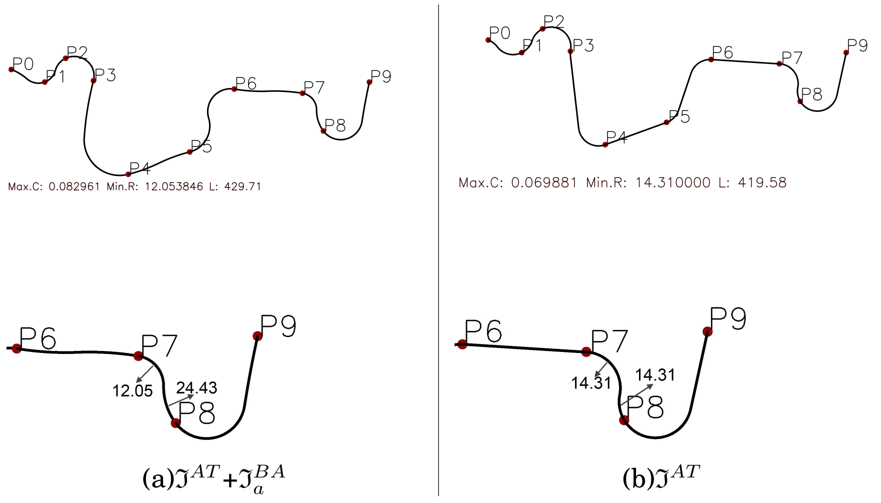


Fig. 25. The failure of $\mathfrak{I}_a^{BA}$ to capture the full quality of the output of $\mathfrak{I}^{AT}$

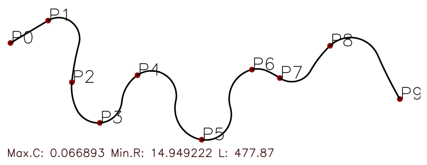


Fig. 26. Combined $\mathfrak{I}_{c1}^{AT}$ and $\mathfrak{I}_a^{BA}$ technique. The solution obtained in this case is already a local optima

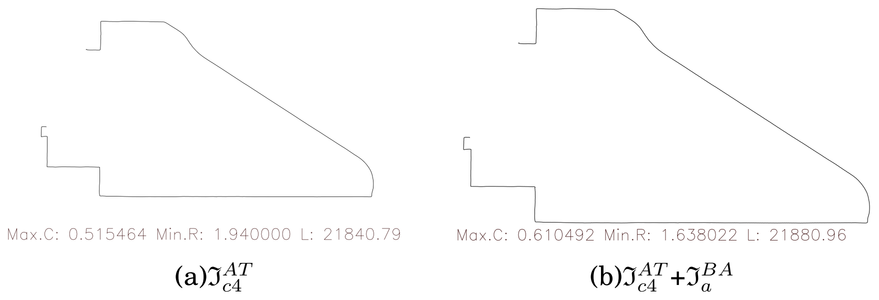


Fig. 27. Solution obtained with $\mathfrak{I}_{c4}^{AT}$ for a set of approximately 1000 GPS points recorded while driving.

Finally we compare the best performing techniques on Type B data. We consider an example of GPS points recorded while driving. We use $\mathfrak{I}_{c4}^{AT}$, $\mathfrak{I}_a^{BA}$, and the combined $\mathfrak{I}_{c4}^{AT}+\mathfrak{I}_a^{BA}$ algorithm to compare the results. The interpolation result for the best performing algorithm is shown in Figure 27. The computation corresponds to the assumption that the recording vehicle is moving at a constant speed (the bound on maximum curvature is constant along the trajectory). $\mathfrak{I}_{c4}^{AT}$ gives the best result of $\frac{1}{1.94\ meters}$ as the maximum curvature, and the combined $\mathfrak{I}_{c4}^{AT}+\mathfrak{I}_a^{BA}$ produce the second best result

Table VII.

600 Scenarios	Maximum curvature		Curve length	
Algorithm	Avg.	Relative Diff.(%)	Avg.	Relative Diff.(%)
$\mathfrak{I}_{15}^{S}$	1.7192	2524.21	447.89	-15.63
$\mathfrak{I}_{35}^{BC}$	0.7002	968.80	469.22	-11.61
$\mathfrak{I}_{a}^{BA}$	0.1064	62.38	499.33	-5.94
$\mathfrak{I}_{c1}^{AT}$	0.0672	2.53	531.17	0.06
$\mathfrak{I}_{c4}^{AT}$	0.0655	0.00	530.87	0.00

The comparison is with respect to $\mathfrak{I}_{c4}^{AT}$

Table VIII.

1200 Scenarios	Maximum curvature		Curve length	
Algorithm	Avg.	Relative Diff.(%)	Avg.	Relative Diff.(%)
$\mathfrak{I}_{a}^{BA}$	0.114	73.005	488.983	-6.231
$\mathfrak{I}_{c4}^{AT}$+$\mathfrak{I}_{a}^{BA}$	0.071	8.226	522.110	0.120
$\mathfrak{I}_{c1}^{AT}$	0.067	2.668	524.449	0.569
$\mathfrak{I}_{c4}^{AT}$	0.066	0.000	521.481	0.000

The comparison is with respect to $\mathfrak{I}_{c4}^{AT}$

with maximum curvature of about $\frac{1}{1.63\ meters}$. The $\mathfrak{I}_{a}^{BA}$ technique produces the curve of maximum curvature $\frac{1}{0.99\ meters}$.

8. CONCLUSIONS

We address the problem of interpolating a set on N data points with C^1 class curves, while minimizing the maximum curvature in all points where it is defined, and minimizing curve length. We extend a family of techniques constructing an interpolation by concatenating arcs and lines. We also construct several solutions based on general interpolation algorithms. A set of reported experiments single out one of the proposed methods using R-geodesics to have a promising trade-off: significantly better curvature for slightly longer path.

For the evaluation of the aforementioned interpolation techniques we use two measurements - global maximum curvature, and total interpolation length. Given two interpolation solutions for a set of data points we prefer the curve with lower maximum curvature. In case of curves with equal maximum curvatures we prefer the one with shorter interpolation length. In the Experiments Section we test and compare the eleven studied algorithms using over 1000 examples of randomly generated scenarios, as well as recorded benchmark data with approximately 1000 GPS points. In this paper we evaluate: 6 variants of new R-geodesics based techniques ($\mathfrak{I}^{AT}$), 2 variants of Bi-Arc interpolation with hill-climbing ($\mathfrak{I}^{BA}$), piece-wise Bézier with hypothesized control points ($\mathfrak{I}_{f}^{BC}$), piece-wise constructions with a version of Splines ($\mathfrak{I}_{v}^{S}$), and Bi-Arc initialized with R-geodesics ($\mathfrak{I}^{AT}$+$\mathfrak{I}^{BA}$).

From the six studied versions of $\mathfrak{I}^{AT}$, two stand out for special trade-offs: a version that allows for loops on intermediary segments ($\mathfrak{I}_{a}^{AT}$) and a version that searches for the locally best parameters of the next R-geodesic ($\mathfrak{I}_{c4}^{AT}$). From described results it is observed that $\mathfrak{I}_{a}^{AT}$ has the best performance on maximum curvature. However, the technique allows self-intersecting loops that make the solution less desirable due to significantly increased total length (214% longer than $\mathfrak{I}_{c4}^{AT}$).

The best technique based on general interpolation $\mathfrak{I}_{a}^{BA}$ is 73% worse than $\mathfrak{I}_{c4}^{AT}$ in curvature with a 6% improvement in curve length. Using $\mathfrak{I}_{c4}^{AT}$ to initialize $\mathfrak{I}_{a}^{BA}$ leads to an improvement of 65% in curvature with the trade-off of 6% on curve length.

We compare $\mathfrak{I}^{AT}$ with methods based on general interpolation approaches such as $\mathfrak{I}_f^{BC}$ and $\mathfrak{I}_v^S$. The version of $\mathfrak{I}_f^{BC}$ with the parameter f defining the closeness of the control points to end-points equal to 35% ($\mathfrak{I}_{35}^{BC}$) is the best performing method among the candidates based on general interpolation techniques (other than $\mathfrak{I}^{BA}$). However, in comparison to $\mathfrak{I}_{c4}^{AT}$, the result on maximum curvature is worse by approximately 968%. The result on total length is nevertheless approximately 11% better compared to $\mathfrak{I}_{c4}^{AT}$.

A remaining open question is whether a global optima can be achieved and proven.

REFERENCES

J. Barraquand and J.C. Latombe. 1989. On nonholonomic mobile robots and optimal maneuvering. In *Intelligent Control, 1989. Proceedings., IEEE International Symposium on*. IEEE, 340–347.

Richard H. Bartels, John C. Beatty, and Brian A. Barsky. 1987. *An introduction to splines for use in computer graphics and geometric modeling*. M. Kaufmann Publishers.

Guarino Lo Bianco and Piazzi. 2001. Optimal Trajectory Planning With Quantic G2-Splines. Proc. of IEEE Inteligent Vehicles Symposium.. (2001).

T. Blu, P. Thévenaz, and M. Unser. 2004. Linear interpolation revitalized. *Image Processing, IEEE Transactions on* 13, 5 (2004), 710–719.

Boissonnat, Cerezo, and Leblond. 1992. Shortest Path of Bounded Curvature in the Plane. Proc. of the IEEE International Conference on Robotics and Automation, pp.:2315-2320. (1992).

L. Dubins. 1957. On curves of minimal length with a constraint on average curvature and with prescribed intital and terminal position and tangents. American journal of mathematics, Vol.79(3):497-517. (1957).

O. D. Evans and Y. Kim. 1998. Efficient implementation of image wraping on a multimedia processor. Real-Time Image., Vol.4:417-428. (1998).

A.R. Forrest. 1972. Interactive interpolation and approximation by Bézier polynomials. *Comput. J.* 15, 1 (1972), 71–79.

Bahattin Koc, Yuan-Shin Lee, and Yawei Ma. 2000a. Max-Fit Biarc Fitting to STL Models for Rapid Prototyping Processes. Proceedings of the sixth ACM symposium on Solid modeling and applications. (2000).

B. Koc, Y. Ma, and Y.S. Lee. 2000b. Smoothing STL files by Max-Fit biarc curves for rapid prototyping. *Rapid Prototyping Journal* 6, 3 (2000), 186–205.

Ivan Krasilnikov. 2010. ACM ICPC training and related sites. http://github.com/infnty/acm/raw/master/lib/geometry/CircleTangentsViz.jar. (2010).

Larson, Hostetler, and Edwards. 2012. *Calculus* (seventh ed.). Cengage Learning.

J.P. Laumond, P.E. Jacobs, M. Taix, and R.M. Murray. 1994. A motion planner for nonholonomic mobile robots. *Robotics and Automation, IEEE Transactions on* 10, 5 (1994), 577–593.

Steven M. LaValle. 2006. *PlanningAlgorithms*. Cambridge University Pres.

T.M. Lee, E. Lee, and M. Yang. 2007. Precise bi-arc curve fitting algorithm for machining an aspheric surface. *The International Journal of Advanced Manufacturing Technology* 31, 11 (2007), 1191–1197.

H. Makino. 1988. Clothoidal InterpolationA New Tool for High-Speed Continuous Path Control. *CIRP Annals-Manufacturing Technology* 37, 1 (1988), 25–28.

J. McCrae and K. Singh. 2009. Sketching piecewise clothoid curves. *Computers & Graphics* 33, 4 (2009), 452–461.

DS Meek and DJ Walton. 1992. Clothoid spline transition spirals. *Math. Comp.* 59, 199 (1992), 117–133.

B. Mirtich and J. Canny. 1992. Using skeletons for nonholonomic path planning among obstacles. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 2533–2540.

L. Piegl. 1987. Interactive data interpolation by rational Bezier curves. *IEEE COMP. GRAPHICS APPLIC.* 7, 4 (1987), 45–58.

L.A. Piegl and W. Tiller. 2002. Data approximation using biarcs. *Engineering with computers* 18, 1 (2002), 59–65.

Jaroslaw R Rossignac and Aristides A. G. Requicha. 1987. Piecewise-Circular Curves for Geometric Modeling. IBM Journal of Research Development, Vol.31(3). (1987).

J. Schönherr. 1993. Smooth biarc curves. *Computer-Aided Design* 25, 6 (1993), 365–370.

L.I. Schumaker. 1983. On shape preserving quadratic spline interpolation. *SIAM J. Numer. Anal.* 20, 4 (1983), 854–864.

Carlo H Sequin, Kiha Lee, and Jane Yen. 2005. Fair,G2- and C2-Continuous Circles Splines for interpolation of Sparse Data Points. Computer Aided Design Vol.37(2):201-11. (2005).

Lejun Shao and Hao Zhou. 1996. Curve Fitting With Bezier Cubics. Graphical Models and Image Processing Vol.58(3):223-32. (1996).

D.H. Shin and S. Singh. 1990. *Path generation for robot vehicles using composite clothoid segments*. Technical Report. DTIC Document.

P. Svestka and M.H. Overmars. 1995. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, Vol. 2. IEEE, 1631–1636.

M. Unser, A. Aldroubi, and M. Eden. 1991. Fast B-plines transforms for continuous image representation and interpolation. IEEE Trans. Pattern Anal. Machine Intell., Vol.13:277-285. (1991).

R. Vishen and M. Silaghi. 2013. Interpolation Minimizing Maximum Curvature. http://cs.fit.edu/proj/interpolation. (2013).

Martin Voshell. 2004. High Acceleration and the Human Body. http://csel.eng.ohio-state.edu/voshell/gforce.pdf. (2004).

Z. Yao and A. Joneja. 2007. Path generation for high speed machining using spiral curves. *Computer-Aided Design & Applications* 4 (2007), 191–198.

Byunggu Yu, Seon Ho Kim, Thomas Bailey, and Ruben Gamboa. 2004. Curve-based Representation of Moving Object Trajectory. Proceedings of the International Database Engineering and Application Symposium. (2004).

Received January 2013; revised xxxx; accepted yyyy