Robotics and Computer–Integrated Manufacturing 72 (2021) 102203

Contents lists available at ScienceDirect

Robotics and Computer-Integrated Manufacturing

journal homepage: www.elsevier.com/locate/rcm

Full length article

η^{3D}-splines for the generation of 3D Cartesian paths with third order geometric continuity[☆]

Andrea Tagliavini, Corrado Guarino Lo Bianco [*]

Dip. di Ingegneria e Architettura, University of Parma, Italy

ARTICLE INFO

Keywords:
Path planning primitive
Third-order geometric continuity
Continuous jerk
Real-time planner
Industrial manipulators

ABSTRACT

The paper proposes a new planning primitive, named η^{3D}-splines, based on 7th order polynomials, which is suited to generate three-dimensional paths characterized by third-order geometric continuity. The third order continuity represents an important property, since it allows continuous-jerk reference signals for the joints actuators of robotic systems. Differently from other approaches in the literature, η^{3D}-splines are efficiently evaluated by means of closed form expressions as function of the assigned interpolation conditions. This allows an intuitive real-time generation of composite paths: from the knowledge of the geometric characteristics of the curve which is currently executed, and by choosing a novel end-point together with the desired interpolating conditions, a new path can be efficiently generated by simultaneously maintaining the overall third order geometric continuity. Additionally, the η^{3D}-splines can be shaped by acting on a set of six free parameters, so as to emulate other planning primitives, like, for example, linear segments, circular arcs, clothoids, helical curves, and conic spirals. Furthermore, by means of the same parameters, all possible 7th order polynomials, which fulfill the given interpolating conditions, can be generated. The accompanying video shows an anthropomorphic manipulator executing a composite trajectory generated by means of the η^{3D}-splines.

1. Introduction

Cartesian paths for robotic systems must be planned by accounting for their continuity properties. Indeed, non-smooth primitives worsen the controllers performances and simultaneously stress the system mechanics. The path geometric continuity problem is widely discussed in the literature and it is managed through apposite planners. Early works were focused on the generation of planar curves mainly conceived for autonomous mobile robots. The emphasis was initially posed on the generation of minimum length paths between assigned points [1–4]. The problem has been continuously revised along the years by also considering alternative cost indexes [5]. The resulting composite routes, obtained by joining linear segments and circular arcs, were generated so as to guarantee the continuity of the path tangent. In the same years, other authors [6–11] introduced the concept of second order geometric continuity, thus including the path smoothness among the criteria to be considered: not only the path tangent, but also its curvature must be continuous. The second order continuity was achieved by means of apposite primitives like clothoids, polar curves or cubic spirals. Later, to the same purpose, other flexible path primitives appeared in the literature. For example, the Bezier curves adopted in [12–14], the B-splines proposed in [15], and, finally, the η^2-splines devised in [16].

The common denominator of the papers just cited is that smoothness reasons motivate the adoption of continuous curvature paths. However, in [17] it was shown that the smooth control of unicycle vehicles actually requires an additional continuity level: the generation of continuous control signals necessarily imposes paths whose curvature-derivative is continuous, so that a novel path planning primitive, named η^3-splines [18,19], was developed, later followed by η^4-splines [20].

In more recent years, the smooth path planning problem has been extended to three-dimensional (3D) Cartesian curves. The fields of application of 3D trajectories are substantially two: the generation of collision-free routes in cluttered environments [21,22] and the generation of paths with specific geometric characteristics, like the ones required, for example, by Computer Numerical Control (CNC) machines [23–25] or for arc welding and laser cutting applications.

In the first case, the commonly adopted solution is based on the generation of collision-free, discrete, Cartesian paths, whose points are chosen by means of algorithms – like, for example, Rapidly-exploring Random Trees (RRT or RRT*) – which optimize a proper cost index.

☆ This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

* Corresponding author.
E-mail address: guarino@ce.unipr.it (C. Guarino Lo Bianco).

https://doi.org/10.1016/j.rcim.2021.102203
Received 13 February 2021; Received in revised form 23 April 2021; Accepted 9 June 2021
Available online 22 June 2021

The desired degree of smoothness is typically achieved by means of spline trajectories, which interpolate the given points directly in the Cartesian space or, alternatively, in the configuration space. For such class of problems, the actual shape of the Cartesian path is not rigidly imposed in advance [26–29].

A similar solution can also be adopted for the management of the second class of problems. In case of applications which allow relatively high computational times, a path with the prescribed geometry can be generated by means of a dense sequence of pass-through points [30–32]. In all the other cases, an actual Cartesian path, with the required geometric continuity, must be planned. For a long time, linear segments and circular arcs were the sole path primitives considered in robotic contexts. As earlier pointed out, they only allow continuous-tangent paths. In advanced applications, like the ones involving CNC machines, such limit cannot be admitted, so that, recently, many works concerning the generation of smooth curves have appeared in the literature: Bezier curves are used in [33,34] for the synthesis of continuous-curvature paths, while B-splines are adopted in [35] for a jerk bounded application requiring continuous curvature derivatives.

The path planning primitive proposed in this work, named η^{3D}-splines and based on 7th order polynomials, is conceived for applications belonging to the second class of problems. They guarantee the same smoothness level of [35], but are conceived for more general application contexts. In particular, while in CNC applications the emphasis is primarily posed on the creation of junction curves which smoothly join a sequence of linear segments, for a general purpose robotic application the target is the generation of flexible curves able to connect generic primitives by guaranteeing the required continuity level. Indeed, if junction curves are not properly designed, reference signals for joint velocities, accelerations, and jerks may be discontinuous, thus worsening the systems performances. Continuity problems can be handled by stopping the manipulator at the end of each segment of the trajectory, but such solution is clearly inefficient. Conversely, η^{3D}-splines, by guaranteeing the required geometric continuity of the path, allow C^3 joint reference signals, so that composite trajectories can be executed with no stops.

The strengths of η^{3D}-splines, if compared with other planning primitives, are essentially two: the planning method, which allows an efficient online evaluation of the curve coefficients directly from the interpolating conditions, and their straightforward emulation capabilities.

Concerning the first point, the third-order geometric continuity can be obtained by assigning, at the start and at the end points of the curve, the desired Frenet-frames, curvatures, curvature-derivatives and torsions. Such interpolating conditions directly appear in the closed form expressions devised for the efficient computation of the spline coefficients: the same continuity level achieved in [35] can be reached at a negligible computational cost. This allows one to easily manage situations in which the Cartesian path must be re-planned on-the-fly during the movement – see for example [36,37] – by simultaneously maintaining the overall geometric continuity. The planning method is more immediate and intuitive than the ones adopted for the synthesis of B-spline or Bezier curves, which require a proper choice of the control points in order to satisfy the assigned continuity properties and to model the curves shape.

As anticipated, η^{3D}-splines can also emulate other common primitives. More precisely, they exactly generate linear segments, while they can emulate, with good approximation, curves like circular arcs, clothoids, helical curves, and conic spirals. Additionally, any 7th order polynomial curve, for example a Bezier curve, satisfying the given interpolating conditions, can be replaced by an analogous η^{3D}-spline with the same coefficients and viceversa [38]. Such emulation capabilities are carried out through a set of six independent parameters which can be used to finely shape the curve.

The emulation capabilities of η^{3D}-splines have been exploited for the implementation of a Cartesian planner entirely based on such primitive, and suited for applications in which the path must be generated on-the-fly. Such planner is able to generate, in a simple and intuitive way, complex composite paths with third-order geometric continuity.

The paper is organized as follows. Section 2 shows that jerk continuous reference signals for robotic manipulators can be obtained by planning paths with third order geometric continuity. The same Section further introduces some preliminary considerations concerning 3D curves and recalls the definitions of geometric continuity. Section 3 proposes the closed-form equations which are used for the evaluation of the η^{3D}-splines coefficients. In the same section, two important properties of the novel primitive are enunciated. In Section 4, a simple method is proposed for the selection of the shaping parameters, and the emulation capabilities of the novel path primitive are discussed. In Section 5, the η^{3D}-splines are experimentally tested by generating a composite 3D path for an industrial manipulator. Final conclusions are drawn in Section 6, while Appendix A demonstrates two propositions given in Section 3. The paper is accompanied by a video attachment concerning the experimental test.

2. Preliminary considerations on the geometric continuity of 3D curves

In robotic applications, paths are typically planned so as to guarantee the smoothness of the actuators reference signals. In particular, if $\mathbf{q} := [q_1\, q_2, \ldots, q_n]^T$ is the vector of the joint variables and n is the number of joints, $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ should be continuous, but advanced applications also impose the jerk continuity. Such additional continuity level is introduced to reduce the mechanical solicitations acting on the manipulator structure and to improve the controller performances. For Cartesian trajectories, such continuity requests naturally lead to equivalent requirements involving the motion of the origin of the tool frame, i.e., $\mathbf{p} := [p_x\ p_y\ p_z]^T$. The conditions which must be satisfied by a Cartesian curve in order to guarantee the continuity of the joints jerks are derived in the reminder of this section. Furthermore, some geometric expressions used in next Section 3 are briefly recalled.

Cartesian trajectories are typically planned by avoiding kinematic singularities, so that the Jacobian matrix of the system, i.e., $\mathbf{J}(\mathbf{q})$, will be supposed non singular in this paper and the inverse kinematic function, i.e., $\mathbf{q} = \mathbf{q}(\mathbf{p})$, will be assumed continuous. By virtue of the last hypothesis, the Cartesian path continuity guarantees, in turn, the continuity of $\mathbf{q}$. Furthermore, for industrial manipulators, $\mathbf{J}(\mathbf{q})$ is a continuously differentiable function of class C^∞.

Linear velocities can always be evaluated through a Jacobian matrix, $\mathbf{J}(\mathbf{q})$, according to the following equation

$$\dot{\mathbf{p}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}\,. \tag{1}$$

Such equation can be reorganized as follows

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\ \dot{\mathbf{p}}\ . \tag{2}$$

According to the premises, the Jacobian matrix is not singular, so that its inverse exists. Moreover, since $\mathbf{J}(\mathbf{q}) \in C^\infty$ and bearing in mind the inverse function theorem, $\mathbf{J}^{-1}(\mathbf{q})$ is certainly a continuous function. Since $\mathbf{q}$ is continuous, (2) makes it possible to conclude that the continuity of $\dot{\mathbf{q}}$ is achieved by assuming a continuous $\dot{\mathbf{p}}$.

The same reasoning can be used with the higher order derivatives. The differentiation of (1) leads to the following equations

$$\ddot{\mathbf{p}} = \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}}\ ,$$

$$\dddot{\mathbf{p}} = \ddot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} + 2\dot{\mathbf{J}}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{J}(\mathbf{q})\dddot{\mathbf{q}}\ ,$$

which can be rewritten as follows

$$\ddot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\ [\ddot{\mathbf{p}} - \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}]\ , \tag{3}$$

$$\dddot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\ [\dddot{\mathbf{p}} - \ddot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} - 2\dot{\mathbf{J}}(\mathbf{q})\ddot{\mathbf{q}}]\ . \tag{4}$$

Since $\mathbf{J}(\mathbf{q}) \in C^\infty$, (3) and (4) allow one asserting that the continuity of $\ddot{\mathbf{q}}$ and $\dddot{\mathbf{q}}$ is obtained by assuming that also $\ddot{\mathbf{p}}$ and $\dddot{\mathbf{p}}$ are continuous. Practically, if $\mathbf{p}$ is C^3 then $\mathbf{q}$ is C^3 as well.

Cartesian trajectories are very commonly planned by adopting the path-velocity decomposition approach [39], so that they are obtained by combining a path, $\mathbf{p}(s)$, with a timing law, $s(t)$. s is the so-called curvilinear coordinate and it coincides with the distance from the beginning of the curve, measured along the curve itself. Consequently, $s \in [0, s_f]$ for a curve whose length is s_f. Bearing in mind such considerations, the time derivatives of $\mathbf{p}$ can be written as follows

$$\dot{\mathbf{p}} = \frac{d}{dt}\mathbf{p}[s(t)] = \frac{d\mathbf{p}}{ds}\frac{ds}{dt} = \frac{d\mathbf{p}}{ds}\dot{s}\,, \tag{5}$$

$$\ddot{\mathbf{p}} = \frac{d^2}{dt^2}\mathbf{p}[s(t)] = \frac{d^2\mathbf{p}}{ds^2}\dot{s}^2 + \frac{d\mathbf{p}}{ds}\ddot{s}\,, \tag{6}$$

$$\dddot{\mathbf{p}} = \frac{d^3}{dt^3}\mathbf{p}[s(t)] = \frac{d^3\mathbf{p}}{ds^3}\dot{s}^3 + 3\frac{d^2\mathbf{p}}{ds^2}\dot{s}\ddot{s} + \frac{d\mathbf{p}}{ds}\dddot{s}\,. \tag{7}$$

Clearly, (5)–(7) imply that the continuity of $\dot{\mathbf{p}}$, $\ddot{\mathbf{p}}$, and $\dddot{\mathbf{p}}$ requires the continuity of $(d\mathbf{p})/(ds)$, $(d^2\mathbf{p})/(ds)^2$, and $(d^3\mathbf{p})/(ds)^3$, as well as of $\dot{s}$, $\ddot{s}$, and $\dddot{s}$. Hence, the continuity problem can be split into two separate sub-problems, the first one involving the timing law, the second one concerning the geometric characteristics of the path. In particular, for paths which are parametrized as function of the curvilinear coordinate, i.e., s, the two concepts of analytic and geometric continuity coincide, so that if $\mathbf{p}(s)$ is a continuous function, i.e., if $\mathbf{p}(s) \in C^0$, then it also admits 0 order geometric continuity – or, equivalently, $\mathbf{p}(s) \in \mathcal{G}^0$. The same concept also applies to higher order derivatives, so that if $\mathbf{p}(s) \in C^1$, then $[d\mathbf{p}(s)]/(ds)$ admits a geometric continuity of the first order, i.e., $\mathbf{p}(s) \in \mathcal{G}^1$, and so on. This implies that, for the problem at hand, the focus is posed on paths belonging to $\mathcal{G}^3$ in order to achieve the continuity of (5)–(7).

For the reader convenience, the reminder of this section will recall some geometric implications of the $\mathcal{G}^3$ continuity concepts. For practical reasons, curves are often defined through a function $\mathbf{p}(u) \in \mathbb{R}^3$, where $u \in [u_0, u_f]$ is a generic scalar parameter which is used instead of s. By construction [40], $\mathbf{p}'(u) := [d\mathbf{p}(u)]/(du)$ is a vector which is tangent in u to the curve. For regular curves, i.e., curves for which $\|\mathbf{p}'(u)\| > 0, \forall u \in [u_0, u_f]$, there always exists a direct, bijective relationship between u and s, which can be expressed as follows

$$s = \int_{u_0}^{u_f} \|\mathbf{p}'(\tau)\|\, d\tau\,. \tag{8}$$

Consequently, if $\mathbf{p}(u)$ is continuous in u, then $\mathbf{p}[u(s)]$ is continuous in s – $u(s)$ is the inverse function of (8) – and the curve is $\mathcal{G}^0$. Eq. (8) immediately allows one writing

$$\frac{ds}{du} = \|\mathbf{p}'(u)\|\,,$$

so that the path derivative w.r.t. s can be written as follows

$$\frac{d\mathbf{p}(u)}{ds} = \frac{d\mathbf{p}(u)}{du}\frac{du}{ds} = \mathbf{p}'(u)\frac{1}{\frac{ds}{du}} = \frac{\mathbf{p}'(u)}{\|\mathbf{p}'(u)\|} = \mathbf{t}(u), \tag{9}$$

Practically, the $\mathcal{G}^1$ continuity implies that unit vector $\mathbf{t}(u)$, which is by construction tangent to the curve (see also Fig. 1), is a continuous function of u.

For which concerns the second order geometric continuity, i.e., the continuity of $(d^2\mathbf{p})/(ds^2)$, the following equation is obtained by differentiating (9):

$$\frac{d^2\mathbf{p}(u)}{ds^2} = \frac{d}{ds}\left[\frac{d\mathbf{p}(u)}{ds}\right] = \frac{d}{ds}[\mathbf{t}(u)] = \kappa(u)\mathbf{n}(u). \tag{10}$$

The last equality in (10) is due to the first Frenet–Serret equation [40]. $\mathbf{n}(u)$ is the *normal unit vector* which points toward the center of the osculating circle (see also Fig. 1), i.e., the circle which best approximates the curve in u. $\kappa(u)$ is the *curvature* in u, i.e., it is the reciprocal of the radius of the osculating circle. They are defined as follows [40]

$$\mathbf{n}(u) = \frac{[\mathbf{p}'(u) \times \mathbf{p}''(u)] \times \mathbf{p}'(u)}{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|\, \|\mathbf{p}'(u)\|}, \tag{11}$$

$$\kappa(u) = \frac{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|}{\|\mathbf{p}'(u)\|^3}. \tag{12}$$

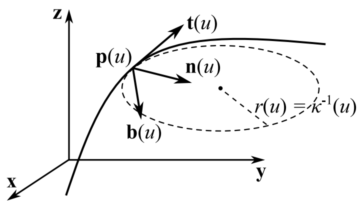

Fig. 1. A generic curve $\mathbf{p}(u)$ (solid line), together with its Frenet frame and its osculating circle (dashed line) shown for a generic u.

Evidently, $\mathbf{n}(u)$ is well defined if the curve is *biregular*, i.e., if it satisfies condition $\|\mathbf{p}'(u) \times \mathbf{p}''(u)\| > 0, \forall u \in [u_0, u_f]$. By virtue of (10), a curve is $\mathcal{G}^2$ if $\mathbf{n}(u)$ and $\kappa(u)$ are continuous functions.

By further differentiating (10), the following equation is obtained

$$\begin{aligned}\frac{d^3\mathbf{p}(u)}{ds^3} &= \frac{d}{ds}[\kappa(u)\mathbf{n}(u)] = \frac{d\kappa(u)}{ds}\mathbf{n}(u) + \kappa(u)\frac{d\mathbf{n}(u)}{ds}\\ &= \frac{\kappa'(u)}{\|\mathbf{p}'(u)\|}\mathbf{n}(u) + \kappa(u)\frac{d\mathbf{n}(u)}{ds}.\end{aligned} \tag{13}$$

The second Frenet–Serret equation [40] asserts that

$$\frac{d\mathbf{n}(u)}{ds} = -\kappa(u)\mathbf{t}(u) + \tau(u)\mathbf{b}(u), \tag{14}$$

where

$$\mathbf{b}(u) := \mathbf{t}(u) \times \mathbf{n}(u) = \frac{\mathbf{p}'(u) \times \mathbf{p}''(u)}{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|} \tag{15}$$

is the so called *binormal unit vector* and

$$\tau(u) = \frac{[\mathbf{p}'(u) \times \mathbf{p}''(u)] \cdot \mathbf{p}'''(u)}{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|^2} \tag{16}$$

is the *torsion* of the curve in u. Unit vectors $\mathbf{t}(u)$, $\mathbf{n}(u)$, and $\mathbf{b}(u)$ are each other orthogonal and form the so-called *Frenet frame* associated to point u along the curve. From (13) and (14) it immediately descends that the $\mathcal{G}^3$ continuity is achieved if a curve is $\mathcal{G}^2$, i.e., $\mathbf{t}(u)$, $\mathbf{n}(u)$, $\kappa(u)$ are continuous, and, furthermore, if the continuity is also guaranteed for

$$\begin{aligned}\frac{d\kappa(u)}{ds} &= \frac{\kappa'(u)}{\|\mathbf{p}'(u)\|} = \frac{[\mathbf{p}'(u) \times \mathbf{p}''(u)] \cdot [\mathbf{p}'(u) \times \mathbf{p}'''(u)]}{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|\, \|\mathbf{p}'(u)\|^4}\\ &\quad - 3[\mathbf{p}'(u) \cdot \mathbf{p}''(u)]\frac{\|\mathbf{p}'(u) \times \mathbf{p}''(u)\|}{\|\mathbf{p}'(u)\|^6}\\ &= \mathbf{b}(u) \cdot \frac{\mathbf{p}'(u) \times \mathbf{p}'''(u)}{\|\mathbf{p}'(u)\|^4} - 3\kappa(u)\frac{\mathbf{p}'(u) \cdot \mathbf{p}''(u)}{\|\mathbf{p}'(u)\|^3}\end{aligned} \tag{17}$$

and for $\tau(u)$. By virtue of (9), (11), (12), and (16), the $\mathcal{G}^3$ continuity imposes that $\mathbf{p}(u)$ and its derivatives w.r.t. u, up to the third order, must be continuous, i.e., $\mathbf{p}(u) \in C^3$.

Summarizing, (2)–(7) allow one asserting that the continuity of $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, and $\dddot{\mathbf{q}}$ can be obtained by planning $\mathcal{G}^3$ curves, i.e., by imposing that $\mathbf{t}(u)$, $\mathbf{n}(u)$, $\mathbf{b}(u)$, $\kappa(u)$, $\kappa'(u)$, and $\tau(u)$ are continuous functions, and, moreover, by generating a timing law such that the continuity is also guaranteed for s, $\dot{s}$, $\ddot{s}$, and $\dddot{s}$. The timing law generation problem is not addressed in this work, but possible approaches can be found in the literature (see for example [41]).

3. The η^{3D}-splines

The proposed primitive is based on a 7th order vector polynomial defined as follows

$$\mathbf{p}(u) := \chi_0 + \chi_1 u + \chi_2 u^2 + \chi_3 u^3 + \chi_4 u^4 + \chi_5 u^5 + \chi_6 u^6 + \chi_7 u^7, \tag{18}$$

where $u \in [0, 1]$, while $\chi_i := [\alpha_i\, \beta_i\, \gamma_i]^T \in \mathbb{R}^3$ are properly defined vector coefficients. The $\mathcal{G}^3$ continuity is certainly achieved $\forall u \in (0, 1)$ since, for constant values of χ_i, the derivatives of $\mathbf{p}(u)$ of any order are continuous in such interval. However, this work aims at generating $\mathcal{G}^3$ composite paths obtained by joining several curves, so that coefficients

Table 1
Interpolating conditions for the η^{3D}-spline curve. (32).

A	*B*
$\mathbf{p}(0)=\mathbf{p}_A$	$\mathbf{p}(1)=\mathbf{p}_B$
$[\mathbf{t}(0)\ \mathbf{n}(0)\ \mathbf{b}(0)]=[\mathbf{t}_A\ \mathbf{n}_A\ \mathbf{b}_A]$	$[\mathbf{t}(1)\ \mathbf{n}(1)\ \mathbf{b}(1)]=[\mathbf{t}_B\ \mathbf{n}_B\ \mathbf{b}_B]$
$\kappa(0)=\kappa_A$	$\kappa(1)=\kappa_B$
$\tau(0)=\tau_A$	$\tau(1)=\tau_B$
$\frac{d\kappa}{ds}(0)=\bar{\kappa}_A$	$\frac{d\kappa}{ds}(1)=\bar{\kappa}_B$

χ_i must be assigned so as to also impose the $\mathcal{G}^3$ continuity in the junction points between adjacent curves. According to the discussion in Section 2, such result can be achieved by imposing that, at the end of each curve, $\mathbf{t}$, $\mathbf{n}$, $\mathbf{b}$, κ, κ', and τ coincide with the homologous terms of the subsequent one. Practically, the $\mathcal{G}^3$ continuity is obtained if the interpolating conditions shown in Table 1 can be arbitrarily imposed to (18) (subscripts A and B indicate the assigned interpolating conditions at the beginning and at the end of the curve, respectively). Evidently, such boundary conditions can be imposed by solving an appropriate system of equations but, more conveniently, this work proposes closed form expressions for the immediate and efficient evaluation of the χ_i parameters. In particular, the coefficients of the seventh order polynomial curve, satisfying the boundary conditions specified in Table 1, can be immediately obtained by means of the following expressions

$$\chi_0 = \mathbf{p}_A, \tag{19}$$

$$\chi_1 = \eta_1 \mathbf{t}_A, \tag{20}$$

$$\chi_2 = (1/2)[\kappa_A \eta_1^2 \mathbf{n}_A + \eta_3 \mathbf{t}_A], \tag{21}$$

$$\chi_3 = \frac{1}{6}[\kappa_A\tau_A\mathbf{b}_A\eta_1^3 + (\bar{\kappa}_A\eta_1^2 + 3\,\kappa_A\eta_1\eta_3)\mathbf{n}_A] + \eta_5\mathbf{t}_A, \tag{22}$$

$$\begin{aligned}\chi_4 =& -(2/3)\,\kappa_A\tau_A\mathbf{b}_A\eta_1^3 - (1/6)\,\kappa_B\tau_B\mathbf{b}_B\eta_2^3\\ &- (1/3)\left[\eta_1^2\left(2\,\bar{\kappa}_A + 15\,\kappa_A\right) + 6\,\kappa_A\eta_1\eta_3\right]\mathbf{n}_A\\ &- (1/6)\left[\eta_2^2\left(\bar{\kappa}_B - 15\,\kappa_B\right) + 3\,\kappa_B\eta_2\eta_4\right]\mathbf{n}_B\\ &- \left(20\,\eta_1 + 5\,\eta_3 + 4\,\eta_5\right)\mathbf{t}_A - 35\,\mathbf{p}_A + 35\,\mathbf{p}_B\\ &- (1/2)\left(30\,\eta_2 - 5\,\eta_4 + 2\,\eta_6\right)\mathbf{t}_B,\end{aligned} \tag{23}$$

$$\begin{aligned}\chi_5 =& \kappa_A\tau_A\mathbf{b}_A\eta_1^3 + \frac{1}{2}\,\kappa_B\tau_B\mathbf{b}_B\eta_2^3\\ &+ \left[\eta_1^2\left(\bar{\kappa}_A + 10\,\kappa_A\right) + 3\,\kappa_A\eta_1\eta_3\right]\mathbf{n}_A\\ &+ (1/2)\left[\eta_2^2\left(\kappa_B - 14\,\kappa_B\right) + 3\,\kappa_B\eta_2\eta_4\right]\mathbf{n}_B\\ &+ \left(45\,\eta_1 + 10\,\eta_3 + 6\,\eta_5\right)\mathbf{t}_A\\ &+ \left(39\,\eta_2 - 7\,\eta_4 + 3\,\eta_6\right)\mathbf{t}_B + 84\,\mathbf{p}_A - 84\,\mathbf{p}_B,\end{aligned} \tag{24}$$

$$\begin{aligned}\chi_6 =& -(2/3)\,\kappa_A\tau_A\mathbf{b}_A\eta_1^3 - (1/2)\,\kappa_B\tau_B\mathbf{b}_B\eta_2^3\\ &- (1/6)\left[\eta_1^2\left(4\,\bar{\kappa}_A + 45\,\kappa_A\right) + 12\,\kappa_A\eta_1\eta_3\right]\mathbf{n}_A\\ &- (1/2)\left[\eta_2^2\left(\bar{\kappa}_B - 13\,\kappa_B\right) + 3\,\kappa_B\eta_2\eta_4\right]\mathbf{n}_B\\ &- (1/2)\left(72\,\eta_1 + 15\,\eta_3 + 8\,\eta_5\right)\mathbf{t}_A - 70\,\mathbf{p}_A + 70\,\mathbf{p}_B\\ &- (1/2)\left(68\,\eta_2 - 13\,\eta_4 + 6\,\eta_6\right)\mathbf{t}_B,\end{aligned} \tag{25}$$

$$\begin{aligned}\chi_7 =& (1/6)(\,\kappa_A\tau_A\mathbf{b}_A\eta_1^3 + \,\kappa_B\tau_B\mathbf{b}_B\eta_2^3)\\ &+ (1/6)\left[\eta_1^2\left(\bar{\kappa}_A + 12\,\kappa_A\right) + 3\,\kappa_A\eta_1\eta_3\right]\mathbf{n}_A\\ &+ (1/6)\left[\eta_2^2\left(\bar{\kappa}_B - 12\,\kappa_B\right) + 3\,\kappa_B\eta_2\eta_4\right]\mathbf{n}_B\\ &+ \left(10\,\eta_1 + 2\,\eta_3 + \eta_5\right)\mathbf{t}_A\\ &+ \left(10\,\eta_2 - 2\,\eta_4 + \eta_6\right)\mathbf{t}_B + 20\,\mathbf{p}_A - 20\,\mathbf{p}_B.\end{aligned} \tag{26}$$

By analyzing (19)–(26), it can be noticed that the χ_i coefficients only depend on the interpolating conditions and on a set of six independent parameters which can be conveniently packed into vector $\boldsymbol{\eta} := [\eta_1\,\eta_2\,\eta_3\,\eta_4\,\eta_5\,\eta_6]^T$, where $\eta_1, \eta_2 \in \mathbb{R}^+$ and $\eta_3, \eta_4, \eta_5, \eta_6 \in \mathbb{R}$. Consequently, given the interpolating conditions and vector $\boldsymbol{\eta}$, the η^{3D}-splines coefficients can be obtained at a negligible computational cost. The selection of $\boldsymbol{\eta}$ influences the curve shape and will be discussed in Section 4.

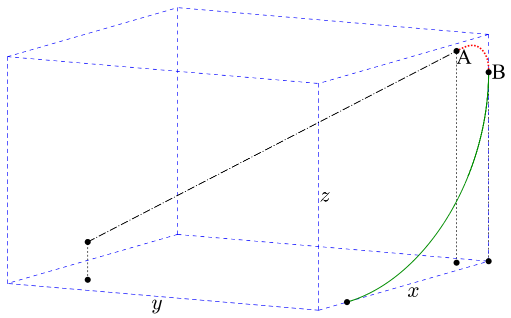

Fig. 2. An η^{3D}-spline (dotted red line) is used to smoothly join a linear segment (dash-dotted black line) with a circular arch (solid green line). The composite path is $\mathcal{G}^3$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

For space reasons the procedure for the synthesis of (19)–(26) is not proposed but, more conveniently, the curve properties and its strengths are pointed out in the following by means of two propositions. In particular, Proposition 1 asserts that η^{3D}-splines can generate, through a proper choice of $\boldsymbol{\eta}$, all possible 7th order polynomial curves which satisfy the assigned interpolating conditions. The proof of Proposition 1, reported in Appendix A, points out another important characteristic which is stated in Proposition 2: the interpolating conditions, which are normally assigned so as to guarantee the $\mathcal{G}^3$ continuity, are always satisfied independently from the choice of $\boldsymbol{\eta}$. This implies that the curve can be modeled by means of $\boldsymbol{\eta}$, but the choice of $\boldsymbol{\eta}$ does not affect the continuity properties of the combined path.

Proposition 1. *Given a generic 7th order polynomial* $\mathbf{p}(u)$*, expressed by means of* (18)*, which satisfies the interpolating conditions of* Table 1*, through a proper choice of vector* $\boldsymbol{\eta}$ *it is always possible to find an* η^{3D}*-spline, namely* $\mathbf{p}_{\boldsymbol{\eta}}(u)$*, such that* $\mathbf{p}_{\boldsymbol{\eta}}(u) = \mathbf{p}(u)$*.*

Proof. See Appendix A.

Proposition 2. *Any path primitive* $\mathbf{p}_{\boldsymbol{\eta}}(u)$*, obtained by selecting the coefficients of* (18) *through* (19)–(26)*, always satisfies the interpolating conditions of* Table 1*, independently from the choice of* $\boldsymbol{\eta}$*.*

Proof. It is an immediate consequence of the demonstration in Appendix A.

Propositions 1 and 2 suggest two possible applications for the η^{3D}-splines. For example, the novel planning primitive can be used to generate composite $\mathcal{G}^3$-paths or, alternatively, through an appropriate choice of vector $\boldsymbol{\eta}$ (see Section 4), for the emulation of 3D curves.

In industrial contexts, the first application is probably the most common one. An example case is proposed in Fig. 2: a composite $\mathcal{G}^3$-path is easily obtained by joining a linear segment (dash-dotted black line) and a circular arc (solid green line), generically located in a 3D environment, through the η^{3D}-splines (dotted red line). The interpolating conditions for the η^{3D}-splines are directly obtained from the path primitives which need to be joined. In particular, for point A, the interpolating conditions are the same of the-end point of the linear segment, while in B they coincide with the initial ones of the circular arc. More in details, $\mathbf{p}_A$ is the end-point of the linear segment, while $\mathbf{t}_A$ coincides with its characteristic unit vector. In any point of a linear segment, including its extremes, curvature, curvature derivative, and torsion are zero, so that $\kappa_A = \bar{\kappa}_A = \tau_A = 0$. Since $\kappa_A = 0$ and $\tau_A = 0$, then $\mathbf{n}_A$ and $\mathbf{b}_A$ can be freely selected: they must only satisfy conditions $\mathbf{n}_A \cdot \mathbf{b}_A = 0$ and $\mathbf{n}_A \times \mathbf{b}_A = \mathbf{t}_A$. Concerning point B, $\mathbf{p}_B$ coincides with the starting point of the circular arc and $\mathbf{t}_B$ is the unit tangent in the same point. For a circular arc the curvature is constant and equal to r^{-1}, where r is the radius of the primitive. Consequently, $\kappa_B = r^{-1}$. Normal vector $\mathbf{n}_B$ points towards the center of the circular arc, while the binormal vector is evaluated as $\mathbf{b}_B = \mathbf{t}_B \times \mathbf{n}_B$. Curvature derivative and torsion are equal to zero, i.e., $\bar{\kappa}_B = \tau_B = 0$. According to Proposition 2, the imposition of such boundary conditions guarantees the $\mathcal{G}^3$ continuity of the composite path. However, the curve shape can still be modeled through the choice of vector $\boldsymbol{\eta}$. Details on the selection of $\boldsymbol{\eta}$ are given in next Section 4. More precisely, possible strategies will be proposed for the generation of generic junction profiles or in order to let η^{3D}-splines emulate other planning primitives.

4. Considerations on the selection of $\boldsymbol{\eta}$

The curve shape can be imposed by means of $\boldsymbol{\eta}$. To this purpose, several alternative strategies can be proposed. For example, it may be chosen by means of nonlinear programming algorithms [42], in order to satisfy some given optimal criteria. However, in many practical cases, simple heuristic strategies are sufficient for the generation of curves with interesting geometric characteristics. The advantage of such heuristic rules is represented by their efficiency, which allows the online generation of complex $\mathcal{G}^3$ paths. In particular, curves with nice geometric properties can be obtained by assigning $\boldsymbol{\eta}$ as follows

$$\eta_1 = \eta_2 = s_f, \tag{27}$$

$$\eta_3 = \eta_4 = \eta_5 = \eta_6 = 0, \tag{28}$$

where s_f is the curve length.

Such rule of thumb emerged during the studies on the emulation capabilities of the η^{3D}-splines. As stated in the Introduction, $\boldsymbol{\eta}$-splines can emulate, with very good approximation, many path primitives like, for example, circular arcs and clothoids.

Let us consider a set of circular arcs with different length $s_f = (r\pi)/(2i)$, where r is the radius and $i = 1, 2, \dots, 6$. For each arc, the inner angle is clearly given by $\tilde{\theta} := s_f/r$ (see also Fig. 3). The emulation capability of η^{3D}-splines can be measured through the following approximation error

$$e(s) := \min_{u\in[0,1]} \left\| \mathbf{p}_{\boldsymbol{\eta}}(u) - \mathbf{p}(s) \right\|, \tag{29}$$

where $\mathbf{p}_{\boldsymbol{\eta}}(u)$ is the spline curve used to emulate the actual arc, i.e., $\mathbf{p}(s)$, $s \in [0, s_f]$. Practically, $e(s)$ is the minimum Euclidean distance between $\mathbf{p}_{\boldsymbol{\eta}}(u)$ and $\mathbf{p}(s)$ measured for a given $s \in [0, s_f]$.

Very good emulation results have been verified even when the available degrees of freedom are only partially exploited. In particular, by assigning interpolating conditions compatible with an arc, $\boldsymbol{\eta}$ can be chosen as follows

$$\eta_1 = \eta_2 = \eta^*(\tilde{\theta}), \tag{30}$$
$$\eta_3 = \eta_4 = \eta_5 = \eta_6 = 0,$$

where $\eta^*(\tilde{\theta})$ is found by solving the following minimax problem

$$\min_{\eta^*\in\mathcal{S}} \max_{s\in[0,s_f]} \{|e(s)|\},$$

where $\mathcal{S}$ is a proper search interval. The problem was solved for different values of r and for $i = 1, 2, \dots, 6$. The nonlinear programming algorithm always converged to optimal values of η^* which were very close to s_f (they are not reported for conciseness).

The optimization results revealed that amplitudes of the emulation errors depend on r and on $\tilde{\theta}$. In particular, the dependence on r is perfectly linear, so that the normalized error, defined as $e_n(s) := e(s)/r$, $s \in [0, s_f]$, is only function of $\tilde{\theta}$. Average and maximum normalized errors for the 6 values of $\tilde{\theta}$ are listed in the first two rows of Table 2: they are generally small and become negligible as $\tilde{\theta}$ decreases.

The optimal values of η^* have been subsequently used to obtain, through a least square nonlinear regression, the following analytic expression for η_1 and η_2

$$\eta_1 = \eta_2 = s_f(\alpha\tilde{\theta}^2 + \beta\tilde{\theta} + \gamma), \tag{31}$$

Table 2
Average and maximum normalized errors $e_n(s)$ for several values of $\tilde{\theta}$ and $\eta_1 = \eta_2$.

$\tilde{\theta}$	$\pi/2$	$\pi/4$	$\pi/6$	$\pi/8$	$\pi/10$	$\pi/12$
$\eta_1 = \eta_2 = s^*$						
avg	$5.5 \cdot 10^{-6}$	$7.7 \cdot 10^{-8}$	$4.6 \cdot 10^{-9}$	$3.2 \cdot 10^{-9}$	$3.1 \cdot 10^{-10}$	$5.5 \cdot 10^{-11}$
max	$9.6 \cdot 10^{-6}$	$1.3 \cdot 10^{-7}$	$9.2 \cdot 10^{-9}$	$6.4 \cdot 10^{-9}$	$1.1 \cdot 10^{-9}$	$1.6 \cdot 10^{-10}$
$\eta_1 = \eta_2 = s_f(\alpha\tilde{\theta}^2 + \beta\tilde{\theta} + \gamma)$						
avg	$5.2 \cdot 10^{-6}$	$2.7 \cdot 10^{-6}$	$1.7 \cdot 10^{-6}$	$8.6 \cdot 10^{-7}$	$2.7 \cdot 10^{-7}$	$2.1 \cdot 10^{-8}$
max	$9.2 \cdot 10^{-6}$	$6.7 \cdot 10^{-6}$	$4.2 \cdot 10^{-6}$	$2.2 \cdot 10^{-6}$	$6.9 \cdot 10^{-7}$	$5.3 \cdot 10^{-8}$
$\eta_1 = \eta_2 = s_f$						
avg	$3.2 \cdot 10^{-3}$	$2.1 \cdot 10^{-4}$	$4 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$	$6.0 \cdot 10^{-6}$	$4.0 \cdot 10^{-6}$
max	$7.8 \cdot 10^{-3}$	$5.0 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$3.3 \cdot 10^{-5}$	$1.4 \cdot 10^{-5}$	$7.0 \cdot 10^{-6}$

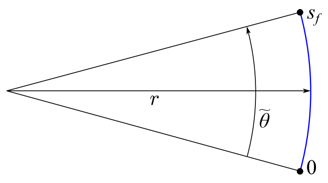

Fig. 3. A circular arc obtained by means of the η^{3D}-splines.

where $\alpha = -0.0099417176196074$, $\beta = -0.0055734866225982$, and $\gamma = 1.00101667238653$. The resulting approximation errors for the 6 test cases, obtained by still assuming (28) and (30), are shown in the third and in the fourth rows of Table 2. For $\tilde{\theta} = \pi/2$, the maximum error is close 10^{-5} m for an arc whose radius is equal to $r = 1$ m. Such error is acceptable for many robotic applications and it further reduces for smaller values of r since, according to the definition of normalized error, $e(s) = r\, e_n(s)$.

Since α and β are very small, while $\gamma \simeq 1$, η^* is generally close to s_f. Such consideration suggested to test solutions obtained by directly assigning $\boldsymbol{\eta}$ according to (27) and (28). As shown by the fifth and the sixth rows of Table 2, emulation errors are still acceptable, especially for small values of $\tilde{\theta}$.

As early anticipated, η^{3D}-splines can also emulate clothoids. Such capability has been tested by considering the same set of curves used in [43,44] to check the emulation capabilities of Bézier curves. The set is composed by 30 clothoids, joined together so as to obtain a composite curve 6.0 m long. The composite path curvature is proportional to the path length, i.e., $\kappa(s) = s$, and each segment is 0.2 m long.

Very good results have been achieved by first imposing condition (28), and then by calculating the remaining 2 parameters through the following optimal problem

$$\min_{\eta_1^*,\eta_2^*\in\mathcal{S}} \max_{s\in[0,s_f]} \{|e(s)|\}.$$

The worst case error of the optimal solutions was equal to $4.655 \cdot 10^{-6}$ m. Errors were only marginally influenced by perturbations of the optimal solutions. Consequently, the worst case error only slightly increases ($5.630 \cdot 10^{-6}$ m), if the 30 optimal values of η_1^* and η_2^* are replaced by the following 2 functions obtained through a nonlinear regression

$$\eta_i(\kappa_{av}) := \lambda_{0i} + \lambda_{1i}\kappa_{av} + \lambda_{2i}\kappa_{av}^2, \quad i = 1, 2, \tag{32}$$

where $\kappa_{av} := (\kappa_A + \kappa_B)/2$ is the average curvature of each segment. The coefficients of (32) are listed in Table 3. The first row of Table 3 points out that also for clothoids, the optimal values of η_1 and η_2 are close to s_f. In facts, if $\boldsymbol{\eta}$ is chosen according to (27) and (28), the maximum error is equal to $5.060 \cdot 10^{-6}$ m for clothoids admitting $\kappa_{av} < 1$ m^{-1}, i.e., it is very close to the value obtained by solving the optimization problem.

A proper selection of $\boldsymbol{\eta}$ is also relevant when η^{3D}-splines are used to create junction curves between other primitives. According to previous discussion (27) and (28) lead to a good emulation of circular arcs and

Table 3
Coefficients of function (32).

	η_1	η_2
λ_1	0.19920352009325834053	0.20097340855985815211
λ_2	0.00067959647624348148	−0.00091915134872076114
λ_3	−0.00018934505601462691	−0.00000857636957731629

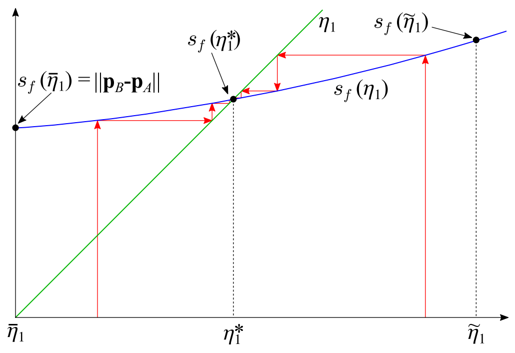

Fig. 4. Convergence of the algorithm if $s_f(\eta_1)$ is monotonically increasing. The transients are indicated through red lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

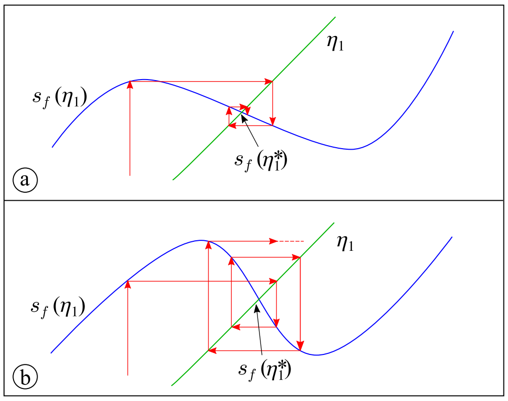

Fig. 5. Convergence properties: (a) the algorithm converges since $(ds_f)/(d\eta_1) > -1$ for $\eta_1 = \eta_1^*$, (b) the algorithm diverges since $(ds_f)/(d\eta_1) < -1$. The transients are indicated through red lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

clothoids, i.e., of curves with zero or constant curvature variability $[\kappa(u) = (\kappa_B - \kappa_A)/s_f]$: apparently, such choice seems to prevent the insurgence of possible oscillatory behaviors in the curve shape. Therefore, (27) and (28) were also tested for the generation of generic profiles. The resulting curves shown, as desired, moderate and slowly variable curvatures and torsions, while oscillatory behaviors were totally absent.

The proposed selection strategy has inevitably a drawback: in case of generic interpolating conditions s_f is not known in advance, so that the imposition of (27) is not straightforward. In order to overcome the problem, the following iterative procedure has been used. An initial curve is generated by imposing $\eta_1 = \eta_2 = \|\mathbf{p}_A - \mathbf{p}_B\|$ and its length $\tilde{s}_f$ is evaluated. Then, a second curve is planned by imposing $\eta_1 = \eta_2 = \tilde{s}_f$ and its length is used for the next iteration. After few iterations – 2 or 3 are normally sufficient – the procedure converges to a value which is very close to s_f.

Evidently, such algorithm can only be used if it convergences toward $\eta_1^* = s_f(\eta_1^*)$, where η_1^* indicates the convergence point. To this purpose, function $s_f(\eta_1)$ must assume a shape similar to the one shown in Fig. 4: after some iterations, the algorithm would necessarily converge toward η_1^* independently from the starting point.

Proposition 3. *For sufficiently small values of $\|\mathbf{p}_B - \mathbf{p}_A\|$, the proposed algorithm converges toward $\eta_1^* = s_f(\eta_1^*)$.*

Proof. The iterative algorithm looks for the intersection point between functions $f_1 = s_f(\eta_1)$ and $f_2 = \eta_1$. As a consequence, it is first necessary to demonstrate the two functions actually intersect each other. Owing to the continuity of function $s_f(\eta_1)$, this hypothesis is verified if there exist $\bar{\eta}_1 < \tilde{\eta}_1$ such that $s_f(\bar{\eta}_1) > \eta_1 > s_f(\tilde{\eta}_1)$. Furthermore, in order to prove the convergence, it is necessary to verify that the slope of $s_f(\eta_1)$ is higher than -1 over the search interval and, in particular, in η_1^*. The reason of this second requirement can be understood with the aid of Figs. 4 and 5. The first one shows two typical convergence sequences occurring when $s_f(\eta_1)$ is a monotonically increasing function. Conversely, Fig. 5 shows two possible situations which may occur if $(ds_f)/(d\eta_1)$ is negative in η_1^*: in case (a) the derivative is greater than -1 and the convergence is achieved, while in case (b) the algorithm does not converge.

- *There exists $\bar{\eta}_1$ such that $s_f(\bar{\eta}_1) > \bar{\eta}_1$* – Closed form expressions for $\mathbf{p}'(u)$ can be found by evaluating its coefficients through (19)-(26) and by assuming that (27) and (28) apply. A few algebraic manipulations make it possible to write $\mathbf{p}'(u)$ as follows

$$\begin{aligned}\mathbf{p}'(u;\eta_1) =& f_1(u)\eta_1^3\mathbf{b}_A + f_2(u)\eta_1^3\mathbf{b}_B + f_3(u)\eta_1^2\mathbf{n}_A + f_4(u)\eta_1^2\mathbf{n}_B \\ &+ f_5(u)\eta_1\mathbf{t}_A + f_6(u)\eta_1\mathbf{t}_B + f_7(u)(\mathbf{p_B} - \mathbf{p}_A),\end{aligned} \tag{33}$$

where $f_i(u), i = 1, 2, \dots, 7$ are proper scalar polynomial functions which also depend on $\kappa_A, \kappa_B, \bar{\kappa}_A, \bar{\kappa}_B, \tau_A$, and τ_B.

By assuming $\eta_1 \to 0$, from (33) it immediately descends that

$$\lim_{\eta_1\to 0} \|\mathbf{p}'(u;\eta_1)\| = |f_7(u)|\, \|\mathbf{p}_B - \mathbf{p}_A\|\,. \tag{34}$$

$s_f(\bar{\eta}_1)$ can be obtained by applying (34) to (8). The following result is obtained

$$\lim_{\eta_1\to 0} s_f(\eta_1) = \|\mathbf{p}_B - \mathbf{p}_A\| \int_0^1 |f_7(u)|\, du = \|\mathbf{p}_B - \mathbf{p}_A\| \geq 0,$$

where $|f_7(u)| = 140(1-u)^3u^3$ is a function whose integral, evaluated over $[0, 1]$, is equal to 1. Practically, by assuming $\bar{\eta}_1 = 0$ condition $s_f(\bar{\eta}_1) > \bar{\eta}_1$ is banally satisfied.

- *There exists $\tilde{\eta}_1$ such that $s_f(\tilde{\eta}_1) < \tilde{\eta}_1$* – By virtue of the triangular inequality, (33) allows one writing the following expression

$$\begin{aligned}\|\mathbf{p}'(u)\| \leq& |f_1(u)|\,\eta_1^3\,\|\mathbf{b}_A\| + |f_2(u)|\,\eta_1^3\,\|\mathbf{b}_B\| + |f_3(u)|\,\eta_1^2\,\|\mathbf{n}_A\| + |f_4(u)|\,\eta_1^2\,\|\mathbf{n}_B\| \\ &+ |f_5(u)|\,\eta_1\,\|\mathbf{t}_A\| + |f_6(u)|\,\eta_1\,\|\mathbf{t}_B\| + |f_7(u)|\,\|\mathbf{p}_B - \mathbf{p}_A\|\,, \\ =& |f_1(u)|\,\eta_1^3 + |f_2(u)|\,\eta_1^3 + |f_3(u)|\,\eta_1^2 + |f_4(u)|\,\eta_1^2 \\ &+ |f_5(u)|\,\eta_1 + |f_6(u)|\,\eta_1 + |f_7(u)|\,\|\mathbf{p}_B - \mathbf{p}_A\|\,,\end{aligned} \tag{35}$$

According to (8), s_f can be obtained by integrating $\|\mathbf{p}'(u)\|$. Consequently, the integrals of both sides of (35) lead, after some algebraic manipulations, to the following inequality

$$s_f(\eta_1) \leq K_1\eta_1^3 + K_2\eta_1^2 + 0.9074\eta_1 + \|(\mathbf{p}_B - \mathbf{p}_A)\|\,, \tag{36}$$

where

$$\begin{aligned}K_1 :=& 0.2798 \cdot 10^{-2}(|\kappa_A\tau_A| + |\kappa_B\tau_B|) \geq 0, \\ K_2 :=& 0.8988 \cdot 10^{-2}(|\kappa_A| + |\kappa_B|) + 0.4663 \cdot 10^{-3}(|\bar{\kappa}_A| + |\bar{\kappa}_B|) \geq 0.\end{aligned}$$

Bearing in mind (36), condition $s_f(\tilde{\eta}_1) < \tilde{\eta}_1$ is satisfied if, in turn, the following inequality holds

$$(K_1\tilde{\eta}_1^2 + K_2\tilde{\eta}_1 + 0.9074)\tilde{\eta}_1 \leq \tilde{\eta}_1 - \|(\mathbf{p}_B - \mathbf{p}_A)\| \tag{37}$$

or, equivalently, if

$$(0,0926 - K_1\tilde{\eta}_1^2 - K_2\tilde{\eta}_1)\tilde{\eta}_1 \geq \|(\mathbf{p}_B - \mathbf{p}_A)\| \tag{38}$$

By recalling that $K_1, K_2, \widetilde{\eta}_1 \geq 0$, two conclusion can be drawn. Firstly, the following inequality must apply in order to satisfy (37)

$$\widetilde{\eta}_1 \geq \|(\mathbf{p}_B - \mathbf{p}_A)\|\,. \tag{39}$$

Secondly, depending on the interpolating conditions (38) may not admit feasible solutions. In that case, the original planning problem can be split into sub-problems, so as to reduce $\|\mathbf{p}_B - \mathbf{p}_A\|$. Owing to the structure of (38), it will always be possible to find reasonably small values for $\|\mathbf{p}_B - \mathbf{p}_A\|$ and $\widetilde{\eta}_1$ such that (38) and (39) are simultaneously satisfied.

It is important to remark that conditions (38) is actually very restrictive, being obtained from a triangular inequality. Normally, condition $s_f(\widetilde{\eta}_1) < \widetilde{\eta}_1$ is satisfied by simply selecting a sufficiently large value of $\widetilde{\eta}_1$ fulfilling (39).

- *Condition* $(ds_f)/(d\eta_1) > -1$ *is satisfied* $\forall \eta_1 \in \mathbb{R}^+$ – Eq. (33) can also be written as follows

$$\mathbf{p}'(u;\eta_1) = \underbrace{[\mathbf{t}_A\ \mathbf{n}_A\ \mathbf{b}_A]}_{{}^0_A\mathbf{R}} \begin{bmatrix} f_5(u)\eta_1 \\ f_3(u)\eta_1^2 \\ f_1(u)\eta_1^3 \end{bmatrix} \underbrace{[\mathbf{t}_B\ \mathbf{n}_B\ \mathbf{b}_B]}_{{}^0_B\mathbf{R}} \begin{bmatrix} f_6(u)\eta_1 \\ f_4(u)\eta_1^2 \\ f_2(u)\eta_1^3 \end{bmatrix} + f_7(u)(\mathbf{p}_B - \mathbf{p}_A), \tag{40}$$

where ${}^0_A\mathbf{R}$ and ${}^0_B\mathbf{R}$ are rotation matrices which describe the orientation of the Frenet frame at the beginning and at the end of the curve. Bearing in mind (8) and (40), the derivative of s_f w.r.t. η_1 can be written as follows

$$\frac{\partial s_f}{\partial \eta_1} = \int_0^1 \frac{\partial \mathbf{p}'(u;\eta_1)}{\partial \eta_1} \mathbf{t}(u;\eta_1) du \tag{41}$$

where

$$\frac{\partial \mathbf{p}'(u;\eta_1)}{\partial \eta_1} = {}^0_A\mathbf{R} \begin{bmatrix} f_5(u) \\ 2f_3(u)\eta_1 \\ 3f_1(u)\eta_1^2 \end{bmatrix} + {}^0_B\mathbf{R} \begin{bmatrix} f_6(u) \\ 2f_4(u)\eta_1 \\ 3f_2(u)\eta_1^2 \end{bmatrix}.$$

Eq. (41) does not admit a closed form representation, so that its minimum value can be found by solving the following optimization problem

$$\min_{\gamma\in\Gamma} \left\{ \frac{\partial s_f(\eta_1)}{\partial \eta_1} \right\}$$

where $\gamma = \{{}^0_A\mathbf{R}, \kappa_A, \bar{\kappa}_A, \tau_A, {}^0_B\mathbf{R}, \kappa_B, \bar{\kappa}_B, \tau_B, \eta_1\}$ and Γ is a proper search space. In particular, ${}^0_A\mathbf{R}, {}^0_B\mathbf{R} \in SO(3)$, $\kappa_A, \kappa_B, \eta_1 \in \mathbb{R}^+$, and $\bar{\kappa}_A, \bar{\kappa}_B, \tau_A, \tau_B \in \mathbb{R}$. If the optimization problem returns a value greater than -1, then the algorithm converges independently from the interpolating conditions. The result does not depend on the direction and on the norm of $\mathbf{p}_B - \mathbf{p}_A$, so that such vector was assumed constant and equal to $\mathbf{p}_B - \mathbf{p}_A = [1\ 0\ 0]^T$.

The optimization and the subsequent analysis of the results evidenced some interesting properties. The problem is nonlinear and multimodal, so that it has been repeatedly solved by starting from randomly chosen points. The minimum cost index obtained over all the runs was equal to $-1.666 \cdot 10^{-2}$, i.e., it was much higher than -1. Similar cost indexes were found in other runs of the algorithm for alternative minimizers. A deeper analysis of the solutions revealed that the derivative of $s_f(\eta_1)$ can be negative for very particular configurations of the interpolating conditions and for values of η_1 close to $\|\mathbf{p}_B - \mathbf{p}_A\|$ (for the specific case, for η_1 close to 1). Furthermore, the derivative is generally positive over $\mathbb{R}^+$: when negative solutions are detected, they span over very narrow intervals of η_1. ■

Remark 1. The most common application for η^{3D}-splines concerns the creation of smooth junctions between linear segments. For example, this is typical planning case occurring for CNC machines. In such a framework, terms K_1 and K_2 are identically zero, so that (38) is banally satisfied if the following condition holds

$$\widetilde{\eta}_1 \geq \frac{\|\mathbf{p}_B - \mathbf{p}_A\|}{0,0926},$$

Table 4
Statistics concerning the computation of η_1 after i iterations. e and m are expressed in %, t is expressed in seconds.

i		avg	dev	min	max
1	e	0.095	0.058	$4.90 \cdot 10^{-5}$	0.288
	t	$4.07 \cdot 10^{-6}$	$2.68 \cdot 10^{-7}$	$4.00 \cdot 10^{-6}$	$5.00 \cdot 10^{-6}$
	m	0.473	0.171	0.099	0.940
2	e	0.032	0.028	$3.60 \cdot 10^{-8}$	0.152
	t	$8.54 \cdot 10^{-6}$	$5.08 \cdot 10^{-7}$	$8.00 \cdot 10^{-6}$	$1.00 \cdot 10^{-5}$
	m	0.507	0.188	0.099	0.940
3	e	0.012	0.014	$2.60 \cdot 10^{-11}$	0.084
	t	$1.30 \cdot 10^{-5}$	$2.54 \cdot 10^{-7}$	$1.20 \cdot 10^{-5}$	$1.50 \cdot 10^{-5}$
	m	0.516	0.193	0.099	0.940
4	e	0.005	0.007	$1.80 \cdot 10^{-14}$	0.047
	t	$1.72 \cdot 10^{-5}$	$4.58 \cdot 10^{-7}$	$1.70 \cdot 10^{-5}$	$2.00 \cdot 10^{-5}$
	m	0.519	0.195	0.099	0.940
5	e	0.002	0.003	0	0.027
	t	$2.14 \cdot 10^{-5}$	$5.79 \cdot 10^{-7}$	$2.10 \cdot 10^{-5}$	$2.40 \cdot 10^{-5}$
	m	0.521	0.196	0.099	0.940

so that the iterative algorithm certainly converges.

As previously asserted, the iterative procedure proposed for the selection of $\boldsymbol{\eta}$ typically converges in 2–3 iterations, so that computational times are compatible with the real-time requirement. In order to prove such assertion, the iterative procedure has been tested by considering the generation of junction curves between circular arcs. A set of 2250 test cases has been generated. The following constant interpolating conditions for the starting point of the $\boldsymbol{\eta}^{3D}$-splines have been assumed:

$$\mathbf{p}_A = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \ \mathbf{R}_A = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \ \kappa_A = 1, \ \bar{\kappa}_A = \tau_A = 0, \tag{42}$$

which are relative to a circular arc whose radius is equal to 1 m. Conversely, variable interpolating conditions have be assumed for the end-point. They have been generated as follows

$$\mathbf{p}_B = \begin{bmatrix} x_B \\ y_B \\ z_B \end{bmatrix}; \begin{cases} x_B & \in \{-0.3,\ 0,\ 0.3\} \\ y_B & \in \{0.3,\ 0.6,\ 0.9\} \\ z_B & \in \{0,\ 0.3\} \end{cases} \tag{43}$$

$$\mathbf{R}_B = R_x(\theta_2)R_z(\theta_1)\mathbf{R}_A;\ \theta_1, \theta_2 \in \left\{0,\ \frac{\pi}{4},\ \frac{\pi}{2},\ \frac{3\pi}{4},\ \pi\right\} \tag{44}$$

$$\kappa_B \in \{0.1,\ 0.5,\ 1,\ 2,\ 10\}, \tag{45}$$

$$\bar{\kappa}_B = \tau_B = 0, \tag{46}$$

where $R_k(\theta) \in SO(3)$ indicates a rotation around the k axis [in (44) $k \in \{x, z\}$]. As usual $\eta_3 = \eta_4 = \eta_5 = \eta_6 = 0$.

The algorithm converged in all the test cases. Table 4 shows some statistics concerning the obtained results. They are expressed as function on i, where i is the number of iterations considered. The statistics are relative the following benchmarks:

- Percentage difference e between η and s_f at the ith iteration $\left(e = \frac{|\eta_1 - s_f|}{s_f}\right)$;
- Computational time t at the ith iteration expressed in seconds;
- Percentage difference m between the initial and the final value of η_1 at the ith iteration $\left(m = \frac{\eta_1 - \|\mathbf{p}_A - \mathbf{p}_B\|}{\eta_1}\right)$.

Table 4 makes it possible to draw some conclusions. The algorithm can be reasonably stopped at the 3rd iteration, since the average values of e is close to 1% and the m terms are very similar to the ones achieved at the 4th iteration, i.e., the last cycle only marginally modifies the path shape. The same table further shows that, in many practical cases, two iterations are actually sufficient. The planning algorithm – executed on a single core of an Intel i7-1165G7 processor running at @2.80 GHz

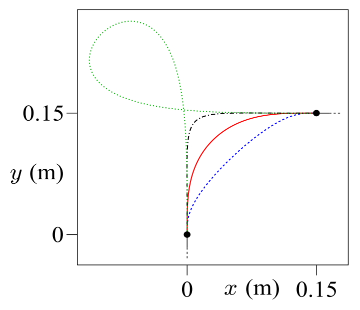

Fig. 6. Curve shapes obtained for: dashed blue line $\eta_1 = \eta_2 = \bar{s}_f/2$; solid red line $\eta_1 = \eta_2 = \bar{s}_f$; dash-dotted black line $\eta_1 = \eta_2 = 1.5\,\bar{s}_f$; dotted green line $\eta_1 = \eta_2 = 4\,\bar{s}_f$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

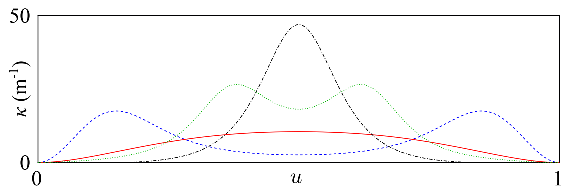

Fig. 7. Curvature profiles corresponding to the 4 curves shown in Fig. 6: dashed blue line $\eta_1 = \eta_2 = \bar{s}_f/2$; solid red line $\eta_1 = \eta_2 = \bar{s}_f$; dash-dotted black line $\eta_1 = \eta_2 = 1.5\,\bar{s}_f$; dotted green line $\eta_1 = \eta_2 = 4\,\bar{s}_f$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

– converges, on average, in $[8\ \ 12] \cdot 10^{-6}$ s, i.e., the execution time is compatible with a wide range of real-time applications. Computational times associated to any i are predictable, since they are characterized by a very low standard deviation: this is another important characteristic in real-time contexts.

An application of the proposed iterative procedure is shown in Fig. 6, in which the η^{3D}-splines have been used for the generation of smooth junctions between two linear segments. The following interpolating conditions have been used, directly derived from the endpoints of the linear segments: $\mathbf{p}_A = [0\,0\,0]^T$, $\mathbf{p}_B = [0.15\,0.15\,0]^T$, $\mathbf{t}_A = [0\,1\,0]^T$, $\mathbf{n}_A = [1\,0\,0]^T$, $\mathbf{b}_A = [0\,0\,-1]^T$, $\mathbf{t}_B = [0\,0\,-1]^T$, $\mathbf{n}_B = [0\,-1\,0]^T$, $\mathbf{b}_B = [0\,0\,-1]^T$, $\kappa_A = \kappa_B = \dot{\kappa}_A = \dot{\kappa}_B = \tau_A = \tau_B = 0$.

The 4 curves shown in Fig. 6 have been obtained by assuming that (28) applies. Furthermore, named $\bar{s}_f$ the path length obtained through the iterative procedure, η_1 and η_2 have been chosen as follows: (a) $\eta_1 = \eta_2 = \bar{s}_f/2$; (b) $\eta_1 = \eta_2 = \bar{s}_f$; (c) $\eta_1 = \eta_2 = 1.5\,\bar{s}_f$; (d) $\eta_1 = \eta_2 = 4\,\bar{s}_f$. Fig. 7 shows the corresponding curvatures and highlights that solution (b) returns the smallest values: $\kappa(u)$ smoothly increases from 0 up to an almost constant value – the central part of the curve is, approximately, a circular arc – and, then, it newly decreases to 0.

Another example is proposed in Fig. 8. It concerns 2 straight segments which are not coplanar. Apart from $\mathbf{p}_B := [0.15\,0.15\,0.15]^T$, the remaining interpolating conditions are the same used for the previous example and, similarly, the same 4 selection rules have been assumed for η_1 and η_2. As shown in Fig. 9, solution (b) is still the one with the smallest curvatures.

No further examples are presented for space reasons, but additional tests have shown that the proposed selection strategy generally returns curves with limited lengths and curvatures, and which avoid oscillatory behaviors. In general, for sufficiently high values of $\eta_1 = \eta_2$ – higher then the ones considered in the examples – maximum curvatures start decreasing, but curve lengths become excessive. The above mentioned characteristics allow one concluding that the strategy proposed for the selection of $\boldsymbol{\eta}$, while not yielding to optimal solutions, returns smooth $\mathcal{G}^3$ curves – curvatures and curvature derivatives are limited – of reasonable length – s_f is generally comparable with $\|\mathbf{p}_B - \mathbf{p}_A\|$.

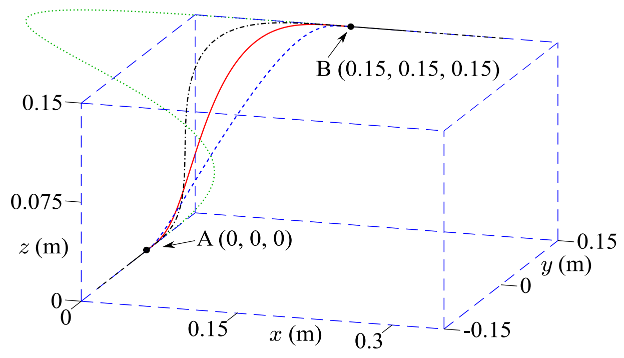

Fig. 8. Curve shapes obtained for: dashed blue line $\eta_1 = \eta_2 = \bar{s}_f/2$; solid red line $\eta_1 = \eta_2 = \bar{s}_f$; dash-dotted black line $\eta_1 = \eta_2 = 1.5\,\bar{s}_f$; dotted green line $\eta_1 = \eta_2 = 4\,\bar{s}_f$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

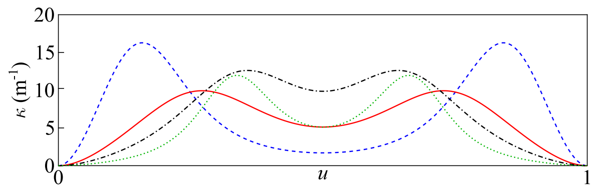

Fig. 9. Curvature profiles corresponding to the 4 curves shown in Fig. 8: dashed blue line $\eta_1 = \eta_2 = \bar{s}_f/2$; solid red line $\eta_1 = \eta_2 = \bar{s}_f$; dash-dotted black line $\eta_1 = \eta_2 = 1.5\,\bar{s}_f$; dotted green line $\eta_1 = \eta_2 = 4\,\bar{s}_f$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

This result is achieved at a negligible computational cost, since the η^{3D}-splines coefficients are immediately obtained from (19)–(26). As already mentioned, if specific optimal conditions were to be satisfied, $\boldsymbol{\eta}$ should be selected through nonlinear programming algorithms.

In next Section 5 the η^{3D}-splines are experimentally tested with the aid of an industrial manipulator.

5. Experimental validation

η^{3D}-splines have been embedded in the path planner of an industrial manipulator and, subsequently, they have been experimentally tested by generating some $\mathcal{G}^3$ composite paths. To this purpose, a Comau Smart SiX 6-1.4 manipulator was used. The manipulator can be remotely controlled by means of an external Linux PC whose kernel was patched with the Real Time Application Interface (RTAI) software [45]. The communication between PC and robot controller exploits a real-time Ethernet connection.

As early mentioned, η^{3D}-splines can also emulate, exactly or with a good approximation, many path primitives commonly used by conventional planners. For this reason, a Cartesian planner, entirely based on the η^{3D}-splines, was implemented and exploited for the generation of a composite $\mathcal{G}^3$ path.

The trajectory shown in Fig. 10 has been specifically synthesized to this purpose. It is made of a set of curves whose interpolating conditions are assigned so as to generate some common path primitives. In particular, the $\mathcal{G}^3$ path contains 3 straight lines (see red segments 1, 3 and 130), 3 adjacent circular arcs (see the green segments from 5 to 7), a helical curve (see the black segments from 9 to 67), and a conic spiral (see the orange segments from 69 to 127). Such curves are joined by means of generic η^{3D}-spline profiles so as to guarantee the overall $\mathcal{G}^3$ continuity of the composite path (see cyan segments 2, 4, 8, 68, 128, 129, 131). Black dots highlight the segments end-points. For all the curves, vector $\boldsymbol{\eta}$ was always selected according to (27) and (28).

The maximum emulation error $(2.7 \cdot 10^{-5}$ m) was detected for curves from 5 to 7: for many actual robotic applications such value

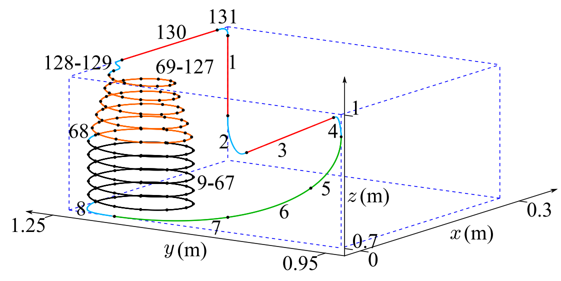

Fig. 10. The composite $\mathcal{G}^3$ path used for Experiment 2.

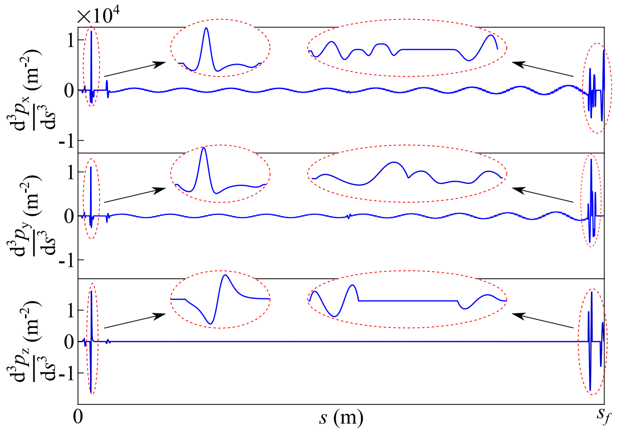

Fig. 11. The 3 components of $(d^3\mathbf{p})/(ds^3)$ are continuous functions.

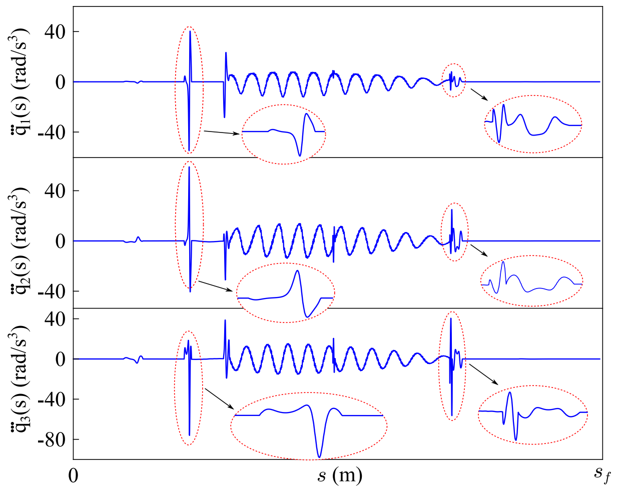

Fig. 12. For the composite $\mathcal{G}^3$ path, the resulting jerk profiles of the first 3 joints are continuous.

is acceptable and, according to the discussion in Section 4, it can be further reduced by shortening the arcs lengths. Fig. 11 shows that, as desired, $(d^3\mathbf{p})/(ds^3)$ is continuous over the entire composite path and, consequently, the joint jerks shown in Fig. 12 are continuous as well.

The accompanying video shows the execution of the path for a longitudinal speed equal to 0.1 ms^{-1}: the $\mathcal{G}^3$ continuity allows a very smooth motion.

6. Conclusions

η^{3D}-splines are an extremely flexible path planning primitive. As shown in the paper, it can easily emulate other planning primitives and its parameters can be obtained, at a negligible computational cost, directly from the assigned interpolating conditions. These properties, combined with the $\mathcal{G}^3$ geometric continuity that the η^{3D}-splines can guarantee, make it possible to smartly create in real time smooth paths of considerable complexity, by only using a single planning primitive.

The research activity is currently focused on two open problems. On the one hand, the effort is posed on the investigation of solutions which exploit the available degrees of freedom for the generation of smoother path, so as to further reduce the errors in the joint space; on the other, η^{3D}-splines are going to be integrated with an additional primitive, so as to allow the generation of motions which also account for the tool orientations.

CRediT authorship contribution statement

Andrea Tagliavini: Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing. **Corrado Guarino Lo Bianco:** Conceptualization, Methodology, Formal analysis, Writing - original draft, Writing - review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A

The coefficients of a generic 7th order spline $\mathbf{p}(u)$ satisfying the interpolating conditions specified in Table 1 will be indicated as $\tilde{\chi}_i, i = 0, 1, \ldots, 7$, while the coefficients of an η^{3D}-spline $\mathbf{p}_\eta(u)$ satisfying the same interpolating conditions will be indicated as χ_i. In order to prove Proposition 1, it is necessary to demonstrate that, through a proper choice of vector $\boldsymbol{\eta}$, it is always possible to obtain $\mathbf{p}_\eta(u) = \mathbf{p}(u)$ or, equivalently, $\chi_i = \tilde{\chi}_i, i = 0, 1, \ldots, 7$.

Interpolating condition $\mathbf{p}(0) = \mathbf{p}_A$ is evidently satisfied for a curve like (18) if $\tilde{\chi}_0 = \mathbf{p}_A$. According to (19), the first coefficient of $\mathbf{p}_\eta(u)$ is given by $\chi_0 = \mathbf{p}_A$, so that, evidently, $\chi_0 = \tilde{\chi}_0$.

By virtue of (9), the following condition applies for any generic curve

$$\mathbf{p}'(u) = \mathbf{t}(u)\left\|\mathbf{p}'(u)\right\|. \tag{A.1}$$

If the same curve satisfies initial condition $\mathbf{t}(0) = \mathbf{t}_A$ then, for $u = 0$, (A.1) can be written as follows

$$\mathbf{p}'(0) = \mathbf{t}_A\left\|\mathbf{p}'(0)\right\|. \tag{A.2}$$

By differentiating (18), it is possible to evince that the first derivative of $\mathbf{p}(u)$, computed for $u = 0$, is given by $\mathbf{p}'(0) = \tilde{\chi}_1$, so that from (A.2) it descends that

$$\mathbf{p}'(0) = \tilde{\chi}_1 = \mathbf{t}_A\left\|\mathbf{p}'(0)\right\|. \tag{A.3}$$

Evidently, coefficient χ_1 of $\mathbf{p}_\eta(u)$ is given by (20). By assuming

$$\eta_1 = \left\|\mathbf{p}'(0)\right\| \tag{A.4}$$

condition $\chi_1 = \tilde{\chi}_1$ is satisfied.

Any orthogonal Frenet frame is a base for the 3D Cartesian space, so that the following expression is certainly true

$$\mathbf{p}''(0) = \alpha\mathbf{t}_A + \beta\mathbf{n}_A + \gamma\mathbf{b}_A, \tag{A.5}$$

where α, β, and γ are proper scalar.

Bearing in mind (A.2) and (A.5), if a generic curve satisfies condition $\mathbf{n}(0) = \mathbf{n}_A$, then (11), after a few algebraic manipulations, can be written as follows

$$\mathbf{n}_A = \frac{[\mathbf{p}'(0) \times \mathbf{p}''(0)] \times \mathbf{p}'(0)}{\|\mathbf{p}'(0) \times \mathbf{p}''(0)\|\,\|\mathbf{p}'(0)\|} = \frac{\beta\mathbf{n}_A + \gamma\mathbf{b}_A}{\|\beta\mathbf{b}_A - \gamma\mathbf{n}_A\|}.$$

Evidently, such expression is true only if

$$\gamma = 0. \tag{A.6}$$

If the curve also fulfills $\kappa(0) = \kappa_A$, then (12) – together with (A.2), (A.5), and (A.6) – allows one writing the following expression

$$\kappa_A = \frac{\|\mathbf{p}'(0) \times \mathbf{p}''(0)\|}{\|\mathbf{p}'(0)\|^3} = \frac{\|\mathbf{p}'(0)\| \, \|\mathbf{t}_A \times (\alpha \mathbf{t}_A + \beta \mathbf{n}_A)\|}{\|\mathbf{p}'(0)\|^3} = \frac{\beta}{\|\mathbf{p}'(0)\|^2}$$

and, consequently,

$$\beta = \kappa_A \left\|\mathbf{p}'(0)\right\|^2. \tag{A.7}$$

By virtue of (A.6) and (A.7), for any curve which satisfies the conditions in Table 1, (A.5) assumes the following form

$$\mathbf{p}''(0) = \alpha \mathbf{t}_A + \kappa_A \left\|\mathbf{p}'(0)\right\|^2 \mathbf{n}_A. \tag{A.8}$$

On the other side, for a generic polynomial curve $\mathbf{p}(u)$ like (18), the following expression holds

$$\mathbf{p}''(0) = 2\widetilde{\chi}_2, \tag{A.9}$$

so that the interpolating conditions are satisfied by imposing

$$\widetilde{\chi}_2 = \frac{1}{2}(\alpha \mathbf{t}_A + \kappa_A \left\|\mathbf{p}'(0)\right\|^2 \mathbf{n}_A). \tag{A.10}$$

From (21), it descends that the same result can be achieved for $\mathbf{p}_{\boldsymbol{\eta}}(u)$ by acting on $\boldsymbol{\eta}$. In particular, condition $\chi_2 = \widetilde{\chi}_2$ is satisfied by imposing (A.4) and by further assigning

$$\eta_3 = \alpha. \tag{A.11}$$

A similar reasoning can be used for χ_3. Vector $\mathbf{p}'''(0)$ can be expressed through its components in the Frenet frame

$$\mathbf{p}'''(0) = \hat{\alpha} \mathbf{t}_A + \hat{\beta} \mathbf{n}_A + \hat{\gamma} \mathbf{b}_A \tag{A.12}$$

where $\hat{\alpha}$, $\hat{\beta}$, and $\hat{\gamma}$ are proper scalars. If a curve fulfills $\tau(0) = \tau_A$, by virtue of (16) the following condition holds

$$\tau_A = \frac{[\mathbf{p}'(0) \times \mathbf{p}''(0)] \cdot \mathbf{p}'''(0)}{\|\mathbf{p}'(0) \times \mathbf{p}''(0)\|^2}. \tag{A.13}$$

By further considering (A.2), (A.8), and (A.12), and after a few algebraic manipulations, (A.13) can be rearranged as follows

$$\tau_A = \frac{\hat{\gamma}}{\|\mathbf{p}'(0)\|^3 \kappa_A},$$

which implies that the interpolating conditions are fulfilled if the following expression is satisfied

$$\hat{\gamma} = \tau_A \left\|\mathbf{p}'(0)\right\|^3 \kappa_A. \tag{A.14}$$

By imposing $\frac{d\kappa}{ds}(0) = \bar{\kappa}_A$ to (17), evaluated for $u = 0$, one can write

$$\bar{\kappa}_A = \mathbf{b}_A \cdot \frac{\mathbf{p}'(0) \times \mathbf{p}'''(0)}{\|\mathbf{p}'(0)\|^4} - 3\kappa_A \frac{\mathbf{p}'(0) \cdot \mathbf{p}''(0)}{\|\mathbf{p}'(0)\|^3}. \tag{A.15}$$

By considering (A.2), (A.8), and (A.12), (A.15) simplifies as follows

$$\bar{\kappa}_A = \frac{\hat{\beta}}{\|\mathbf{p}'(0)\|^3} - 3\kappa_A \frac{\alpha}{\|\mathbf{p}'(0)\|^2},$$

so that, necessarily, $\hat{\beta}$ must assume the following structure

$$\hat{\beta} = \bar{\kappa}_A \left\|\mathbf{p}'(0)\right\|^3 + 3\kappa_A \alpha \left\|\mathbf{p}'(0)\right\|. \tag{A.16}$$

Expressions (A.14) and (A.16) are valid for generic curves, but can be specialized for polynomial functions. In particular, from (18) it descends that

$$\mathbf{p}'''(0) = 6\widetilde{\chi}_3, \tag{A.17}$$

so that, bearing in mind (A.12), (A.14), and (A.16), the following expression for $\widetilde{\chi}_3$ is obtained

$$\widetilde{\chi}_3 = \frac{1}{6}\hat{\alpha}\mathbf{t}_A + \left(\frac{1}{6}\bar{\kappa}_A \left\|\mathbf{p}'(0)\right\|^3 + \frac{1}{2}\kappa_A \alpha \left\|\mathbf{p}'(0)\right\|\right)\mathbf{n}_A + \frac{1}{6}\tau_A \left\|\mathbf{p}'(0)\right\|^3 \kappa_A \mathbf{b}_A. \tag{A.18}$$

The same value can be assumed by an η^{3D}-spline by assigning $\eta_5 = \hat{\alpha}/6$, and by recalling that (A.4) and (A.11) simultaneously hold: condition $\chi_3 = \widetilde{\chi}_3$ is certainly satisfied.

An analogous procedure can be used to demonstrate that $\chi_i = \widetilde{\chi}_i$, for $i = 4, 5, 6, 7$. To this purpose, it is necessary to consider the interpolating conditions at the curve endpoint ($u = 1$). The process requires much more algebraic manipulations, so that, for space reasons, it is omitted.

It is worth highlighting that the demonstration also allows one asserting that the choice of parameters $\boldsymbol{\eta}$ never affect interpolating conditions, which are always satisfied independently from the values it assumes: this property is exploited in the paper in order to obtain curves with the desired shape.

Appendix B. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.rcim.2021.102203.

References

[1] L. Dubins, On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents, Amer. J. Math. 79 (1957) 497–517, http://dx.doi.org/10.2307/2372560.

[2] J. Reeds, R. Shepp, Optimal paths for a car that goes both forward and backward, Pac. J. Math. 145 (2) (1990) 367–393, http://dx.doi.org/10.2140/pjm.1990.145.367.

[3] J.-D. Boissonnat, A. Cérézo, J. Leblond, Shortest paths of bounded curvature in the plane, in: Proc. of the 1992 IEEE Int. Conf. Robot. and Autom., Nice, France, 1992, pp. 2315–2320, http://dx.doi.org/10.1007/BF01258291.

[4] P. Souères, J.-P. Laumond, Shortest paths synthesis for a car-like robot, IEEE Trans. Automat. Control 41 (5) (1996) 672–688, http://dx.doi.org/10.1109/9.489204.

[5] W. Yao, N. Qi, C. Yue, N. Wan, Curvature-bounded lengthening and shortening for restricted vehicle path planning, IEEE Trans. Autom. Sci. Eng. 17 (1) (2020) 15–28, http://dx.doi.org/10.1109/TASE.2019.2916855.

[6] W. Nelson, Continuous-curvature paths for autonomous vehicles, in: IEEE Int. Conf. Robot. and Autom., Vol. 3, ICRA89, Scottsdale, AZ, 1989, pp. 1260–1264, http://dx.doi.org/10.1109/ROBOT.1989.100153.

[7] W. Nelson, Continuous steering-function control of robot carts, IEEE Trans. Ind. Electron. 36 (3) (1989) 330–337, http://dx.doi.org/10.1109/41.31495.

[8] Y. Kanayama, B. Hartman, Smooth local path planning for autonomous vehicles, in: Proc. IEEE Int. Conf. Robot. and Autom., Vol. 3, ICRA89, Scottsdale, AZ (US), 1989, pp. 1265–1270, http://dx.doi.org/10.1109/ROBOT.1989.100154.

[9] S. Fleury, P. Souères, J. Laumond, R. Chatila, Primitives for smoothing paths of mobile robots, in: Proc. IEEE Int. Conf. Robot. and Autom., Atlanta, GA, 1993, pp. 832–839, http://dx.doi.org/10.1109/ROBOT.1993.292080.

[10] J.-D. Boissonnat, A. Cérézo, J. Leblond, A Note on Shortest Paths in the Plane Subject to a Constraint on the Derivative of the Curvature, Tech. Rep. 2160, INRIA, Rocquencourt, France, 1994.

[11] J. Reuter, Mobile robots trajectories with continuously differentiable curvature: an optimal control approach, in: Proc. 1998 IEEE/RSJ Int. Conf. Intell. Robots and Syst., Vol. 1, Victoria, B.C., Canada, 1998, pp. 38–43, http://dx.doi.org/10.1109/IROS.1998.724593.

[12] G. Yang, B. Choi, Smooth trajectory planning along bezier curve for mobile robots with velocity constraints, Int. J. Control Autom. 6 (2013) 225–234, http://dx.doi.org/10.1007/978-3-642-35485-4_18.

[13] L. Zhang, L. Sun, S. Zhang, J. Liu, Trajectory planning for an indoor mobile robot using quintic Bezier curves, in: 2015 IEEE Int. Conf. on Robot. and Biom., ROBIO, 2015, http://dx.doi.org/10.1109/ROBIO.2015.7418860.

[14] J. Kim, Trajectory generation of a two-wheeled mobile robot in an uncertain environment, IEEE Trans. Ind. Electron. 67 (7) (2020) 5586–5594, http://dx.doi.org/10.1109/TIE.2019.2931506.

[15] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, I. Soderkvist, Planning smooth and obstacle-avoiding B-spline paths for autonomous mining vehicles, IEEE Trans. Autom. Sci. Eng. 7 (1) (2010) 167–172, http://dx.doi.org/10.1109/TASE.2009.2015886.

[16] A. Piazzi, C. Guarino Lo Bianco, Quintic G^2-splines for trajectory planning of autonomous vehicles, in: Procs IEEE Int. Veh. Symp., Dearborn (MI), USA, 2000, pp. 198–203, http://dx.doi.org/10.1109/IVS.2000.898341.

[17] C. Guarino Lo Bianco, A. Piazzi, M. Romano, Smooth motion generation for unicycle mobile robots via dynamic path inversion, IEEE Trans. Robot. 20 (5) (2004) 884–891, http://dx.doi.org/10.1109/TRO.2004.832827.

[18] A. Piazzi, M. Romano, C. Guarino Lo Bianco, G^3-splines for the path planning of wheeled mobile robots, in: Proc. 2003 Eur. Contr. Conf., ECC 2003, Cambridge, UK, 2003, http://dx.doi.org/10.23919/ECC.2003.7085234.

[19] A. Piazzi, C. Guarino Lo Bianco, M. Romano, $\boldsymbol{\eta}^3$-splines for the smooth path generation of wheeled mobile robots, IEEE Trans. Robot. 23 (5) (2007) 1089–1095, http://dx.doi.org/10.1109/TRO.2007.903816.

[20] F. Ghilardelli, G. Lini, A. Piazzi, Path generation using $\boldsymbol{\eta}^4$-splines for a truck and trailer vehicle, IEEE Trans. Autom. Sci. Eng. 11 (1) (2014) 187–203, http://dx.doi.org/10.1109/TASE.2013.2266962.

[21] W. Ding, W. Gao, K. Wang, S. Shen, An efficient B-spline-based kinodynamic replanning framework for quadrotors, IEEE Trans. Robot. 35 (6) (2019) 1287–1306, http://dx.doi.org/10.1109/TRO.2019.2926390.

[22] S. Zhang, L. Sun, Z. Chen, X. Lu, J. Liu, Smooth path planning for a home service robot using η^3-splines, in: IEEE Int. Conf. on Rob. and Biom., ROBIO, 2014, pp. 1910–1915, http://dx.doi.org/10.1109/ROBIO.2014.7090615.

[23] Q.-B. Xiao, M. Wan, Y. Liu, X.-B. Qin, W.-H. Zhang, Space corner smoothing of CNC machine tools through developing 3D general clothoid, Robot. Comput.-Integr. Manuf. 64 (2020) 101949, http://dx.doi.org/10.1016/j.rcim.2020.101949.

[24] X. Huang, F. Zhao, T. Tao, X. Mei, A newly developed corner smoothing methodology based on clothoid splines for high speed machine tools, Robot. Comput.-Integr. Manuf. 70 (2021) 102106, http://dx.doi.org/10.1016/j.rcim.2020.102106.

[25] W. Wang, C. Hu, K. Zhou, S. He, L. Zhu, Local asymmetrical corner trajectory smoothing with bidirectional planning and adjusting algorithm for CNC machining, Robot. Comput.-Integr. Manuf. 68 (2021) 102058, http://dx.doi.org/10.1016/j.rcim.2020.102058.

[26] D. Simon, C. Isik, Optimal trigonometric robot joint trajectories, Robotica 9 (4) (1991) 379–386, http://dx.doi.org/10.1017/S0263574700000552.

[27] A. Gasparetto, V. Zanotto, A new method for smooth trajectory planning of robot manipulators, Mech. Mach. Theory 42 (2007) 455–471, http://dx.doi.org/10.1016/j.mechmachtheory.2006.04.002.

[28] S. Kucuk, Maximal dexterous trajectory generation and cubic spline optimization for fully planar parallel manipulators, Comput. Electr. Eng. 56 (2016) 634–647, http://dx.doi.org/10.1016/j.compeleceng.2016.07.012.

[29] S. Kucuk, Optimal trajectory generation algorithm for serial and parallel manipulators, Robot. Comput.-Integr. Manuf. 48 (2017) 219–232, http://dx.doi.org/10.1016/j.rcim.2017.04.006.

[30] S. Patil, J. Pan, P. Abbeel, K. Goldberg, Planning curvature and torsion constrained ribbons in 3D with application to intracavitary brachytherapy, IEEE Trans. Autom. Sci. Eng. 12 (4) (2015) 1332–1345, http://dx.doi.org/10.1109/TASE.2015.2475121.

[31] C. Dai, S. Lefebvre, K. Yu, J.M.P. Geraedts, C.C.L. Wang, Planning jerk-optimized trajectory with discrete time constraints for redundant robots, IEEE Trans. Autom. Sci. Eng. (2020) 1–14, http://dx.doi.org/10.1109/TASE.2020.2974771.

[32] Y.-A. Lu, K. Tang, C.-Y. Wang, Collision-free and smooth joint motion planning for six-axis industrial robots by redundancy optimization, Robot. Comput.-Integr. Manuf. 68 (2021) 102091, http://dx.doi.org/10.1016/j.rcim.2020.102091.

[33] Z. Xin, H. Zhao, S. Wan, l. Xiangfei, H. Ding, An analytical decoupled corner smoothing method for five-axis linear tool paths, IEEE Access 7 (2019) 22763–22772, http://dx.doi.org/10.1109/ACCESS.2019.2898703.

[34] Q. Bi, J. Shi, Y. Wang, L. Zhu, H. Ding, Analytical curvature-continuous dual-Bezier corner transition for five-axis linear tool path, Int. J. Mach. Tools Manuf. 91 (2015) 96–108, http://dx.doi.org/10.1016/j.ijmachtools.2015.02.002.

[35] J. Yang, A. Yuen, An analytical local corner smoothing algorithm for five-axis CNC machining, Int. J. Mach. Tools Manuf. 123 (2017) 22–35, http://dx.doi.org/10.1016/j.ijmachtools.2017.07.007.

[36] A. Pekarovskiy, T. Nierhoff, S. Hirche, M. Buss, Dynamically consistent online adaptation of fast motions for robotic manipulators, IEEE Trans. Robot. 34 (1) (2018) 166–182, http://dx.doi.org/10.1109/TRO.2017.2765666.

[37] J. Kim, E.A. Croft, Online near time-optimal trajectory planning for industrial robots, Robot. Comput.-Integr. Manuf. 58 (2019) 158–171, http://dx.doi.org/10.1016/j.rcim.2019.02.009.

[38] T. Hsu, J. Liu, Design of smooth path based on the conversion between η^3 spline and Bezier curve, in: 2020 Amer. Contr. Conf., ACC, 2020, pp. 3230–3235, http://dx.doi.org/10.23919/ACC45564.2020.9147319.

[39] K. Kant, S. Zucker, Toward efficient trajectory planning: The path-velocity decomposition, Int. J. Robot. Res. 5 (3) (1986) 72–89, http://dx.doi.org/10.1177/027836498600500304.

[40] E. Kreyszig, Differential Geometry, Dover Publications, New York, 1991, http://dx.doi.org/10.3138/9781487589455.

[41] Y. Fang, J. Hu, W. Liu, Q. Shao, J. Qi, Y. Peng, Smooth and time-optimal S-curve trajectory planning for automated robots and machines, Mech. Mach. Theory 137 (2019) 127–153, http://dx.doi.org/10.1016/j.mechmachtheory.2019.03.019.

[42] C. Guarino Lo Bianco, O. Gerelli, Generation of paths with minimum curvature derivative with η^3-splines, IEEE Trans. Autom. Sci. Eng. 7 (2) (2010) 249–256, http://dx.doi.org/10.1109/TASE.2009.2023206.

[43] M. Brezak, I. Petrović, Real-time approximation of clothoids with bounded error for path planning applications, IEEE Trans. Robot. 30 (2) (2014) 507–515, http://dx.doi.org/10.1109/TRO.2013.2283928.

[44] Y. Chen, Y. Cai, J. Zheng, D. Thalmann, Accurate and efficient approximation of clothoids using Bézier curves for path planning, IEEE Trans. Robot. 33 (5) (2017) 1242–1247, http://dx.doi.org/10.1109/TRO.2017.2699670.

[45] L. Dozio, P. Mantegazza, Linux real time application interface (RTAI) in low cost high performance motion control, in: Motion Control 2003, a Conference of ANIPLA National Italian Association for Automation, Milano, Italy, 2003, http://hdl.handle.net/11311/267537.