Autonomous Racecar Project Report
(Algorithmically Complete Motion Planner)

Dharini Dutia
July 30, 2018

1 INTRODUCTION

In an autonomous driving system, path planning is a critical component as it is responsible for complex maneuvers and safe operation. It provides the ability to adapt in different environments by accommodating both non-holonomic vehicle motion constraints and obstacle avoidance constraints, efficiently. Along with that, the trajectory generation should be continuous and smooth which can ensure required behaviour. Representing trajectories using NURBS curves is the best alternative as they are computationally stable, continuous, derivable and highly flexible.

Generally, NURBS curves are widely used for CAD systems, but they have not been thoroughly explored in robotic applications. Some of the prior research includes trajectory approximation and reconstruction [1], and smoothing [2] using NURBS. The thesis [3] first suggested to represent the robot’s feasible trajectories using NURBS basis functions. In his work, he transformed an Optimal Control (OC) problem of obstacle avoidance into a Non-Linear Programming (NLP) problem by completely eliminating trajectory constraints. He argued that by exploiting the continuity property of the representation the constraints from the OC problem are reduced, making the ”original problem tractable”. Although, this work does not talk about real-time abilities or obtain a deterministic guarantee for a feasible global solution.

Additionally, [4] suggests most suitable control points for path planning of a non-holonomic 7DOF humanoid manipulator. The method provides a continuous human-like motion given a start and end location with its orientation in real-time and the system transitions to it iteratively in a smooth, computationally efficient manner. They set the orientations according to the minimum radius of curvature allowed by the robot. This method works well for the mentioned application but cannot be scaled or reused as the number of control points are fixed and no obstacle avoidance capabilities.

Similarly, [5] applies NURBS for a mobile robot to track a moving object in a known environment. The new control points of constant spacing are added after observing the moving target given a unit weight vector. Simulations show smooth tracking, although there is no discussion about optimally choosing control points. [6] experiments with control points selection and knots vectors for feasible trajectory generation using NURBS. A global shortest plan is an input to obtain control points, whose weights are optimized and curvature is bounded for a collision-free curve. They provide an analysis of the effects of modulating weights, curvature on the curve formation but the algorithm is essentially offline.

Observing the potential of using geometrically stable NURBS to represent a robot trajectory, we propose an algorithmically complete motion planner which uses Non-Uniform Rational B-Splines (NURBS) to represent the collision-free trajectory. Unlike previous work, we formulate and utilize the relationship between derivatives of NURBS curve and the dynamics of the vehicle to generalize the selection of weights and knots.

2 PROBLEM STATEMENT & FORMULATION

Consider a mobile robot, moving in environment $E$ where each configuration of the mobile robot is represented by $(x(t), y(t), \theta(t))$ in configuration space $Q$. The objective of the motion planner is to generate collision-free, dynamically feasible trajectory for local planning of the system.

**Definition 1** (Unicycle dynamics). For the first iteration of the proposed algorithm, we use unicycle robot dynamics (Chapter 13, Planning Algorithms [8]). The unicycle wheeled-robot model is given by state variables...
x, y, θ and input variables v, ω, which are related by the following state equations:

\[ \dot{x} = v \cos \theta \]  
\[ \dot{y} = v \sin \theta \]  
\[ \dot{\theta} = \omega \]  

The magnitude of the robot’s velocity is v and direction is given by the angle θ. This model impose dynamical constraints of maximum velocity \( v_{max} \) and acceleration \( acc_{max} \), while the trajectory generation.

Remark. The robot/vehicle cannot rotate in its current position, the turn is bounded by a minimum radius of curvature \( p_{min} \).

Definition 2 (NURBS). A NURBS curve [7] of degree \( p \) is defined as a mapping \( r : [0, 1] \rightarrow \mathbb{R}^n \) given by

\[ τ(u) = \sum_{i=0}^{k} R_{i,p}(u) P_i \]  

where \( P_i \) represent the \( k + 1 \) control points of the NURBS curve. The functions \( R_{i,p} \) are called the NURBS basis functions, which are themselves defined using the B-spline basis functions, \( N_{i,p} \) given by

\[ R_{i,p}(u) = \frac{N_{i,p}(u)}{\sum_{j=0}^{k} N_{j,p}(u)} w_i \]  
\[ N_{i,0}(u) = \begin{cases} 1 & \text{if } u_i \leq u < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \]  
\[ N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \]

Remark. The B-spline basis functions are defined recursively, and depend only on the knot vector \( u \). Any change in the knot vector will require recomputing the basis vectors.

The NURBS have several interesting properties [7] which we will exploit for generating a set of feasible trajectories.

Property 1 (Continuity). A NURBS curve of degree \( p \) is \( C^p \) smooth.

Property 2 (Degree-Knots Relation). If the number of knots is \( m+1 \), the degree of the basis functions is \( p \), and the number of degree \( p \) basis functions is \( k+1 \), then

\[ m = p + k + 1 \]

Property 3 (Strong Convex Hull). Let \( τ_p \) represent a NURBS curve segment of degree \( p \) and control polygon \( \mathbf{P} = \{ \mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_p \} \). Then, for \( u \in [u_i, u_{i+1}] \), then \( τ_p(u) \in \text{Convex-Hull}(\mathbf{P}) \). This property holds only if all the weights, \( w_i \), are non-negative.

NURBS have the ability to form positional, tangential and curvature continuity, thus making it a desirable tool for smooth trajectory generation. Consequently, we formulate the planning problem:

Problem 1 (NURBS Trajectory Planning). Let \( q_t, q_G \) represent the start and desired goal configurations of the robot in \( \mathcal{O} \). Let \( \mathcal{P}_{\text{Obs}} = \{ \mathcal{P} \in \text{Obs} \mid \text{Obs is an C-space obstacle} \} \) represent the set of boundary points of the C-space obstacles. Let \( \tau \) denote a parameterized NURBS curve with control points \( \mathbf{C} \) such that \( τ(0) = q_t \) and \( τ(1) = q_G \). Then find a subset \( \mathbf{C} \in \mathcal{P}_{\text{Obs}} \cup \mathcal{C}_{\text{free}} \) such that the NURBS, \( τ \) satisfies the constraint - for every \( u \in [0, 1] \), \( τ(u) \cap \mathcal{C}_{\text{Obs}} = \emptyset \), to avoid obstacles. Following, this trajectory the robot’s velocity is \( v_t \leq -v_{max}, +v_{max} \)∀t.

It must be noted that the control point selection for \( \mathbf{C} \) is restricted to the set \( \mathcal{P}_{\text{Obs}} \cup \mathcal{C}_{\text{free}} \), which does not include the interior of the C-space obstacles \( \mathcal{C}_{\text{Obs}} \). The \( \mathcal{P}_{\text{Obs}} \subset \partial(\mathcal{C}_{\text{Obs}}) \) can be viewed as a sample of the C-space obstacle boundary. Also, the weight and knot vector of this subset \( \mathbf{C} \) satisfies the dynamical feasibility constraint of the robotic system.
Definition 3 (Derivatives of a NURBS Surface). The derivative of a the NURBS curve \( r(u) \) is defined as:

\[
\begin{align*}
\mathbf{r}'(u) &= \sum_{i=1}^{k} R'_{i,m}(u) \mathbf{p}_i \\
\end{align*}
\]

which can be transformed as:

\[
\begin{align*}
\mathbf{r}'(u) &= \sum_{i=1}^{k} \lambda_i(u) (\mathbf{p}_i - \mathbf{p}_{i-1}) \\
\end{align*}
\]

where

\[
\begin{align*}
\lambda_i(u) &= \frac{1}{w_j^2} \sum_{j=q}^{i-1} \sum_{q=i}^{k} (N_{j,n}'(u)N_{j,n}(u) - N_{q,n}(u)N_{j,n}'(u)) w_j \cdot w_q \\
\end{align*}
\]

and \( n \) is the multiplicity of knots vector.

**Proof.** Proposition 2 of [12].

To include the dynamic constraints into trajectory planning considerations, we try to equate the derivative of NURBS with the velocity of the robot. The first derivative of a NURBS curve intuitively means the tangent at a particular point, this tangent gives the direction of the robot’s velocity. Similarly acceleration is related to the second derivative of the curve.

\[
\begin{align*}
v &= |\mathbf{r}'(u)|
\end{align*}
\]

For dynamical feasibility, an upper limit to the velocity and accelerations imposed by the motors is to be considered. The derivative (Equation 2.9) is bounded by:

\[
||\mathbf{r}'(u)|| \leq n. \frac{W^2}{w^2} \max_{i=1,...,k} \left\{ \frac{||\mathbf{p}_i - \mathbf{p}_{i-1}||}{u_{i+p+1} - u_{i+1}} \right\}
\]

where \( w = \min_{i=0,...,k} \{W_i\} \) and \( W = \max_{i=0,...,k} \{W_i\} \).

**Proof.** Proposition 7 of [12].

This upper bound on the first derivative is set as the maximum velocity \( v_{\text{max}} \) of the vehicle.

\[
\begin{align*}
n. \frac{W^2}{w^2} \max_{i=1,...,k} \left\{ \frac{||\mathbf{p}_i - \mathbf{p}_{i-1}||}{u_{i+p+1} - u_{i+1}} \right\} \leq v_{\text{max}}
\end{align*}
\]

**Assumption 1 (Uniform Knots sequence).** The knots vector \( u \) is assumed to be uniformly distributed.

**Assumption 2 (Minimum Weight).** The minimum weight on the control point is taken as \( w = 1 \).

Thus, solving the above equation for the unknown maximum weight, \( W \):

\[
\begin{align*}
W \leq \frac{v_{\text{max}} \cdot (p, \delta u)}{n. \max_{i=1,...,k} ||\mathbf{p}_i - \mathbf{p}_{i-1}||}
\end{align*}
\]

If we select the weights of the control points using Eq. (2.14), it automatically takes care of the dynamical constraints. Finally, Eq. (2.11) can be used to as an input to the motors at each time interval. Thus, we formulate the reduced problem as:

**Problem 2 (Reduced Problem).** Given the NURBS with quartic polynomial basis and the strong control hull property, the NURBS trajectory planning problem reduces to choosing the control point set \( C \) such that for every \( i \in \{0,1,2,...,k\} \), we have \( C_0 = q_I, C_k = q_C \) and Convex-hull\((C_{i-3}, C_{i-2}, C_{i-1}, C_i) \cap C_{\text{obs}} = \emptyset \). The first derivative, second derivative and maximum curvature of the curve is bounded by the upper limits \( v_{\text{max}}, a_{\text{acc max}}, \) and \( a_{\text{max}} \), respectively.

**Assumption 3 (Quartic Polynomial NURBS).** We shall assume that the degree of polynomial to be \( p = 3 \).

**Remark.** A trivial solution for above problem could begin with \( C_0 = q_I \) and run a variant of iterative deepening DFS with an explicit intersection check. The worst case performance of such an algorithm could be \( O(b^d) \), where \( b \) is the branching factor of the graph while \( d \) is the depth of shallowest solution [10].
Solution 2.1. To solve the problem, we exploit the concept of Delaunay triangulation. In general, we may use any arbitrary space partitioning algorithm, but we use Delaunay triangulation for its useful properties and close relations to other geometric objects such as Voronoi diagrams.

Definition 4 (Delaunay Triangulation [11]). Let \( P_{\text{Obs}} \) be a set of points in C-space. A triangulation of \( P_{\text{Obs}} \) is said to be Delaunay triangulation if no point \( P \in P_{\text{Obs}} \) lies inside the circumcircle of any triangle.

Property 4 (Size of triangulation). The number of triangles in the DT of \( n \) points rows as \( O(n^{\frac{d}{2}}) \), where \( d \) is the dimension of space. This property ensures that for a point robot with a 2-D workspace, the number of triangles grows linearly with the number of points! We shall next define a graph using DT.

Definition 5 (Graph on DT). Let \( G_D = (S, E) \) be a graph such that every triangle \( s \) is a node in graph \( G_D \), i.e. \( s \in S \). A pair of triangles \( s, s' \) defines an edge, \( (s, s') \in E \), if and only if they are adjacent triangles in DT.

Assumption 4 (Map of the Environment). The boundary points of all the C-space obstacles are given prior to the start of the algorithm. Also, the current location which is the starting point and the goal location are known.

1. The point sets \( P_{\text{Obs}}, P_{\text{boundary}} \) are given, where the latter denotes the boundary of the polygonal world. \( q_I \) and \( q_G \) are the start and goal configuration of the robot, respectively.
2. Let DT represent the Delaunay triangulation of the points in \( P_{\text{Obs}} \cup P_{\text{boundary}} \). Let \( S \) denote the set of triangles in \( DT \) and \( D_G \) denote the corresponding graph on \( DT \) as defined in def. 3.
3. Let \( S_O \subset S \) denote the triangles lying inside the C-space obstacles.
4. Let \( q_I \in s_I \) and \( q_G \in s_G \), where \( s_I, s_G \in S \setminus S_O \).
5. Let \( \pi = \{s_1, s_2, \ldots, s_k, s_G\} \) be a path of length \( k + 1 \) in the graph \( D_G \) found using any shortest path algorithm (eg. A*, Dijkstra) such that for every \( s \in \pi \cdot s \in S \setminus S_O \).
6. Then choose the control points as follows,
   (a) Choose \( C_0 = q_I \).
   (b) Choose \( C_1 \) as the unshared vertex of \( s_I \).
   (c) Choose \( C_2, C_3 \) as the 2 vertices of shared edge between \( s_I, s_1 \).
   (d) Choose next two \( C_4, C_5 \) as 2 vertices of shared edge between \( s_I, s_{i+1} \).
   (e) Choose \( C_k = q_G \).
7. Let \( u = [u_0, u_1, u_2, \ldots, u_k] \) be the uniformly distributed knots vector.
8. Let \( w_i = [w, w + W', \ldots, W] \) be the Weight vector.
9. Generate \( \tau(u) \) as the smooth trajectory, with \( \tau'(u) \) and \( \tau''(u) \) the derivatives

3 IMPLEMENTATION

The proposed architecture of the NURBS based motion planner is divided into: \( i \) searching for all feasible paths, \( ii \) obtaining the control points and \( iii \) generating a 3rd degree NURBS curve. There are existing libraries to represent and compute on NURBS curves and surfaces in Octave [13], MATLAB [14] and Python [15]. The layered architecture is implemented in Python given a map of the environment, an input goal and output being the NURBS trajectory object.

First, environment with an convex obstacle is used as a testbed Figure 1(a). Using the obstacle boundary points and map boundary, delaunay triangulation (Figure 1(b)) divides the map which can be manipulated as obstacle/free regions. After a path search, the generated trajectory looks like Figure 1(c).

To satisfy, dynamical constraints, the weights are selected as per the proposition. A parking lot environment Figure 2(a) is considered. The process is similar to previous example (see Figure 2(b) & 2(d)). With the weight vector \( W_I = \{1.95, 6.75, 0.1, 4.27, 0.1, 10.49, 1.92, 1.1\} \), generated trajectory looks like Figure 2(d). Further refinement of these weights produces desired curves.
4 CONCLUSION

The proposed algorithm generates a smooth and feasible trajectory used as the local planning of the autonomous vehicle. Several experiments are conducted to demonstrate the effect of control point modulations, weights and knots selection on the generated curve. The formulation presents an generalized, scalable and robust approach for trajectory generation using NURBS representation.

LIST OF SYMBOLS

The next list describes several symbols that were used within the body of the document.

\( \tau(u) \)  NURBS curve

\( k + 1 \)  Number of control points

\( n \)  Multiplicity of the knots vector

\( p \)  Degree of the NURBS curve
Figure 3.2: Parking lot trajectory generation

REFERENCES


