

Cubic Bézier Local Path Planner for Non-holonomic Feasible and Comfortable Path Generation

Guillaume Vailland, Valérie Gouranton, Marie Babel

▶ **To cite this version:**

Guillaume Vailland, Valérie Gouranton, Marie Babel. Cubic Bézier Local Path Planner for Non-holonomic Feasible and Comfortable Path Generation. ICRA 2021 - IEEE International Conference on Robotics and Automation, May 2021, Xian, China. pp.7894-7900. hal-03188287

HAL Id: hal-03188287
https://hal.science/hal-03188287

Submitted on 1 Apr 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Cubic Bézier Local Path Planner for Non-holonomic Feasible and Comfortable Path Generation

Guillaume Vailland
Univ Rennes, INSA Rennes, Inria, CNRS, Irisa
Rennes, France

Valérie Gouranton
Univ Rennes, INSA Rennes, Inria, CNRS, Irisa
Rennes, France

Marie Babel
Univ Rennes, INSA Rennes, Inria, CNRS, Irisa
Rennes, France

***Abstract*—In the case of non-holonomic robot navigation, path planning algorithms such as Rapidly-exploring Random Tree (RRT) rarely provide feasible and smooth paths without the need of additional processing. Furthermore, in a transport context like power wheelchair navigation, passenger comfort should be a priority and influence path planning strategy. In this paper, we propose a local path planner which guarantees bounded curvature value and continuous Cubic Bézier piecewise curves connection. To simulate and test this Cubic Bézier local path planner, we developed a new RRT version (CBB-RRT*) which generates on-the fly comfortable path adapted to non-holonomic constraints.**

***Index Terms*—Non-holonomic Motion Planning; Motion and Path Planning; Human Factors and Human-in-the-Loop**

I. Introduction

Path planning is an important field of robotics, control theory and artificial intelligence [1]. It refers to the process of finding a way from a start state to a goal state under constraints. In particular, path planning is crucial for autonomous and semi-autonomous robot navigation in unknown environments. It allows the robot to find safe control states or a safe path to reach a given goal state. Among the multiple path planning algorithms that exist, the Rapidly-exploring Random Tree (RRT) [2] algorithm stands out for its efficiency in quickly exploring unknown spaces and for its low structure complexity, which allows it to be easily optimized or upgraded. Indeed, over the years, many versions of RRT have been proposed [3]–[5] among which an asymptotically optimal version RRT* [6] that has largely contributed to its widespread use. RRT and RRT* both generate a collision-free path composed of piecewise linear paths between states.

However, a trajectory composed of linear piecewise paths is not suitable for non-holonomic robots, in particular for differential robots such as power wheelchairs. Indeed, their kinematic and dynamic constraints prevent them from to follow path with too much curvature. In addition, within a differential robot transport scenario, user comfort is closely linked to angular accelerations and jerk energy [7]. As such, comfort also depends on the smoothness of the trajectory i.e. the trajectory curvature [8], [9]. Even when walking, people tend to limit curvature maximum value and variations [10]. In order for a path generated by a path planner to be both feasible by a differential drive robot and comfortable for a transported user, it must be smoothed by limiting curvature variation and curvature maximum value. To this end, 3 different approaches exist.

The first one consists in handling curvature constraints at the control level while following a reference path. In [11], a smoothly path following method respecting kinematic constraints and based on Clothoid is presented. Authors in [12] propose a Lyapunov-based feedback control law along with a path following strategy consisting in reaching reference path states one after the other. Their control law regenerates a feasible local path between each state and provides comfortable motion based on curvature velocity adaptation. However, relying on control to ensure path smoothness can lead to a sub-optimal path, including unnecessary turns as the reference path is not smooth.

The second common approach applies a smoothing algorithm on the reference path after its planning and before the control step. Post-smoothing algorithms take a reference path composed of different states or control points and tweak piecewise edges to generate a path respecting smoothing constraints. Multiple smoothing methods exist based on different interpolation mechanics and typical curves like Dubin's, Clothoid or Bézier curves. In [13], low degree Bézier curves are used in corners and sharp turns to obtain a continuous curvature path *i.e.* $G2\text{-}continuous$. The works [14] and [15] present a method based on Bézier transition curves, also known as Bézier spirals, to smooth a series of states into a $G2\text{-}continuous$ path. In [16], authors also use Bézier curves to smooth sharp turns and connect piecewise linear paths with each other. The algorithm in [9] generates a comfortable path by joining the states while minimizing curvature variation, integral and curve length. It uses the $G1$ Hermite Interpolation Problem with Clothoid and then tweaks angles to achieve $G2\text{-}continuity$. Authors in [17] generate a $C2\text{-}continuous$ and locally controllable path from a set of points by inserting Quintic Bézier curves between segments. In [18], authors propose an innovative method called Gradient Informed Path Smoothing (GRIPS). Unlike other algorithms, GRIPS deforms the path shape by locally optimizing the state locations. As such, this algorithm is able to correct locally sub-optimal states.

However, post-smoothing algorithms require extra execution time and sometimes additional expensive collision checks [9] while the path planner already spent time to ensure collision-free states. In addition, a post-smoothing generated path might

be globally sub-optimal as states disrespecting curvature constraints are selected in the first place. Starting from GRIPS, if locally correcting state locations to integrate more constraints is a viable solution, one could ask why not directly integrate these constraints when generating states and path. This is the main idea behind the third approach to enhance path smoothness *i.e.* incorporating curvature constraints inside the path planner algorithm to generate feasible and comfortable path on-the-fly. Many solutions adapted to different path planners exist [19], [20]. Focusing on RRT, multiple solutions emerged over the years using the same tools *i.e.* interpolation curves and post-smoothing methods, but in a realtime approach.

The solution proposed in [21] executes a local check of path feasibility before state connection and generates a valid path based on Bézier piecewise curves with null curvature at each states, thus ensuring $G2\text{-}continuity$. Authors also present different sampling, collision-checking and pruning techniques to enhance performances of the RRT, then defining the Spline RRT (SRRT). However SRRT fails to generate piecewise path on-the-fly as each new state connection requires the complete re-parameterization of the path from the start to the new state instead of simply appending a new independent piecewise Bézier curve path as presented in [22]. Indeed, their solution uses independent Bézier curve as piecewise path after approximating the cost to go to a new state using Dubin's curves. However, while the use of seventh-order Bézier curves ensures curvature continuity with null curvature at each state, it also leads to heavy polynomial calculations. Furthermore, the Bézier curves and their curvature values are not involved in the evaluation of the distance cost to a new state: this can still lead to inconsistent state choices in terms of comfort. In [23], authors proposed a new local path planner based on a Lyapunov control law to generate piecewise curves and to evaluate distance between states. The method takes into account comfort constraints and eliminates inaccessible ones. [24] also proposed a different local path planner based on Clothoid $G1$ fitting to extend constraints and to ensure the elimination of inaccessible states. However, discontinuities between piecewise Clothoid path remain, due to Clothoid representation complexity and to $G1$ fitting algorithm. Authors in [25] and [26] include the Line-of-Sight algorithm (LOS) in their path planning process to check if two states can be connected under non-holonomic constraints in addition to the usual collision check.

All those methods proved themselves efficient and functional. Yet, they are still suffering from drawbacks, either because of the use of complex piecewise curves or high degree splines or because they do not fulfill all comfort and non-holonomic kinematic constraints. As such, in this paper, we propose a new local path planner based on low degree Cubic Bézier curves. Combined with path planning algorithms such as RRT*, it generates comfortable and non-holonomic robot compliant paths on-the-fly. We first present the local path planner in section II. Section III presents a new RRT* version, called CBB-RRT*, based on our local path planner. CBB-RRT* performances are exposed and compared to existing RRT* ones. Finally, we discuss our local path planner and implementations limitations and usability in section IV.

II. Cubic Bézier Based Local Path Planner

A. Problem Statement

Considering a two-dimensional space X with obstacles whose position and geometry are known and 2 states S_{init}, S_{goal}. X_{free} refers to X space free of obstacle. Path planners like RRT address the problem of finding a valid path from S_{init} to S_{goal} if it exists. A path is considered valid if it meets all path planner constraints. The most usual global constraint is for the path to be collision-free *i.e.* belonging to X_{free}. However, in order for the path to be feasible by a non-holonomic robot and comfortable for a transported user, more constraints need to be satisfied. A path is comfortable for a user if its maximum curvature value and curvature variations are minimized as a pedestrian would naturally do [10]. Indeed, curvature is inversely proportional to angular acceleration and is linked to jerk energy, which are known to be discomfort factors [8]. A path is feasible by a non-holonomic robot if it fulfills its kinematic and dynamic constraints. Those constraints are also represented by the maximum curvature of the path as the maximum curvature is inversely proportional to the minimum curvature radius that the robot is able to achieve. Finally, to enhance path stability, the path should be $G2\text{-}continuous$, meaning that its curvature should be continuous.

If the path is defined as a parametric curve $g(t)$ where t stands for the curve parameter in $\mathbb{R}$, then the curvature $\kappa(t)$ can be calculated as

$$\kappa(t) = \frac{|\dot{g}(t) \times \ddot{g}(t)|}{|\dot{g}(t)|^3}. \tag{1}$$

As shown in (1), the path parameterization should be at least second order differentiable. In order to obtain a valid path, its absolute maximum curvature value must be less than the maximum value achievable by the robot and tolerated by the user. To summarize, the generated path must fulfill 3 main constraints :

$$\begin{cases} |\kappa(t)| \leq \min(\kappa_{robot}, \kappa_{user}), \\ g(t) \text{ is } G2\text{-}continuous, \\ g(t) \in X_{free}, \end{cases} \tag{2}$$

with κ_{robot} the maximum curvature achievable by the robot and κ_{user} the maximum curvature tolerated by the user. Note that the curve should also globally limit curvature variations, thus limiting sharp turns. While the third constraint is generally fulfilled by the path planner, we present in the following section a new local path planner to produce piecewise paths that meet curvature constraints and result in a global $G2\text{-}continuous$ path once merged together.

B. Proposed Local Path Planner

To generate a path, several path planners, like RRT, connect piecewise curves between states. In most of them, those piecewise curves are linear and do not comply with any

constraints other than not crossing an obstacle. We here propose a new local path planner based on Cubic Bézier curves to validate new states and to connect them to the tree. Bézier curves are parametric curves whose shape can be modified by tweaking a set of control points. The main advantage of Bézier curves lies in their simple parametrization based on Bernstein polynomial $B_i(t)$. Cubic Bézier curves are defined by a set of four control points P_0, P_1, P_2, P_3 and a third order Bernstein basis polynomial such as $P(t)$ the Cubic Bézier curve parametrization is defined by

$$P(t) = w_0 * B_0(t) * P_0 + w_1 * B_1(t) * P_1 + w_2 * B_2(t) * P_2 + w_3 * B_3(t) * P_3, \quad (3)$$

$$B_0(t) = (1-t)^3, \qquad B_1(t) = 3 * t * (1-t)^2,$$
$$B_2(t) = 3 * t^2 * (1-t), \qquad B_3(t) = t^3,$$

with w_i weighting basis coefficients. In this paper $w_i = 1$ for each i in order to simplify calculations. Using equations (3) and (1) we can compute the Cubic Bézier curve's curvature for any value of t. In particular, as shown in [27], its curvature at $t = 0$ is given by

$$\kappa(t_0) = \frac{w_0 w_2}{w_1^2} \frac{n-1}{n} \frac{h}{|P_0 - P_1|^2} = \frac{2}{3} \frac{h}{|P_0 - P_1|^2}, \quad (4)$$

with n being the Bézier curve order and h the perpendicular distance from P_2 to the line P_0P_1 as shown in Fig. 1.

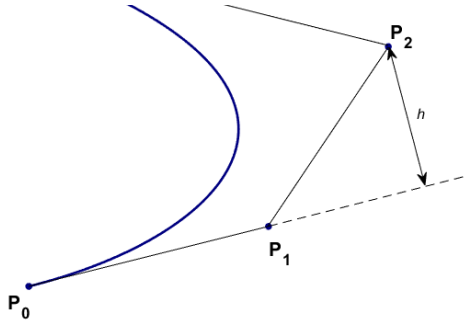


Fig. 1: Initial curvature. The Cubic Bezier curve is in blue and its three first control points are noted as P_0, P_1 and P_2

Using those expressions, we can ensure that each piecewise curve composing the final path meets the maximum curvature constraint stated in (2). Indeed, we can evaluate any piecewise curve curvature and invalidate the curve if its value exceed maximum curvature value κ_{max}. As the final path is a composition of those piecewise Bézier curves, its final curvature will not exceed the maximum predefined value.

The last constraint to fulfill is the curvature continuity of the final path *i.e.* the curvature continuity between piecewise curves. Considering a new state S_{new} and a possible state $S_{candidate}$ with which S_{new} can be connected without crossing any obstacle (Fig. 2). $S_{candidate}$ is already accessible through a piecewise Cubic Bézier curve $P(t_p)$, $t_{p_0} \leq t_p \leq t_{p_3}$ with t_p the curve P parameter in $\mathbb{R}$ and t_{p_i} the parameter value so that $P(t_{p_i}) = P_i$. S_{new} could be connected to $S_{candidate}$ using one piecewise Cubic Bézier curve $Q(t_q)$, $t_{q_0} \leq t_q \leq t_{q_3}$ with t_q the curve Q parameter in $\mathbb{R}$ and t_{q_i} the parameter value so that $Q(t_{q_i}) = Q_i$.

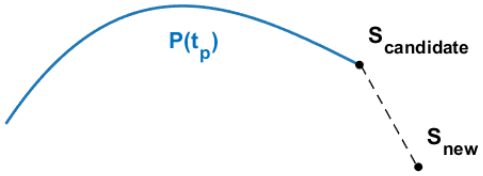


Fig. 2: New default state connection case with $P(t_p)$ a C-Shaped Bézier Curve in blue, $S_{candidate}$ an accessible state and S_{new} the new state to connect.

We need to ensure that at a link state S_{link} between the two piecewise Bézier curves, both $P(t_p)$ curvature $\kappa_p(t_p)$ and $Q(t_q)$ curvature $\kappa_q(t_q)$ are equal (Fig. 3). In other terms that $\kappa_p(t_{link}) = \kappa_q(t_{q_0})$ with $P(t_{link}) = Q(t_{q_0}) = S_{link}$. In most of related works, piecewise curves are connected at each state such as S_{link} is then equal to $S_{candidate}$.

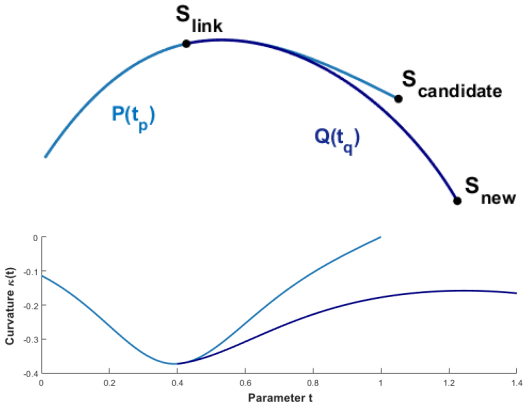


Fig. 3: Default case solution and its curvature profile. Vertical axis gives curvature $\kappa(t)$ and horizontal axis gives parameter t. In this solution, $t_{link} = 0.4$ *i.e.* $P(0.4) = S_{link}$.

The particularity of our work is that $S_{candidate}$ and S_{new} positions are not start and end point of the Bézier piecewise curve. Indeed, they are rather used as third and fourth control point Q_2 and Q_3 respectively. As a result, the final heading of the robot stays tangent to line $S_{candidate}S_{new}$ as a classical RRT path with linear piecewise curves would suggest. But more importantly, the first control point Q_0, equal to the link state S_{link} also belongs to the curve $P(t_p)$ at a given t_{link} with $t_{p_0} \leq t_{link} \leq t_{p_3}$. This configuration allows turns to be anticipated and reduces path curvature as a human would do.

Thus, at a given t_{link}, we know the position of three of the four $Q(t_q)$ control points (Q_0, Q_2 and Q_3). However, $Q(t_q)$ and $P(t_p)$ still need to be $G2$-*continuous* meaning that position, derivative and curvature values of $Q(t_{q_0})$ and $P(t_{link})$ must be equal. To do so, we can tweak the last unknown control point Q_1.

Firstly, in order for $Q(t_{q_0})$ and $P(t_{link})$ positions to be equal, S_{link} must belong to both $P(t_p)$ and $Q(t_q)$ curves. This constraint is fulfilled as far as we take a point on $P(t_p)$ curve as first $Q(t_q)$ curve control point Q_0. Secondly, we can

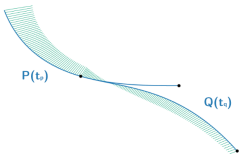
ensure that the derivative of $Q(t_{q_0})$ is equal to $P(t_{link})$ if the Q_1 control point relies on the $P(t_{link})$ tangent. Finally, using equation (4), we can express the $G2\text{-}continuity$ constraint as a system to solve for the Q_1 position according to

$$\begin{cases} Q_0 = P(t_{link}) = Q(t_{q_0}), t_{p_0} \leq t_{link} \leq t_{p_3}, \\ Q_1 \in \dot{P}(t_{link}), \\ k_0 = \frac{2}{3}\frac{h_0}{|Q_0 - Q_1|^2}, \end{cases} \tag{5}$$

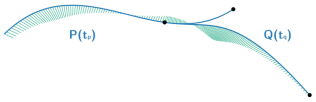
with $\dot{P}(t_{link})$ being the P tangent at t_{link}, k_0 the curvature value of $P(t_p)$ at t_{link} and h_0 the perpendicular distance from Q_2 to the line Q_0Q_1. Regardless of the curvature sign, we find 2 solutions to the system (5) :

$$\begin{cases} x_{q_1} = \pm\sqrt{\frac{2}{3}|\frac{h_0}{k_0}|\frac{1}{1+m_{link}^2}} + x_{q_0}, \\ y_{q_1} = m_{link} * (x_{q_1} - x_{q_0}) + y_{q_0}, \end{cases} \tag{6}$$

with x_{q_i} and y_{q_i} the coordinates of the Q_i control point and m_{link} the slope of $\dot{P}(t_{link})$. Once the correct solution sign is selected depending on the $P(t_p)$ current direction, the final control point Q_1 position is known and the curve $Q(t_q)$ can be drawn as in Fig. 3. Now, the maximum curvature of $Q(t_q)$ can be evaluated and the new state and piecewise curve can be validated or rejected depending on the curvature constraint. Note that the computation of $Q(t_q)$ does not require any change in the $P(t_p)$ curve. This ancestor independence provides more flexibility to the path planner algorithm and better performance as the path does not need to be fully reparametrized with each new state.



(a) C-Shape P Curve, $t_{link} = 0.5$



(b) S-Shape P Curve, $t_{link} = 0.7$

Fig. 4: Both S and C Cubic Bézier shapes P Curve cases and arbitrary set t_{link} value. Curve curvature direction, sign and value are represented by green normals.

It is yet to set the convenient t_{link} which determines S_{link} and Q_0 positions and greatly influences the $Q(t_q)$ curvature profile and maximum. Fig. 4 shows different possible configurations and the resulting $P(t_p)$ and $Q(t_q)$ curves. Furthermore, depending on the $P(t_q)$ curve shape, a certain range of t_{link} values will produce a curvature discontinuity (see Fig. 5). Indeed, when P is S-shaped, for a certain range of t_{link} near

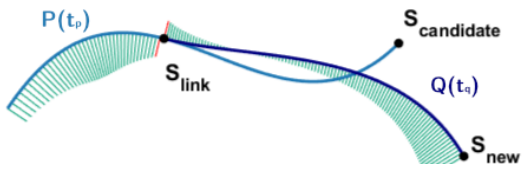


Fig. 5: S-Shape P Curve with $t_{link} = 0.4$ producing an invalid solution.

the inflection point, neither of the two solutions in equation (6) is both in the right direction and of correct sign. This is due to the configuration of the control points and can easily be expressed as conditions. If P is S-shaped, t_{link} will provide a valid solution if the configuration meets the conditions

$$\begin{cases} \text{sign}(x_{\dot{q}_0}) = \text{sign}(k_0) \text{ if } y_{p_3} \geq y_{q_0} + m_{link}(x_{p_3} - x_{q_0}), \\ \text{sign}(x_{\dot{q}_0}) \neq \text{sign}(k_0) \text{ if } y_{p_3} < y_{q_0} + m_{link}(x_{p_3} - x_{q_0}), \end{cases} \tag{7}$$

with x_{q_i}, y_{q_i}, x_{p_i}, y_{p_i} the $P(t_p)$ and $Q(t_q)$ control points coordinates and $x_{\dot{q}_0}$ the x derivative of $P(t_{link})$ whose sign gives P x direction.

Another impossible configuration happens when curvature k_0 is null. In that case we can set Q_1 equal to Q_2 and the resulting path will remain $G2\text{-}continuous$ as long as previously stated conditions are fulfilled (equation (7)). Note that the validity of t_{link} only depends on $P(t_p)$ curve's shape and control points. Thus, a consistent t_{link} value for future cubic Bézier curve connection can be computed regardless new state position. Note that our method is applicable in any dimension as it relies on n-dimensional properties. The use of a two-dimensional space in this paper is only to facilitate understanding.

This proposed local path planner allows to generate a Cubic Bézier piecewise curve between 2 states and to evaluate its curvature. Then, this curvature value can be used as a piecewise curve validation criterion to fulfill constraints (equation (2)), as well as a local piecewise path cost value in the overall algorithm process.

III. Cubic Bézier Based RRT* (CBB-RRT*)

RRT is a path planner that allows fast exploration of an unknown environment [2]. RRT generates a collision-free path by combining piecewise linear paths between random states. Over the years, many enhanced versions have been proposed. One of the most used is the RRT* version that generates an asymptotically optimal tree by introducing a rewiring step to the RRT tree extension method [6]. In RRT*, a random state is found but not directly linked to the nearest in-tree state. State proximity is determined using euclidian distance *i.e.* a state is considered to be near to another if its euclidian distance is under a certain value η. First, the nearest in-tree state is use to create a new state in the direction of the random state at η maximum distance. Then, the new state is connected to the near in-tree state that provides the cheapest local piecewise path cost becoming the new in-tree state. This two step approach allows the use of different evaluation cost methods to find the nearest state, the near states and the final parent state of the steered state. Finally, once the new state is in tree, the rewiring step occurs *i.e.* the steered state becomes the parent of its near in-tree states if the cost to reach them is lower than the actual.

A. CBB-RRT* algorithm and rewiring strategy

We implemented a Cubic Bézier based RRT* version (CBB-RRT*) using the proposed Cubic Based local path planner (algorithm 1) to test it in a real case scenario. RRT* was

chosen because it is the simplest asymptotically optimal RRT version and largely used in various domains. However, the presented path planner could be implemented in the exact same way in various RRT versions as a replacement of the linear local path planner [25], [28]–[34]. As in RRT* [6], $InitTree()$ method initialises the tree $\mathcal{T}$ with a root state and $RandomState()$ method uniformly generates a random state s_{random}. $Extend()$ method extends $\mathcal{T}$ by one new state s_{new} as described earlier and the $Near()$ method finds all the s_{new} near in-tree states S_{near} based on euclidian distance like RRT*. In its overall process, CBB-RRT* only differs from

Algorithm 1: CBB-RRT* body

```
1  T ← InitTree(s_init); S ← ∅;
2  while i < N do
3      if S.Empty() then
4          s_random ← RandomState();
5          i ← i + 1;
6      else
7          s_random ← S.PopBack();
8      s_new ← Extend(T, s_random);
9      S_near ← Near(T,s_new);
10     for s_near ∈ S_near do
11         if HasBeenRewired(s_near) then
12             UpdateDescendants(T, s_near);
```

RRT* algorithm by its rewiring strategy in order to be compliant with the proposed local path planner. Indeed, the Cubic Bézier based local path planner generates piecewise paths dependent from their parent curve shape and parametrization. Thus, rewiring a state, meaning changing its piecewise curve, will recursively invalidate descendant piecewise path. In other word, if a piecewise path changes, continuity with next piecewise paths is broken which invalidates all path. Therefore, using our proposed local path planner requires another rewiring strategy than the RRT* one.

Algorithm 2: UpdateDescendants Method

```
1  S_children ← GetChildren(T, s_near);
2  for s_child ∈ S_children do
3      if MeetConstraints(s_near, s_child) then
4          T.UpdatePiecewiseCurve(s_near, s_child);
5          UpdateDescendants(T, s_child);
6      else
7          S_descendant ← GetDescendants(T, s_child);
8          T.RemoveVertices(S_descendants ∪ s_child);
9          S ← S_descendants ∪ s_child
```

The CBB-RRT* rewiring strategy is an extension of the RRT* rewiring strategy that occurs inside $Extend()$ method. For all near in-tree state, we verify if it has been rewired with $HasBeenRewired()$ method. If it has been rewired, the $UpdateDecendant()$ method is applied. All s_{near} children are gathered in $S_{children}$ using the $GetChildren()$. For all children, if a new piecewise path from s_{near} that meets all constraints can be found using the $MeetConstraints()$ method, then the piecewise curve is updated using $UpdatePiecewieCurve()$ method and a recursive call happens to try to rewire all descendants. Otherwise, all child descendants are gathered in $S_{descendant}$ using $GetDescendants()$ method, and removed from the tree using the $RemoveVertices()$ method and saved in a state stack S. To avoid losing too many states, all states in S will be reprocessed by CBB-RRT* as a random state. Indeed, if S is not empty (Alg. 1 Line 3), the last state from S will be used as the new random state (Alg. 1 Line 7)

B. Simulation results

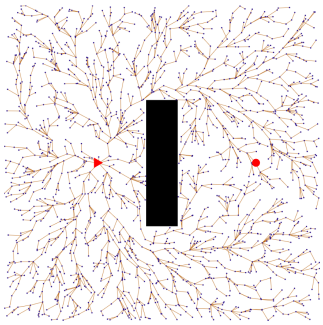
To show the impact of our local path planner once implemented in a RRT version, we compared CBB-RRT* to RRT* by running both algorithms in similar known 2D environments For both algorithms the random states came in the same order from the same uniformly sampled set. In all simulations, the maximum distance and near criterion has been arbitrary set to $\eta = 4$ and the number of iterations to $N = 2000$. In CBB-RRT*, the link parameter value has been set to $t_{link} = 0.5$ for every C-Shaped Cubic Bézier piecewise curve and set to a multiple value of curve inflection t when S-Shaped. In a dynamic path planning scenario, maximum curvature value κ_{max} can be set according to robot linear velocity to ensure respecting robot kinematic constraints and comfort standard like the ones stated in [8]. For those static simulation, the maximum curvature was arbitrary set to $\kappa_{max} = 1.0$.

As we can see in Fig. 6, trees generated by CBB-RRT* and their overall exploration area are smaller. This is due to a more constrained state validation but also to the different rewiring strategy. Furthermore, overall CBB-RRT* tree structure and descendants appeared different even though composing states are similar. Indeed, using our local path planner as a cost evaluation function, CBB-RRT* prioritizes reachable states with smaller curvature turns and generates a curvature bounded continuous path as shown in Fig. 7. As no curvature constraints is considered in RRT* and as its generated path is unfeasible by a non-holonomic robot without further process, no curvature profile for RRT* is available.

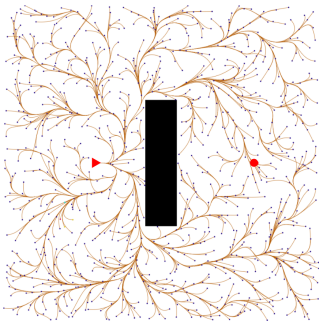
IV. Discussion

A. Usability and Limitations

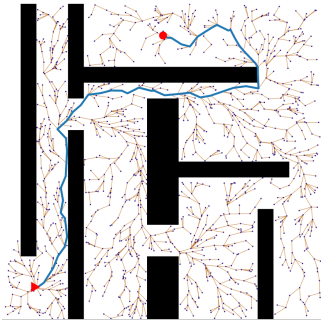
While relying on RRT* features like fast exploration and asymptotical optimallity, CBB-RRT* generates a smooth and comfortable path that meets all previously stated constraints (equation (2)). Indeed, the use of our local path planner guarantees path curvature continuity and limitation on which comfort and feasibility depend. Because path curvature is bounded by a specified value, it is achievable by any non-holonomic robots. In addition, the path provided by CBB-RRT* is parametrized. Path can be discretized and position, heading and curvature are known at any time. This is a huge advantage at control step as the robot can adapt its velocity to respect comfort constraints. Furthermore, our local path planner could be used within any other path planning, path



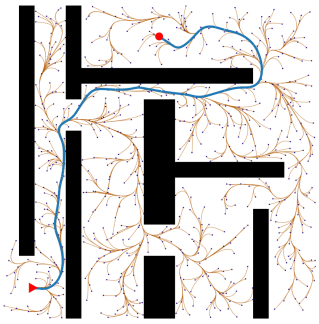
(a) RRT* in a simple environment.



(b) CBB-RRT* in a simple environment.



(c) RRT* in a complex environment.



(d) CBB-RRT* in a complex environment.

Fig. 6: RRT* and CBB-RRT* with $\eta = 4.0$, $N = 2000$, $\kappa_{max} = 1.0$. Red triangle represents the start state and its direction and red circle represents the goal. Blue lines represent complex environment solution paths.

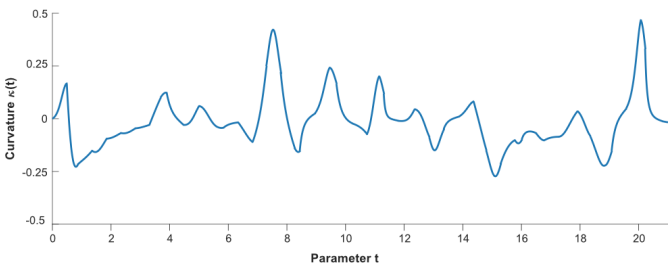


Fig. 7: Fig 4.(d) solution path curvature profile. Vertical axis gives curvature $\kappa(t)$ and horizontal axis gives parameter t.

smoothing and control algorithms such as more advanced RRT versions like NC-RRT [28], Risk-DTRRT [25], DT-RRT* [30], BIT* [32] or RRT# [34]. Beyond control and path planning domains, our local path planner could be useful in data interpolation or design fields [35] as a solution to $G2\text{-}continuous$ piecewise connection with only third degree polynomial curves [36].

However, using our local path planner, CBB-RRT* breaks piecewise curve symmetry and requires a specific rewiring strategy to work. This is due to the ancestor dependency of Cubic Bézier piecewise curves. As such, it takes more iterations than RRT* for the same amount of exploration and its rewiring complexity is higher. Also, generating piecewise Cubic Bézier curves instead of linear paths is more expensive in terms of control points calculations, collision checks and data transfers. Nonetheless, many optimizations can be developed to enhance CBB-RRT* overall performance.

B. Optimizations

Note that for comparative purposes, each CBB-RRT* algorithm step has been modified as little as possible whereas it could be optimized for the specific use of our local path planner. For example, a more precise proximity evaluation function would allow to refine the nearest state selection and connect random states more often. To this end, the proximity function could take into account heading difference between states, or piecewise curve length approximation like in [22] with Dubin's curve, or piecewise curvature value using our local path planner. Even though the rewiring strategy has been extended and differs from the original,it can be optimized by a profound redesign based on more recent strategies [5], [33]. Finally, the sampling strategy that generates random states can also be improved by using or combining various solutions, such as goal oriented sampling [29], [37], incremental direct sampling [31], [32], ellipsoidal direct sampling [38] and weighted sampling [4].

Regarding our local path planner, some choices would deserve further attention and investigation. For example, tweaking the weight of the Cubic Bézier basis coefficients w_i could change the curve shape and provide smoother local piecewise curve while still meeting constraints as they are involved in the initial curvature (equation (4)). Also, the use of polynomial based Cubic Bézier curve was motivated by calculation simplicity but using other Cubic Bézier based curves could be considered such as T-Bézier curves [39] or H-Bézier curves [40]. Indeed, those non-polynomial Bézier based curves provide more flexibility and tweaking parameters than polynomial based ones. Finally, evaluation of piecewise curve maximum curvature and the finding of the ideal t_{link} value could be optimized using lookup tables or more advanced strategies.

V. Conclusion

This paper presents a new Cubic Bézier based local path planner that generates $G2\text{-}continuous$ Cubic Bézier piecewise curves. This local path planner could find interests in many different fields. In a non-holonomic transport robot path planning context like power wheelchair navigation, trajectory curvature is directly linked to feasibility and comfort. As such, we presented the CBB-RRT* algorithm, a classical RRT* version using the proposed local path planner. While some CBB-RRT* computational steps have to be further investigated to improve its performance, it generates smooth, comfortable paths, feasible by non-holonomic robots without any additional steps.

VI. Aknowledgement

This work is carried out as part of the INTERREG VA FMA ADAPT project "Assistive Devices for empowering disAbled People through robotic Technologies" http://adapt-project.com/index.php. The Interreg FCE Programme is a European Territorial Cooperation program that aims to fund high quality cooperation projects in the Channel border region between France and England. The Program is funded by the European Regional Development Fund (ERDF)

REFERENCES

[1] S. M. LaValle, "Planning algorithms," *Cambridge University Press*, 2006.

[2] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning," vol. 1946, 1998.

[3] S. M. LaValle and J. J. Kuffner, "Rapidly-Exploring Random Trees: Progress and Prospects," *Algorithmic and computational robotics: new directions*, vol. 5, pp. 293–308, 2001.

[4] L. Janson, E. Schmerling, and M. Clark, AshleyPavone, "Fast marching trees: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[5] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[6] S. Karaman and E. Frazzoli, "Incremental Sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, no. 2, 2010.

[7] L. Labakhua, U. Nunes, R. Rodrigues, and F. S. Leite, "Smooth trajectory planning for fully automated passengers vehicles," *Third International Conference on Informatics in Control, Automation and Robotics*, 2006.

[8] R. Solea and U. Nunes, "Trajectory planning with velocity planner for fully-automated passenger vehicles," in *IEEE Intelligent Transportation Systems Conference*, pp. 474–480, IEEE, 2006.

[9] P. Bevilacqua, M. Frego, E. Bertolazzi, D. Fontanelli, L. Palopoli, and F. Biral, "Path planning maximising human comfort for assistive robots," in *IEEE Conference on Control Applications*, pp. 1421–1427, IEEE, 2016.

[10] G. Arechavaleta, J. P. Laumond, H. Hicheur, and A. Berthoz, "An optimality principle governing human walking," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 5–14, 2008.

[11] M. D. Cecco, E. Bertolazzi, G. Miori, R. Oboe, and L. Baglivo, "PC-sliding for vehicles path planning and control : Design and Evaluation of Robustness to Parameters Change and Measurement Uncertainty," in *4th International Conference on Informatics in Control, Automation and Robotics*, pp. 11–18, 2007.

[12] J. J. Park and B. Kuipers, "A smooth control law for graceful motion of differential wheeled mobile robots in 2D environment," in *IEEE International Conference on Robotics and Automation*, pp. 4896–4902, IEEE, 2011.

[13] J. W. Choi, R. E. Curry, and G. H. Elkaim, "Continuous Curvature Path Generation Based on Bézier Curves for Autonomous Vehicles," *International Journal of Applied Mathematics*, vol. 40, no. 2, 2010.

[14] K. Yang and S. Sukkarieh, "An Analytical Continuous-Curvature Path-Smoothing Algorithm," *IEEE Transaction On Robotics*, vol. 26, no. 3, pp. 561–568, 2010.

[15] K. Yang, D. Jung, and S. Sukkarieh, "Continuous curvature path-smoothing algorithm using cubic Bézier spiral curves for non-holonomic robots," *Advanced Robotics*, vol. 27, no. 4, pp. 247–258, 2013.

[16] F. Zhou, B. Song, and G. Tian, "Bézier curve based smooth path planning for mobile robot," *Journal of Information and Computational Science*, vol. 8, no. 12, pp. 2441–2450, 2011.

[17] K. Renny Simba, N. Uchiyama, and S. Sano, "Real-time smooth trajectory generation for nonholonomic mobile robots using Bézier curves," *Robotics and Computer-Integrated Manufacturing*, vol. 41, pp. 31–42, 2016.

[18] E. Heiden, L. Palmieri, S. Koenig, K. O. Arras, and G. S. Sukhatme, "Gradient-Informed Path Smoothing for Wheeled Mobile Robots," in *IEEE International Conference on Robotics and Automation*, pp. 1710–1717, IEEE, 2018.

[19] S. McLeod and J. Xiao, "Real-time adaptive non-holonomic motion planning in unforeseen dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4692–4699, IEEE, 2016.

[20] J. A. Silva and V. Grassi, "Clothoid-based global path planning for autonomous vehicles in urban scenarios," in *IEEE International Conference on Robotics and Automation*, pp. 4312–4318, IEEE, 2018.

[21] K. Yang, "An efficient Spline-based RRT path planner for non-holonomic robots in cluttered environments," in *International Conference on Unmanned Aircraft Systems*, pp. 288–297, IEEE, 2013.

[22] A. A. Neto, D. G. Macharet, and M. F. Campos, "Feasible RRT-based path planning using seventh order Bézier curves," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1445–1450, IEEE, 2010.

[23] J. J. Park and B. Kuipers, "Feedback motion planning via non-holonomic RRT* for mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4035–4040, IEEE, 2015.

[24] M. Sobue and H. Fujimoto, "RRT-Based Path Planning Considering Initial and Final Pose under Curvature Constraints for Nonholonomic Wheeled Robot," in *45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 724–731, IEEE, 2019.

[25] W. Chi, C. Wang, J. Wang, and M. Q. Meng, "Risk-DTRRT-Based Optimal Motion Planning Algorithm for Mobile Robots," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 3, pp. 1271–1288, 2018.

[26] "Autonomous Robotic Exploration by Incremental Road Map Construction," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1720–1731, 2019.

[27] Thomas W. Sederberg, "Computer aided geometric design," 2012.

[28] X. Wang, X. Luo, B. Han, Y. Chen, G. Liang, and K. Zheng, "Collision-free path planning method for robots based on an improved rapidly-exploring random tree algorithm," *Applied Sciences*, vol. 10, no. 4, 2020.

[29] G. Kang, Y. B. Kim, Y. H. Lee, H. S. Oh, W. S. You, and H. R. Choi, "Sampling-based motion planning of manipulator with goal-oriented sampling," *Intelligent Service Robotics*, vol. 12, no. 3, pp. 265–273, 2019.

[30] L. Chen, Y. Shan, W. Tian, B. Li, and D. Cao, "A Fast and Efficient Double-Tree RRT*-Like Sampling-Based Planner Applying on Mobile Robotic Systems," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 6, pp. 2568–2578, 2018.

[31] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, "Regionally accelerated batch informed trees (RABIT*): A framework to integrate local information into optimal path planning," in *IEEE International Conference on Robotics and Automation*, pp. 4207–4214, IEEE, 2016.

[32] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *IEEE International Conference on Robotics and Automation*, pp. 3067–3074, IEEE, 2015.

[33] C. B. Moon and W. Chung, "Kinodynamic planner dual-tree RRT (DT-RRT) for two-wheeled mobile robots using the rapidly exploring random tree," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 2, pp. 1080–1090, 2015.

[34] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2421–2428, IEEE, 2013.

[35] S. M. Hu, C. L. Tai, and S. H. Zhang, "An extension algorithm for B-splines by curve unclamping," *CAD Computer Aided Design*, vol. 34, no. 5, pp. 415–419, 2002.

[36] G. Wolberg and I. Alfy, "Monotonic cubic spline interpolation," in *Computer Graphics International*, pp. 188–195, 1999.

[37] B. Akgun and M. Stilman, "Sampling Heuristics for Optimal Motion Planning in High Dimensions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2640–2645, IEEE, 2011.

[38] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997–3004, IEEE, 2014.

[39] J. Li, "Planar T-Bézier curve with approximate minimum curvature variation," *Journal of Advanced Mechanical Design, Systems and Manufacturing*, vol. 12, no. 1, 2018.

[40] H. Zhao and G. Wang, "Shape control of cubic H-Bézier curve by moving control point," *Journal of Information and Computational Science*, vol. 4, no. 2, pp. 871–878, 2007.