2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Navigation with polytopes and B-spline path planner

Ngoc Thinh Nguyen[1], Pranav Tej Gangavarapu[1], Arne Sahrhage[1], Georg Schildbach[2], Floris Ernst[1]

***Abstract*— This paper firstly presents our optimal path planning algorithm within a 2D non-convex, polytopic region defined as a sequence of connected convex polytopes. The path is a B-spline curve but being parametrized with its equivalent Bézier representation. By doing this, the local convexity bound of each curve's interval is significantly tighter. Thus, it allows many more possibilities for constraining the entire curve to remain inside the region by using only linear constraints on the control points of the curve. We further guarantee the existence of the valid path by pointing out an algebraic solution.**
We integrate the algorithm, together with our previously published results, into the *Navigation with polytopes* toolbox which can be used as a global path planner, compatible with ROS navigation tools. It provides a framework for constructing a polytope map from a standard occupancy gridmap, searching for an appropriate sequence of connected polytopes and finally, planning a minimal-length path with different options on B-spline or Bézier parametrizations. The validation and comparison with existing methods are done using gridmaps collected under Gazebo simulations and real experiments.

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

I. Introduction

Autonomous navigation of unmanned vehicles has been of great interest to both the research and industrial communities for several decades [1]–[7]. Among a large variety of navigation problems such as localization, mapping, safe driving, etc, the planning task plays an important role as it provides an overall guidance for sequential movements [6]–[8]. Numerous path planning approaches make use of discrete spatial data (e.g. gridmap, discrete dynamics) such as traditional grid search methods like A*, Dijkstra [8], or approaches using random discrete movements like RRT (Rapidly exploring Random Tree) [7] and discrete models in combination with optimal planning [6]–[9]. However, any direct usage of discrete data implies a black box hiding what really happens in between the predefined time steps (e.g., for discrete models) or positions (e.g., for gridmaps). One can enhance the accuracy by increasing the grid resolution but at the cost of increasing the computational burden.
Therefore, we consider an approach of adapting continuously interpolating motions with the standard discrete setups [1], [10]. In [1], we construct the polytope map (i.e., continuous space) within a standard gridmap (i.e., discrete space) and then, plan a reference B-spline path which *entirely* and *continuously* stays inside the polytopic region. For dealing with both continuous and geometrical constraints, B-spline and its close relative, Bézier curves, are usually considered as promising candidates [1]–[5]. This is thanks to their interesting property of each interval being bounded by a *control boundary* defined as the convex hull of their local control points [1]–[3]: in Fig. 1, the green triangle is the control boundary of the yellow section (seen from the inset) of the second-degree B-spline curve (solid red line). We further constrain a B-spline path with its equivalent Bézier representation as inspired by the Computer Aided Design Community [11], [12]. It allows us to obtain a tighter control boundary of each interval of the B-spline curve (in Fig. 1, the purple triangle is the Bézier control boundary of the aforementioned yellow B-spline section) and thus, to reduce the conservativeness when formulating the geometrical constraints. In [1], we were able to constrain each interval of the B-spline curve to stay within one polytope among the connected sequence which is impossible to achieve with the original B-spline format. However, the method still has its drawback of occasionally failing due to an *unsolvable* result in certain cases since the optimal path-planning problem does not have any guarantee of its solvability. Therefore, as a continuation and completion of

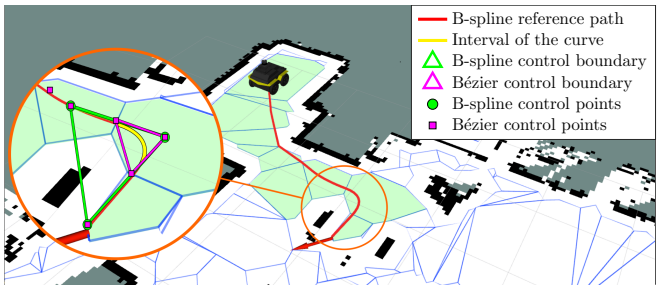


Fig. 1: B-spline path planned within a polytope map with the *Navigation with polytopes* toolbox (visualization in Rviz).

our work in [1], this paper presents the following novelties:
• New path planning constraints for a B-spline path to stay inside a sequence of connected polytopes in 2D: employ the equivalent Bézier representation due to the advantages of the tighter local boundaries as similar to [1], and guarantee the existence of a valid path by providing an algebraic solution.
• *Navigation with polytopes* toolbox: a complete package to perform our path planning algorithms in any standard gridmap (c.f. Fig. 1). It mainly includes:

1) a function to construct the polytope map from a given gridmap. The polytopes are built far away from any occupied grids at a defined safety distance which

[1] University of Luebeck (UzL), Institute for Robotics and Cognitive Systems, Luebeck, Germany.
{nguyen,sahrhage,ernst}@rob.uni-luebeck.de
pranav.gangavarpu@student.uni-luebeck.de
[2] UzL, Institute for Electrical Engineering in Medicine, Luebeck, Germany.
georg.schildbach@uni-luebeck.de.
N. T. Nguyen and A. Sahrhage are funded by the German Ministry of Food and Agriculture (BMEL) Project No. 28DK133A20.

accounts for the robot's dimension and possible noises.
2) a function to search for a sequence of connected polytopes connecting two given points with minimal distance.
3) a minimal-length path planning algorithm with three different options: i) using the minimum number of control points [1], ii) using more control points but with guaranteed solution and iii) the algebraic solution.
4) an extra library for calculating and storing the transformation matrix between the B-spline curve and its equivalent Bézier representation.

II. Preliminaries

This paper is a continuation with significant improvements of our previously published work [1]. For consistency and easy reference, various fundamental setups are kept the same as in [1] such as definitions of problem, basic elements, symbols, as well as the path models (i.e. B-spline with equivalent Bézier representation).

A. Problem statement

We consider the global path planing problem in which the reference path is defined as a smooth geometric path in 2D space as follows:

$$p(t) : [t_s, t_f] \rightarrow \mathbb{R}^2, \tag{1}$$

with $t \in \mathbb{R}$ the curve parameter (e.g. path length, pseudo-time increment, etc. depending on specific circumstances). The path starts at the point P_s (e.g. current robot's position) and ends at the point P_f (e.g. the selected goal point):

$$p(t_s) = P_s, \quad p(t_f) = P_f. \tag{2}$$

Furthermore, the path needs to remain inside the safe region $\mathcal{S}$ continuously (w.r.t. the variable $t \in [t_s, t_f]$) which is a sequence of q connected polytopes S_i, $i \in \{1, \dots, q\}$:

$$p(t) \in \mathcal{S} \triangleq S_1 \cup S_2 \cup \dots \cup S_q, \ \forall t \in [t_s, t_f], \tag{3}$$

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

$$E_i = S_i \cap S_{i+1}, \ \forall i \in \{1, \dots, q-1\}. \tag{4}$$

For the problem to be well-defined, we consider that the starting and ending points (P_s, P_f) from (2) also belong to the first and last polytopes, respectively:

$$P_s \in S_1, \quad P_f \in S_q. \tag{5}$$

Note that, in Section V, we will discuss how to obtain the prerequisite sequence of connect polytopes $\mathcal{S}$ from a real gridmap (c.f. Fig. 1) as well as some relaxations of (5).

B. Transition zone and extended polytope

Regarding the sub-problem of navigation through two connected polytopes among the sequence $\mathcal{S}$ (3), one simple solution is to find a *transition zone* whose union with any of the two polytopes is convex (cf. Fig. 2). Then, enforcing a transit at the transition zone before moving to the next polytope can ensure the line-of-sight navigation.

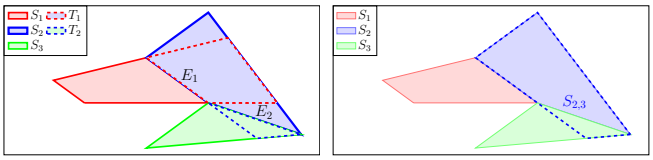


Fig. 2: Illustration of connected polytopes, transition zones and extended polytope according to Definitions 1–2.

Definition 1 (Transition zone [1]): The transition zone T_i is defined for two connected polytopes S_i and S_{i+1} from (4) as follow:

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

in which E_i is the common edge as defined in (4), and the operation $(S_i|E_i)$ gives the (possibly unbounded) polytope formed by the half-space representation of S_i without the constraint corresponding to the edge E_i. □

Definition 2 (Extended polytope [1]): $S_{i,i+1}$ is defined 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). □

For consistency, the last extended polytope is considered to be the last polytope, i.e. $S_{q,q+1} \triangleq S_q$.

Note that, any extended polytope $S_{i,i+1}$ as defined in (7) is convex and it is also straightforward to achieve the transition zone from the corresponding extended polytope:

$$T_i = S_{i,i+1} \cap S_{i+1} = S_{i,i+1} \cap S_{i+1,i+2}. \tag{8}$$

In the next section, we will introduce our tool to counteract the path planning problem with both *continuous* and *geometrical* constraints (3) which is to sketch the path as a B-spline curve but parametrize it with the equivalent Bézier representation. This method has proven its effectiveness in our previous study [1], i.e., it possesses all the advantageous properties of a B-spline curve as well as provides a better approximation (i.e. a tighter control boundary) for each interval of the whole curve thanks to its Bézier formulation.

III. B-spline and equivalent Bézier curves

This section gives a brief introduction to B-spline curves and their equivalent Bézier representation. We only focus on defining the essential formulations needed for later use while more details could be found in our previous works [1], [2]. We intentionally keep the same notations and symbols as used in [1] for easy reference.

A. Definition of B-spline curves

A clamped uniform B-spline curve $z(t) : [t_s, \ t_f] \rightarrow \mathbb{R}^2$ 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:

$$z(t) = \sum_{i=1}^{n} P_i B_{i,d,\xi}(t) = \boldsymbol{P}\boldsymbol{B}_{d,\xi}(t), \ t \in [t_s, t_f], \tag{9}$$

in which, $\boldsymbol{P} \triangleq [P_1 \ \dots \ P_n] \in \mathbb{R}^{2\times n}$ and $\boldsymbol{B}_{d,\xi}(t) \triangleq [B_{1,d,\xi}(t) \ \dots \ B_{n,d,\xi}(t)]^\top : \mathbb{R} \rightarrow \mathbb{R}^n$ contains the B-spline

basis functions[1] of the same degree. The basis functions are constructed with a *clamped* and *uniformly distributed* knot vector ξ [1] which allows the curve $z(t)$ from (9) to have exactly $(n-d)$ consecutive intervals equally distributed within $[t_s, t_f]$ among which, a partial curve within the j^{th} interval ($j \in \{1, \dots, n-d\}$) is given by:

$$z(j,t) \triangleq z(t),\ t \in [t_s + (j-1)\Delta, t_s + j\Delta), \tag{10}$$

with $\Delta = (t_f - t_s)/(n-d)$. More interestingly, the j^{th} interval $z(j,t)$ only depends on its $(d+1)$ neighbor control points and also stays within their convex hull:

$$z(j,t) = \sum_{i=j}^{j+d} P_i B_{i,d,\xi}(t) \in \text{Conv}\{\boldsymbol{P}_j\}, \tag{11}$$

with $\boldsymbol{P}_j \triangleq [P_j \cdots P_{j+d}]$ containing $(d+1)$ control points from (9). Furthermore, the B-spline curve $z(t)$ starts and ends at the first and last control points P_1 and P_n from (9), respectively:

$$z(t_s) = P_1, \quad z(t_f) = P_n. \tag{12}$$

B. Local equivalent Bézier representation

As brought to attention and widely used by CAD (Computer Aided Design) community [11], [12], any interval of a B-spline curve as defined in (9), e.g. $z(j,t)$ from (10) is a Bézier curve of the same degree:

$$z(j,t) = \sum_{i=1}^{d+1} \overline{P}_{(j-1)d+i} B_{i,d,\overline{\xi_j}}(t), \tag{13}$$

in which $\overline{P}_k$ are the Bézier control points ($k \in \{(j-1)d+1, \dots, jd+1\}$), $B_{i,d,\overline{\xi_j}}$ is the same basis function as used in the B-spline definition (9) but with a new knot vector $\overline{\xi_j}$ defined by repeating only two different time values:

$$\overline{\xi_j} = \{\underbrace{t_s + (j-1)\Delta, \dots}_{d+1 \text{ knots}}, \underbrace{t_s + j\Delta, \dots}_{d+1 \text{ knots}}\}, \tag{14}$$

with $(t_s + (j-1)\Delta, t_s + j\Delta)$ the starting and ending of the j^{th} interval (10). The Bézier control points $\overline{P}_k$ ($k \in \{(j-1)d+1, \dots, jd+1\}$) as used in (13) is calculated by:

$$\overline{\boldsymbol{P}}_j = \boldsymbol{P}_j A(d,n,j), \tag{15}$$

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 $A(d,n,j) \in \mathbb{R}^{(d+1)\times(d+1)}$ can be calculated with the algorithm defined in [11]. As the total number of intervals is fixed at $(n-d)$ from (10), we can calculate $A(d,n,j)$ for all $j \in [1, \dots, n-d]$, then reformulate the transformation between the Bézier and B-spline control points as follows:

$$\overline{\boldsymbol{P}} = \boldsymbol{P} A(d,n), \tag{16}$$

with $\overline{\boldsymbol{P}} \triangleq [\overline{P}_1 \cdots \overline{P}_{\overline{n}}]$ consisting of all the Bézier control points and $\boldsymbol{P}$ as in (9). The total number of Bézier control points needed for expressing the whole B-spline curve of degree d is:

$$\overline{n} = (n-d)d + 1, \tag{17}$$

with n the number of the B-spline control points from (9).

Remark 1: The calculation of the matrix $A(d,n,j)$ is recursively defined in [11], which takes a considerable amount of time w.r.t. solving a complete path planning problem (28). We propose a new direct calculation method in [13]. We use it to compute the values of $A(d,n)$ in (16) beforehand for a large range of n (e.g., up to hundreds) and several values of the degree d (e.g., $2, 3, 4$), and then store them (c.f. Section V). During the online process, if (d,n) are matched, we make use of their stored values. Otherwise, a new value of $A(d,n)$ is calculated and added to the bank. Our algorithm takes maximum $\{20, 10, 4\}$ milliseconds to calculate the matrix $A(d,n)$ for the $\{4,3,2\}$-degree curve regardless the number of control points [13]. □

As a Bézier curve is also a B-spline curve, the same properties of local convex hull container (11) and endpoint interpolation (12) are applied to (13):

$$z(t_s + (j-1)\Delta) = \overline{P}_{(j-1)d+1}, \tag{18}$$

$$z(j,t) \in \text{Conv}\{\overline{\boldsymbol{P}}_j\} \subset \text{Conv}\{\boldsymbol{P}_j\}, \tag{19}$$

for all $j \in \{1, \dots, n-d\}$, with $z(j,t)$ as in (10) and $(\overline{\boldsymbol{P}}_j, \boldsymbol{P}_j)$ from (15). The convexity property (19) is significantly tighter than the standard one as given in (11), as proven in [12] and clearly demonstrated in our previous work [1]. The Reader is referred to [1] for illustrative comparisons between the B-spline and Bézier control boundaries. There, it was shown that constraining the B-spline curve $z(t)$ (9) with its equivalent Bézier control points allows to accept more possibilities of the safety condition (3) than the direct use of the B-spline points $\boldsymbol{P}$ from (9).

IV. B-spline path planning with guaranteed solution in sequence of polytopes

This section presents our novel approach for optimally placing the control points $\{P_1, \dots, P_n\}$ such that the B-spline curve $z(t)$ from (9) satisfies all the requirements of our path generation problem proposed in Section II-A and has its minimal length (by using the same optimal cost function as introduced in our previously published result [1]). In comparison with [1], the approach presented here still makes use of the equivalent Bézier control points which provides a much larger solution space w.r.t. the direct use of its B-spline control points, but guarantee the existence of the solution.

The problem to counteract is to constrain the B-spline curve $z(t)$ from (9) to satisfy the endpoint constraints while staying inside the safe region $\mathcal{S}$:

$$z(t_s) = P_s, \quad z(t_f) = P_f, \quad z(t) \in \mathcal{S}, \tag{20}$$

with $\mathcal{S} = S_1 \cup S_2 \cup \cdots \cup S_q$ ($q \geq 2$) from (3) and $\{P_s, P_f\}$ start and end poses from (12). Before we start describing the approach, let us clarify a notation: $\boldsymbol{P}_i \in S_j$ with $\boldsymbol{P}_i$ an array of points as in (15) means every point in $\boldsymbol{P}_i$ stays within the polytope S_j.

[1] A recursive definition of $\boldsymbol{B}_{d,\xi}(t)$ can be found in [1], [2].

A. Constrained B-spline path planning with guaranteed solution using equivalent Bézier control points

Proposition 1: The requirements (20) are satisfied if the B-spline control points of $z(t)$ are chosen according to the following conditions:

C1) Number of B-spline control points:

$$n = d(q-1) + 2, \tag{21}$$

which allows the curve to have $d(q-2)+2$ intervals as given in (10).

C2) Start and end points:

$$P_1 = P_s, \quad P_n = P_f. \tag{22}$$

with n as in (21).

C3) First and last intervals stay in the first and last (i.e. $S_{q,q+1} \equiv S_q$) extended polytopes (7), respectively:

$$\overline{\boldsymbol{P}}_1 \in S_{1,2}, \ \overline{\boldsymbol{P}}_{d(q-1)+2} \in S_{q,q+1}, \tag{23}$$

C4) and every other extended polytope contains d consecutive intervals:

$$\overline{\boldsymbol{P}}_j \in S_{k,k+1}, \forall j \in \{d(k-2)+2, \dots, d(k-1)+1\}, \\ \forall k \in \{2, \dots, q-1\}, \tag{24}$$

with $\overline{\boldsymbol{P}}_j$ consisting of $(d+1)$ Bézier control points (which control the j^{th} interval) given in terms of $(d+1)$ B-spline control points $\boldsymbol{P}_j$ as in (15) and $S_{k,k+1}$ the extended polytope as in (7). □

Proof: The proof is similar to our previous result [1], except for a feasible solution described at the end. Therefore, only the sketch of a proof is presented in this paper:

1) Condition **C2** (22) helps to ensure the start and end points of the path.

2) Condition **C1** provides a sufficient number of the intervals of the curve for the existence of a feasible solution. All the intervals are then constrained to stay within the safe region $\mathcal{S}$ by two conditions **C3**–**C4** because each Bézier control boundary (i.e. convex hull of the corresponding $(d+1)$ Bézier control points (19)) is inside one extended polytope.

3) Solutions for the complete problem (21)–(24) always exist. One can be found by placing the $d(q-1)+2$ original B-spline control points according to two conditions:

i) first and last points chosen according to (22),

ii) having d points in every transition zone T_k:

$$\{P_{(k-1)d+2}, \dots, P_{kd+1}\} \in T_k, \forall k \in \{1, \dots, q-1\}, \tag{25}$$

which is feasible since the transition T_k is not empty for all $k \in \{1, \dots, q-1\}$ as defined in (6). We then prove that the condition (25) ensures the satisfaction of the two conditions **C3**–**C4** (23)–(24) on the Bézier control points.

For **C3**, regarding the first interval of the curve, we have that $P_1 = P_s \in S_{1,2}$ from (22) and $\{P_2, \dots, P_{d+1}\} \in T_1 \subseteq S_{1,2}$ which ensure $\overline{\boldsymbol{P}}_1 \in S_{1,2}$ as $\text{Conv}\{\overline{\boldsymbol{P}}_1\} \subset \text{Conv}\{\boldsymbol{P}_1\}$ from (15). Similar argument is applied for the last interval of the curve and together, they lead to the satisfaction of (22).

Regarding **C4**, every extended polytope $S_{k,k+1}$ with $k \in \{2, \dots, q-1\}$ contains $2d$ B-spline control points:

$$\{P_{(k-2)d+2}, \dots, P_{kd+1}\} \in S_{k,k+1}, \tag{26}$$

which is due to (25) and the fact that both $T_k \subset S_{k,k+1}$ and $T_{k+1} \subset S_{k,k+1}$ (6)–(8). Then, for d consecutive intervals $(k-2)d+2, \dots, (k-1)d+1$, their Bézier control points satisfy:

$$\text{Conv}\{\overline{\boldsymbol{P}}_i\} \subset \text{Conv}\{\boldsymbol{P}_i\} \subset S_{k,k+1}, \\ \forall i \in [(k-2)d+2, \dots, (k-1)d+1]. \tag{27}$$

with $\overline{\boldsymbol{P}}_i$, $\boldsymbol{P}_i$ as in (15). Note that, we have $\text{Conv}\{\boldsymbol{P}_i\} \subset S_{k,k+1}$ since $\boldsymbol{P}_i \in S_{k,k+1}$, $\forall i \in [(k-2)d+2, \dots, (k-1)d+1]$ due to (26). Finally, we can ensure the condition (24) and the resulted B-spline path satisfies all the requirements (20). This also completes the proof. ■

Remark 2: In comparison with our previously published approach using a "minimal number of control points" [1], Proposition 1 adds d intervals to a middle polytope instead of using only 1 as in [1]. It allows to control the curve's shape within each polytope completely and independently and thus, always guarantee the existence of the solution. Furthermore, the feasible solution (25) is built upon the B-spline format. It is obviously only a subset of the Bézier constraints (24) while both of them can ensure the path planning requirements (20). Thus, we actually gain more feasibility and flexibility when switching to the equivalent Bézier format as also proven in our previous results [1]. □

B. Complete path planning optimization problem

Next, in order to minimize the B-spline curve's length as similar to our previous works [1], [2], we combine the constraints (21)–(24) with the quadratic cost function $g(\boldsymbol{P})$ (given by equations (35)–(37) Section IV.B in [1]) to formulate the complete path planning optimization problem:

$$\boldsymbol{P}^* = \arg\min_{\boldsymbol{P}} \quad g(\boldsymbol{P}), \\ \text{subject to constraints (21)–(24)}, \tag{28}$$

with the decision variables $\boldsymbol{P} = [P_1 \cdots P_n]$.

Finally, the reference path $p(t)$ as required in (1) is taken as:

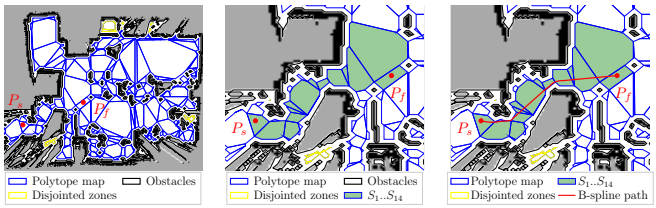
$$p(t) = \boldsymbol{P}^* \boldsymbol{B}_{d,\xi}(t), \tag{29}$$

in which the optimal control points $\boldsymbol{P}^*$ are obtained from solving the optimization problem (28) and the B-spline basis functions $\boldsymbol{B}_{d,\xi}$ as used in (9) is defined with $[t_s, t_f] = [0, 1]$.

V. Navigation with Polytopes Toolbox

The functionalities described in this paper and in our previous work [1] are made publicly available and maintained as toolbox *Navigation with polytopes*[2]. The implementation is done in Python which can be used as stand-alone scripts for research purposes or as a global planner in ROS1 for practical usages. Currently, the toolbox provides three main tasks that can be used independently:

[2] https://gitlab.rob.uni-luebeck.de/robPublic/navigation_with_polytopes



(a) Polytope map from gridmap (b) Finding sequence of polytopes (c) Planning B-spline reference path

Fig. 3: Main tasks of *Navigation with polytopes* toolbox.

- Construction of polytope map from gridmap
- Finding of appropriate sequence of polytopes
- B-spline path planning algorithms.

An example of the outcomes for each task can be seen in Fig. 3 for a given gridmap. Next, we will present the core ideas of each functionality into more detail.

A. Construction of polytope map from gridmap

The following procedure is used for constructing a polytope map from a standard gridmap (c.f. Fig. 3a):

1) Extract the outer boundary of the whole free space by using the command *findContours* with option RETR_EXTERAL of OpenCV toolbox[3]. If there are disjointed regions, the one containing the current position would be selected.
2) Extract the boundaries of all the inner obstacles by using again *findContours* with option RETR_LIST.
3) Simplify the aforementioned contours by using RDP (Ramer–Douglas–Peucker) algorithm[4] [14].
4) Shrink the outer workspace polygon and enlarge all the obstacle boundaries by a safety offset o_p by using Gdspy toolbox[5]. Apply the Boolean operation to remove all the obstacles from the workspace polygon.
5) Partition the obstacle-free polygon (possibly with holes) into connected polytopes by using Mark Bayazit's algorithm[6] [15].

B. Finding of appropriate sequence of polytopes

After obtaining polytope map, a sequence of connected polytopes allowing to travel safely from P_s to P_f is obtained with the function *gen_way_with_minimal_distance* which performs a graph search through all the polytopes:

1) Seek for two polytopes S_1 and S_q from (5) that contain P_s and P_f respectively by using function *find_suitable_zone* in the toolbox (c.f. Remark 3).
2) Search for all traversable routes starting from S_1 by using the function *gen_map*. It takes the polytope map and P_s as inputs and returns a list of all the possible sequences of connected polytopes starting from S_1 to every other regions (if possible).
3) Select sequences which can lead to S_q (the polytope which contains the goal P_f) and return the one which allows to travel from P_s to P_f with minimal distance.

[3] https://opencv.org/
[4] https://github.com/biran0079/crdp
[5] https://github.com/heitzmann/gdspy
[6] https://github.com/wsilva32/poly_decomp.py

Remark 3: If the polytope map consists of some disjointed regions, the one containing the starting point P_s (i.e., the current robot's position in our implementation) is selected as the primary region. Next, if the goal P_f is not inside the polytope map or belongs to an isolated region, then the nearest polytope within the primary region is selected. □

C. B-spline path planning algorithms

For planning the reference B-spline path, the toolbox provides the function *bspline_path_planner_polytope_map* which receives five important parameters: P_s, P_f, the polytope map, the degree d of the curve and *method*. There are three options for $method$:

1) *bezier_min* calls the algorithm which uses a minimal number of control points from [1],
2) *bezier_guarantee* (default option) is an implementation of Proposition 1 with guaranteed solution,
3) *bspline_guarantee* returns the feasible solution of Proposition 1 as given in (25).

The function returns both the path defined as a list of points, and the B-spline control points $\boldsymbol{P}$ for constructing the analytical formulation $z(t)$ (9) of the path if needed. Furthermore, around 250 values of the B-spline-to-Bézier transformation matrix $A(d,n)$ as used in (16) are calculated up to degree 5 and stored in the folder *conversion_data*. New values of $A(d,n)$ can be calculated from the script *save_conversion_matrix.py* given in the same folder. We implement the optimization problem (28) in Pyomo [16] and solve it with the solver IPOPT [17].

D. Discussions

We have finished presenting the main functionalities of the toolbox with necessary information of inputs/outputs as well as adjustable parameters. It is worth mentioning that the toolbox can serve as a framework for applying various existing control techniques to realistic navigation problems given the gridmap data. The reconstruction of a gridmap into a polytope map as well as the finding of an appropriate polytope sequence allow a simplified and efficient way to represent the obstacle-free environment with only linear constraints. They actually are applicable for usages in optimal control such as MPC (Model Predictive Control) and Mixed-Integer-Programming [18]–[20]. E.g., we introduce in [18] an MPC controller for safe navigaton of a mobile robot within a polytope which can push the system far away from the selected boundaries such as walls. This controller as well as further improvements will be added to the toolbox in the future.

VI. Instruction for Usage & Validation

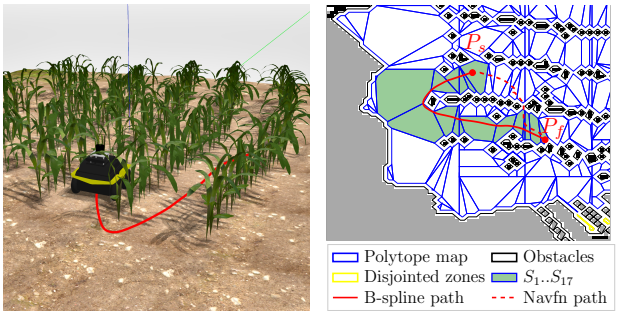
This section presents illustrative validation results of the *Navigation with polytopes* toolbox. We firstly introduce the path planning results within several gridmaps collected from complex and realistic environments and then give a brief introduction on how to integrate the toolbox with ROS. We further explain the parameters required for each application.

A. Path planning results in gridmaps

In order to validate the whole path planning process (i.e. consisting of three main tasks as illustrated in Fig. 3), we consider two gridmaps named as follows:
i) *Field Map*: from a simulated agricultural corn field (c.f. Fig. 4b) provided by Field Robot Event [7],
ii) *Lab Map*: from our laboratory with office furniture.
In both scenarios, the maps are obtained using the Clearpath Jackal mobile robot equipped with a OS1-16 lidar [1] using ROS Gmapping package [21]. The parameters required for the path planning process are given in Table I. Note that, $\varepsilon_{rdp,o}$ and $\varepsilon_{rdp,i}$ are used for simplifying the outer and inner contours with the RDP algorithm (c.f. Section V-A).

TABLE I: Parameters used for constructing polyotope map and planning B-spline path

Parameter	Notation	Value	
		Field Map	Lab Map
Degree	d as in (9)	2	2
No. of B-spline points	n as in (21)	34	32
No. of Bézier points	$\overline{n}$ as in (17)	65	61
Outer RDP simplification	$\varepsilon_{rdp,o}$	0.1	0.5
Inner RDP simplification	$\varepsilon_{rdp,i}$	0.5	0.5
Safety offset	o_p	1	2
Map size in pixels	undefined	320×320	384×384



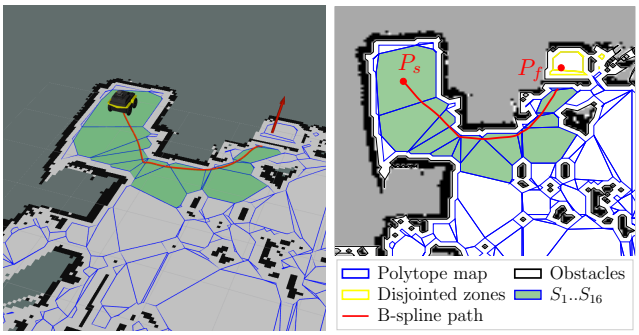
(a) Simulated field in Gazebo. (b) Results in Python.

Fig. 4: Path planning results on *Field Map*.

The results for both scenarios are shown in Figures 4-5. The polytope maps are plotted in blue while the sequences of polytopes from P_s to P_f are filled with green. From Fig. 4, it can be seen that *Field Map* is a complicated case for constructing the polytope map because of numerous scattered obstacles. Our previous method used in [1] actually failed for this situation but thanks to the upgrade (i.e. using *findContours* as presented in Section V), the current toolbox can handle this complexity level. For both scenarios, our toolbox has found the appropriate sequence of polytopes (green regions in Figures 4b and 5) which actually allows for the shortest route to the goal. It is also worth mentioning that in Fig. 5, when the goal is selected in a disconnected region (yellow polytopes), the algorithm chooses the closest polytope in the primary region, thus validating the robustness of the package. Finally the reference path (plotted in red) is planned using our B-spline path planning function provided in the toolbox with option *bezier_guarantee* (i.e., the algorithm presented in Proposition 1) and with the minimal-length objective. In both Figures 4-5, the planned paths seem to be short (as their turnings are close to obstacles but keeping a safety distance) and completely remain inside the polytopes which make the reference paths both safe and optimal. For comparison, in Fig. 4b, the dotted red path is planned using the default global planner *Navfn* of ROS Navigation Stack [6]–[8] which inapplicably guides the robot across the field. Regarding the computation, the average time for generating the polytope map is 600 ms and 1200 ms while the whole process takes 1000 ms and 3800 ms for for *Lab Map* and *Field Map* respectively on a laboratory computer (Intel Core i7-9750H, 2.60GHz).

B. ROS integration

For ease of use and integration into existing projects, we provide a ROS1 package as part of the toolbox which provides two ROS nodes. The first node *poly_map_construct*



(a) Visualization in Rviz. (b) Results in Python.

Fig. 5: Path planning results on *Lab Map*.

creates a polytope map from a given gridmap according to the procedure given in Section V-A. The latter one named *bspline_path_planner_node* performs the whole path planning process given the current pose and a goal. It executes the same functions as the first node and furthermore, seeks for an optimal sequence of polytopes and plans a B-spline path as described in Section V. Fig. 5a shows the results of the whole procedure on the *Lab Map* which is obtained with ROS using the aforementioned nodes. All the polytopes (plotted in blue) are visualized in Rviz using *jsk_visualization*[8] package.

VII. Conclusions and Future Works

In this paper, we firstly introduced a new B-spline path planning algorithm in a sequence of connected polytopes. The algorithm uses the equivalent Bézier format due to its great advantages but adds a guarantee for the solution. Next, we introduced the toolbox *Navigation with polytopes* to facilitate the use of the functions presented in [1] and this paper, given as a global path planner package for ROS. The main functionalities and their implementation were presented with illustrative results in complex simulated and real-world scenarios. Future works include integration of our local controller [18] to the toolbox and consideration of more constraints such as heading angles and curvatures.

[7] https://fieldrobot.nl/event/

[8] https://github.com/jsk-ros-pkg/jsk_visualization

REFERENCES

[1] N. T. Nguyen, L. Schilling, M. S. Angern, H. Hamann, F. Ernst, and G. Schildbach, "B-spline path planner for safe navigation of mobile robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 339–345.

[2] 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.

[3] S. G. Manyam, D. W. Casbeer, I. E. Weintraub, and C. Taylor, "Trajectory optimization for rendezvous planning using quadratic bézier curves," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1405–1412.

[4] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist, "Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles," *IEEE transactions on automation science and engineering*, vol. 7, no. 1, pp. 167–172, 2009.

[5] 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.

[6] 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.

[7] 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.

[8] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[9] 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.

[10] N. T. Nguyen and I. Prodan, "Stabilizing a multicopter using an nmpc design with a relaxed terminal region," *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 126–132, 2021.

[11] 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.

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

[13] N. T. Nguyen and P. T. Gangavarapu, "B-spline-to-bezier conversion and applications on path planning," *submitted to the 2023 European Control Conference (ECC)*, Under Review.

[14] 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.

[15] 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.

[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] 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.

[18] N. T. Nguyen and G. Schildbach, "Tightening polytopic constraint in mpc designs for mobile robot navigation," in *2021 25th International Conference on System Theory, Control and Computing (ICSTCC)*. IEEE, 2021, pp. 407–412.

[19] K. Worthmann, M. W. Mehrez, M. Zanon, G. K. Mann, R. G. Gosine, and M. Diehl, "Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 4, pp. 1394–1406, 2015.

[20] 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.

[21] 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.