2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636612

B-spline path planner for safe navigation of mobile robots

Ngoc Thinh Nguyen[1], Lars Schilling[1], Michael Sebastian Angern[2], Heiko Hamann[3], Floris Ernst[1], Georg Schildbach[2]

***Abstract*— We propose a 2D path planning algorithm in a non-convex workspace defined as a sequence of connected convex polytopes. The reference path is parameterized as a B-spline curve, which is guaranteed to entirely remain within the workspace by exploiting the local convexity property and by formulating linear constraints on the control points of the B-spline. The novelties of the paper lie in the use of the equivalent Bézier representation of the B-spline curve, which significantly reduces the conservatism in the local convexity bound and in the integration of these constraints into a convex quadratic optimization problem, which minimizes the curve length. The algorithm is successfully validated in both simulations and experiments, by providing obstacle-free reference paths on real occupancy grid maps obtained from the laser scan data of a mobile robot platform.**

***Index Terms*— Path planner, B-spline, Bézier, Convexity.**

I. Introduction

Over the last decade, research and industrial communities have shown increasing interest in autonomous mobile robots. For safe navigation, the robot is equipped with different types of sensors, such as cameras or LIDAR (Light Detection and Ranging) sensors, in order to incrementally build and/or improve a map of its environment, and to localize itself in that map, via SLAM (simultaneous localization and mapping). Then a motion planning strategy is applied in order to navigate the robot from an initial position to a goal position, while moving strictly inside obstacle-free regions [1]–[3]. To this end, numerous path-planning approaches have been proposed in the robotics literature, and successfully applied in various problem settings. The most popular ones are different incarnations based on the A* and RRT (Rapidly exploring Random Tree) algorithms [4]. Recently, approaches using numerical optimization have gained increasing popularity [1]–[3]. A common drawback of all of these methods is that they rely on discrete spatial data (e.g. grid map, discrete dynamics) for path planning and collision checking, which necessarily leads to a trade-off between computational tractability (coarse mesh) and accuracy (fine mesh). Hence, a different line of research has considered continuous interpolating curves with geometric constraints for obstacle-free path planning. A common type of curve used for this is B-splines, which have been successfully employed for path planning problems of a wide range of systems such as autonomous passenger vehicles [5], for unmanned heavy-duty vehicles in mines [6], and for autonomous drones [7], [8]. This is thanks to their well-known property of being locally bounded by a convex polygon defined through their control points, called a *control polygon* [7]–[10]. Therefore, by choosing all control polygons to lie entirely within obstacle-free areas of the map and enforcing the control points to be contained in the control polygons, the resulting B-spline path is guaranteed to be collision-free; see Fig. 1 for an illustration.

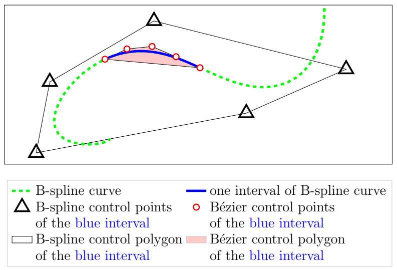

B-spline curve
B-spline control points of the blue interval
B-spline control polygon of the blue interval
one interval of B-spline curve
Bézier control points of the blue interval
Bézier control polygon of the blue interval

Fig. 1: Comparison between the local B-spline control boundary (black-plotted polytope with triangle vertices) versus the Bézier boundary (red-filled polytope with circle vertices) of the blue section of a fourth-order B-spline curve.

However, the approach is still conservative as the B-splines do not actually fill the entire control polygons. It is known from the Computer-Aided-Design (CAD) community since the 1980s that there exist much tighter bounds of B-spline curves (w.r.t. the standard control polygon), namely the control polygon of an equivalent Bézier curve [11], as illustrated in Fig. 1. This fact is used in this paper in order to significantly reduce the conservatism compared to previous B-spline based path planners [6]–[9] and helps to bring the following novelties to the state of the art:

- We propose the constraints for a B-spline curve to stay inside a sequence of connected polytopes using the equivalent Bézier representation (in comparison with the direct use of B-spline control points [6]–[10]). This provides a better approximation of the curve's shape (cf. Fig. 1), more choices of the control points (i.e. serving as the decision variables) and further allows to formulate the constraints with clear interpretation (i.e. enforcing each interval inside each polytope).
- By first defining a sequence of polytopes to move and then constraining a B-spline curve to fully stay within this sequence, we can formulate the final path planning problem as a convex optimization problem (with quadratic cost and linear constraints). This can

[1] University of Luebeck (UzL), Institute for Robotics and Cognitive Systems, Luebeck, Germany.
{nguyen,schilling,ernst}@rob.uni-luebeck.de.
[2] UzL, Institute for Electrical Engineering in Medicine, Luebeck, Germany.
{m.angern,georg.schildbach}@uni-luebeck.de.
[3] UzL, Institute of Computer Engineering, Luebeck, Germany.
hamann@iti.uni-luebeck.de.

be solved efficiently by standard convex solvers which are also readily available on embedded platforms with limited computing capability.

- The algorithm is validated using a real mobile robot platform. The implementation includes: simplifying the surrounding environment (e.g. obtained from LIDAR data or the occupancy grid map) into polygon region, adding safety offset, decomposing the region into convex polytopes, finding a suitable sequence of polytopes and finally planning the B-spline path.

II. Preliminaries

A. Problem statement

The considered problem is to navigate between two points in a safe region, which is assumed to be a non-convex, connected region consisting of an ordered list of $q \geq 2$ connected polytopes $\{S_1, \dots, S_q\}$ where S_1 contains the starting point P_s and S_q contains the goal point P_f:

$$P_s \in S_1, \quad P_f \in S_q. \tag{1}$$

Our main concern is to generate a smooth geometric path $p(t)$ defined in terms of a parameter t (note that t can represent path length, pseudo-time increment, etc. depending on specific circumstances) as follows:

$$p(t) : [t_s, t_f] \to \mathbb{R}^2, \tag{2}$$

which connects two points P_s and P_f, both in $\mathbb{R}^2$, i.e.:

$$p(t_s) = P_s, \quad p(t_f) = P_f. \tag{3}$$

The path $p(t)$ is required to continuously stay in the region $\mathcal{S}$ which is defined as the union of the q connected polytopes S_i, $i \in \{1, \dots, q\}$:

$$p(t) \in \mathcal{S} \triangleq S_1 \cup S_2 \cup \dots \cup S_q, \ \forall t \in [t_s, t_f], \tag{4}$$

in which, two consecutive polytopes S_i, S_{i+1} share exactly one edge E_i connecting two common vertices (cf. Fig. 2):

$$E_i = S_i \cap S_{i+1}, \ \forall i \in \{1, \dots, q-1\}. \tag{5}$$

B. Transition zones of two connected polytopes

In order to conveniently constrain the path in between the aforementioned connected polytopes (i.e. each pair has only one sharing edge), we define the so-called *transition zone* which, defined for two connected polytopes, represents a subset in one polytope whose union with the other connected polytope is convex (cf. Figure 2).

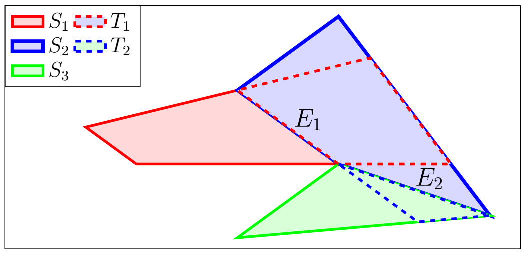


Fig. 2: Three connected polytopes and their transition zones according to Definition 1.

Definition 1 (Transition zone): Given two connected polytopes S_i and S_{i+1} as used in (5), we define the transition zone $T_i \subseteq S_{i+1}$ as follows:

$$T_i = S_{i+1} \cap (S_i | E_i), \tag{6}$$

in which E_i is the common edge as defined in (5), and the operation $(S_i|E_i)$ gives the (possibly open) polytope formed by the half-space representation of S_i without the constraint corresponding to the edge E_i. □

Definition 2 (Extended polytope): We define $S_{i,i+1}$ as the extension of the polytope S_i towards the polytope S_{i+1}:

$$S_{i,i+1} = S_i \cup T_i, \tag{7}$$

with T_i the transition zone defined as in (6). Since we are considering only q polytopes $\{S_1, \dots, S_q\}$ as in (4), we directly define $S_{q,q+1} \triangleq S_q$. Note that any extended polytope $S_{i,i+1}$ as defined in (7) is convex. □

Regarding our path generation problem defined in (4), one of the most promising solutions which allows us to validate both *continuous* and *geometrical* constraints (4) is to employ B-spline curves for constructing the reference path [8], [10], [12]. Therefore, the next section will give a brief definition of general B-spline curves and their local convex boundary formed by the equivalent Bézier curves of the same degree.

III. B-spline and equivalent Bézier curves

A. Definition and properties of B-spline curves

In this section, we recapitulate the definition and the standard convexity properties of B-spline curves [9], [10], [12], [13] using notations adapted from [10], [12].

A B-spline curve $z(t) : [t_s, \ t_f] \to \mathbb{R}^2$ of degree d is constructed by using n control points $P_i \in \mathbb{R}^2$ ($i \in \{1, \dots, n\}$) with $n \geq d+1$ as follows:

$$z(t) = \sum_{i=1}^{n} P_i B_{i,d,\xi}(t) = \boldsymbol{P}\boldsymbol{B}_{d,\xi}(t), \ t \in [t_s, \ t_f], \tag{8}$$

in which, $\boldsymbol{P} \triangleq [P_1 \ \cdots \ P_n] \in \mathbb{R}^{2 \times n}$ and $\boldsymbol{B}_{d,\xi}(t) \triangleq [B_{1,d,\xi}(t) \ \dots \ B_{n,d,\xi}(t)]^\top : \mathbb{R} \to \mathbb{R}^n$. The so-called *knot vector* ξ is a non-decreasing sequence of $(n+d+1)$ time instants:

$$\xi = \{\tau_1 \leq \tau_2 \leq \dots \leq \tau_{n+d+1}\}, \tag{9}$$

and $B_{i,d,\xi}(t)$ is the i^{th} B-spline basis function of degree d constructed using the knot vector ξ from (9) and is recursively defined as follows:

$$B_{i,0,\xi}(t) = \begin{cases} 1, & t \in [\tau_i, \tau_{i+1}), \\ 0, & \text{otherwise}, \end{cases} \tag{10}$$

$$B_{i,d,\xi}(t) = \frac{t - \tau_i}{\tau_{i+d} - \tau_i} B_{i,d-1,\xi}(t) + \frac{\tau_{i+d+1} - t}{\tau_{i+d+1} - \tau_{i+1}} B_{i+1,d-1,\xi}(t). \tag{11}$$

The aforementioned time instants τ_j ($j \in \{1, \dots, n+d+1\}$) of ξ can be chosen following either periodic uniform, open-uniform or non-uniform methods [10]. In this work, we employ the open-uniform knot vector ξ in which the time instants τ_j ($j \in \{1, \ldots, n+d+1\}$) from (9) are *clamped* and *uniformly* distributed by:

$$\tau_j = \begin{cases} t_s, & 1 \leq j \leq d, \\ t_s + (j-d-1)\Delta, & d+1 \leq j \leq n+1, \\ t_f, & n+2 \leq j \leq n+d+1, \end{cases} \tag{12}$$

with $\Delta = (t_f - t_s)/(n-d)$. An important remark is that by distributing ξ as in (12), the B-spline curve $z(t)$ from (8) actually varies in $(n-d)$ equal intervals among which the j^{th} interval (with $j \in \{1, \ldots, n-d\}$) is given by:

$$[\tau_{j+d}, \tau_{j+d+1}) \triangleq [t_s + (j-1)\Delta, t_s + j\Delta), \tag{13}$$

with $\tau_{d+1} = t_s$ and $\tau_{n+1} = t_f$. Also, the partial curve of $z(t)$ within the j^{th} interval (13) is denoted by:

$$z(j,t) \triangleq z(t),\ \ t \in [t_s + (j-1)\Delta, t_s + j\Delta). \tag{14}$$

The B-spline curve $z(t)$ as defined in (8)–(12) exhibits the following properties [9], [10], [13]:

P1) Endpoint interpolation:

$$z(t_s) = P_1, \quad z(t_f) = P_n, \tag{15}$$

where $[t_s, t_f]$ is the considered time range of the B-spline curve $z(t)$ as in (8) and (P_1, P_n) are the first and last control points, respectively.

P2) Local support and local convexity within the j^{th} interval defined in (13) ($j \in \{1, \ldots, n-d\}$):

$$z(j,t) = \sum_{i=j}^{j+d} P_i B_{i,d,\xi}(t), \tag{16}$$

$$z(j,t) \in \text{Conv}\{\boldsymbol{P}_j\}, \tag{17}$$

in which Conv$\{\boldsymbol{P}_j\}$ represents the convex hull of $(d+1)$ control points $\boldsymbol{P}_j \triangleq [P_j \cdots P_{j+d}]$.

P3) Derivatives of B-spline basis functions can be expressed as a linear combination of B-spline basis functions:

$$\frac{\partial \boldsymbol{B}_{d,\xi}(t)}{\partial t} = M_{d,d-1} L_{d,d-1} \boldsymbol{B}_{d,\xi}(t), \tag{18}$$

with $\boldsymbol{B}_{d,\xi}$ as in (8). The two matrices $M_{d,d-1} \in \mathbb{R}^{n \times (n-1)}$ and $L_{d,d-1} \in \mathbb{R}^{(n-1) \times n}$ are given in Theorems 4.1–4.3 of [9].

B. *Local equivalent Bézier curves*

It is well-known in the field of CAD (Computer-Aided-Design) that the B-spline curve in its j^{th} interval, $z(j,t)$ as in (14), is actually a Bézier curve of the same degree [11], [14] which is defined as follows:

$$z(j,t) = \sum_{i=1}^{d+1} \overline{P}_{(j-1)d+i} B_{i,d,\overline{\xi_j}}(t), \tag{19}$$

in which $\overline{P}_k$ are the Bézier control points ($k \in \{(j-1)d+1, \ldots, jd+1\}$) and $B_{i,d,\overline{\xi_j}}$ is the basis function as defined in (10)–(11) with the new knot vector $\overline{\xi_j}$:

$$\overline{\xi_j} = \{\underbrace{\tau_{j+d}, \ldots, \tau_{j+d}}_{d+1 \text{ knots}}, \underbrace{\tau_{j+d+1}, \ldots, \tau_{j+d+1}}_{d+1 \text{ knots}}\}, \tag{20}$$

with $[\tau_{j+d}, \tau_{j+d+1})$ denoting the considered j^{th} interval as in (13).

For the j^{th} interval (13), the Bézier control points $\overline{P}_k$ ($k \in \{(j-1)d+1, \ldots, jd+1\}$) as used in (19) can be obtained from $(d+1)$ original B-spline control points $\{P_j, \ldots, P_{j+d}\}$ by using the following matrix transformation:

$$\overline{\boldsymbol{P}}_j = \boldsymbol{P}_j A(d,n,j), \tag{21}$$

with $\overline{\boldsymbol{P}}_j \triangleq [\overline{P}_{(j-1)d+1} \cdots \overline{P}_{jd+1}]$ and $\boldsymbol{P}_j \triangleq [P_j \cdots P_{j+d}]$ consisting of $(d+1)$ Bézier and B-spline control points, respectively. The matrix $A(d,n,j) \in \mathbb{R}^{(d+1)\times(d+1)}$ is calculated with the algorithm defined in [14] (at page 7 and another recursive formulation of $A(d,n,j)$ is also given in Remark 2 of [14]). Note that the total number of Bézier control points $\overline{P}_k$ as in (19) needed for expressing the whole B-spline curve of degree d is:

$$\overline{n} = (n-d)d + 1, \tag{22}$$

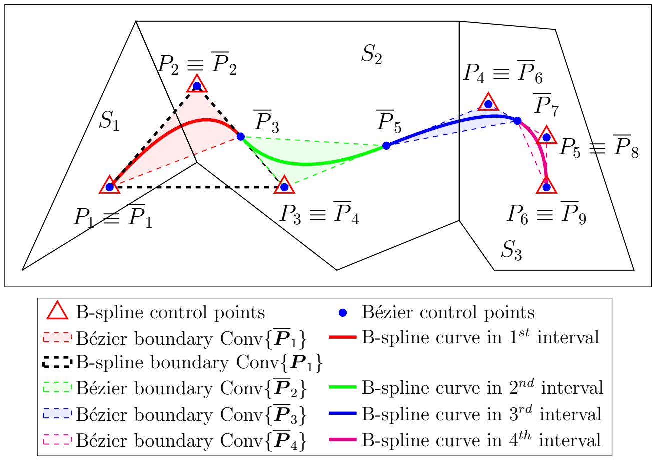
with n the number of the B-spline control points from (8). By applying the two properties **P1** and **P2** given in (15)–(17) to the local Bézier curve from (19), we have that:

$$\overline{P}_{(j-1)d+1} = z(\tau_{j+d}) = z(t_s + (j-1)\Delta), \tag{23}$$

$$z(j,t) \in \text{Conv}\{\overline{\boldsymbol{P}}_j\}, \tag{24}$$

for all $j \in \{1, \ldots, n-d\}$, with $z(j,t)$ as in (14) and $\overline{\boldsymbol{P}}_j$ from (21). The convexity property (24) is significantly tighter than the standard one as given in (17) as proven in [11]. This will be illustrated through the following example.

Illustrative example: Fig. 3 represents the safe region $\mathcal{S} = S_1 \cup S_2 \cup S_3$ as defined in (4) and the second-order B-spline as in (8) constructed with six control points (plotted by triangle red marks). In one interval j, the partial curve $z(j,t)$



S_1
S_2
S_3
$P_1 \equiv \overline{P}_1$
$P_2 \equiv \overline{P}_2$
$\overline{P}_3$
$P_3 \equiv \overline{P}_4$
$\overline{P}_5$
$P_4 \equiv \overline{P}_6$
$\overline{P}_7$
$P_5 \equiv \overline{P}_8$
$P_6 \equiv \overline{P}_9$
B-spline control points
Bézier control points
Bézier boundary Conv$\{\overline{\boldsymbol{P}}_1\}$
B-spline curve in 1st interval
B-spline boundary Conv$\{\boldsymbol{P}_1\}$
Bézier boundary Conv$\{\overline{\boldsymbol{P}}_2\}$
B-spline curve in 2nd interval
Bézier boundary Conv$\{\overline{\boldsymbol{P}}_3\}$
B-spline curve in 3rd interval
Bézier boundary Conv$\{\overline{\boldsymbol{P}}_4\}$
B-spline curve in 4th interval

Fig. 3: Second-order B-spline curve with its Bézier boundaries fitting inside connected polytopes.

from (14) is plotted in solid red, green, blue and magenta lines, corresponding to each value of $j \in \{1, \ldots, 4\}$. Each partial curve $z(j,t)$ is also a Bézier curve of degree two which is constructed by using three Bézier control points $\{\overline{P}_{2(j-1)+1}, \ldots, \overline{P}_{2j+1}\}$ as defined in (21) with the matrix

$A(d, n, j)$ given as:

$$A(2, n \geq 4, j) = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0 & 0.5 \end{bmatrix}, & j = 1, \\ \begin{bmatrix} 0.5 & 0 & 0 \\ 0.5 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, & j = n-2, \\ \begin{bmatrix} 0.5 & 0 & 0 \\ 0.5 & 1 & 0.5 \\ 0 & 0 & 0.5 \end{bmatrix}, & \text{otherwise.} \end{cases} \quad (25)$$

The property $z(j, t) \in \text{Conv}\{\overline{\boldsymbol{P}}_j\}$ as in (24) is also illustrated with each Bézier boundary $\text{Conv}\{\overline{\boldsymbol{P}}_j\}$ given by four triangles filled with red, green, blue and magenta corresponding to $j \in \{1, \ldots, 4\}$.
Furthermore, for the first interval, the B-spline boundary $\text{Conv}\{\boldsymbol{P}_1\}$ from (17) fails to stay inside the safe region $\mathcal{S}$ (i.e., the presenting B-spline curve will not be validated if one uses the original control polygons to check the constraint $z(t) \in \mathcal{S}$) while the Bézier boundary $\text{Conv}\{\overline{\boldsymbol{P}}_1\}$ (with $\overline{\boldsymbol{P}}_1 = [\overline{P}_1, \overline{P}_2, \overline{P}_3]$ as in (21)) fits well and therefore, admits that $z(t) \in \mathcal{S}, \forall t \in [t_s, t_f]$. □

Remark 1: Regarding the matrix $A(2, n, j)$ as given in (25), for the case of $n = 3$, there is only one value of $j = 1$. The B-spline curve becomes the Bézier curve and hence, the matrix $A(2, 3, 1)$ is the identity matrix in $R^{3\times3}$. A similar argument can be applied for any degree d which leads to:

$$A(d, d+1, 1) = I_{d+1}, \quad (26)$$

with I_{d+1} the identity matrix of dimension $(d+1)$. □

IV. B-SPLINE PATH PLANNING WITH MINIMAL LENGTH

This section presents the approach for optimally placing the control points $\{P_1, \ldots, P_n\}$ such that the B-spline curve $z(t)$ from (8) satisfies all the requirements of our path generation problem proposed in Section II-A and has its minimal length. We will start by discussing the constraints on the control points in the following.

A. Safe B-spline path through a sequence of polytopes

In this section, we present our approach to constrain the control points of the B-spline curve $z(t)$ from (8) such that the curve stays inside the safe region $\mathcal{S}$ and also satisfies the endpoint constraints:

$$z(t) \in \mathcal{S}, \quad z(t_s) = P_s, \quad z(t_f) = P_f, \quad (27)$$

with $\mathcal{S} = S_1 \cup S_2 \cup \cdots \cup S_q$ $(q \geq 2)$ from (4) and $\{P_s, P_f\}$ start and end poses from (15).

Proposition 1: The requirements (27) are satisfied if the following conditions are guaranteed:

C1) Number of control points:

$$n = q + d. \quad (28)$$

C2) Start and end points:

$$P_1 = P_s, \quad P_{q+d} = P_f. \quad (29)$$

C3) All the Bézier control points in one interval belong to one extended polytope (7):

$$\overline{P}_k \in S_{j,j+1}, \ \forall \, \overline{P}_k \in \overline{\boldsymbol{P}}_j \text{ and } \forall \, j \in \{1, \ldots, q\}, \quad (30)$$

with $\overline{\boldsymbol{P}}_j$ consisting of $(d+1)$ Bézier control points given in terms of $(d+1)$ B-spline control points $\boldsymbol{P}_j$ as in (21) and $S_{j,j+1}$ the extended polytope as in (7). □

Proof: At first, the starting and ending constraints from (27) are satisfied by condition C2 (29) due to the endpoint interpolation property (15) of B-spline curves.
Next, by using $n = d + q$ control points as in (28), the curve $z(t)$ from (8) has q intervals. Within each interval j, $j \in \{1, \ldots, q\}$, we have that:

$$z(j, t) \in \text{Conv}\{\overline{\boldsymbol{P}}_j\} \subseteq S_{j,j+1}, \quad (31)$$

in which the convexity property is given in (24) and the latter is due to the fact that all $(d+1)$ points in $\overline{\boldsymbol{P}}_j$ stay inside $S_{j,j+1}$ as constrained by (30). Finally, we arrive at:

$$z(t) \in \bigcup_{j=1}^{q} S_{j,j+1} \equiv \mathcal{S}, \ t \in [t_s, t_f]. \quad (32)$$

This completes the proof. ■

Remark 2: Using the Bézier representation (19) allows us to formulate the constraint (30) such that we can enforce "each interval $z(j,t)$ to be inside each extended polytope" as proven in (31). This cannot be done if the original B-spline convexity property (17) is employed instead. The reason is that two consecutive B-spline boundaries share d common points (e.g. $\boldsymbol{P}_j = [P_j \cdots P_{j+d}]$ and $\boldsymbol{P}_{j+1} = [P_{j+1} \cdots P_{j+d+1}]$ from (17)). This leads to the fact that if the B-spline control points are employed in condition C3 (30), i.e. $P_k \in S_{j,j+1}, \forall P_k \in \boldsymbol{P}_j, \ \forall j \in \{1, \ldots, q\}$, then, the following necessary condition is required:

$$\bigcap_{i=j}^{j+d} S_{i,i+1} \neq \varnothing, \forall j \in \{1, \ldots, q\}, \quad (33)$$

which is clearly not guaranteed for the extended polytopes defined in (7).
On the other hand, there is only one common point for the Bézier representation (24) (e.g. $\overline{\boldsymbol{P}}_j$ and $\overline{\boldsymbol{P}}_{j+1}$ share one common point $\overline{P}_{jd+1}$). Therefore, the necessary condition for the solution of (30) to exist is:

$$S_{j,j+1} \cap S_{j+1,j+2} \neq \varnothing, \forall j \in \{1, \ldots, q-1\}, \quad (34)$$

with $S_{j,j+1} \cap S_{j+1,j+2} = T_j$ as defined in (6)–(7). □

Remark 3: In condition C3 (30), one can replace the extended polytope $S_{j,j+1}$ with the polytope S_j itself but the result will be more conservative as the two connected polytopes share only one edge as defined in (5). □

B. Path generation problem with minimal length

In this section, we present the complete optimization problem used to solve the B-spline reference path satisfying the constraints (27) and minimizing the curve's length. We exploit the property **P3** in (18) of the B-spline curve $z(t)$ from (8) in order to formulate the length cost into a quadratic function of the control points $P_i \in \mathbb{R}^2$, $i \in \{1,\dots,n\}$ (with n chosen as in (28). By denoting $\boldsymbol{P} \triangleq [P_1 \cdots P_n]$ from (8), the optimization problem is given by:

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} \int_{t_s}^{t_f} \|\dot{z}(t)\|^2 dt, \tag{35}$$
subject to constraints (28)–(30).

By using property (18) of the B-spline curve, we have that:

$$\dot{z}(t) = \boldsymbol{P} M_{d,d-1} L_{d,d-1} \boldsymbol{B}_{d,\xi}(t) = \sum_{i=1}^{n} Q_i B_{i,d,\xi}(t), \tag{36}$$

with $Q_i \in \mathbb{R}^2$ being the i^{th} column of $Q = \boldsymbol{P} M_{d,d-1} L_{d,d-1} \in \mathbb{R}^{2\times 2q}$. Therefore, the optimization problem (35) is re-formulated into:

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} \sum_{i=1}^{n}\sum_{j=1}^{n} Q_i^\top Q_j \int_{t_s}^{t_f} B_{i,d,\xi}(t) B_{j,d,\xi}(t) dt, \quad \text{subject to constraints (28)–(30),} \tag{37}$$

which clearly has a quadratic cost function since the integral terms are independent of the decision variables $\boldsymbol{P} = [P_1 \cdots P_n]$.
Finally, the reference path $p(t)$ as required in (2) is taken as:

$$p(t) = \boldsymbol{P}^* \boldsymbol{B}_{d,\xi}(t), \tag{38}$$

in which the optimal control points $\boldsymbol{P}^*$ are obtained from solving the optimization problem (37) and the B-spline basis functions $\boldsymbol{B}_{d,\xi}$ as used in (8) is defined with $[t_s, t_f] = [0, 1]$.

C. Discussion

We can only concentrate on the geometric problem of the path planning problem within this paper due to the space limit but our proposed algorithm can be easily extended for trajectory generation cases by replacing t_s and t_f in (8) with the actual start and end time instants. Then, the B-spline framework allows us to consider more constraints on the velocity and on higher derivatives of the trajectory by adding more linear constraints as discussed in [8], [9]. Also note that our proposed approach for placing the control points as given in Proposition 1 is not a unique solution for the problem (27). In general, there exist mixed-integer constraints formulations for generating similar obstacle-free B-spline curves [7], [12] but we avoid using mixed-integer programming in our work due to its high computational burden. Instead, our constraints proposed in Proposition 1 only consist of equalities (28)–(29) and linear matrix inequalities (30) which, in combination with a quadratic cost (37) for minimizing the curve length, finally form the convex and quadratic programming problem which is relatively easy to solve even with hardware with limited computing capability.

V. Simulation and Experiment Results

Next, we present the validation of our proposed path planning method. We first show a simulation result that demonstrates the advantages of using the equivalent Bézier representation to constrain the B-spline curve as in (30) as well as the effectiveness of the minimal-length optimization problem (37). Next, we give details on the implementation of the proposed method on a real mobile robot platform and the evaluation on two real-world datasets (i.e. one single laser scan and one occupancy grid map) using an onboard computer of a real robot platform.

A. Simulation results

For the simulation, we consider a safety region $\mathcal{S}$ from (4) consisting of seven connected polytopes as plotted with black edges (no fill) in Fig. 4. The start and end poses are $P_s = (1, 2.5)^\top$ and $P_f = (1, 0)^\top$, respectively. Information about the used B-spline curve is shown in Table I.

TABLE I: Parameters of B-spline curve used in simulation

Parameter	Notation	Value
Degree	d as in (8)	4
Number of B-spline control points	n as in (8)	11
Number of Bézier control points	$\overline{n}$ as in (22)	29

In Fig. 4, we provide two simulation results, among which the green curve is one feasible solution (manually chosen for clear demonstration) of the proposed constraints (28)–(30) while the red solid curve shows the optimal path (38). It can be seen that the non-optimal green path is much longer than the optimal red path which always approaches all corners closely due to the effect of the minimal-length optimization problem (35). Both paths stay inside the safety region $\mathcal{S}$ (i.e. the union of seven polytopes) which proves the effectiveness of the proposed approach on constraining a B-spline curves by using its equivalent Bézier representation (28)–(30).

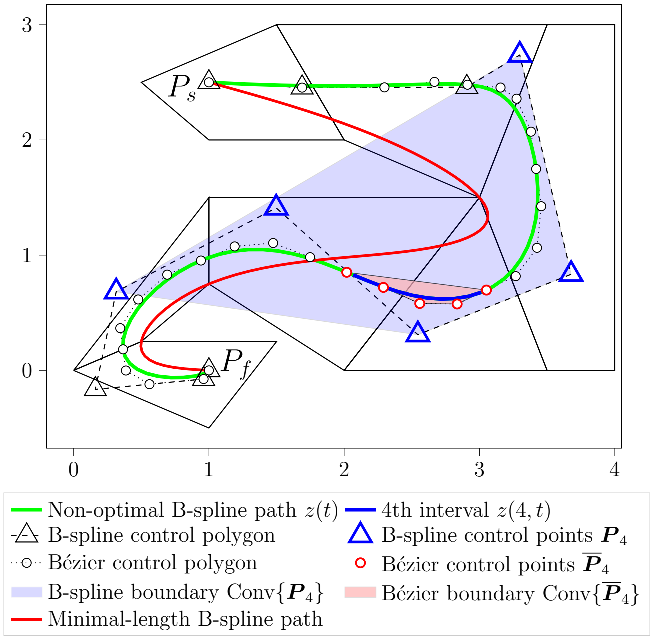


Fig. 4: Path planning results using fourth-order B-spline curves: comparisons between B-spline versus Bézier boundaries and non-optimal versus minimal-length paths.

Furthermore, we would like to demonstrate the comparison between the Bézier and the original B-spline boundaries. Let us consider the non-optimal green curve $z(t)$ and its 4^{th} partial curve $z(4,t)$ from (14) (plotted with blue solid line) as shown in Fig. 4. This blue curve $z(4,t)$ stays within both the B-spline boundary $\mathrm{Conv}\{\boldsymbol{P}_4\}$ as in (17) (big blue polytope) and the Bézier boundary $\mathrm{Conv}\{\overline{\boldsymbol{P}}_4\}$ as in (24) (small red polytope) with the Bézier control points $\overline{\boldsymbol{P}}_4 = \boldsymbol{P}_4 A(4,11,4)$ from (21) with the matrix $A(4,11,4)$ as in (21) given by (i.e. using algorithm given in [14], page 7):

$$A(4,11,4) = \begin{bmatrix} 1/24 & 11/24 & 11/24 & 1/24 & 0 \\ 0 & 1/3 & 7/12 & 1/12 & 0 \\ 0 & 1/6 & 2/3 & 1/6 & 0 \\ 0 & 1/12 & 7/12 & 1/3 & 0 \\ 0 & 1/24 & 11/24 & 11/24 & 1/24 \end{bmatrix}. \quad (39)$$

As seen in Fig. 4, the Bézier boundary is significantly tighter. It provides a better approximation of the curve's shape and a more efficient way to constrain the curve fitting inside the safety region as introduced in (30).

For implementing the optimization problem (37), we use the solver IPOPT [15] with Pyomo [16] in Python. With our laboratory computer (Intel Core i7-9750H processor), the average computation time is 45 ms for 100 samples. In the next section, we will present the implementation and experimental results of the proposed path planner on a mobile robot platform.

B. *Experimental validation on mobile robot platform*

The proposed path planner is implemented on the differential wheeled Jackal robot from Clearpath Robotics [3] equipped with the Ouster OS1-16 LIDAR sensor (cf. Fig. 5) and an on-board computer (Intel Core i5-4750TE processor). In order to evaluate the method, we consider two scenarios:

1) Local planner: a single LIDAR scan was collected and then an obstacle-free path was planned within the scanned area.
2) Global planner: an occupancy grid map[1] (which is obtained beforehand by using the *gmapping* method [17]) is given to the Jackal and the robot needs to plan an obstacle-free path within the map.

For both scenarios, the path leads from the robot's current position P_s to the desired goal P_f. In order to obtain the sequence of connected polytopes $\{S_1, \ldots, S_q\}$ as required in (4), we adapt the procedure given in [2] using several existing Python toolboxes. Note that the robot's size is taken into account by using a safety offset when calculating those polytopes so that the robot does not hit any obstacles even when reaching the edges of the polytopes. The complete workflow is given as follows:

1) Approximate the obstacle-free space (i.e. obtained from the laser scan data and from the occupancy grid map corresponding to two scenarios) by a polygon region (possibly being non-convex and containing holes): using Ramer–Douglas–Peucker algorithm[2] [18].

[1]This occupancy grid map (cf. Fig. 7) does not include the scanned area used in local planner scenario.

[2]https://github.com/biran0079/crdp

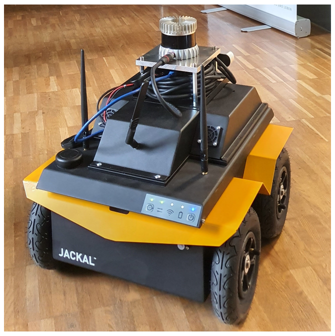


Fig. 5: Jackal robot equipped with a LIDAR sensor.

2) Shrink the obstacle-free polygon region by a safety offset (can be taken as the maximum length of the Jackal as it is differential wheeled robot) in order to account for the vehicle's size and possible detection noises: using Gdspy toolbox [3].
3) Partition the shrunk polygon into connected polytopes: using Mark Beyazit's algorithm[4] [19].
4) Perform a graph search in order to obtain the sequence of polytopes $\{S_1, \ldots, S_q\}$ leading from the initial pose P_s to the desired goal P_f [2]. We relax the conditions, i.e. if there is no polytope containing P_f, then P_f is re-chosen as the closest point to the original P_f but within the safety region.
5) Solve the path-planning problem (37) and return the optimal path (using solver IPOPT [15] in Pyomo [16]).

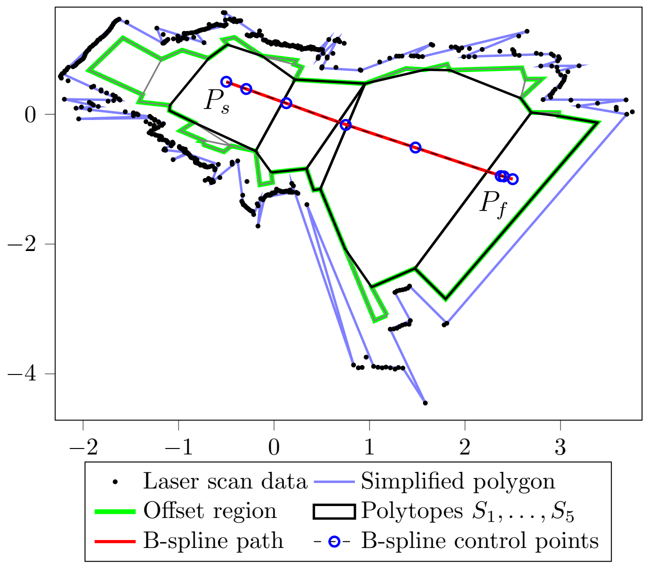


Fig. 6: Experimental result in the local planner scenario: from laser scan data to optimal B-spline path.

The results under the local and global planner scenarios are given in Figures 6 and 7, respectively. In both figures, the safety regions after performing the safety offset (i.e. step 2 from the aforementioned procedure) are plotted with green solid lines. Then, at step 3, those regions are partitioned into

[3]https://github.com/heitzmann/gdspy

[4]https://github.com/wsilva32/poly_decomp.py

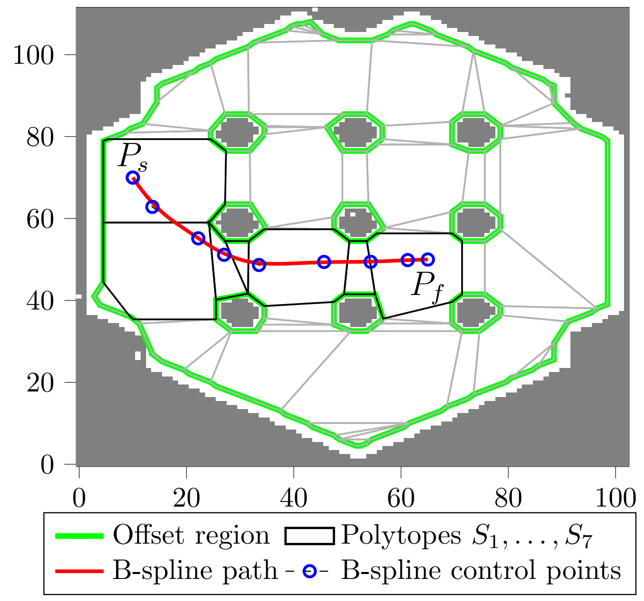


Fig. 7: Experimental result in the global planner scenario: from occupancy grid map to optimal B-spline path.

numerous connected polytopes among which the sequences of connected polytopes leading from the start point P_s to the end goal P_f are emphasized with black solid edges (i.e. resulted at step 4) while the other polytopes in the workspace are plotted with gray edges. Finally, the two optimal B-spline paths of degree four (obtained from solving the optimization problem (35) at step 5) are plotted in solid red with their B-spline control points marked with blue circle markers. The total processing times are 67 ms in the local planner scenario and 100 ms in the global planner case using the on-board computer of the Jackal robot. Note that, even though the resulted straight path shown in Figure 6, which is obtained in local planner scenario, seems trivial it strongly demonstrates the effectiveness of the minimizing-length goal proposed in the optimization problem (35). The two reference paths (red curves in Figures 6 and 7) obtained with our proposed planning method under two different scenarios are both obstacle-free, smooth and differentiable up to degree 4 (with one under local planner case even being straight as shown in Figure 6). Therefore, with recent advances on path following control techniques [4], it is capable for our differential wheeled Jackal robot to track those references. We do not show tracking results here as it is also out of this paper's scope but this is the logical next step of our future research.

VI. Summary

We addressed a path planning method for navigation in a non-convex, connected region. We parameterize the path as a B-spline curve but control the shape by using its equivalent Bézier representation which provides a significantly tighter bound on the curve. By using this property, our method constrains the curve to continuously stay inside the safety region using only linear constraints. The final path planning problem is reformulated as a convex optimization problem with the objective of minimizing the path length. The algorithm is validated under both simulation and experiment on real datasets with an on-board computer of a mobile robot platform. In the future we will take more constraints into account, such as initial and final heading angles, way-points, curvature boundedness, and extend the method for trajectory generation problems.

References

[1] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International Journal of Physical Sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.

[2] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A nonlinear model predictive control formulation for obstacle avoidance in high-speed autonomous ground vehicles in unstructured environments," *Vehicle System Dynamics*, vol. 56, no. 6, pp. 853–882, 2018.

[3] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[4] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, pp. 33–55, 2016.

[5] M. Elbanhawi and M. Simic, "Examining the use of b-splines in parking assist systems," *Applied Mechanics and Materials*, vol. 490, pp. 1025–1029, 2014.

[6] T. Maekawa, T. Noda, S. Tamura, T. Ozaki, and K.-i. Machida, "Curvature continuous path generation for autonomous vehicle using b-spline curves," *Computer-Aided Design*, vol. 42, no. 4, pp. 350–359, 2010.

[7] F. Stoican, I. Prodan, E. I. Grøtli, and N. T. Nguyen, "Inspection trajectory planning for 3d structures under a mixed-integer framework," in *Proceedings of the 2019 IEEE International Conference on Control & Automation (ICCA'19)*. IEEE, 2019, pp. 1349–1354.

[8] N. T. Nguyen, I. Prodan, and L. Lefèvre, "Flat trajectory design and tracking with saturation guarantees: a nano-drone application," *International Journal of Control*, vol. 93, no. 6, pp. 1266–1279, 2020.

[9] F. Suryawan, J. De Doná, and M. Seron, "Splines and polynomial tools for flatness-based constrained motion planning," *International Journal of Systems Science*, vol. 43, no. 8, pp. 1396–1411, 2012.

[10] T. Lyche, C. Manni, and H. Speleers, "Foundations of spline theory: B-splines, spline approximation, and hierarchical refinement," in *Splines and PDEs: From Approximation Theory to Numerical Linear Algebra*. Springer, 2018, pp. 1–76.

[11] W. Böhm, "Generating the bézier points of b-spline curves and surfaces," *Computer-aided Design*, vol. 13, no. 6, pp. 365–366, 1981.

[12] I. Prodan, F. Stoican, and C. Louembet, "Necessary and sufficient lmi conditions for constraints satisfaction within a b-spline framework," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 8061–8066.

[13] L. Piegl and W. Tiller, "B-spline curves and surfaces," in *The NURBS Book*. Springer, 1995, pp. 81–116.

[14] L. Romani and M. A. Sabin, "The conversion matrix between uniform b-spline and bézier representations," *Computer Aided Geometric Design*, vol. 21, no. 6, pp. 549–560, 2004.

[15] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[16] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in python," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 219–260, 2011.

[17] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[18] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: the international Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[19] Y. Kulkarni, A. Sahasrabudhe, and M. Kale, "Midcurves generation algorithm for thin polygons," in *National Conference on Emerging Trends in Engineering and Science (ETES)*, 2014, pp. 76–82.