J Intell Robot Syst (2015) 80 (Suppl 1):S23–S56
DOI 10.1007/s10846-014-0172-0

Continuous Path Smoothing for Car-Like Robots Using B-Spline Curves

Mohamed Elbanhawi · Milan Simic · Reza N. Jazar

Received: 14 April 2014 / Accepted: 16 December 2014 / Published online: 8 January 2015

Abstract A practical approach for generating motion paths with continuous steering for car-like mobile robots is presented here. This paper addresses two key issues in robot motion planning; path continuity and maximum curvature constraint for nonholonomic robots. The advantage of this new method is that it allows robots to account for their constraints in an efficient manner that facilitates real-time planning. B-spline curves are leveraged for their robustness and practical synthesis to model the vehicle's path. Comparative navigational-based analyses are presented to selected appropriate curve and nominate its parameters. Path continuity is achieved by utilizing a single path, to represent the trajectory, with no limitations on path, or orientation. The path parameters are formulated with respect to the robot's constraints. Maximum curvature is satisfied locally, in every segment using a smoothing algorithm, if needed. It is demonstrated that any local modifications of single sections have minimal effect on the entire path. Rigorous simulations are presented, to highlight the benefits of the proposed method, in comparison to existing approaches with regards to continuity, curvature control, path length and resulting acceleration. Experimental results validate that our approach mimics human steering with high accuracy. Accordingly, efficiently formulated continuous paths ultimately contribute towards passenger comfort improvement. Using presented approach, autonomous vehicles generate and follow paths that humans are accustomed to, with minimum disturbances.

Keywords Path planning · Path smoothing · Nonholonomic robots · C^2 continuity · Maximum curvature · Real time planning

M. Elbanhawi (✉) · M. Simic · R. N. Jazar
School of Aerospace, Mechanical, and Manufacturing Engineering (SAMME), RMIT University.Bundoora East Campus, Corner of Plenty Road, McKimmies Road, Bundoora VIC 3083, Melbourne, Australia
e-mail: mohamed.elbenhawi@rmit.edu.au

M. Simic
e-mail: milan.simic@rmit.edu.au

R. N. Jazar
e-mail: reza.jazar@rmit.edu.au

1 Introduction

Path planning is an essential task in robotics. It is development of a collision-free, continuous sequence of feasible actions, or path segments, from the start to the goal position. An autonomous robot must be capable of sensing its surroundings to achieve obstacle detection. It must perform localization in order to be aware of its current position, and goal position within a global map, as well as, sense obstacles on

the projected trajectory. Adequate sensors, localization, actuation, and environment knowledge is often assumed in planning. However, robotic research is also conducted in dynamic and stochastic environments. For planning purposes, the environment is represented as a configuration space (c-space) or as a discrete set of neighboring cells. Path planning algorithms attempt to systematically generate an optimal path with respect to a predefined metric and avoid occupied cells, or occupied c-space regions. Deterministic planning methods are computationally exhaustive, even for simplified planning problems [1]. Planning is followed by execution of the desired actions. This process is generally achieved using low level controls.

Planning algorithms have been actively researched for applications in robotics, animation and computational biology [2]. They are often categorized into roadmaps, cell-decompositions, potential fields, and sampling-based planners [3]. Roadmap algorithms and cell decompositions attempt to capture the connectivity of the search space and employ a graph-search algorithm [4, 5].

Consequently, heuristic search algorithms were proposed to accelerate planning process, although they may also have some weaknesses. Potential field methods, in some situations, generate oscillating suboptimal paths [6–8]. Sampling-based planners rely on stochastic sampling to effectively search c-space [9]. However, random sampling is sacrificing the optimality of the path. The attained paths are suboptimal and contained redundant movements that require further processing as proposed in [10].

Underactuated and nonholonomic motion planning is commonly achieved by firstly ignoring the constraints, to generate a trajectory, and then modifying the trajectory [11]. Simplified solutions generate piecewise linear paths that ignore robots constraints. Cheng [12] highlighted side effects caused by ignoring the robots differential constraints in planning and generating linear paths.

Trajectory tracking algorithms are needed to execute the planned paths. Consequently, a mathematical formulation of the path is required. In the simplest forms, tracking is executed using separate PID control loops, for heading and velocity [13]. Fuzzy logic controllers were proposed for both car-like and differential drive robots [14, 15]. Algorithms, proposed in [16, 17], require feasible and continuous paths, to minimize high-speed dynamic tracking error, by defining the steady state steering commands. Cheein, Scaglia [18] proposed a trajectory tracking algorithm which outputs velocity and steering commands, for car-like robots, that minimize tracking error and controller effort. Additionally, their proposed tracking algorithm required the path to be kinematically feasible.

Curvature discontinuities lead to overshooting, as shown in [19, 20], control system instability [21], and passenger discomfort [22]. In applications that involve heavy machinery, curvature discontinuity has been related to mechanical wear and failure [23, 24]. In agriculture, robots are required to closely follow paths and prevent crop damage. Clothoids were proposed to generate continuous maneuvers for a farming vehicle [25]. Continuous curvature paths application improve stability and control of industrial vehicles [26].

Combined circular arc segments of minimum turning radius and straight lines were proposed to generate feasible paths for car-like robots. Dubin's paths were commonly used for mobile robots with minimum turning radius path smoothing [27]. Similar approaches have been proposed for Unmanned Aerial Vehicles (UAV) in planar [28] and 3D scenarios [29]. Computing the appropriate Dubin's set for two configurations was shown to be intractable and challenging in real-time scenarios .[30]. Approximate models, and coarse discretization of the control space, were employed to simplify the planning problem and perform it in a timely manner [31, 32]. The discretization led to path discontinuities and poor quality paths with respect to the planning metric.

Fraichard, Scheuer [33] highlighted the shortcomings of arcs and straight-lines use, resulting in discontinuous curvatures. Straight-lines have no curvature and circular arcs have a constant curvature value, which results in discontinuities when combining them. Following that, Clothoids were proposed for smoothing. They are characterized with curvature that increases with arc-length. Subsequently, they are often employed in civil engineering applications, particularly for road design, as continuous curvature is suitable for human steering and passenger comfort [34].

Clothoids have no closed form and have to be approximated by high-order polynomials and splines, which limits their use for real-time robotic applications [35–37]. Recent advances allow Clothoid synthesis with limited length and orientation [38].

Bézier curves, B-splines and NURBS (Non-Uniform Rational B-Splines) are curves commonly used in Computer Aided Design (CAD) applications. Their use in CAD applications has been surveyed in depth [39–41]. Bézier curves were developed for car design applications. Those curves with different number of control points were studied and used for path planning [42]. A similar approach was used for path planning and obstacle avoidance for multiple robot agents [43]. Bézier curves have properties that hinder their use in path planning. For example, their order depends on the number of control points. These limitations will be addressed later when they are assessed for navigational purposes.

B-Splines were originally created by Schoenberg [44]. They have more properties that are particularly desirable for navigation, which will be discussed later. Their use in robotics, specifically for wheeled robots, is still limited. Applications for industrial manipulators have been studied for trajectory generation and modification [45, 46].

NURBS have wide range of applications in various fields, but a limited use in robotics. They are essentially a weighed extension of Non-Uniform B-splines. They offer a high level of flexibility and produce natural smooth curves. NURBS are considered as a standard in several CAD applications. Natural trajectories generated by humans, as they are moving, were modeled [47] and generated using NURBS [48]. As a result of the highly desirable features of NURBS, they have been used in applications were accuracy, suppleness and computational efficiency are needed, such as generating paths for tools, [49, 50] blood vessel modeling [51], reverse engineering, [52, 53] and finite element analysis (FEM) [54].

1.1 Related Work

Characteristics of parametric curves have motivated their use in mobile robot applications. Related work can be divided into smoothing, continuous smoothing and maximum curvature smoothing.

Smoothing applications use parametric splines to generate smooth paths and often attempt to minimize curvature, but they do not address continuity, or maximum curvature. B-splines are used for potential field planning smoothing [19] and for assistive mobile robots [22]. In [55] B-spline smoothing is combined with a sampling-based planning algorithm for UAVs. Bézier curves have been implemented for smoothing applications in [21, 43, 56].

Some parametric spline algorithms consider maximum curvature. B-spline smoothing algorithms, for offline optimization, with conditions for maximum curvature, are presented in [23, 24]. Bézier smoothing algorithms, in [57, 58], satisfy maximum curvature. However, these algorithms suffer from path discontinuity.

Continuous smoothing applications are generally limited to geometric continuity. They do not guarantee velocity or acceleration continuity, which are more relevant for robotic applications. The work by Kwangjin, Sukkarieh [57] relies on the Bézier curve planar for the second-order geometric continuity condition [59]. Similarly, the algorithm presented in [60] guarantees geometric continuity using polynomial interpolation. A path primitive was introduced to generate third-order geometrically continuous paths [61]. Recently, a second-order, C^2, continuous shortcutting algorithm was proposed, but it could not ensure the condition satisfaction along the entire path and did not satisfy maximum curvature constraint [62]. B-spline continuous paths were generated using a genetic algorithm, however, limitations were imposed on the number of control points [63].

Several solutions were presented for bounding velocity, acceleration and jerk, for robots navigating predefined geometric paths. The work explained here is a preceding step to those algorithms development. It is needed to generate feasible paths, which could then be combined with bounding algorithms prior to tracking. Algorithms for longitudinal jerk minimization [64], acceleration bounding [65] and time optimization with acceleration bounds are given in [66, 67]. Our earlier work dealt with bounding acceleration and velocity for a given continuous path [68]. It did not consider smoothing, or curvature bounding process.

In this paper we present a smoothing algorithm that generates continuous paths with maximum curvature constraints. No control point, length or orientation limitations are imposed on the path. Smoothing is proposed using a single B-spline curve.

1.2 Contribution

The novelty in the presented approach is in the combination of the following:

1) Guaranteed C^2 continuity, for a cubic curve, through the entire path.
2) Paths generated for nonholonomic car-like robots are with minimum turning radius.
3) Smoothing process is implemented in an efficient real-time manner, with a response time in the order of *milliseconds*. Human drivers, like in the presented approach, instinctively follow continuous paths [34], which improves passenger comfort. Drivers react in real-time whilst considering the vehicles constraints.
4) We have experimentally shown that our approach mimics human steering.

Since Clothoids still lack an efficient approximation method, we proposed an approach in here that relies on parametric curves. Different parametric splines were analyzed in order to select a curve with appropriate parameters for real-time path synthesis. Discontinuities arise from joining different path segments. An algorithm is proposed to allow robust and accurate path representation, using a single path segment. This guarantees continuity throughout the entire path. We define a path segment and formulate the curvature with respect to the segment's parameters. An algorithm is presented to satisfy maximum curvature at every segment.

The majority of the related works do not present discussions on the time performances of their algorithms, despite its significance for robotic applications. It is shown here that, our algorithm is capable of running, and allowing a response, in the time frame of milliseconds, using current commercially available hardware (defined in experimental section). In cases where online replanning would be needed, it is essential to employ an efficient smoothing process and maintain the performances and safety of the robot [69]. Finally our method, unlike all discussed approaches, is validated experimentally, on a robotic platform, to show its close resemblance to human steering.

2 Preliminaries

Parametric curves, investigated for path synthesis, are presented in this section. Unique characteristics, advantageous in navigation are highlighted.

2.1 Bézier Curves

An n-th degree Bézier curve, $c(u)$, is defined by Eq. 1 where $u$is the normalized curve parameter and $B_{n,i}(u)$ is the Bézier blending function for the i-th control point P_i. Bézier blending functions are defined as shown in Eq. 2.

$$c(u) = \sum_{i=0}^{n} B_{n,i}(u)\, P_i \tag{1}$$

$$B_{n,i}(u) = \frac{n!}{i!\,(n-i)!} u^i (1-u)^{n-i} \tag{2}$$

Blending functions do not exert local control on the generated curve. This can be limiting in situations where path modification is required, for example, when an obstacle is detected. The number of points, in the control polyline, defines the degree of a Bézier curve. As a result, the predefinition of the number of control points prior to planning is often enforced [42, 43, 70]. Otherwise, high-order curves will be needed to generate paths, which are computationally inefficient, for long paths. Multiple Bézier curves, with limited number of control points, must be connected for smoothing. Unfortunately, this is resulted in discontinuities.

2.2 B-Spline Curves

A p-th degree B-spline curve, $c(u)$, is defined by n control points and a knot vector $\hat{u}$.The number of knots, m, is *equal to* $n + p + 1$. The knot vector, $\hat{u}$, consists of m non-decreasing real numbers, and u is a normalized curve length parameter, given as follows.

$$c(u) = \sum_{i=0}^{n} N_{i,p}(u)\, P_i \tag{3}$$

P_i refers to i-th control point.$N_{i,p}(u)$ is the *i-th* B-spline basis function of a $p-$ degree curve, which is defined using the Cox-de Boor algorithm [71]. For $n$number of control points there are $n$basis functions, each of which exerts local control at that specific point. Eqs. 4 and 5 define recursive algorithm, proposed by de Boor, for calculating basis functions. First order basis functions are evaluated based on their corresponding knot vectors, $\hat{u_i}, \widehat{u_{i+1}}$, as shown in Eq. 4. By recursive substitution in Eq. 5, basis functions of the higher orders, from 2 to p, are calculated. This recursive approach can be represented as a triangle structure of basis functions where the base is first order till the *p-th* degree is reached.

$$N_{i,0} = \begin{cases} 1 u \epsilon \left[\hat{u_i}, \widehat{u_{i+1}}\right) \\ 0 \qquad else \end{cases} \tag{4}$$

$$N_{i,p}(u) = \frac{u - \hat{u_i}}{\widehat{u_{i+p}} - \hat{u_i}} N_{i,p-1}(u) + \frac{\widehat{u_{i+p+1}} - u}{\widehat{u_{i+p+1}} - \widehat{u_{i+1}}} N_{i+1,p-1}(u) \tag{5}$$

Unlike Bézier curves, the order of a B-spline curve is independent from the number of control points, n. Basis functions have local control of the curve, which allows modifications of any path segment, without affecting the neighboring segments, or changing the shape of the entire curve.

2.3 NURBS

NURBS are a form of B-splines that have weights assigned to their control points. This allows the curve to be shifted towards one, or more points, as desired. They maintain the degree independence and local modification properties that characterize B-splines. Incorporation of control point weights improves flexibility and enables NURBS to synthesize free form and analytical curves, by changing control points, knots and weights.

A p-th degree B-spline curve, $c(u)$, defined by n control points and m knots, is given by Eq. 6, where, $N_{n,i}(u)$ is the B-spline basis function and w_i is the weight of a i-th control point P_i.

$$c(u) = \frac{\sum_{i=0}^{n} w_i N_{n,i}(u) P_i}{\sum_{i=0}^{n} w_i P_i} \tag{6}$$

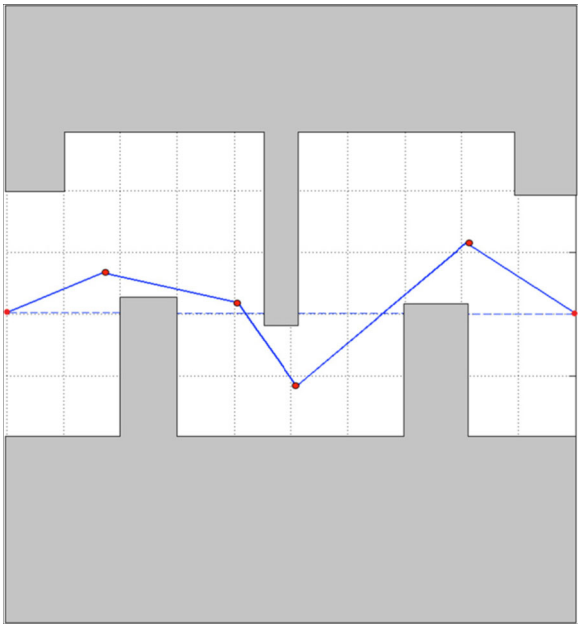

Fig. 1 Piecewise linear path connecting six waypoints

2.4 Problem Description

A piecewise linear path, see Fig. 1 as an example, is generated using a simple path planning algorithm. The linear path consists of a set of consecutive waypoints, $P_i = [P_x, P_y]^T$ shown as red circles and defined in Eqs. 1, 3 and 6, connected by straight lines. It is required to provide an algorithm that modifies set of waypoints to generate a feasible path for the robot. Generated smooth path must obey constraints given by Eqs. 7–9.

A car-like vehicle, as shown in Fig. 2, is referred to as a nonholonomic robot as it has a non-integrable

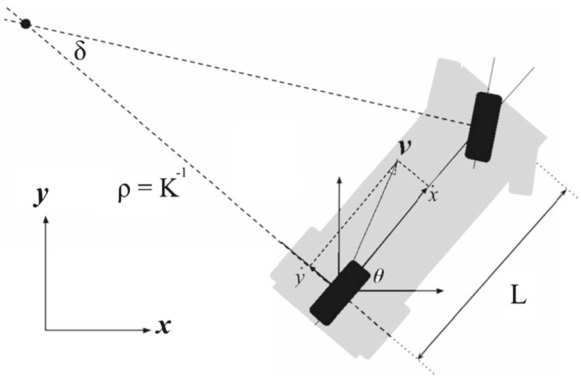

Fig. 2 Kinematic robot model

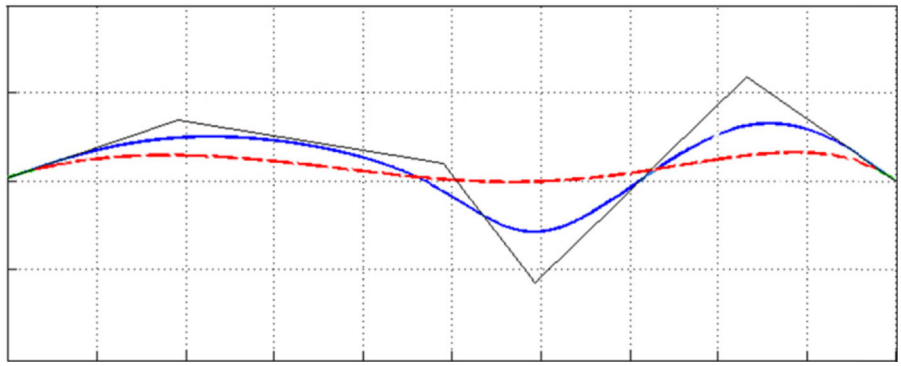

Fig. 3 A clamped B-spline curve (blue) and fifth order Bézier (dashed red) used for path smoothing

constraint expressed, where the velocity components are constrained as given in Eq. 7. This simplified kinematic model is widely accepted and known as *bicycle model*. In Fig. 2 x and y are representing axes in a global Cartesian system, while $\dot{x}$ and $\dot{y}$ are vehicle's velocities with respect to its local coordinate system axis. Path curvature is K, turning radius is ρ, while steering angle is labeled as δ. Angle θ is heading direction relative to global x-axis. The resulting vehicle velocity vector is $\boldsymbol{v}$, while L is the wheelbase. Two-wheel and four-wheel steering models have been extensively studied, as well [3, 16].

Constraints are imposed on the turning maneuver of the vehicle. The steering angle is limited to δ_{max}, which, in turn, limits the turning radius and path curvature to ρ_{min} and K_{max}, given by Eq. 8. The path of the vehicle is constrained by the limited steering angle however it is common to describe the path constraint in terms of curvature. C^{i-1} path continuity for i^{th}-degree neighboring curve segments, $S(u)$ and $R(u)$, is defined in Eq. 9, given that the start of segment $S(u)$ is connected to the end of segment $R(u)$. As known, G^{i-1} continuity [72] requires only the parameterized values of the derivatives to be continuous.

$$\dot{x}sin\theta - \dot{y}cos\theta = 0 \tag{7}$$

$$K_{max} = \frac{\tan\delta_{max}}{L} = \frac{1}{\rho_{min}} \tag{8}$$

$$\frac{d^k S(0)}{du^k} = \frac{d^k R(1)}{du^k}, \forall k = 1, 2 \ldots (i-1) \tag{9}$$

3 Parametric Curve Evaluation

In this section a comparative analysis was conducted between various parametric curves. This investigation justified the selection of B-spline curves in path planning. The piecewise linear path in Fig. 1 was considered for the analysis. We mainly focused on the accuracy of B-spline in following the original path and robustly maintaining the path topology. The ability to quickly generate reliable feasible plans is

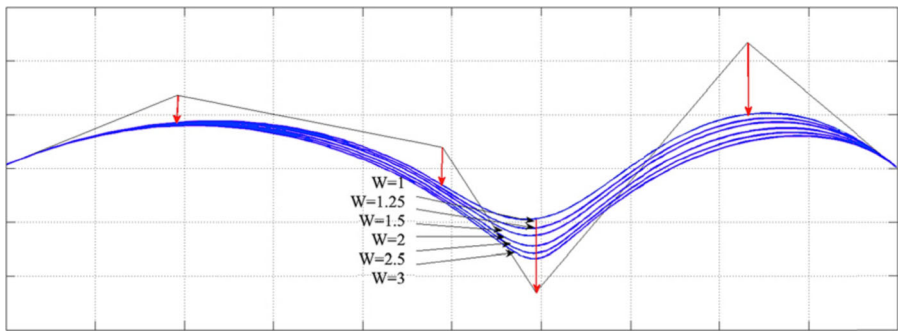

Fig. 4 NURBS curves of different weights are used for path smoothing

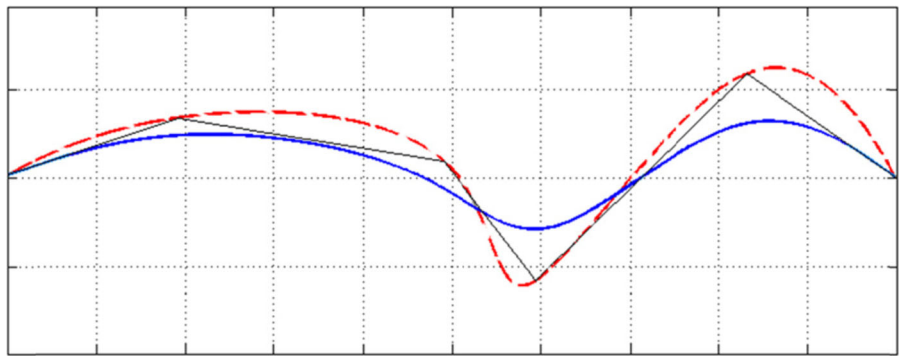

Fig. 5 Clamped cubic B-spline interpolation (dashed red) and smoothing (blue)

essential for any cognitive robot. The advantages of using B-spline curve, in local replanning scenarios, were also considered. Finally, a discussion is provided on the suitable parameters for the curve in robot path planning scenarios.

3.1 Precise Path Modeling

The smoothing curve must interpolate the start and finish points of the piecewise linear path. It is required to closely follow the original path whilst obeying the

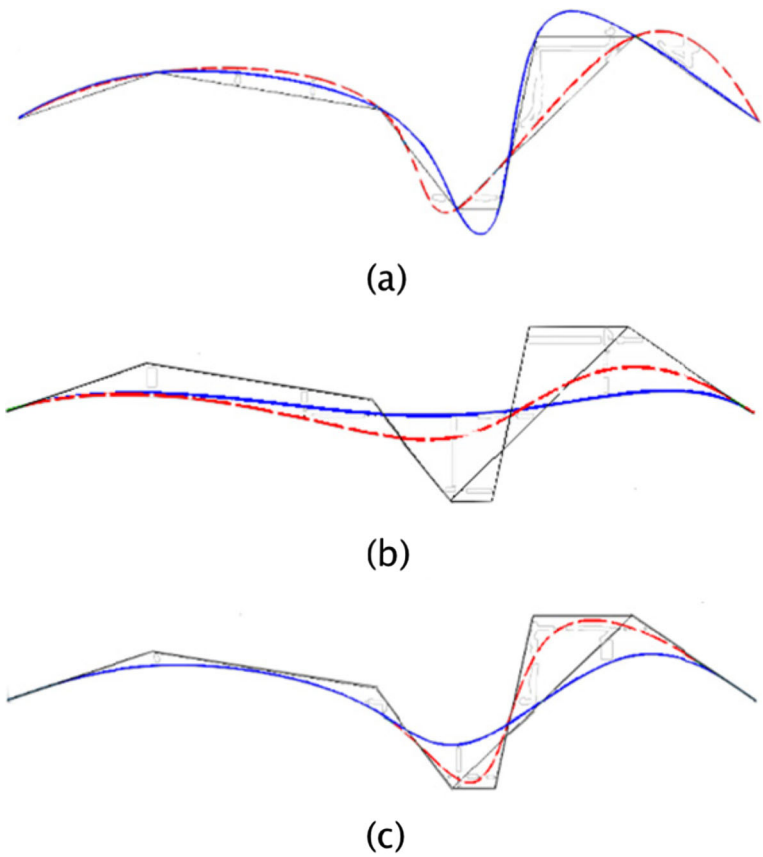

Fig. 6 Original paths are shown in blue and replanned in red (dashed). **a** Interpolation using B-spline curve **b** Smoothing using Bézier curve **c** Smoothing using B-spline curve

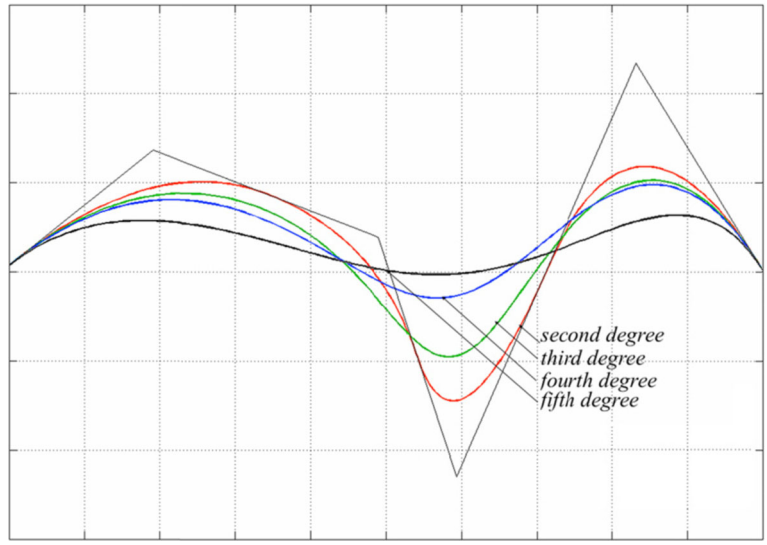

Fig. 7 Path smoothing using all possible clamped B-spline curve degrees

curvature constraints. We compare a Bézier curve with a clamped B-spline curve shown in Fig. 3. Clamped B-splines are used to interpolate the start and end points of the linear path. Fifth order Bézier curves must be used as the path has six waypoints. It was proposed to use multiple Bézier segments to maintain a fixed order. However, it results in path discontinuity. It can be noticed later, that B-splines follow the path more robustly whilst maintaining continuity and curve order.

It is possible to use a clamped NURBS curve of the same order for path smoothing, as shown in Fig. 4. Increasing a point's relative weight, w_i in Eq. 6, will lead shifting the curve towards that point. On the other hand, the curve will shift away from the other points, as the point's weights are relative. The path 6, incorporates a divisor that is sum of all weighted points. The shift away from the other points may lead to obstacle collision. It might be useful to investigate the exploitation of weights for path planning, but in this paper we focus on clamped B-splines (or NURBS curves of equal weights).

Six NURBS curves of different weights are used for path smoothing. The path shifts towards the control point with the largest relative weights, as shown by the red arrows. As previously mentioned, NURBS are a weighted variant of B-spline curves. In other words, a B-spline is a special case of NURBS where all points

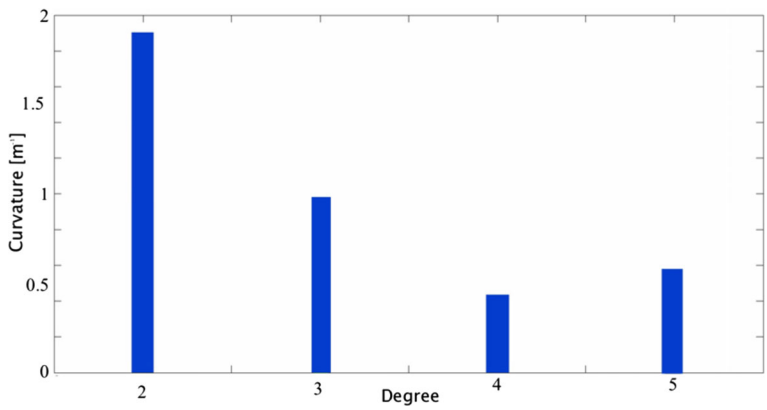

Fig. 8 Maximum path curvature for all possible clamped B-spline curve degrees

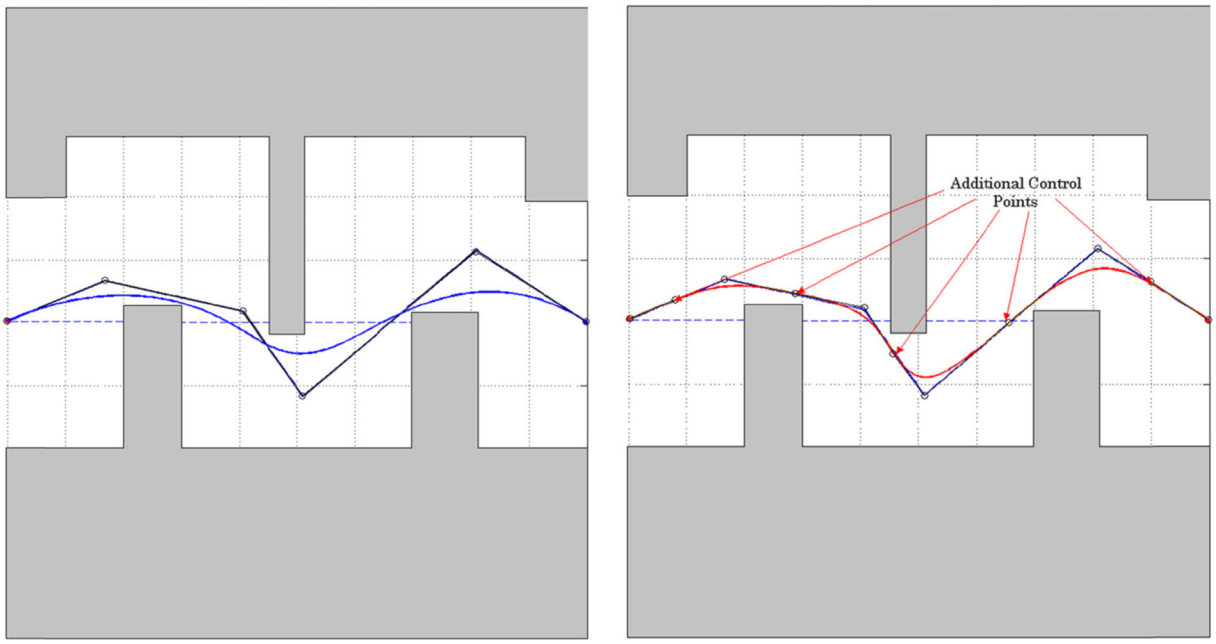

Fig. 9 Midpoint insertion forces the B-splines tangency following to all control polygons and improves path following

are equally weighed. The ability to locally control path is essential, but, uncontrolled change, in case of NURBS weights, may lead to collision, or generation of unfeasible paths (exceeding the vehicle's maximum curvature). Adjusting NURBS weights may be beneficial if the rest of the path can be controlled, with a reasonable effort and processing time. However, it does not appear to be advantageous to the path's continuity, which can be achieved using B-splines, or efficiency, as it has the same synthesis methods based on de-Boor's algorithm.

Some works suggested path interpolation as a method of smoothing [60]. The difference between B-spline smoothing and interpolation is illustrated in

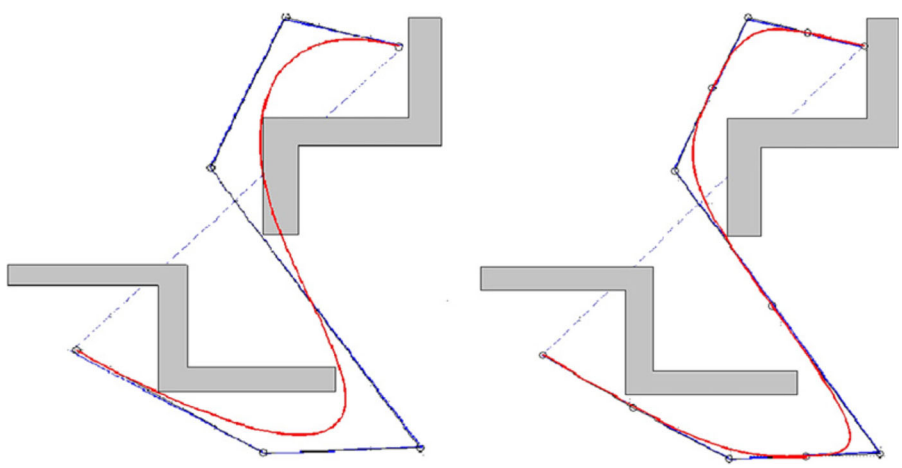

Fig. 10 Midpoint insertion example

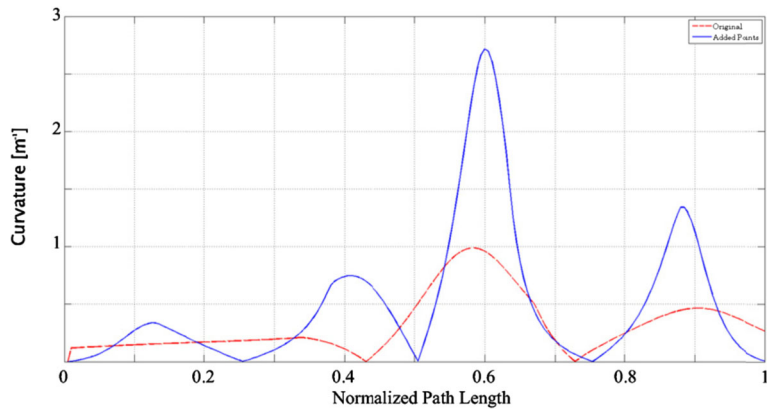

Fig. 11 Effect of midpoint insertion on path curvature

Fig. 5. We can see that interpolated path, shown as a red dashed line, is longer. It has sharper movements, which increases the path curvature and so defeat the purpose of path smoothing. Interpolated path does not lie within the convex hull of the collision free control polyline, increasing the likelihood of obstacle collision.

3.2 Local Modification

Path segment replanning is common whilst executing the path. Perception systems are constantly updating robot's model of the environment. Update rates are set to be high enough, to ensure safe navigation, responsiveness and prevent collision [69]. As a result, the replanning procedure should be capable of running at a timely rate. B-splines are locally controllable, i.e. changing a segment of the curve will not affect the other segments, which makes them suitable for replanning in dynamic environments.

The replanning ability of B-spline curves is highlighted in Fig. 6c, in comparison to a Bézier curve, Fig. 6b, and B-spline interpolation, see Fig. 6a. The B-spline curve is locally modified where the original path is modified. In case of Bézier curves the order of the curve must be changed when the number of control points is changed and the entire path is changed.

3.3 Curve Parameters

Clamped cubic B-splines outperform other parametric curves and smoothing methods, in path following and replanning. Consequently, they have been selected for continuous path smoothing. . Clamping is achieved

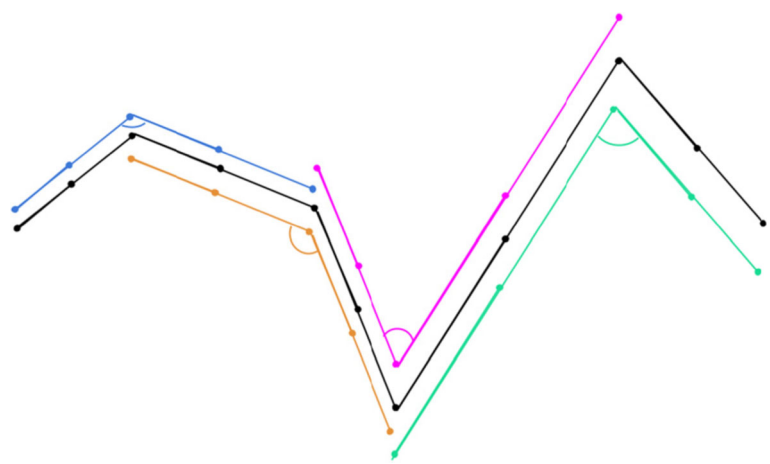

Fig. 12 Subdiving the path into repeaing segments of five succesive points

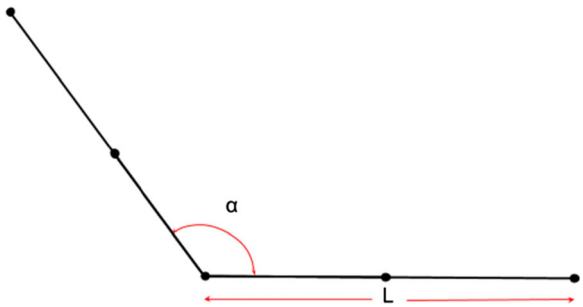

Fig. 13 Segment parameters; Length (L) and angle ($\propto$)

by changing the initial and final knots' multiplicity in Eq. 5. In this section we attempt to select the appropriate curve order. A p^{th} degree clamped B-spline curve can smooth a path of minimum, *p-1*, number of control points. We compare the allowable B-spline curve degrees (2^{nd} to 5^{th}) for a six-point path. The lower the path degree, the better it follows the original linear path, as shown in Fig. 7. On the other hand, lower order paths have an adverse effect on the curvature along the path, illustrated in Fig. 8. A cubic path is selected to balance accurate path representation and smooth curvature. Maintaining a lower order also required less iterations of the de-Boor algorithm, (5), which is the bottleneck of the B-spline and NURBS synthesis.

4 Continuous Smoothing

The novelty of this approach, as previously mentioned, is guaranteed path continuity, given by Eq. 9, along the entire path whilst satisfying maximum curvature, defined by Eq. 8. In this section a clamped cubic B-spline curve is used to generate a path that satisfies required constraints.

4.1 Path Continuity

Path discontinuities will arise when two segments are arbitrary joined together. Continuous paths were generated using a single B-spline curve in [63]. That algorithm relied on the fixed number of control points used for smoothing. Whereas in here, we proposed using a single B-spline curve, with unlimited number of control points. Even though B-splines outperform Bézier curves in path following, they can still cause collision, see Fig. 9 (left) and Fig. 10 (left). B-splines are tangent to control line, with three or more collinear

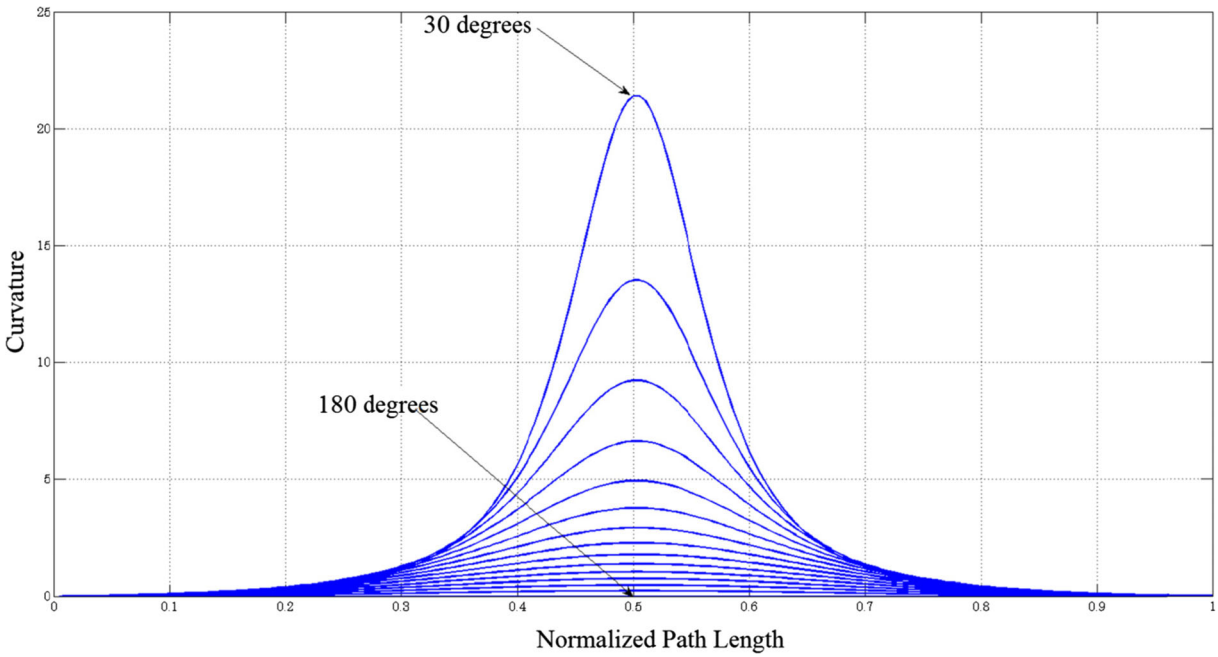

Fig. 14 Effect of varying segment angle α on path curvature k

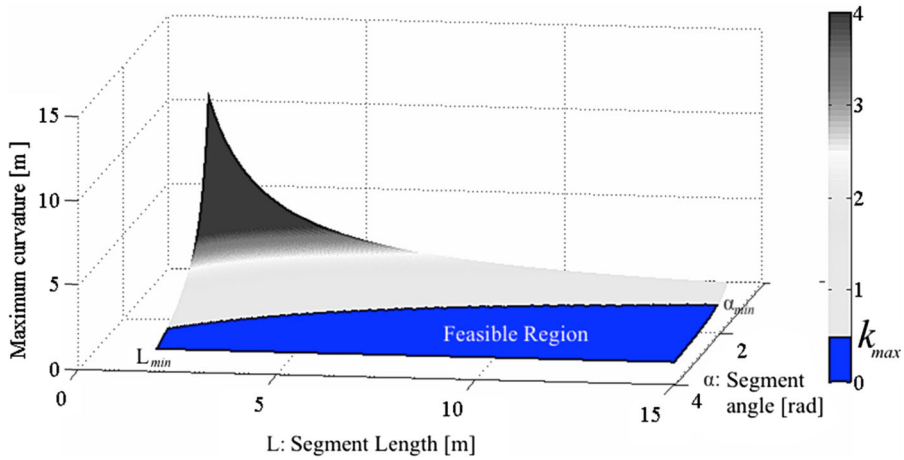

Fig. 15 Combined effects of segment parameters variables, segment length L and angle α, on the curvature k of the corresponding clamped cubic B-spline curve. The feasible parameter region is highlighted in blue

control points [41]. The tangency property is utilized to allow a single B-spline curve to accurately represent the path and maintain continuity. This is achieved by inserting a point in between every two successive control points, referred to as *midpoint insertion*. This effectively forces the B-spline curve to follow the path more accurately using the tangency property. The benefit of midpoint insertion, in improving path following, is illustrated in Figs. 9 and 10. Prior to path planning the environment is expanded, with respect to the robot size. Consequently it is possible to represent the robot as a point in the environment (configuration space), as shown in Figs. 9 and 10. In those examples, when the path is close to obstacles, in reality the robot is still further away. Benefit of midpoint insertion is not simply the safe distance from obstacles, which has been achieved by Voronoi diagram, Minkowski sum, Delaunay triangulation [73] planning methods. It also contributes to maintaining a resemblance to the originally planned linear path. The advantage of midpoint insertion is made clear in Fig. 9 as the path ended up in collision after smoothing without midpoint insertion.

4.2 Maximum Curvature Condition

Midpoint insertion forces the path's tangency to the control lines and leads to increasing the curvature, as shown in Fig. 11. Any path, as shown in Fig. 12, can be subdivided into segments of five successive points

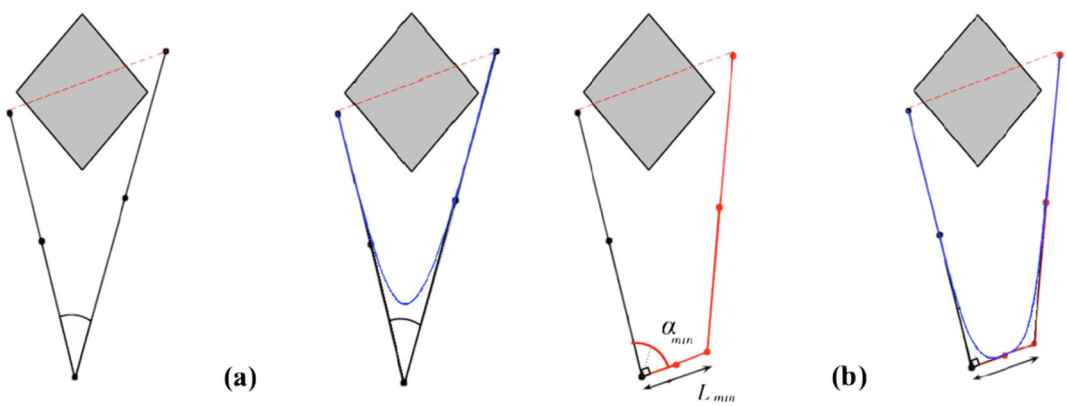

Fig. 16 **a** Segment whose angle exceeds α_{min} and the resulting B-spline path curvature exceeds k_{max} **b** By adding a path segment with parameters, α_{min} and L_{min} curvature continuity is guaranteed and maximum curvature constraint is satisfied

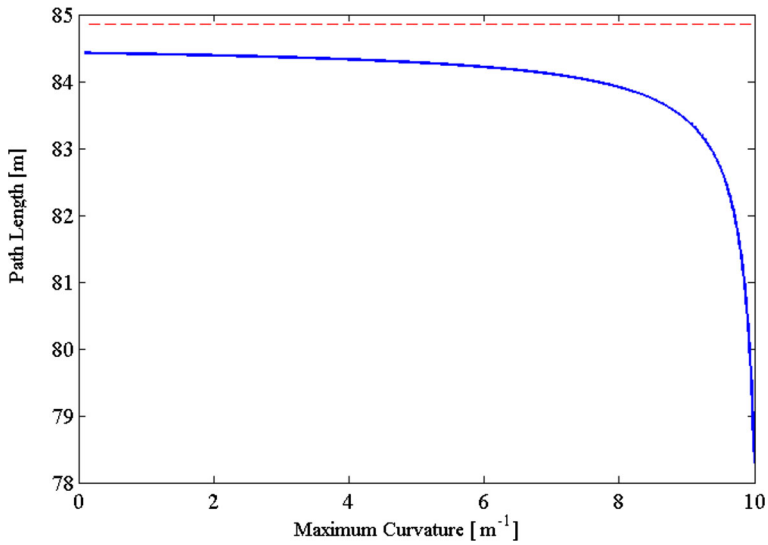

Fig. 17 Path length for different kmax values using the proposed B-spline smoothing algorithm

(three control points and two midpoints), as shown in Fig. 13. In this section we take advantage of B-splines local control property, to satisfy the maximum curvature constraints in each segment. First, the curvature of a segment was formulated with respect to its parameters. Segment parameters were set based on the robot's maximum curvature, defined by Eq. 7. Finally, we introduced a smoothing algorithm that modifies segment parameters, to satisfy maximum curvature constraint.

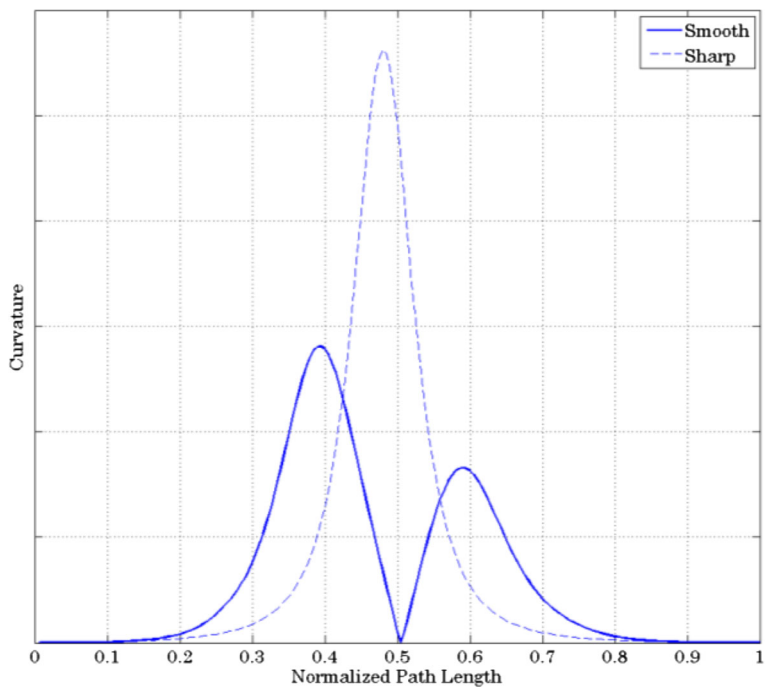

Fig. 18 Curvature profile before (dashed) and after (solid line) adding a path segment with $\alpha_{\min}$ and $L_{\min}$

Table 1 Benchmarking experiments summary

Algorithm	Curve	Curvature limits	Continuity
[60]	Polynomials	No	G^2 continuity
[56]	Bézier	No	Not continuous
[58]	Bézier	Yes	G^2 continuity
Proposed	B-spline	Yes	C^2 continuity

4.2.1 Segment Curvature Formulation

Segments consist of five points, which construct two intersecting lines. Two parameters of any segment are the segment length, L, and segment angle, α, as illustrated in Fig. 13. L is taken as the shorter segment side length. Five control points $P_i = (P_{xi}, P_{yi})$, *for* $i = 1$*-5,* can be formulated in terms of the segment parameters as given in Eqs. 10 and 11.

$$P_x = \left[L, \frac{L}{2}0, \frac{L}{2}\cos\propto, L\cos\propto\right] \tag{10}$$

$$P_y = \left[0, \quad 0, \quad 0, \quad \frac{L}{2}\sin\propto, \quad L\sin\propto\right] \tag{11}$$

In order to fully satisfy maximum curvature constraint, path curvature was derived as a function of the segment parameters. The curve equation, corresponding to the primitive segment, was computed by substituting control points,Eqs. 10 and 11, in the B-spline curve and basis functions (3), (4) and (5). The curve has n *=5* control points, p *=3* (cubic curve) and a knot vector, with m *=9*, and with quadruple initial and final multiplicities [*0,0,0,0,0.5,1,1,1,1*]. The resulting path, $c(u) = [x(u), y(u)]^{\mathrm{T}}$, is given by Eqs. 12 and 13. The derivation is detailed in Appendix A.

Table 2 Section 5.1.1: Path length and curvature results

Algorithm	Length	Curvature	Continuity
Linear	161 m	∞	None
[60]	256 m	1.5 m^{-1}	G^2
Proposed	137 m	3.0 m^{-1}	C^2
Proposed (bounded)	148 m	1.1 m^{-1}	C^2

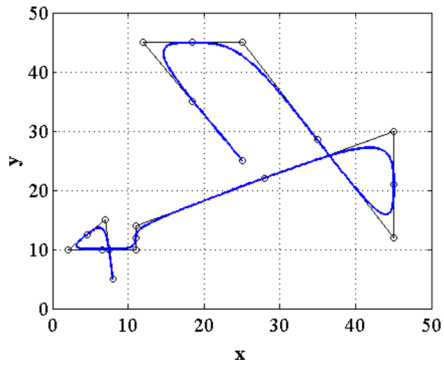

Fig. 19 B-spline smoothing for waypoints in [60]

$$\begin{aligned} x(u) = \; & L\left[\frac{(1-2u)^3}{2} + \left(3u^3 - 3u^2 + \frac{1}{2}\right)\right. \\ & \left. + \left(-3u^3 + 6u^2 - 3u + \frac{1}{2}\right)\cos\alpha + \frac{(2u-1)^3}{2}\cos\alpha\right] \\ y(u) = \; & L\sin\alpha\left[\left(-3u^{3+}6u^2 - 3u + \frac{1}{2}\right) + \frac{(2u-1)^3}{2}\right] \end{aligned} \tag{12}$$

Path equation, $c(u)$, its first and second order derivatives, $c'(u)$ and $c''(u)$, were substituted in path curvature, k, 13. The segment curvature was defined as a function of the corresponding segment parameters in Eq. 14.

$$k = \frac{x(u)'y(u)'' - y(u)'x(u)''}{\left(x(u)'^2 + y(u)'^2\right)^{3/2}} \tag{13}$$

$$k = \frac{2(u\sin\propto)(1-u)}{3L(2u^2(1-\cos\propto)(u^2-2u+1) + (2u-1)^2)^{3/2}} \tag{14}$$

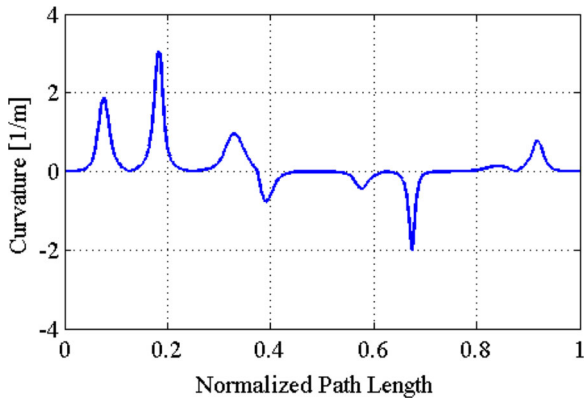

Fig. 20 Corresponding B-spline curvature for waypoints

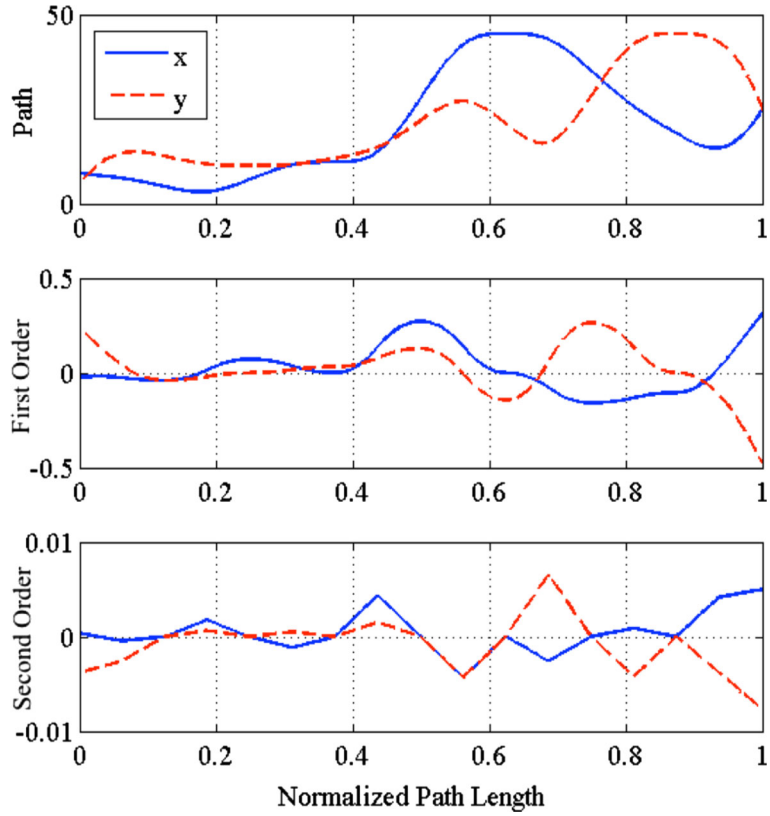

Fig. 21 21. Path is C^2 continuous since its first and second order derivatives are continuous

4.2.2 Segment Parameters

Segment curvature analysis was performed to gain insight on the effect of the parameters on the curvature and define parameters that satisfy the maximum curvature condition. Based on this analysis we will propose a smoothing algorithm to generate feasible B-spline paths for car-like robots.

Increasing the segment angle reduce the curvature of the path, as shown in Fig. 13. It can be noted that a

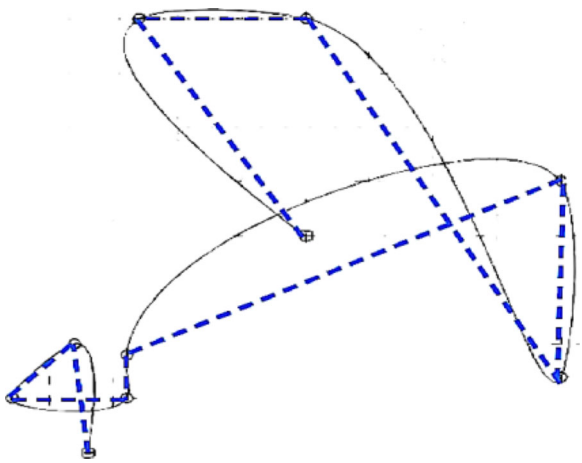

Fig. 22 Polynomial G^2 interpolation reproduced from [60]

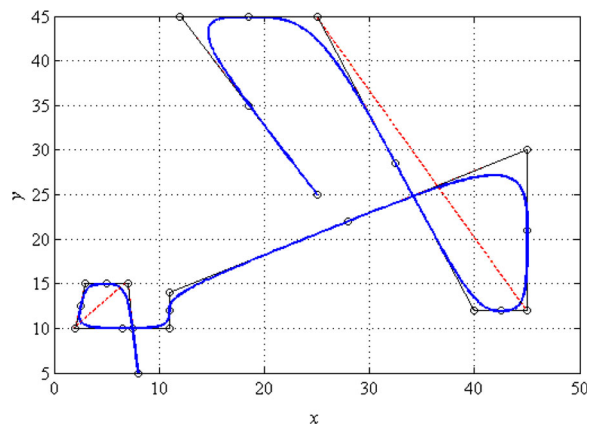

Fig. 23 B-spline smoothing with bounded curvature. Two path smoothing maneuvers were needed

continuous curvature profile is obtained for all cases. In Fig. 14, the region in which the segment parameters satisfies curvature constraints is highlighted in blue. In this example the maximum curvature was 0.5 m^{-1} and the minimum segment parameters were taken as 90 degrees and 2 meters. The goal of the smoothing algorithm is to select parameters, within that range, to ensure that the segment curvature is less than the maximum vehicle curvature. We refer to it as the feasible region.

In order to ensure that the curvature along the path is less than the maximum curvature, the combined effect of the segment parameters must be understood. Feasible segment parameters must be selected in order to ensure that the robot can execute the generated path. Segment feasibility could be assessed by comparing the segment curvature, (14), with the vehicle's curvature constraint, given by Eq. 8.

Analytically solving (14) for the required angle, in each segment, to satisfy the maximum curvature constraint is intractable, considering that it would be evaluated multiple times in the smoothing process. In cases of offline-planning (14) could be solved but this is not sustainable for efficient online replanning situations. Online planners were proposed to replan linear paths [74, 75], which require rapid smoothing. Defining a minimum segment angle, $\alpha_{\min}$, and computing a corresponding minimum segment length, $L_{\min}$, is proposed to circumvent this solution. If a segment's angle exceeds its thresholds it must be modified.

For a given, $\alpha_{\min}$, it is required to find the minimum segment length at which $k = k_{max}$. From Fig. 14, or by solving $\frac{\partial k}{\partial u} = 0$ for u, we can see that $k = k_{max}$ at $u = 0.5$. $L_{\min}$ can be calculated by substituting in Eq. 14 by $u = 0.5$, $k = k_{max}$ and $\alpha = \alpha_{min}$, as given in Eq. 15.

$$L_{\min} = \frac{1}{6}\sin\alpha_{min}.\rho_{min}.\left(\frac{1-\cos\alpha_{\min}}{8}\right)^{-3/2} \quad (15)$$

4.2.3 Smoothing Algorithm

A segment, whose angle, α, exceeds the threshold, α_{min}, is considered infeasible and required curvature bounding. An example is illustrated in Fig. 16a with its corresponding path. As a result, the corresponding B-spline path will exceed the maximum curvature. It is required to modify the segment, to satisfy that curvature constraint whilst maintaining path continuity, i.e. without adding another curve.

Extending the segment angle to $\alpha_{\min}$ and adding another segment of length L_{min}, as shown in Fig. 15b, ensures the curvature will be equivalent to k_{max}. This maneuver relies on the local control feature of B-splines, so segments eecan be modified with minimal effect to other segments and a single curve can be used to maintain path continuity. The effect of smoothing maneuver on the curvature is shown in Fig. 18. The curvature continuity was maintained and maximum curvature was converted into two curvature peaks representing two segments with feasible parameters. The smoothing procedure is outlined in Algorithm 1.

Algorithm 1: B-spline_Smoothing

1 **input:** Maximum curvature k_{max}, Waypoints coordinates P=[P_x, P_y], angle threshold $\alpha_{min.}$
2 **output:** B-spline path coordinates, c(u) = [x(u), y(u)]
3 **for** every two successive waypoints **do**
4 Insert midpoint P_{mid} between P_i and P_{i+1}
5 Update waypoints P'_x, P'_y
6 end for
7 $L_{min} = \frac{1}{6}\sin\alpha_{min}.\rho_{min}.(\frac{1-\cos\alpha_{min}}{8})^{-3/2}$
8 **for** every segment **do**
9 if $\alpha > \alpha_{min}$ do
10 Construct segment with L_{min} and α_{min}
11 Insert midpoint in new segment
12 Update waypoints P'_x, P'_y
13 end if
14 end for
15 $N(u)$ = deBoor_Algorithm(P'_x, P'_y, *degree* = 3, *clamped*)
16 $x(u) = \sum_{i=0}^{n} N_{n,i}(u).P_{xi}$
17 $y(u) = \sum_{i=0}^{n} N_{n,i}(u).P_{yi}$
18 **return** $x(u)$, $y(u)$

Algorithm 1, illustrated in Fig. 15, bounds the curvature by adding a path segment with predetermined bounding parameters, $\alpha_{\min}$ and $L_{\min}$ (Algorithm 1 lines 10-12). The resulting B-spline path length is consequently increased. This behavior is expected, as the original linear path did not consider the constrained

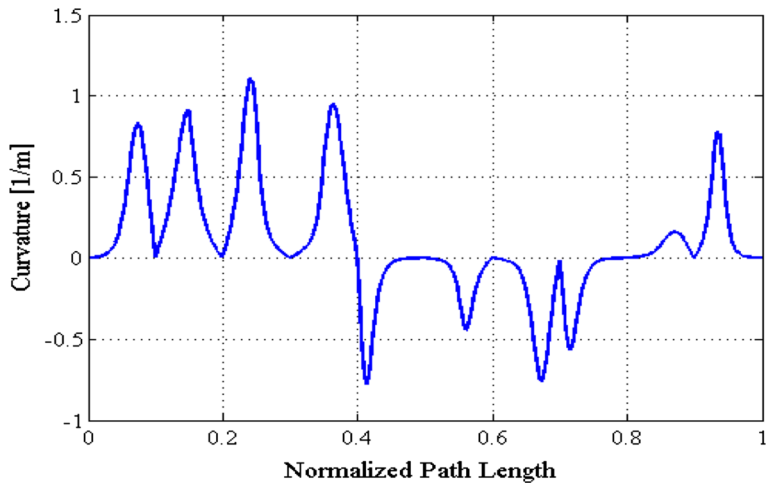

Fig. 24 Bounded B-spline continuous curvature profile

motion of the robot. In general, most planners attempt to minimize the path length. For differentially constrained kinodynamic constraints cannot be simply captured by the path distance metric. Selecting an appropriate metric for a nonholonomic system, or an underactuated system, is challenging task, see chapter 5 [76]. By examining Eq. 1, as the minimum turning radius is smaller (more restricted vehicle's capabilities), expected maneuver segment length would increase.

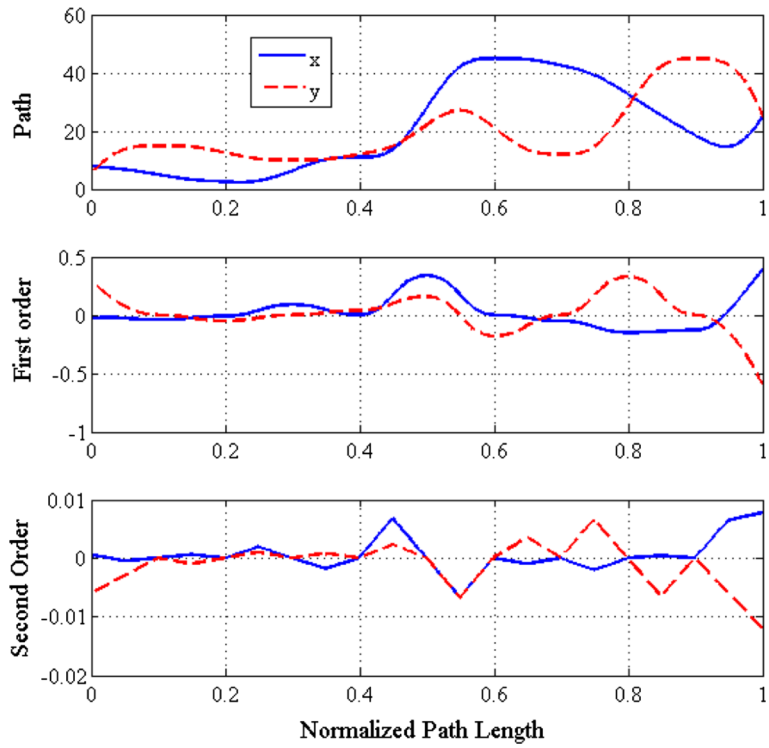

Fig. 25 Path is C^2 continuity is maintained after curvature bounding

Table 3 Section 5.1.2: Curvature and path length results

Algorithm	Maximum Curvature	Continuity
Hermite	0.5 m^{-1}	None
Cubic Splines	0.27 m^{-1}	None
[56]	0.065 m^{-1}	None
Proposed	0.065 m^{-1}	C^2
Proposed (bounded)	0.04 m^{-1}	C^2

Algorithm	Path Length
Linear	7 679m
Proposed	595 m
Proposed (bounded)	657 m

In Fig. 17, the effect of maximum curvature limits on the total B-spline length is presented. As expected, the path length increases as the vehicle is more constrained (lower curvature bound). The original path is a segment with two straight lines separated by an angle less than $\alpha_{\min}$. The control points coordinates are $P_x = [0, 30, 60]$ and $P_y = [0, 30, 30]$, and the total path length is 84.8 m. It is interesting to note that in all cases, the re-planned B-spline path length did not exceed the originally planned straight lines length. This property was further investigated in the numerical experiments (see Sections 5.1.1 and 5.1.2).

5 Results and Discussion

The newly proposed method was tested in three sets of experiments. We compared it with existing smoothing algorithms, to highlight the novelty of our approach.

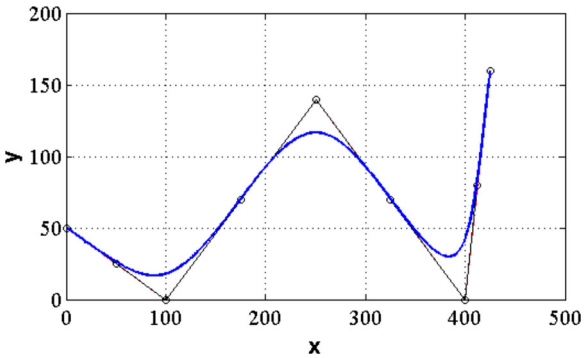

Fig. 26 B-spline smoothing with no curvature limits

The first sets of experiments prove that the proposed method improves smoothing, path following and minimizes disturbances, when compared to recent contributions in the available literature.

We followed by combining our smoothing approach with a randomized path planning algorithm, for a robot with maximum curvature constraint, to illustrate the general algorithm behavior. Random planning algorithms were nominated, as they often result in suboptimal paths with redundant turns. Therefore, we showed that our method would be beneficial for sampling based planning implementations.

Finally, we used a real remote-controlled robot to highlight the resemblance between our algorithm and human steering in four different situations with varying complexity.

The smoothing algorithm used in this section was implemented in Matlab® on a 2.8 GHz machine with 8 GB of Memory running OS X® 10.9.2. The mobile robot control and data acquisition programs were written in C and executed on an onboard microcontroller. The logged data was transferred serially to a host computer and post processed using Matlab. The manual operation control was implemented over two separate 2.4 GHz radio channels for speed and direction.

5.1 Benchmarking Experiments

We have shown that novel algorithm guarantees C^2 continuity and bounds curvature limits along the entire path. Related work was discussed in Section 1.1. In this section we compare the most recent and relevant smoothing algorithms, with our proposed methods. Examples used in this section were implemented in the

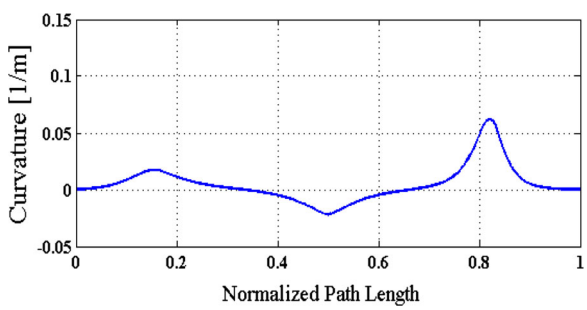

Fig. 27 Corresponding curvature is continious and has a maximum value of $0.07m^{-1}$

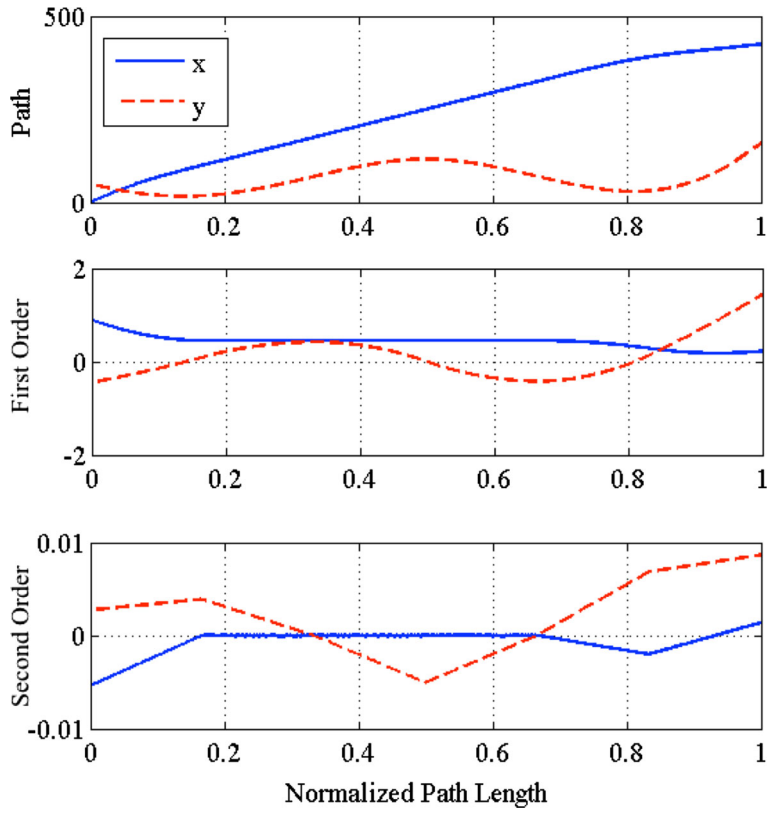

Fig. 28 Path is C^2 continuous; its first and second order derivatives are continuous

respective papers to provide a common benchmark. A summary of the results is given in Table 1.

5.1.1 Case 1:Huh, Chang [60]

The algorithm in [60] used polynomials to interpolate waypoints with G^2 continuous curves and did not consider maximum curvature constraints. The experiment presented in [60] is repeated twice using our proposed algorithm with and without curvature bounds. The path length and curvature results comparing the two B-spline solutions with the polynomial results are

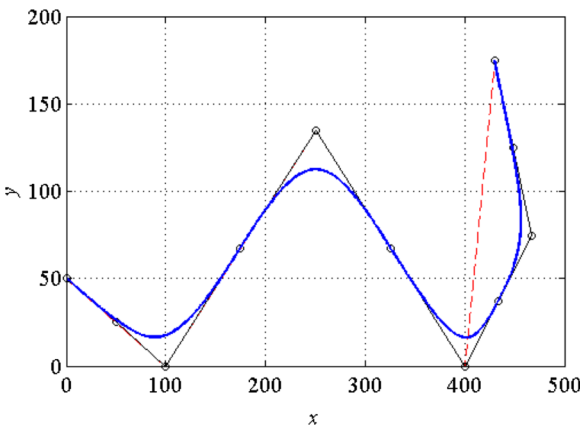

Fig. 29 B-spline smoothing with curvature limit

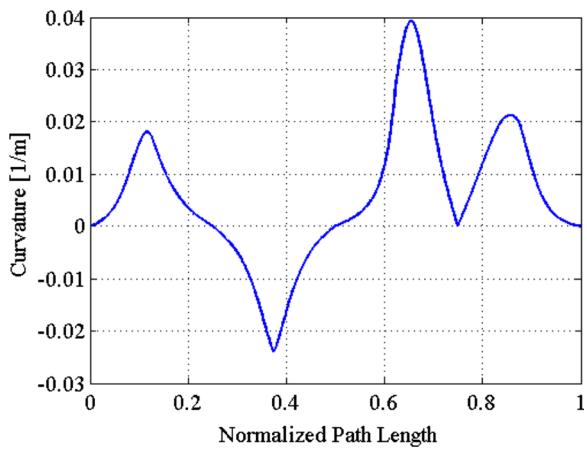

Fig. 30 Corresponding curvature obeys the $0.04m^{-1}$ curvature limit

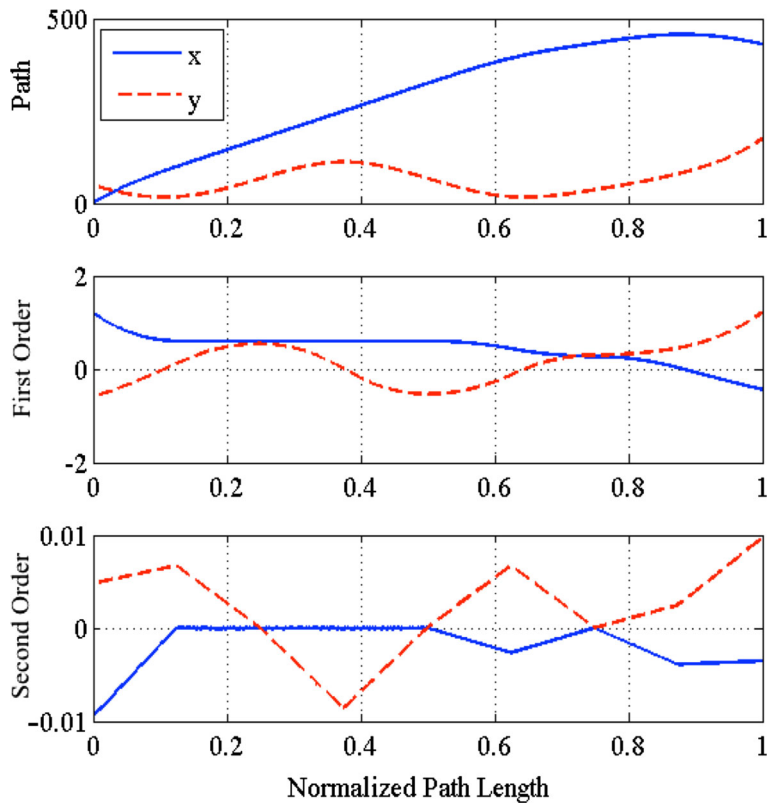

Fig. 31 C^2 continiuty is maintained after modifiying the path to satisfy maximum curvature

listed in Table 2. They show that our method is capable of generating smoother, shorter and continuous paths.

The resulting B-spline curve for the same waypoints, using our approach, is shown in Fig. 19 and its corresponding curvature is given in Fig. 20 A C^2

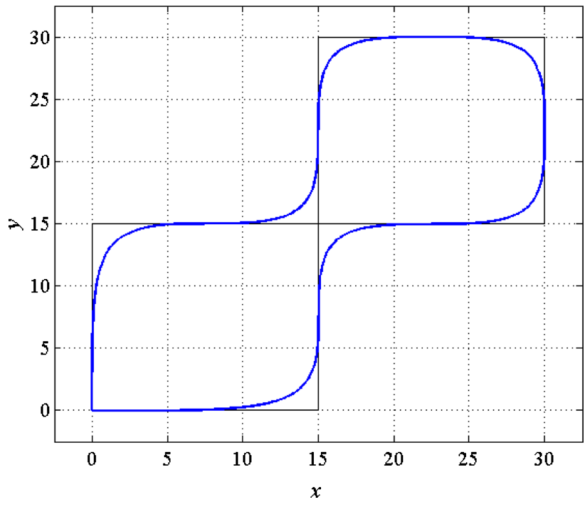

Fig. 32 B-spline smoothing for seven consecutive and opposing turns

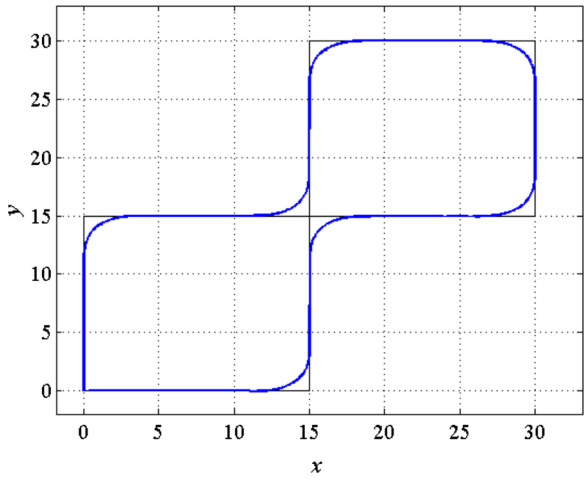

Fig. 33 Bezier G^2 smoothing for seven consecutive and opposing turns

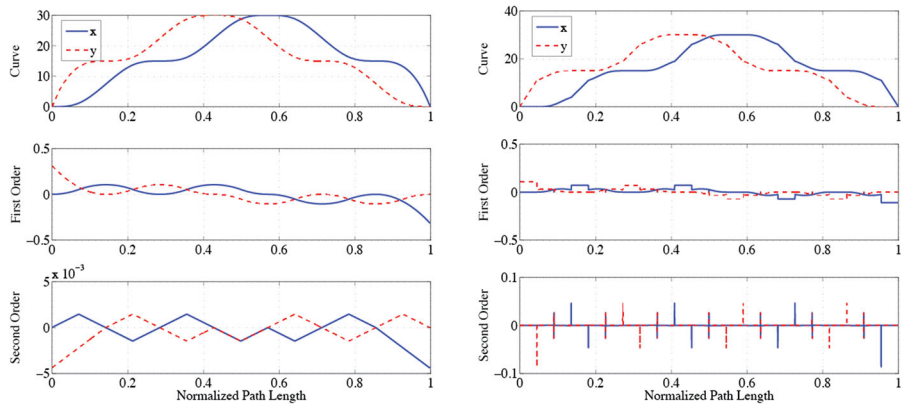

Fig. 34 Seven segment turns resulting path (top), velocity (middle) and acceleration (bottom) profiles with proposed algorithm (left) and with [45] (right)

continuous path was obtained as opposed to a geometric G^2 path that does not guarantee velocity, or acceleration continuity. This is illustrated by the continuity of the path and its first and second order derivatives as shown in Fig. 21. The results shown in Fig. 21 are obtained from Eq. 3 and its derivatives; they serve to show the continuity of the resulting trajectory with respect to the global coordinate system.

The proposed path, in Fig. 19, lied within the convex hull of the control polygon, which led to the decrease in length in comparison to interpolation. Interpolation using G^2 polynomials, in Fig. 20, resulted in oscillating paths that strayed from the original linear path and may have potentially led to obstacle collision. This deviation is apparent when examining the path lengths in Table 2. In contrast, our approach was strict in path following as a result of midpoint insertion procedure. Subsequently it generated higher curvature values when no maximum curvature limit was set, see Fig 20.

Improved smoothing was achieved through bounding the curvature of the B-spline path to a desired value. In this case the curvature was bounded to 1.1 m^{-1} which was significantly lower than the achieved 1.5 m^{-1} in [60]. The resulting path is shown in Fig. 23 and the corresponding bounded continuous curvature in Fig. 24. Two segments were needed to bounding the curvature. Modified segments are highlighted as dotted red lines in Fig. 23.

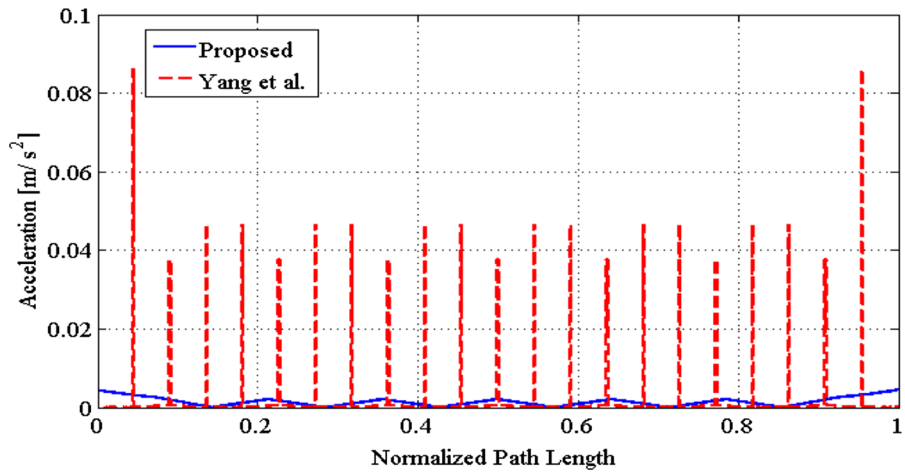

Fig. 35 Seven segment turns resulting acceleration from proposed algorithm (blue solid lines) and from [45] (red dashed)

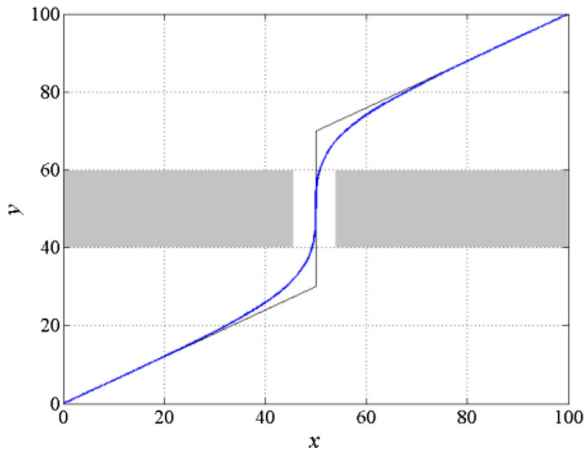

Fig. 36 Narrow passage smoothing using the proposed B-spline

Figure 25 shows that the proposed method was capable of maintaining C^2 continuity when bounding curvature. Despite the additional two segments for smoothing, it can be seen in Table 2 that the bounded B-spline path length was still shorter than both the polynomial and linear paths, which agrees with the results presented in Fig. 17.

5.1.2 Case 2: Zhou et al. [56]

Bézier curves were used for path smoothing and were shown to generate lower curvature values in comparison to cubic Hermite and cubic splines, as defined in [56]. However, that approach did not consider path continuity and curvature limits. The comparative experiments in [56] were repeated twice using our proposed method with and without curvature bounding. The results are summarized in Table 3. First, we generated B-spline path as shown in Fig. 26 without setting limits on the curvature. In this case when examining Fig. 27, the B-spline curvature peaked at $0.065\ m^{-1}$, which was equivalent to using Bézier curves [56]. However the proposed path was still C^2 continuous, see Fig 28.

The experiment was repeated with B-spline path smoothing and a $0.04\ m^{-1}$curvature bound. It can be seen in Fig. 29, that the final path segment was modified to satisfy the curvature condition. Regardless of the added path segment, the B-spline path was still shorter than the original planned linear path, see Table 3. The desired maximum curvature, $0.04\ m^{-1}$, was satisfied, as shown in Fig. 30. The path's C^2 continuity was maintained after bounding, as shown Fig. 30. It is clear that new approach generated smoother continuous paths.

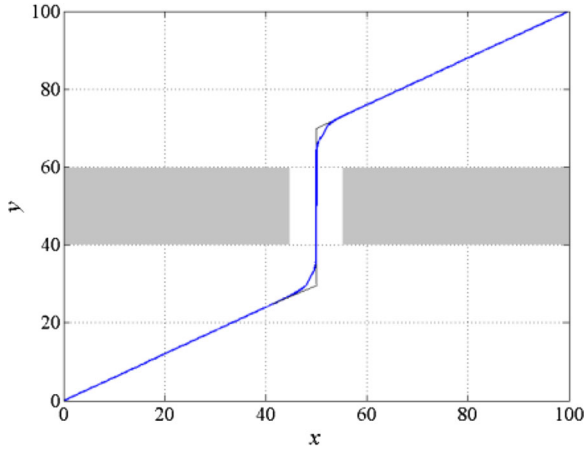

Fig. 37 Narrow passage smoothing using [45]

5.1.3 Case 3: Kwangjin et al. [58]

Bézier curves geometric G^2 condition was used for path smoothing by [57, 58]. It forced the vehicle to reach maximum curvature limit in every corner and could not guarantee velocity and acceleration continuity. The G^2 smoothing method was compared to our proposed B-spline algorithm in two different scenarios. We also illustrated the significance of maintaining velocity and acceleration continuity.

The first example involves seven consecutive turns of opposing orientation (clockwise and counterclockwise). This was chosen, as it is a challenging task for a smoothing algorithm to generate a feasible path whilst minimizing the abrupt changes acting on the robot. The resulting B-spline and Bezier paths are given in Fig. 32 and Fig. 33. It is clear that both methods generate similar paths topology with continuous curvatures. This is more evident when examining the path profile for both methods in Fig. 34 (top). Further analysis of the velocity and acceleration profiles validates that the proposed method is C^2

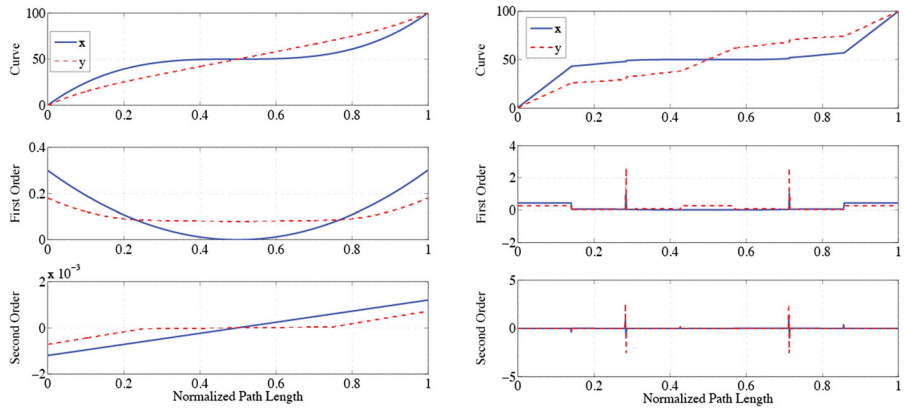

Fig. 38 Narrow passage smoothing resulting path (top), velocity (middle) and acceleration (bottom) profiles for proposed algorithm (left) and [45])(right)

continuous where the Bezier method exhibited sudden local changes when Beziers and straight lines were coalesced. The benefit of generating a C^2 continuous path is highlighted when comparing the resulting accelerations for both paths in Fig. 35. The sudden discontinuities resulting from joining Beziers and straight lines led to acceleration peaks in contrast to the proposed method, which resulted in minimal disturbances.

The second experiment was a narrow passage manoeuver. Such scenarios are particularly challenging for path smoothing algorithms. If discontinuous paths, oscillating poorly interpolated paths, or non-realizable paths are generated they might lead to overshooting and ultimately passage collision. The linear paths in Figs. 36 and Fig. 37 were generated using a Visibility graph variant proposed in [24]. The paths were smoothed using the proposed methand

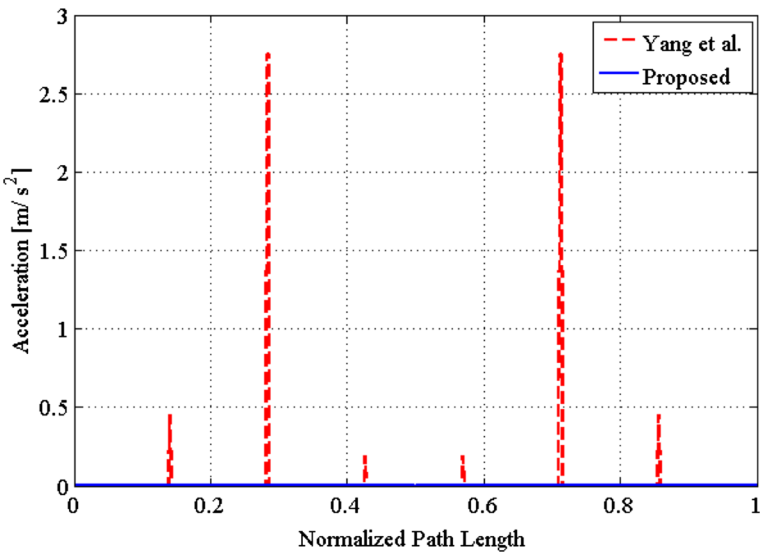

Fig. 39 Narrow passage resulting acceleration from proposed algorithm (blue solid lines) and from [45] (red dashed)

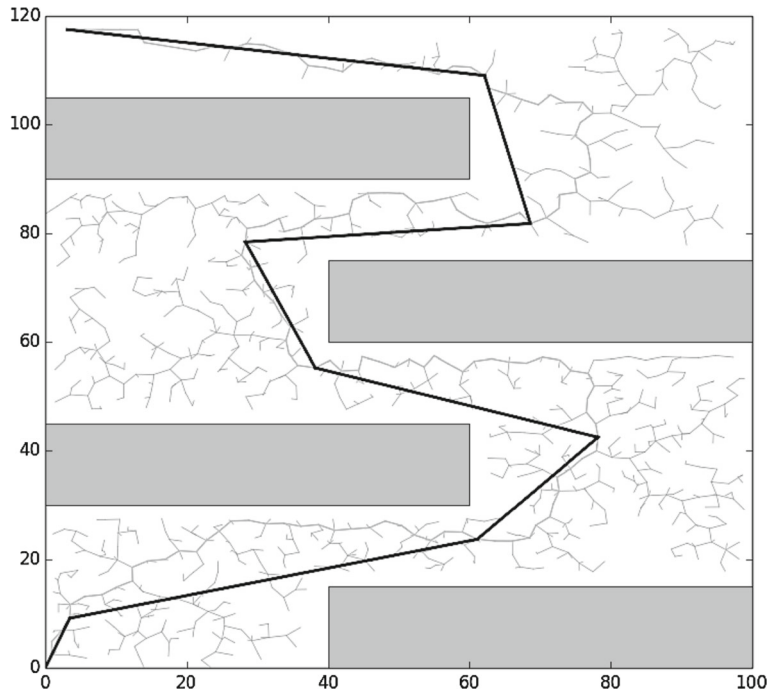

Fig. 40 RRT (1): Environment and piecewise linear path

in [57, 58]. The results validated the significance of C^2 continuity. Both paths had similar curvatures, see Fig. 38 (top), whereas our method has maintained C^2 continuity and minimal disturbances, as shown in Fig. 39.

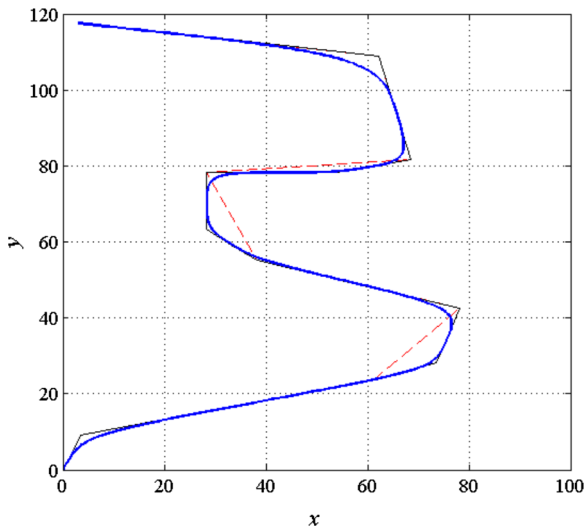

Fig. 41 RRT (1): B-spline path required three smoothing maneuvers. The original sharp turn segments are shown as dashed red lines

5.2 Path Planning Experiments

In this section, B-spline smoothing is implemented with rapidly-exploring random tree (RRT) path planner [77], in two different environments. RRTs were selected as a result of their increasing popularity in robot planning [9]. Despite the randomized algorithms

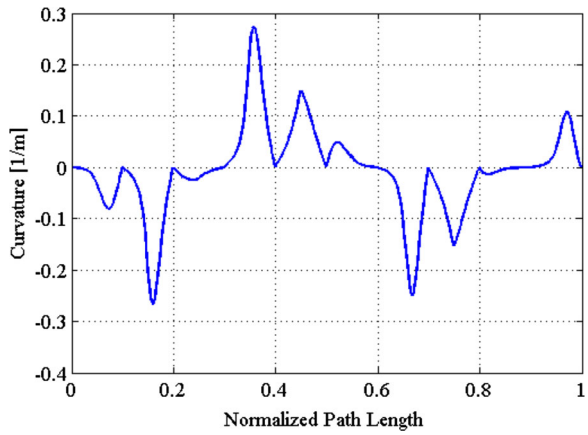

Fig. 42 RRT (1): Continuous curvature obeying the maximum constraint of 0.3 m^{-1}

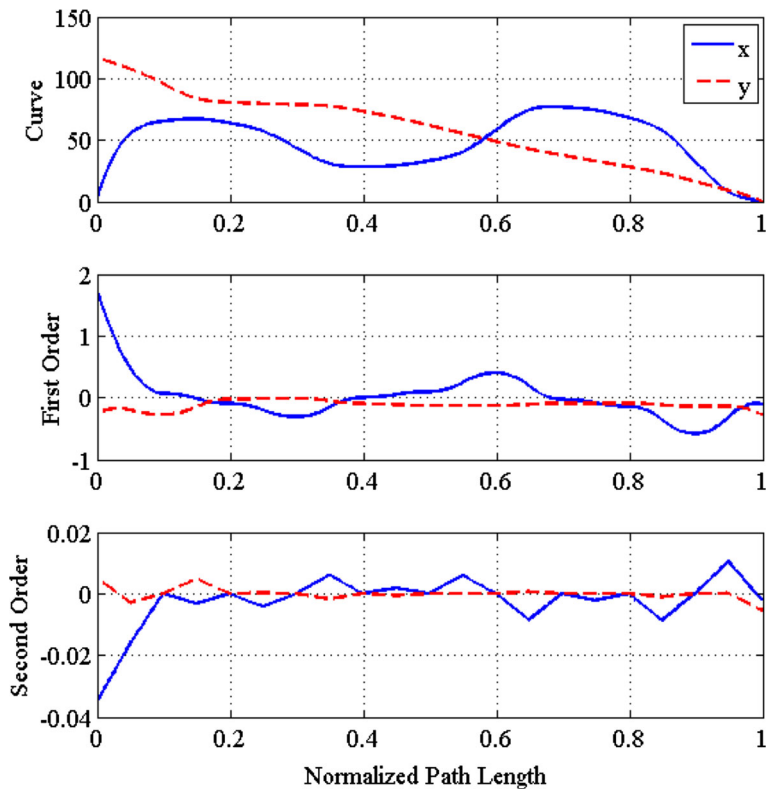

Fig. 43 RRT (1): B-spline path is C^2 continuous

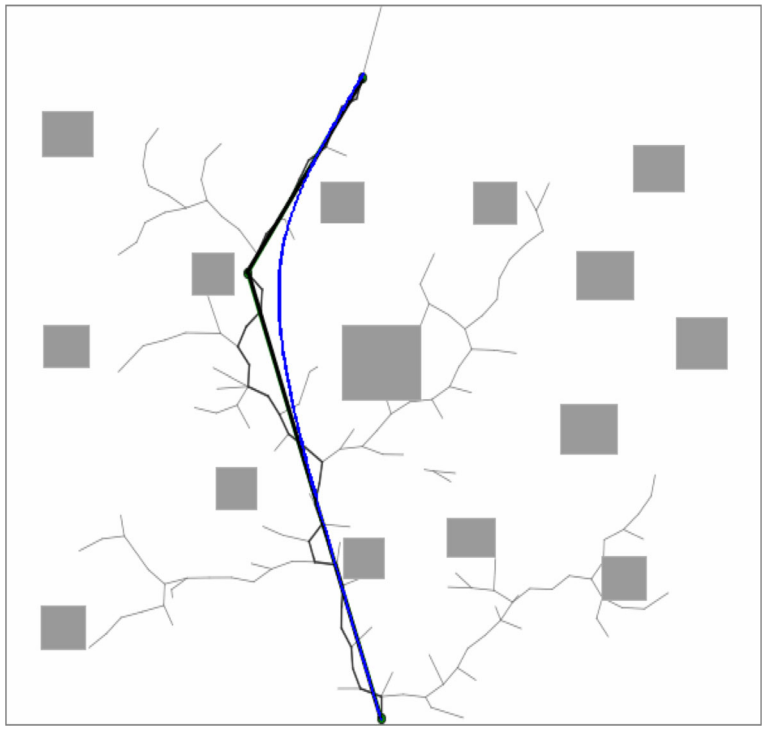

Fig. 44 RRT (2): Piecewise linear and B-spline path in an obstacle cluttered environment

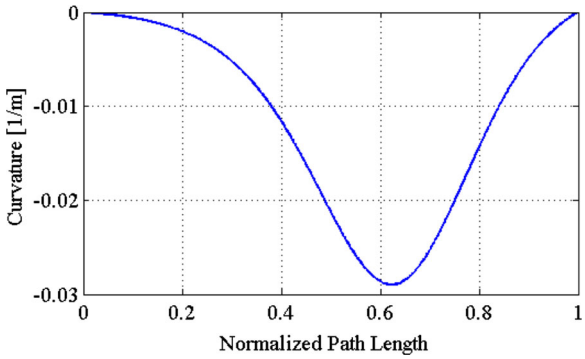

Fig. 45 RRT (2): Continuous curvature obeying the maximum curvature constraint of $0.3m^{-1}$

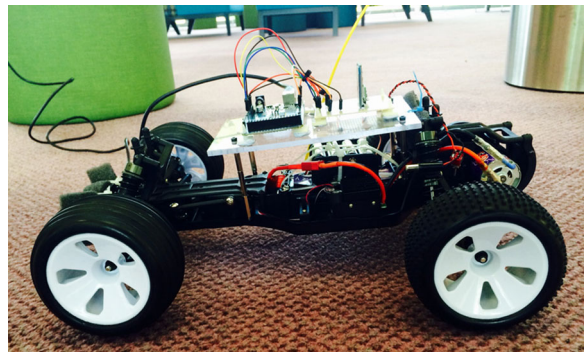

Fig. 47 Experimental scaled car-like robot

efficiency in solving planning problems, their solutions are widely regarded as suboptimal and include redundant motions. This can be attributed to stochastic sampling schemes used for configuration space exploration [10, 78].

Suboptimal paths generated by the RRT planners presented a challenge to post processing 2algorithms, especially in narrow passages and cluttered environments [10, 62, 78]. Therefore we selected a similar narrow passage and cluttered environment, as examples, to highlight the advantages of the proposed method. We refer to two scenarios as RRT (1) and (2).

The first example was selected to combine redundant RRT turns, with the difficulty of smoothing,

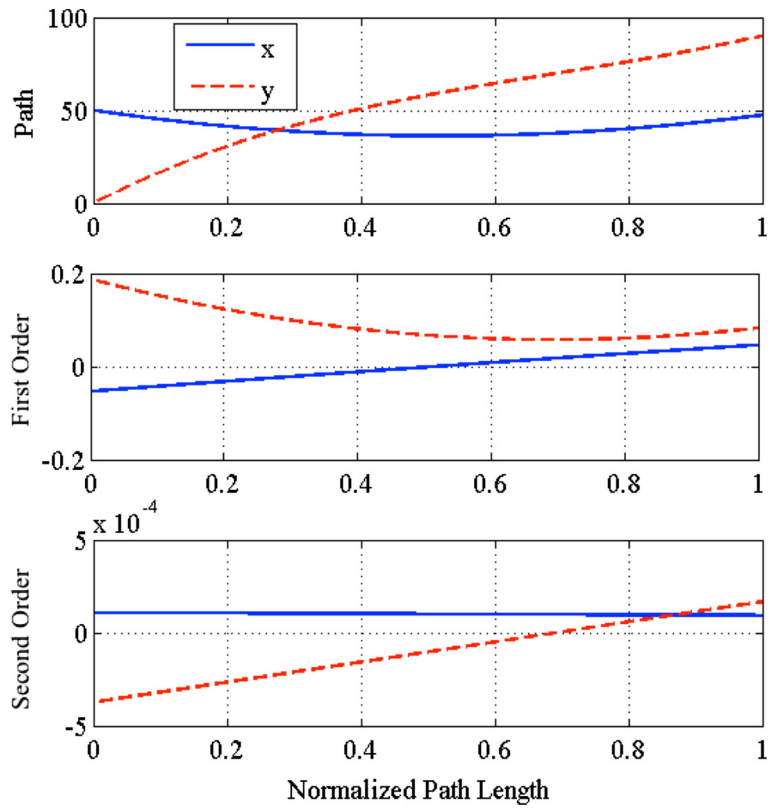

Fig. 46 RRT (2): B-spline path is C^2 continuous

Table 4 Experimental robot parameters

Parameter	Value
Maximum steering angle (δ)	30°
Minimum turning radius (ρ)	0.33 m
Maximum curvature (k)	3 m^{-1}
Wheel base (w)	0.19 m

within a narrow passage. The resulting path contained subsequent sharp turns in the vicinity of obstacles, which required smoothing and curvature bounding maneuvers. The second example showed the advantage of closely following the original linear path using midpoint insertion. The randomly allocated obstacles pose a challenge to the smoothing algorithm, as any large deviation from the path could lead to collision.

In both cases curvature limits were set to 0.3 m^{-1}. RRT paths are shown as a black solid line and obstacles are grey colored boxes, see Figs. 40 and 44 for illustration. Resulting B-spline paths are shown in Figs. 41 and 44. The resulting curvature, in both cases, satisfied the maximum constraint as illustrated in Figs. 42 and 45. In the narrow passage RRT (1) scenario, three additional segments were needed as a result of the sharp RRT path. On the other hand, RRT (2) required no additional segments and the resulting curvature is well below its bounds. This highlights an important feature of the proposed method. It did not force the vehicle to reach the curvature limit and led to improved smoothing. In Figs. 43 and 46 the resulting B-spline paths were also shown to be C^2 continuous. The average algorithm runtime was *80 milliseconds* for both examples. Due to its recursive nature, Cox-De Boor algorithm was the bottleneck of the path synthesis procedure. All experiments were implemented in Matlab® on an i7 2.8GHz machine with 8GB of memory running OSX®.

5.3 Human Steering Experiment

In this experiment we used a human operated, scaled car-like robot, to highlight the resemblance between our approach and natural human steering. The robot, pictured in Fig. 47, was used in the presented experiments. Vehicle's parameters are given in Table 4. It was remote-controlled by a human operator, who was required to steer the robot through four different maneuvers as illustrated in Fig. 48. Steering commands sent to the robot were logged by an on-board

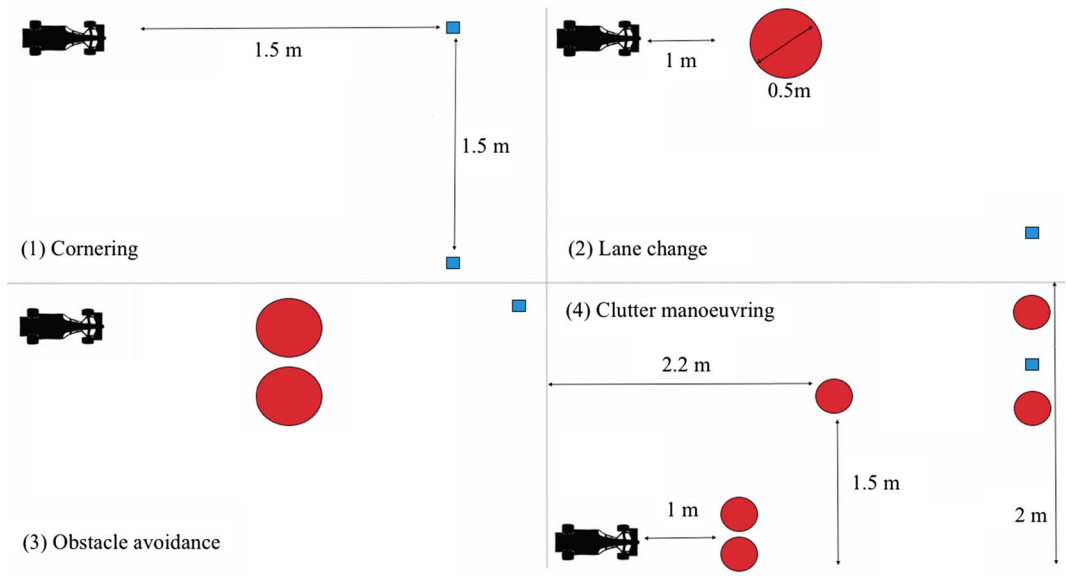

Fig. 48 Setup used for different scenarios. In each case the circular objects are cones with 0.5 m base diameter and the rectangles are waypoints specified for the operator to reach

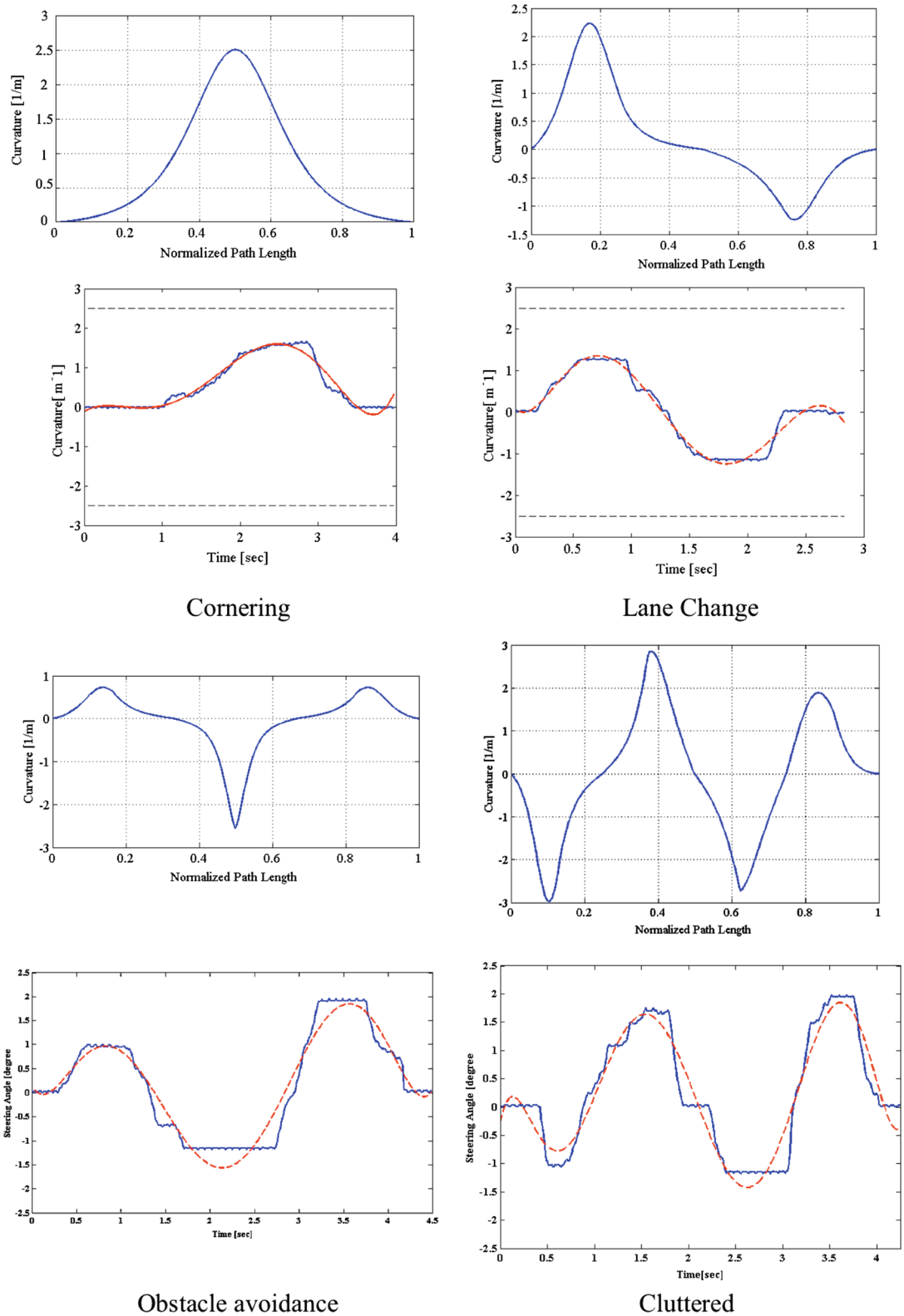

Fig. 49 Comparing human steering and simulated B-spline curvatures

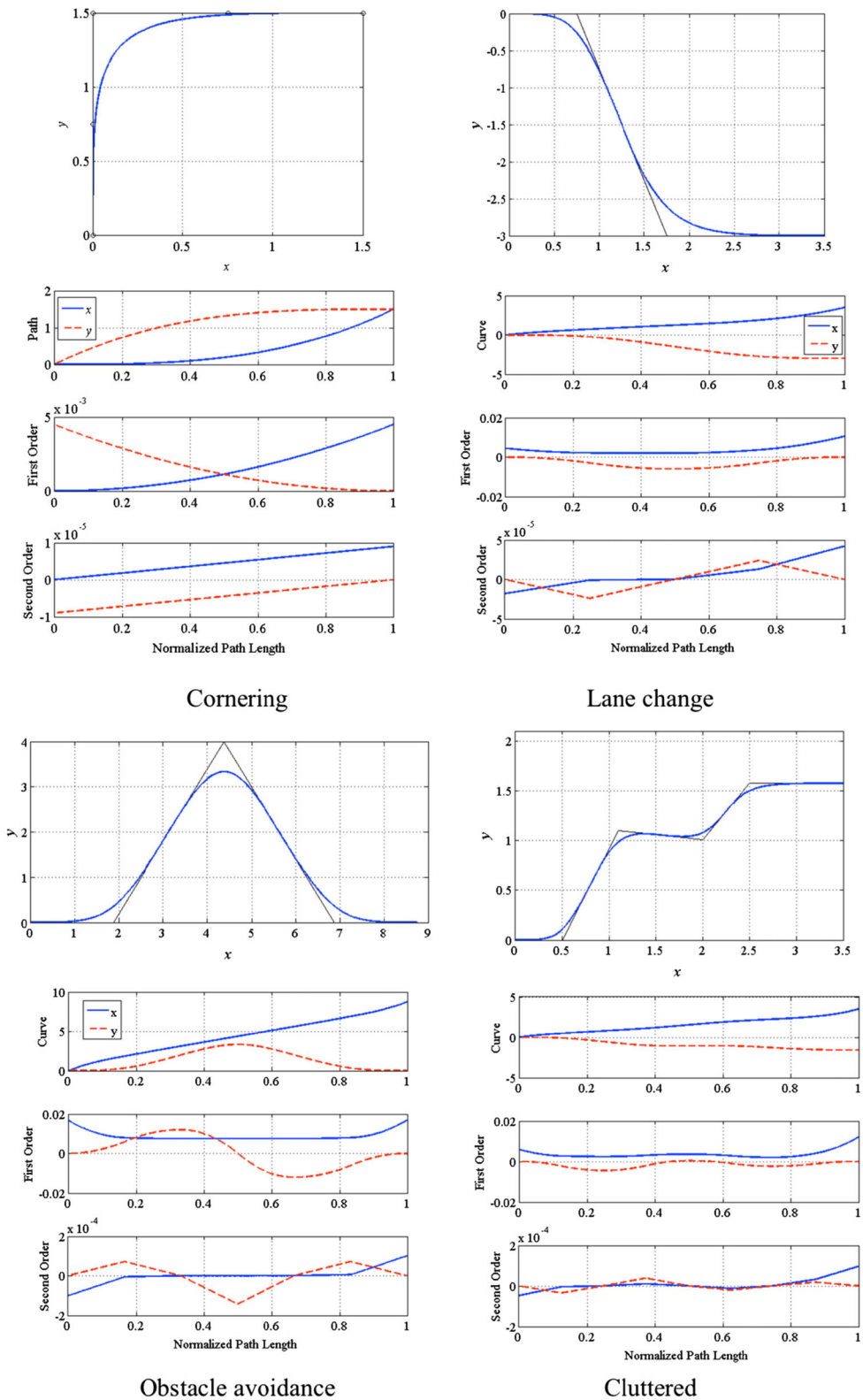

Fig. 50 Proposed B-spline paths and their corresponding C^2 continuous trajectories

microcontroller and then serially transferred to a host computer to be post processed and compared to the corresponding paths generating using the proposed method.

The first experiment (cornering) was a simple right angle turn. This segment resembled the repeating path segment, illustrated in Fig. 13, which was used in Algorithm 1 for path smoothing. It was essential to show that, our method was capable of mimicking human operation, in that simple segment, as it is repeating frequently through any path. We followed this situation with three examples of increasing difficulty and obstacle numbers.

The second example was a lane change maneuver. The width of the lane change was 3m, with a single obstacle placed at a distance of 1 m in front the vehicle, at the start. The path required the operator to generate two successive turns with alternating turning directions.

The third example (obstacle avoidance) involved a similar setup, with an additional obstacle, and a more challenging scenario of avoiding the obstacles and then returning to the same lane. The third path required three consecutive and alternating turns. The first turn was particularly challenging as the width of the obstacles was increasing, by adding a second cone, that needed a sharper turn, followed by a subsequent sharp turn to return to the original, lateral position. Finally, an unstructured example was included, referred to as cluttered maneuvering. It was required to navigate through obstacles and reach the waypoint placed between two of them.

The resulting curvatures, from the proposed algorithm and human operator commands, are compared in Fig. 49. The raw signals from the human operator are shown as solid blue lines and the best-fit curve is shown as a dotted red line. It is shown that the proposed method is capable of mimicking human steering in different experimental setups. The simulated B-spline paths and resulting C^2 continuous trajectories are shown in Fig. 50.

6 Conclusions and Future Work

This paper presents, analytical formulation and algorithm for robotic path smoothing using continuous B-splines. The method was validated using a wide range of benchmarks, numerical and real examples. We proposed an algorithm that guaranteed C^2 continuity and maximum curvature bounding through the entire path. Investigation was based on the use of a single clamped cubic B-spline curve, which was shown to be more suited for planning than other curves implementation.

Numerical experiments showed that our approach outperforms recently developed robotic path smoothing algorithms presented in available reviewed literature. Comparing to [56, 60], we were capable of generating shorter paths, with smoother and controlled curvature. Even when an additional segment was required, for curvature bounding, the path length was shorter than the originally planned linear path.

The advantage of C^2 continuity was evident when comparing with geometrically continuous paths using [57, 58]. Continuous trajectory eliminated any abrupt changes that resulted from the process of different curves combination. It also led to minimization of the resulting accelerations. Smoothing algorithm was presented as a solution for randomized planning algorithms such as RRT, for car-like robots, in both narrow and cluttered environments. Experimental results showed that, B-spline curvature mimic continuous human steering, with smoother curvature values, in different environments, with increasing complexity.

The promising findings, presented here, provide a basis for further investigations. An alternative to using a pre-designed maneuver for maximum curvature condition is in providing an analytical solution. At this stage, planning and smoothing are decoupled and in many cases replanning may be required to generate a feasible trajectory [55]. An efficient approach is planning with B-spline curvature feasibility considerations. We plan to integrate our smoothing algorithm within a sampling-based planning algorithm to present efficient kinodynamic planners. Further laboratory experiments are needed to gain understanding of human steering and effects of C^2 continuity in combination with different tracking algorithms on the robot performance. We will conduct full scale steering experiments for different vehicles on a variety of paths. This will uncover the effects of the operator's perception of the path, human response and

the vehicle constraint's on the resulting curvature. Additional experiments to highlight the effects of C^2 continuity on localization errors, passenger comfort, mechanical wear and overactuation will be conducted on different robotic platforms.

Acknowledgments The authors would like to thank the anonymous reviewers for their invaluable feedback that improved the quality of this article. Mohamed Elbanhawi acknowledges the financial support of the Australian Postgraduate Award (APA) and the Research Training Scheme (RTS).

Appendix: A

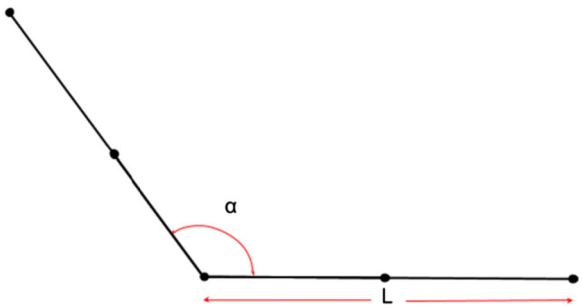

Fig. 51 Segment parameters; Length (L) and angle ($\propto$)

The curve has n =5 control points. It is a cubic curve; $p = 3$

Therefore a knot vector, with m = 9 knots, and quadruple initial and final multiplicities are needed:

$$\hat{u} = [0, 0, 0, 0, 0.5, 1, 1, 1, 1] \tag{A1}$$

We implement the Cox-deBoor algorithm by substituting in Eq. 4 and then recursively updating (5). We get the following 3rd degree basis function: (A2.1–A2.5)

$$N_{0,3} = \frac{(1-2U)^3}{2} \tag{A2.1}$$

$$N_{1,3} = 6u^3 - 6u^2 + 1 \tag{A2.2}$$

$$N_{2,3} = 6u^2 - 6u - 1 \tag{A2.3}$$

$$N_{3,3} = 6u^3 + 12u^2 6u + 1 \tag{A2.4}$$

$$N_{4,3} = \frac{(2u-1)^3}{2} \tag{A2.5}$$

The x and y coordinates for the control points are given by Eqs. A3.1 and A3.2.

$$P_x = \left[L, \frac{L}{2}, 0, \frac{L}{2}\cos\propto, L\cos\propto\right] \tag{A3.1}$$

$$P_y = \left[0, 0, 0, \frac{L}{2}\sin\propto L\sin\propto\right] \tag{A3.2}$$

The summation of the product (A2) by Eq. A3.1 as defined in Eq. A4.1 results in Eq. A4.2.

$$x(u) = \sum_{i=0}^{n} N_{i,p} P_i \tag{A4.1}$$

$$\begin{aligned} x(u) = {} & \frac{(1-2u)^3}{2} * L + (6u^3 - 6u^2 + 1) * \frac{L}{2} \\ & + (-6u^3 + 12u^2 - 6u + 1) * \frac{L\cos(\alpha)}{2} \\ & + \frac{(2u-1)^3}{2} * L\cos(\alpha) \end{aligned} \tag{A4.2}$$

The first and second order derivatives with respect to u (segment parameters L and $\propto$ are constants for a specific segment) are given in Eqs. A5 and A6

$$x'(u) = 3L(u^2(\cos(\alpha) - 1) + 2u + 1) \tag{A5}$$

$$x'(u) = 6L(u(\cos(\alpha) - 1)+) \tag{A6}$$

Similarly for $y(u)$, by substituting (A3.2) in Eq. A7.1 we get (A7.2) and the derivatives (A8) and (A9)

$$y(u) = \sum_{i=0}^{n} N_{i,p} P_i \tag{A7.1}$$

$$\begin{aligned} y(u) = {} & (-6u^3 + 12u^2 - 6u + 1) * \frac{L\sin(\alpha)}{2} \\ & + \frac{(2u-1)^3}{2} * L\sin(\alpha) \end{aligned} \tag{A7.2}$$

$$y'(u) = 3L\sin(\alpha)u^2 \tag{A8}$$

$$y'(u) = 6L\sin(\alpha)u \tag{A9}$$

The curvature $k(u)$ can be defined as a function of segment parameters by substituting Eqs. A5, A6, A8 and A9 in the curvature (A11.1) to obtain (A11.2)

$$k(u) = \frac{x'(u)y''(u) - x''(u)y'(u)}{(x'(u)^2 + y'(u)^2)^{3/2}} \tag{A11.1}$$

$$k(u) = \frac{2(u\sin(\alpha))(1-u)}{3L(2u^2(1-\cos(\alpha))(u^2-2u+1) + (2u-1)^2)^{3/2}} \tag{A11.2}$$

References

1. Reif, J.H.: Complexity of the mover's problem and generalizations. In: Foundations of Computer Science, 1979., 20th Annual Symposium on, pp. 421–427 (1979)
2. Latombe, J.-C.: Motion Planning: A journey of robots, molecules, digital actors, and other artifacts. Int. J. Robot. Res. **18**(11), 1119–1128 (1999). doi:10.1177/02783649922067753
3. Choset, H.M.: Principles of Robot Motion, Theory, Algorithms, and Implementation. Prentice Hall of India (2005)
4. Brooks, R.A., Lozano-Perez, T.: A subdivision algorithm in configuration space for findpath with rotation. IEEE Trans. Syst. Man Cybern. **15**(2), 224–233 (1985). doi:10.1109/TSMC.1985.6313352
5. Canny, J.: A Voronoi method for the piano-movers problem. In: Robotics and Automation. Proceedings. 1985 IEEE International Conference on, pp. 530–535 (1985)
6. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**(1), 90–98 (1986). doi:10.1177/027836498600500106
7. Arkin, R.C.: Motor Schema Based Mobile Robot Navigation. Int. J. Robot. Res. **8**(4), 92–112 (1989). doi:10.1177/027836498900800406
8. Koren, Y., Borenstein, J.: Potential field methods and their inherent limitations for mobile robot navigation. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, vol. 1392, pp. 1398–1404 (1991)
9. Elbanhawi, M., Simic, M.: Sampling-based robot motion planning: a review. IEEE Access **2**, 56–77 (2014). doi:10.1109/ACCESS.2014.2302442
10. Geraerts, R., Overmars, M.H.: Creating high-quality paths for motion planning. Int. J. Robot. Res. **26**(8), 845–863 (2007). doi:10.1177/0278364907079280
11. Laumond, J.P., Sekhavat, S., Lamiraux, F.: Guidelines in nonholonomic motion planning for mobile robots. In: J.P. Laumond (ed.) Robot Motion Planning and Control, vol. 229. Lecture Notes in Control and Information Sciences, pp. 1–53. Springer Berlin Heidelberg (1998)
12. Cheng, P.: Sampling-based motion planning with differential constraints. Ph.D. University of Illinois at Urbana-Champaign (2005)
13. Wallace, R., Stentz, A., Thorpe, C., Moravec, H., Whittaker, W., Kanade, T.: First Results in Robot Road Following. In: 1985, pp. 381–387
14. Antonelli, G., Chiaverini, S., Fusco, G.: A fuzzy-logic-based approach for mobile robot path tracking. IEEE Trans. Fuzzy Syst. **15**(2), 211–221 (2007)
15. Perez, J., Milanes, V., Onieva, E.: Cascade architecture for lateral control in autonomous vehicles. IEEE Intell. Transp. Syst. **12**(1), 73–82 (2011). doi:10.1109/TITS.2010.2060722
16. Jazar, R.N.: Mathematical theory of autodriver for autonomous vehicles. J. Vib. Control. **16**(2), 253–279 (2010)
17. Marzbani, H., Ahmad Salahuddin, M.H., Simic, M., Fard, M., Jazar, R.N.: Steady-state dynamic steering. In: Frontiers in Artificial Intelligence and Applications, vol. 262 (2014)
18. Cheein, F.A., Scaglia, G.: Trajectory Tracking Controller Design for Unmanned Vehicles: A New Methodology. Journal of Field Robotics, n/a-n/a. doi:10.1002/rob.21492 (2013)
19. Magid, E., Keren, D., Rivlin, E., Yavneh, I.: Spline-Based Robot Navigation. In: Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on, pp. 2296–2301 (2006)
20. Roth, S., Batavia, P.: Evaluating Path Tracker Performance for Outdoor Mobile Robots. Paper presented at the Automation Technology for Off-Road Equipment, Chicago, Illinois, USA, 26–27/07
21. Lau, B., Sprunk, C., Burgard, W.: Kinodynamic motion planning for mobile robots using splines. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, pp. 2427–2433 (2009)
22. Gulati, S., Kuipers, B.: High performance control for graceful motion of an intelligent wheelchair. In: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pp. 3932–3938 (2008)
23. Berglund, T., Brodnik, A., Jonsson, H., Staffanson, M., Soderkvist, I.: Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles. IEEE Trans. Autom. Sci. Eng. **7**(1), 167–172 (2010). doi:10.1109/TASE.2009.2015886
24. Maekawa, T., Noda, T., Tamura, S., Ozaki, T., Machida, K.-i.: Curvature continuous path generation for autonomous vehicle using B-spline curves. Comput. Aided Des. **42**(4), 350–359 (2010). doi:10.1016/j.cad.2009.12.007
25. Sabelhaus, D., Röben, F., Meyer zu Helligen, L.P., Schulze Lammers, P.: Using continuous-curvature paths to generate feasible headland turn manoeuvres. Biosyst. Eng. **116**(4), 399–409 (2013). doi:10.1016/j.biosystemseng.2013.08.012
26. Girbés, V., Armesto, L., Tornero, J.: Path following hybrid control for vehicle stability applied to industrial forklifts. Robot. Auton. Syst. **0** (2014). doi:10.1016/j.robot.2014.01.004
27. Xuan-Nam, B., Boissonnat, J.-d., Soueres, P., Laumond, J.P.: Shortest path synthesis for Dubins non-holonomic robot. In: Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on, 8–13 1994, vol. 1, pp. 2–7
28. Anderson, E.P., Beard, R.W., McLain, T.W.: Real-time dynamic trajectory smoothing for unmanned air vehicles.

IEEE Trans. Control Syst. Technol. **13**(3), 471–477 (2005). doi:10.1109/TCST.2004.839555
29. Myung, H., Kuffner, J., Kanade, T.: Efficient Two-phase 3D Motion Planning for Small Fixed-wing UAVs. In: Robotics and Automation, 2007 IEEE International Conference on, pp. 1035–1041 (2007)
30. LaValle, S.: Planning Algorithms. Cambridge University Press (2006)
31. Suzuki, Y., Kagami, S., Kuffner, J.J.: Path Planning with Steering Sets for Car-Like Robots and Finding an Effective Set. In: Robotics and Biomimetics, 2006. ROBIO '06. IEEE International Conference on, pp. 1221–1226 (2006)
32. Pivtoraiko, M., Knepper, R.A., Kelly, A.: Differentially constrained mobile robot motion planning in state lattices. J. Field Robot. **26**(3), 308–333 (2009)
33. Fraichard, T., Scheuer, A.: From Reeds and Shepp's to continuous-curvature paths. Robot. IEEE Trans. **20**(6), 1025–1035 (2004). doi:10.1109/TRO.2004.833789
34. Wang, L.Z., Miura, K.T., Nakamae, E., Yamamoto, T., Wang, T.J.: An approximation approach of the clothoid curve defined in the interval [0, $\pi/2$] and its offset by free-form curves. Comput. Aided Des. **33**(14), 1049–1058 (2001). doi:10.1016/S0010-4485(00)00142-1
35. Meek, D.S., Walton, D.J.: An arc spline approximation to a clothoid. J. Comput. Appl. Math. **170**(1), 59–77 (2004). doi:10.1016/j.cam.2003.12.038
36. Montes, N., Herraez, A., Armesto, L., Tornero, J.: Real-time clothoid approximation by Rational Bezier curves. In: In: Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pp. 2246–2251 (2008)
37. McCrae, J., Singh, K.: Sketching piecewise clothoid curves. Comput. Graph. **33**(4), 452–461 (2009). doi:10.1016/j.cag.2009.05.006
38. Brezak, M., Petrovic, I.: Real-time approximation of clothoids with bounded error for path planning applications. IEEE Trans. Robot. **PP**(99), 1–9 (2013). doi:10.1109/TRO.2013.2283928
39. Farin, G.: From conics to NURBS: A tutorial and survey. IEEE Comput. Graph. Appl. **12**(5), 78–86 (1992). doi:10.1109/38.156017
40. Piegl, L.: On NURBS: a survey. IEEE Comput. Graph. Appl. **11**(1), 55–71 (1991). doi:10.1109/38.67702
41. Farin, G.: Curves and Surfaces for CAGD. Computing. Morgan Kaufmann (2002)
42. Lepetič, M., Klančar, G., Škrjanc, I., Matko, D., Potočnik, B.: Time optimal path planning considering acceleration limits. Robot. Auton. Syst. **45**(3–4), 199–210 (2003). doi:10.1016/j.robot.2003.09.007
43. Jolly, K.G., Sreerama Kumar, R., Vijayakumar, R.: A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. Robot. Auton. Syst. **57**(1), 23–33 (2009). doi:10.1016/j.robot.2008.03.009
44. Schoenberg, I.J.: Contributions to the problem of approximation of equidistant data by analytic functions - b. on the problem of osculatory interpolation - a 2nd class of approximation formulae. Q. Appl. Math. **4**(2), 112–141 (1946)
45. Thompson, S.E., Patel, R.V.: Formulation of joint trajectories for industrial robots using b-splines. IEEE Trans. Ind. Electron. **34**(2), 192–199 (1987). doi:10.1109/TIE.1987.350954
46. Dyllong, E., Visioli, A.: Planning and real-time modifications of a trajectory using spline techniques. Robotica **21**(05), 475–482 (2003). doi:10.1017/S0263574703005009
47. Hodgins, J.K., O'Brien, J.F., Tumblin, J.: Perception of human motion with different geometric models. IEEE Trans. Vis. Comput. Graph. **4**(4), 307–316 (1998). doi:10.1109/2945.765325
48. Schmid, A.J., Woern, H.: Path planning for a humanoid using NURBS curves. In: Automation Science and Engineering, 2005. IEEE international conference on, pp. 351–356 (2005)
49. Sungchul, J., Taehoon, K.: Tool-path generation for NURBS surface machining. In: American control conference, 2003. Proceedings of the 200, vol. 2613, pp. 2614–2619 (2003)
50. Cheng, M.Y., Kuo, J.C.: Real-time NURBS command generators for CNC servo controllers. Int. J. Mach. Tools Manuf. **42**(7), 801–813 (2002). doi:10.1016/S0890-6955(02)00015-9
51. Zhang, Y., Bazilevs, Y., Goswami, S., Bajaj, C.L., Hughes, T.J.R.: Patient-specific vascular NURBS modeling for isogeometric analysis of blood flow. Comput. Methods Appl. Mech. Eng. **196**(29–30), 2943–2959 (2007). doi:10.1016/j.cma.2007.02.009
52. Ma, W., Kruth, J.P.: NURBS curve and surface fitting for reverse engineering. Int. J. Adv. Manuf. Technol. **14**(12), 918–927 (1998). doi:10.1007/BF01179082
53. Piegl, L.A., Tiller, W.: Parametrization for surface fitting in reverse engineering. Comput. Aided Des. **33**(8), 593–603 (2001). doi:10.1016/S0010-4485(00)00103-2
54. Hughes, T.J.R., Reali, A., Sangalli, G.: Duality and unified analysis of discrete approximations in structural dynamics and wave propagation: Comparison of p-method finite elements with k-method NURBS. Comput. Methods Appl. Mech. Eng. **197**(49–50), 4104–4124 (2008). doi:10.1016/j.cma.2008.04.006
55. Koyuncu, E., Inalhan, G.: A probabilistic B-spline motion planning algorithm for unmanned helicopters flying in dense 3D environments. In: Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on, pp. 815–821 (2008)
56. Zhou, F., Song, B., Tian, G.: Bézier curve based smooth path planning for mobile robot. J. Inf. Comput. Sci. **8**(12), 2441–2450 (2011)
57. Kwangjin, Y., Sukkarieh, S.: An analytical continuous-curvature path-smoothing algorithm. Robot. IEEE Trans. **26**(3), 561–568 (2010). doi:10.1109/TRO.2010.2042990
58. Kwangjin, Y., Jung, D., Sukkarieh, S.: Continuous curvature path-smoothing algorithm using cubic Bezier spiral curves for non-holonomic robots. Adv. Robot. **27**(4), 247–258 (2013). doi:10.1080/01691864.2013.755246
59. Walton, D.J., Meek, D.S., Ali, J.M.: Planar G2 transition curves composed of cubic Bézier spiral segments. J. Comput. Appl. Math. **157**(2), 453–476 (2003). doi:10.1016/s0377-0427(03)00435-7
60. Huh, U.-Y., Chang, S.-R.: A G^2 continuous path-smoothing algorithm using modified quadratic polynomial interpolation. Int. J. Adv. Robot. Syst. **25**(11) (2014). doi:10.5772/57340

61. Piazzi, A., Bianco, C.G.L., Romano, M.: η3-Splines for the smooth path generation of wheeled mobile robots. robotics-splines for the smooth path generation of wheeled mobile robots. Robot. IEEE Trans. **23**(5), 1089–1095 (2007). doi:10.1109/TRO.2007.903816
62. Pan, J., Zhang, L., Manocha, D.: Collision-free and smooth trajectory computation in cluttered environments. Int. J. Robot. Res. **31**(10), 1155–1175 (2012). doi:10.1177/0278364912453186
63. Nikolos, I.K., Valavanis, K.P., Tsourveloudis, N.C., Kostaras, A.N.: Evolutionary algorithm based offline/online path planner for UAV navigation. Systems, Man, and Cybernetics, Part B. Cybern. IEEE Trans. **33**(6), 898–912 (2003). doi:10.1109/TSMCB.2002.804370
64. Guarino Lo Bianco, C.: Minimum-jerk velocity planning for mobile robot applications. Robot. IEEE Trans. **29**(5), 1317–1326 (2013). doi:10.1109/TRO.2013.2262744
65. Kunz, T., Stilman, M.: Time-optimal trajectory generation for path following with bounded acceleration and velocity. Robotics: Science and Systems, p. 209 (2013)
66. Velenis, E., Tsiotras, P.: Minimum-time travel for a vehicle with acceleration limits: theoretical analysis and receding-horizon implementation. J. Optim. Theory Appl. **138**(2), 275–296 (2008). doi:10.1007/s10957-008-9381-7
67. Johnson, J., Hauser, K.: Optimal acceleration-bounded trajectory planning in dynamic environments along a specified path. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on, pp. 2035–2041 (2012)
68. Elbanhawi, M., Simic, M., Jazar, R.: Continuous-curvature bounded trajectory planning using parametric splines. In: Frontiers in Artificial Intelligence and Applications, vol. 262, pp. 513–522 (2014)
69. Kelly, A., Stentz, A.: Rough terrain autonomous mobility—part 1: A theoretical analysis of requirements. Auton. Robot. **2**(5), 129–161 (1998). doi:10.1023/A:1008801421636
70. Ahmed, F., Deb, K.: Multi-objective path planning using spline representation. In: Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on, pp. 1047–1052 (2011)
71. De Boor, C.: On calculating with B-splines. J. Appro. Theory **6**(1), 50–62 (1972)
72. Barsky, B.A., Derose, T.D.: Geometric continuity of parametric curves: constructions of geometrically continuous splines. IEEE Comput. Graph. Appl. **10**(1), 60–68 (1990). doi:10.1109/38.45811
73. Jan, G.E., Sun, C.C., Tsai, W.C., Lin, T.H.: An O(n log n) Shortest Path Algorithm Based on Delaunay Triangulation, Mechatronics. IEEE/ASME Trans. **PP**(99), 1–7 (2013). doi:10.1109/TMECH.2013.2252076
74. Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., Thrun, S.: Anytime search in dynamic graphs. Artif. Intell. **172**(14), 1613–1643 (2008). doi:10.1016/j.artint.2007.11.009
75. Bruce, J.R., Veloso, M.M.: Safe multirobot navigation within dynamics constraints. IEEE Proc. **94**(7), 1398–1411 (2006). doi:10.1109/JPROC.2006.876915
76. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006)
77. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. In. Iowa state university (1998)
78. Wein, R., van den Berg, J., Halperin, D.: Planning high-quality paths and corridors amidst obstacles. Int. J. Robot. Res. **27**(11-12), 1213–1231 (2008). doi:10.1177/0278364908097213