49th IEEE Conference on Decision and Control
December 15-17, 2010
Hilton Atlanta Hotel, Atlanta, GA, USA

Curvature-Continuous Trajectory Generation with Corridor Constraint for Autonomous Ground Vehicles

Ji-wung Choi, Renwick E. Curry and Gabriel Hugh Elkaim

Abstract— **We present a practical path planning algorithm based on Bézier curves for autonomous vehicles operating under waypoints and corridor constraints. Bézier curves have useful properties for the trajectory generation problem. This paper describes how the algorithm apply these properties to generate the reference trajectory for vehicles to satisfy the path constraints. The algorithm generates the piecewise-Bézier-curves path such that the curves segments are joined smoothly with C^2 constraint which leads to continuous curvature along the path. The degree of the curves are minimized to prevent them from being numerically unstable. Additionally, we discuss the constrained optimization problem that optimizes the resulting path for a user-defined cost function.**

I. INTRODUCTION

Exploiting the versatility of autonomous vehicles for academic, industrial, and military applications will have a profound effect on future applications. Current research on control systems for autonomous vehicles demonstrates that trajectory generation is hardly a "solved" problem [1]. For vehicle viability, it is imperative to be able to generate safe paths in real time. In this paper, we study a practical path planning algorithm for autonomous ground vehicles operating under waypoints and corridor constraints.

Many path planning techniques for autonomous vehicles have been discussed in the literature. Among them, much of work on search algorithms [2], [3], [4], [5] provide computationally efficient way in discrete state spaces. However, the resulting paths by such algorithms do not tend to be smooth (piecewise linear) and hence, do not satisfy kinematic feasibility of the vehicle.

Dubins path is one of the most well-known methods to generate a smooth path. Dubins showed that the shortest paths for a car-like robot consist of a set of two circular arcs and line segments [6]. Even though the method has been widely used and extended by many research [7], [8], the path has a drawback of discontinuity of curvature at the joint nodes connecting the lines and arcs. Tracking such paths requires a vehicle to stop and reposition in order to alter its steering angle. So curvature continuity is an important requirement for trajectory guidance applications.

Our path planning algorithm is based on Bézier curves. Since the Bézier curves have useful properties for the path generation problem, many path planning techniques have been discussed based on Bézier Curves. Cornell University Team for 2005 DARPA Grand Challenge used a path planner based on Bézier curves of degree 3 in a sensing/action feedback loop to generate smooth paths that are consistent with vehicle dynamics [9]. Skrjanc et al. proposed a new cooperative collision avoidance method for multiple robots with constraints and known start and goal velocities based on Bézier curves of degree 4, [10]. In this method, four control points out of five are placed such that desired positions and velocities of the start and the goal point are satisfied. The fifth point is obtained by minimizing penalty functions. Jolly et al. described Bézier curve based approach for the path planning of a mobile robot in a multi-agent robot soccer system [11]. The resulting path is planned such that the initial state of the robot and the ball, and an obstacle avoidance constraints are satisfied. The velocity of the robot along the path is varied continuously to its maximum allowable levels by keeping its acceleration within the safe limits. When the robot is approaching a moving obstacle, it is decelerated and deviated to another Bézier path leading to the estimated target position.

Our path planning algorithm joins a set of low-degree of Bézier curve segments smoothly to generate the reference trajectory. The algorithm imposes constraints such that curve segments are C^2 continuous in order to have curvature continuous for every point on the path. The optimized resulting path is obtained by computing the constrained optimization problem. Furthermore, the algorithm satisfy the initial and goal orientation as well as position constraint. A nonlinear path following guidance method is used to track the planned path. The numerical simulation results demonstrate a successful routes generation and tracking result of a vehicle.

II. BÉZIER CURVE

A Bézier Curve of degree n, $\mathbf{P}(t)$ is defined by $n+1$ control points $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$:

$$\mathbf{P}(t) = \sum_{i=0}^{n} B_i^n(t)\mathbf{P}_i, \quad t \in [0,1], \tag{1}$$

where Bernstein polynomial $B_i^n(t)$ is given by

$$B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i, \quad i = 0,1,\ldots,n. \tag{2}$$

Bézier Curves have useful properties for path planning:

- They always start at $\mathbf{P}_0$ and stop at $\mathbf{P}_n$.
- They are always tangent to $\overline{\mathbf{P}_0\mathbf{P}_1}$ and $\overline{\mathbf{P}_{n-1}\mathbf{P}_n}$ at $\mathbf{P}_0$ and $\mathbf{P}_n$ respectively.

J. Choi is a Ph.D. candidate in Computer Engineering Department at the University of California, Santa Cruz, 95064, USA. jwchoi@soe.ucsc.edu

R. Curry is an adjunct professor in Computer Engineering Department at the University of California, Santa Cruz, 95064, USA. rcurry@ucsc.edu

G. Elkaim is an associate professor in Computer Engineering Department at the University of California, Santa Cruz, 95064, USA. elkaim@soe.ucsc.edu

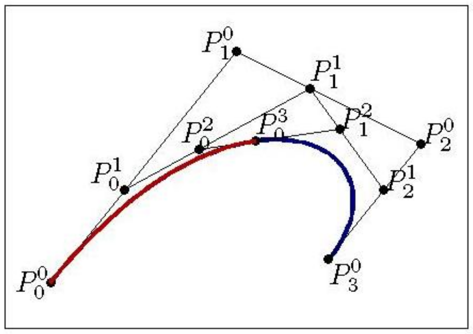
- They always lie within the convex hull of their control points.

A. The de Casteljau Algorithm

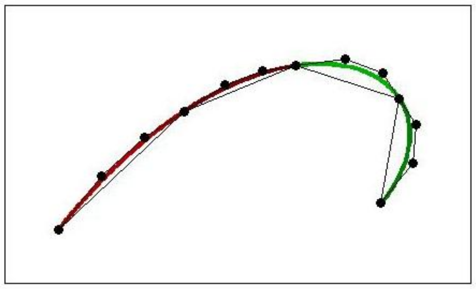
The de Casteljau algorithm describes a recursive process to subdivide a Bézier curve $\mathbf{P}(t)$ into two segments. The subdivided segments are also Bézier curves. Let $\{\mathbf{P}_0^0, \mathbf{P}_1^0, \dots, \mathbf{P}_n^0\}$ denote the control points of $\mathbf{P}(t)$. The control points of the segments can be computed by

$$\begin{aligned}\mathbf{P}_i^j =& (1-\tau)\mathbf{P}_i^{j-1} + \tau\mathbf{P}_{i+1}^{j-1}, \\ & j = 1,\dots,n,\ i = 0,\dots,n-j,\ \tau \in (0,1)\end{aligned} \tag{3}$$

Then, $\{\mathbf{P}_0^0, \mathbf{P}_0^1, \dots, \mathbf{P}_0^n\}$ are the control points of one segment and $\{\mathbf{P}_0^n, \mathbf{P}_1^{n-1}, \dots, \mathbf{P}_n^0\}$ are the another. (See an example in Figure 1(a).) It is interesting to note that the collection of control polygons converge to the Bézier curve as the curve is repeatedly subdivided, as shown in Figure 1(b).



(a) Subdividing a cubic Bézier curve with $\tau = .4$



(b) Recursive subdivision

Fig. 1. Subdividing a cubic Bézier curve with $\tau = .4$ by the de Casteljau Algorithm (a). As a Bézier curve is repeatedly subdivided, the collection of control polygons converge to the curve (b).

B. Derivatives, Continuity and Curvature

The derivatives of a Bézier curve can be determined by its control points. For a Bézier curve $\mathbf{P}(t)$ represented as (1), the first derivative is given by

$$\dot{\mathbf{P}}(t) = \sum_{i=0}^{n-1} B_i^{n-1}(t)\mathbf{D}_i, \tag{4}$$

where $\mathbf{D}_i$, control points of $\dot{\mathbf{P}}(t)$ are

$$\mathbf{D}_i = n(\mathbf{P}_{i+1} - \mathbf{P}_i). \tag{5}$$

The higher order derivative can be obtained by using the relationship of (4), iteratively.

Two Bézier curves $\mathbf{P}(t)$ and $\mathbf{Q}(t)$ are said to be C^k at t_0 continuous if

$$\mathbf{P}(t_0) = \mathbf{Q}(t_0),\ \dot{\mathbf{P}}(t_0) = \dot{\mathbf{Q}}(t_0),\ \dots,\ \mathbf{P}^{(k)}(t_0) = \mathbf{Q}^{(k)}(t_0). \tag{6}$$

The curvature of a Bézier curve $\mathbf{P}(t) = \begin{bmatrix} x(t) & y(t)\end{bmatrix}^T$ at t is given by

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \dot{y}(t)\ddot{x}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{\frac{3}{2}}}. \tag{7}$$

III. Problem Statement

Consider the control problem of an autonomous ground vehicle to operate for the route in-bounds area specified by a sequence of waypoints, $\mathbf{W}_i \in \mathbb{R}^2$, $i = 1, 2, \dots, N$ and corridor widths of each segment between two neighboring waypoints, $l_j \in \mathbb{R}^+$, $j = 1, \dots, N-1$ in a two-dimensional free-space. The route is defined as of DARPA Grand Challenge 2005 [12] as shown in Figure 2(a). The shaded grey area in the figure illustrates the route specified by four waypoints. Our goal is to develop and implement an algorithm for navigation that satisfies the constraint. That is divided into two parts: path planning and path following.

To describe the problem that we consider, some notations need to be introduced. For the dynamics of the vehicle, the state and the control vector are denoted $\mathbf{q}(t) = (x_c(t), y_c(t), \psi(t))^T$ and $\mathbf{u}(t) = (v(t), \omega(t))^T$ respectively, where (x_c, y_c) represents the position of the center of gravity of the vehicle. The yaw angle ψ is defined to the angle from the X axis. v is the longitudinal velocity of the vehicle. $\omega = \dot{\psi}$ is the yaw angular rate. State space is denoted as $\mathscr{C} = \mathbb{R}^3$. We assume that the vehicle follows that

$$\dot{\mathbf{q}}(t) = \begin{pmatrix} \cos\psi(t) & 0 \\ \sin\psi(t) & 0 \\ 0 & 1 \end{pmatrix} \mathbf{u}(t)$$

With the notations defined above, the path planning problem is formulated as: Given an initial position and orientation and a goal position and orientation of the vehicle, generate a path λ specifying a continuous sequence of positions and orientations of the vehicle satisfying the path constraints, [13]. In other words, we are to find a continuous map

$$\lambda : [0,1] \to \mathscr{C}$$

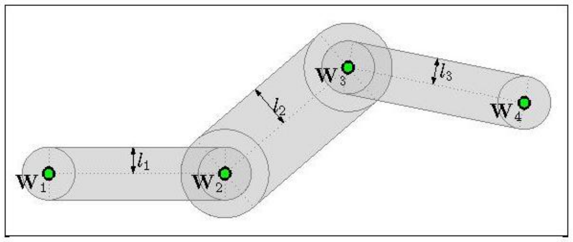
with

$$\lambda(0) = \mathbf{q}_{init} \ \text{ and } \ \lambda(1) = \mathbf{q}_{goal}$$

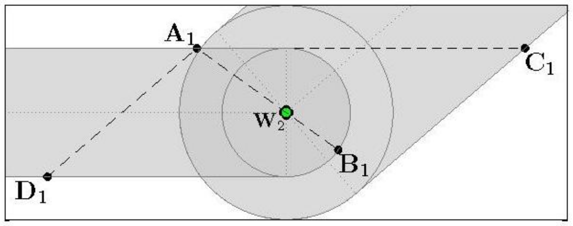
where $\mathbf{q}_{init} = (\mathbf{W}_1, \psi_0)$ and $\mathbf{q}_{goal} = (\mathbf{W}_N, \psi_f)$ are the initial and goal states of the path, respectively.

IV. Path Planning Algorithm

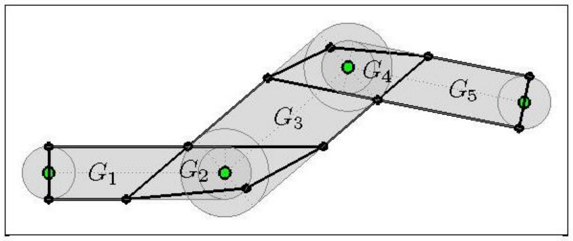
In this section, the path planning method based on Bézier curves is proposed. Bézier curves constructed by large numbers of control points are not only computationally expensive but also numerically unstable. For this reason, it is desirable to join low-degree Bézier curves together in a smooth way for path planning. Thus we use a set of low-degree Bézier



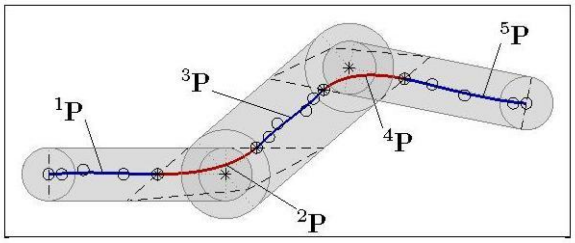
(a) Route geometry; the shaded grey area is permitted for vehicles to traverse.



(b) Points constituting a route segment in corner area



(c) Route segmentation



(d) Bézier curves placement; '*' and 'o' indicate control points of a curve segment at an even number index and at an odd number index.

Fig. 2. Geometry of route and path planning.

curves such that the neighboring curves are C^2 continuous at their end nodes. So curvature is continuous on every points along the resulting path. The Bézier curves used for the path plannings are denoted as

$$ {}^i\mathbf{P}(t) = \sum_{k=0}^{n_i} B_k^{n_i}(t) \cdot {}^i\mathbf{P}_k, \quad i = 1,\dots,M,\ t \in [0,1], $$

where M is the total number of the Bézier curves and n_i is the degree of ${}^i\mathbf{P}(t)$. The planned path λ is a concatenation of all ${}^i\mathbf{P}(t)$ such that

$$ \lambda\left((i-1+t)/M\right) = {}^i\mathbf{P}(t), \quad i = 1,\dots,M,\ t \in [0,1]. $$

The first step of the algorithm is to segment the route into straight areas and corner areas. To describe the route segmentation, let us denote $\mathbf{A}_j$, $j = 1,\dots,N-2$ as the inner pivot point that is the intersection of inner boundary of the path segment between $\mathbf{W}_j$ and $\mathbf{W}_{j+1}$ and that between $\mathbf{W}_{j+1}$ and $\mathbf{W}_{j+2}$. $\mathbf{B}_j$ denotes the one of the intersection points, further from $\mathbf{A}_j$, between $\overrightarrow{\mathbf{A}_j\mathbf{W}_{j+1}}$ and the circle with center $\mathbf{W}_{j+1}$ and smaller radius of l_j and l_{j+1}. $\mathbf{C}_j$ is the intersection point of inner boundary between $\mathbf{W}_j$ and $\mathbf{W}_{j+1}$ and outer boundary between $\mathbf{W}_{j+1}$ and $\mathbf{W}_{j+2}$. $\mathbf{D}_j$ is the intersection point of outer boundary between $\mathbf{W}_j$ and $\mathbf{W}_{j+1}$ and inner boundary between $\mathbf{W}_{j+1}$ and $\mathbf{W}_{j+2}$. (See Figure 2(b).) The route is divided into segments G_i, $i = 1,\dots,2N-3$ by $\overline{\mathbf{A}_j\mathbf{D}_j}$ and $\overline{\mathbf{A}_j\mathbf{C}_j}$. (See Figure 2(c).) Note that each G_i is a convex polygon and that the union of G_i lies within the route in-bound-area.

After segmenting the route, each Bézier curve ${}^i\mathbf{P}(t)$ is placed within G_i. (See Figure 2(d).) The quadratic Bézier curves that have single signed curvature are good fits to the segments of corner areas, G_i, $i = 2,4,\dots,2N-4$. The degrees of curves for the other segments are determined by the minimum number of control points to satisfy C^2 continuity constraint, independently. At each junction, three control points of each adjacent Bézier curve are dedicated to the C^2 continuity constraint by (4) and (7). So the minimum number of control points to satisfy the constraints independent on the others are six for intermediate curve segments ${}^i\mathbf{P}(t)$, $i = 3,5,\dots,2N-5$. On the other hand, the first segment ${}^1\mathbf{P}(t)$ needs five: three to connect ${}^2\mathbf{P}(t)$ plus two to connect $\mathbf{W}_1$ with slope of ψ_0. So does the last curve ${}^{2N-3}\mathbf{P}(t)$.

$$ n_i = \begin{cases} 4, & i = 1, 2N-3 \\ 2, & i = 2,4,\dots,2N-4 \\ 5, & i = 3,5,\dots,2N-5 \end{cases} $$

By the definition of route segmentation and curves placement, the joint nodes between the quadratic Bézier curves and others must be on $\overline{\mathbf{A}_j\mathbf{D}_j}$ or $\overline{\mathbf{A}_j\mathbf{C}_j}$. In other words, the first and the last control points of the curves must be on $\overline{\mathbf{A}_j\mathbf{D}_j}$ or $\overline{\mathbf{A}_j\mathbf{C}_j}$ and are defined by scalers, $\alpha \in [0,1]$ and $\beta \in [0,1]$ such as

$$ \begin{aligned} {}^{2j}\mathbf{P}_0 &= (1-\alpha_j)\mathbf{A}_j + \alpha_j\mathbf{C}_j, \quad \alpha_j \in [0,1],\ j = 1,2,\dots,N-2, \\ {}^{2j}\mathbf{P}_2 &= (1-\beta_j)\mathbf{A}_j + \beta_j\mathbf{D}_j, \quad \beta_j \in [0,1],\ j = 1,2,\dots,N-2. \end{aligned} $$

Recall that lateral acceleration a_r is represented as

$$ a_r = \frac{v^2}{R} = \kappa v^2. $$

In practice, there must be a limitation on feasible a_r. Longitudinal velocity v is assume to be constant in this paper. So, to guarantee kinematic feasibility of the vehicle, the curvature on every point of ${}^i\mathbf{P}(t)$, denoted as ${}^i\kappa(t)$, needs to be bounded relying on the limitation, denoted as a_r^{max}.

$$ |{}^i\kappa(t)| \le \frac{|a_r^{max}|}{v^2}, \quad a_r^{max} \in \mathbb{R}^+. \tag{8} $$

The control points of each curve, ${}^i\mathbf{P}_k$, $k = 0,\dots,n_i$ are determined to satisfy the constraints imposed on the planned path as follows:

- The beginning and end position of λ are $\mathbf{W}_1$ and $\mathbf{W}_N$.

$$ {}^1\mathbf{P}_0 = \mathbf{W}_1, \quad {}^{2N-3}\mathbf{P}_4 = \mathbf{W}_N. \tag{9} $$

- The beginning and end orientation of λ are ψ_0 and ψ_f.

$$
\begin{aligned}
{}^{1}\mathbf{P}_1 &= \mathbf{W}_1 + c_0 \begin{bmatrix}\cos\psi_0 & \sin\psi_0\end{bmatrix}^T, \; c_0 \in \mathbb{R}^+, \\
{}^{2N-3}\mathbf{P}_3 &= \mathbf{W}_N - c_f \begin{bmatrix}\cos\psi_f & \sin\psi_f\end{bmatrix}^T, \; c_f \in \mathbb{R}^+.
\end{aligned} \tag{10}
$$

- ${}^{i-1}\mathbf{P}$ and ${}^{i}\mathbf{P}$ are C^2 continuous at the junctions.

$$
\begin{aligned}
&{}^{i-1}\mathbf{P}_{n_{i-1}} = {}^{i}\mathbf{P}_0, \\
&n_{i-1}({}^{i-1}\mathbf{P}_{n_{i-1}} - {}^{i-1}\mathbf{P}_{n_{i-1}-1}) = n_i({}^{i}\mathbf{P}_1 - {}^{i}\mathbf{P}_0), \\
&n_{i-1}(n_{i-1}-1)({}^{i-1}\mathbf{P}_{n_{i-1}} - 2\cdot{}^{i-1}\mathbf{P}_{n_{i-1}-1} + {}^{i-1}\mathbf{P}_{n_{i-1}-2}) \\
&\quad = n_i(n_i-1)({}^{i}\mathbf{P}_2 - 2\cdot{}^{i}\mathbf{P}_1 + {}^{i}\mathbf{P}_0), \\
&\quad i = 2,3,\ldots,2N-3.
\end{aligned} \tag{11}
$$

- The joints are within the route in-bound-area.

$$0 \le \alpha_j \le 1, \; 0 \le \beta_j \le 1, \quad j = 1,2,\ldots,N-2. \tag{12}$$

- The curvature is bounded relying on ω_{max} and v_{max}.

$$
\begin{aligned}
|{}^{i}\kappa(k/m)| &\le \frac{|a_r^{max}|}{v^2}, \quad a_r^{max} \in \mathbb{R}^+, \\
k &= 0,1,\ldots,m, \; i = 1,2,\ldots,2N-3.
\end{aligned} \tag{13}
$$

- ${}^{i}\mathbf{P}_1,\ldots,{}^{i}\mathbf{P}_{n_i-1}$ always lie within the area of G_i.

$${}^{i}\mathbf{P}_1 \in G_i, \; \ldots, \; {}^{i}\mathbf{P}_{n_i-1} \in G_i, \quad i = 1,2,\ldots,2N-3. \tag{14}$$

Equation (10) is derived by using the tangent property of Bézier curves at their end points. Equation (11) is obtained by applying (4) and (6). Equation (13) discretizes (8) by $m+1$ sample points on ${}^{i}\mathbf{P}(t)$. Equation (14) makes the resulting Bézier curve satisfy the corridor constraint by the convex hull property. The efficient way to compute (14) is presented in the following subsection. Note that ${}^{1}\mathbf{P}_0$ and ${}^{2N-3}\mathbf{P}_4$ are fixed in (9). ${}^{1}\mathbf{P}_1$ and ${}^{2N-3}\mathbf{P}_3$ are given by scalers c_0 and c_f respectively, in (10). The last three control points of ${}^{1}\mathbf{P}(t)$ rely on those of ${}^{2}\mathbf{P}(t)$ in (11). Likewise, the first three control points of ${}^{2N-3}\mathbf{P}(t)$ rely on those of ${}^{2N-4}\mathbf{P}(t)$. Also, all control points of ${}^{i}\mathbf{P}(t)$, $i = 3,5,\ldots,2N-5$ rely on those of ${}^{i-1}\mathbf{P}(t)$ and ${}^{i+1}\mathbf{P}(t)$. So the set of free variables, $\mathbf{X}$ is given by

$$\mathbf{X} = \begin{bmatrix}\alpha_1 & {}^{2}\mathbf{P}_1 & \beta_1 & \ldots & \alpha_{N-2} & {}^{2N-4}\mathbf{P}_1 & \beta_{N-2} & c_0 & c_f\end{bmatrix}^T$$

The number of the elements of $\mathbf{X}$ or the degrees of freedom is $4N-6$. The $\mathbf{X}$ is computed by minimizing the constrained optimization problem:

$$\min_{\mathbf{X}} J = \sum_{i=1}^{N-1} J_i \tag{15}$$

subject to (12), (13), and (14), where J_i is the cost function of ${}^{i}\mathbf{P}(t)$.

A. Corridor Constraint

This subsection provides a computationally efficient way of determining if a Bézier curve ${}^{i}\mathbf{P}$ lies inside of segment G_i, i.e. (14). For the sake of simplicity, we drop the index i from now on.

It is interesting to notice that, given a line segment from $\begin{bmatrix}x_a & y_a\end{bmatrix}^T$ to $\begin{bmatrix}x_b & y_b\end{bmatrix}^T$, a point $\begin{bmatrix}x & y\end{bmatrix}^T$ is to the left of the line segment if

$$(y - y_a)(x_b - x_a) - (x - x_a)(y_b - y_a) \tag{16}$$

is greater than zero. It is to the right, if the above is less than zero. It is on the line segment, if equal to zero. Let $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_m$ be the points constituting G such as counter-clock-wise in order. From the property of (16), the condition for a point $\mathbf{p} = \begin{bmatrix}x & y\end{bmatrix}^T$ to lie within G is represented as

$$
\begin{aligned}
&(y - g_{y_i})(g_{x_{i+1}} - g_{x_i}) - (x - g_{x_i})(g_{y_{i+1}} - g_{y_i}) < 0, \\
&\quad i = 1,\ldots,m-1 \\
&(y - g_{y_m})(g_{x_1} - g_{x_m}) - (x - g_{x_m})(g_{y_1} - g_{y_m}) < 0
\end{aligned}
$$

This can be rewritten as

$$\mathbf{Lp} - \mathbf{M} < \mathbf{0},$$

where

$$
\mathbf{L} = \begin{bmatrix}
g_{y_1} - g_{y_2} & g_{x_2} - g_{x_1} \\
g_{y_2} - g_{y_3} & g_{x_3} - g_{x_2} \\
\vdots & \vdots \\
g_{y_{m-1}} - g_{y_m} & g_{x_m} - g_{x_{m-1}} \\
g_{y_m} - g_{y_1} & g_{x_1} - g_{x_m}
\end{bmatrix},
$$

$$
\mathbf{M} = \begin{bmatrix}
g_{x_2}g_{y_1} - g_{x_1}g_{y_2} \\
g_{x_3}g_{y_2} - g_{x_2}g_{y_3} \\
\vdots \\
g_{x_m}g_{y_{m-1}} - g_{x_{m-1}}g_{y_m} \\
g_{x_1}g_{y_m} - g_{x_m}g_{y_1}
\end{bmatrix}.
$$

As described in Section II, the Bézier curve $\mathbf{P}(t)$ lies inside of the convex hull determined by its control points. So, it suffice that every control point lies within G for $\mathbf{P}(t)$ to lie within G.

$$
\begin{bmatrix}
\mathbf{L} & \mathbf{0} & \ldots & \mathbf{0} \\
\mathbf{0} & \mathbf{L} & \ddots & \vdots \\
\vdots & \ddots & \ddots & \mathbf{0} \\
\mathbf{0} & \ldots & \mathbf{0} & \mathbf{L}
\end{bmatrix} \mathbf{P}_c - \begin{bmatrix}\mathbf{M} \\ \mathbf{M} \\ \vdots \\ \mathbf{M}\end{bmatrix} < \mathbf{0}, \tag{17}
$$

where $\mathbf{P}_c$ is the set of control points of $\mathbf{P}(t)$

$$\mathbf{P}_c = \begin{bmatrix}\mathbf{P}_0 & \mathbf{P}_1 & \ldots & \mathbf{P}_n\end{bmatrix}^T.$$

Recall that the collection of control polygons converge to a Bézier curve as the curve is repeatedly subdivided. By this attribute, as the test is done for the collection of subdivided control points, the test result provides more accurate detection. The collection of subdivided control points with τ can be represented as the following by using (3):

$$\begin{bmatrix}\mathbf{F} & \mathbf{H}\end{bmatrix}^T \mathbf{P}_c$$

$$\mathbf{F} = \begin{bmatrix} 1 & & & & \mathbf{0} \\ 1-\tau & \tau & & & \\ (1-\tau)^2 & 2(1-\tau)\tau & \tau^2 & & \\ \vdots & \vdots & \vdots & \ddots & \\ (1-\tau)^n & \binom{n}{1}(1-\tau)^{n-1}\tau & \binom{n}{2}(1-\tau)^{n-2}\tau^2 & \dots & \tau^n \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} (1-\tau)^n & \dots & \binom{n}{2}(1-\tau)^2\tau^{n-2} & \binom{n}{1}(1-\tau)\tau^{n-1} & \tau^n \\ & \ddots & \vdots & \vdots & \vdots \\ & & (1-\tau)^2 & 2(1-\tau)\tau & \tau^2 \\ & & & 1-\tau & \tau \\ \mathbf{0} & & & & 1 \end{bmatrix}$$

Note that the first row of **F** is same as the last row of **H**. Repeatedly, the set of points can be subdivided by multiplying existing matrix by **F** and **H** and adding the results. So the set of points by k subdivision can be represented as

$$\mathbf{S}_k\mathbf{P}_c,$$

where $\mathbf{S}_k$ is defined as

$$\begin{aligned} \mathbf{S}_1 &= \begin{bmatrix}\mathbf{F} & \mathbf{H}\end{bmatrix}^T \\ \mathbf{S}_2 &= \begin{bmatrix}\mathbf{F}^2 & \mathbf{FH} & \mathbf{HF} & \mathbf{H}^2\end{bmatrix}^T \\ \mathbf{S}_3 &= \begin{bmatrix}\mathbf{F}^3 & \mathbf{F}^2\mathbf{H} & \dots & \mathbf{H}^2\mathbf{F} & \mathbf{H}^3\end{bmatrix}^T \\ &\vdots \end{aligned}$$

Finally, the corridor constraint is represented as the following by replacing $\mathbf{P}_c$ in (17) with $\mathbf{S}_k\mathbf{P}_c$:

$$\begin{bmatrix} \mathbf{L} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{L} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{L} \end{bmatrix} \mathbf{S}_k\mathbf{P}_c - \begin{bmatrix} \mathbf{M} \\ \mathbf{M} \\ \vdots \\ \mathbf{M} \end{bmatrix} < \mathbf{0}, \qquad (18)$$

V. Path Following Algorithm

Given a planned path, we use the nonlinear path following guidance technique proposed by Park et al [14]. In steering ground vehicles, the method generates a command for lateral acceleration given by

$$a_r = 2\frac{v^2}{L_1}\sin\eta, \qquad (19)$$

where L_1 is the look-ahead distance at which the reference point is on the desired path forward of the vehicle. η determines the central angle of the circular arc trajectory of the vehicle at each time step. (See Figure 3.) In this paper, we use constant velocity for all tracking time. So the motion of the vehicle is solely exerted by yaw rate ω. We can rewrite (19) with respect to ω:

$$\omega = \sqrt{a_r\kappa} = 2\frac{v}{L_1}\sin\eta,$$

where note that $\kappa = 2\sin\eta/L_1$ from the diagram of Figure 3. With the guidance law, ω is computed every 50 *ms*.

An important choice in the guidance law is the look-ahead distance L_1. With the help of linear system analysis, Park has shown the following remark.

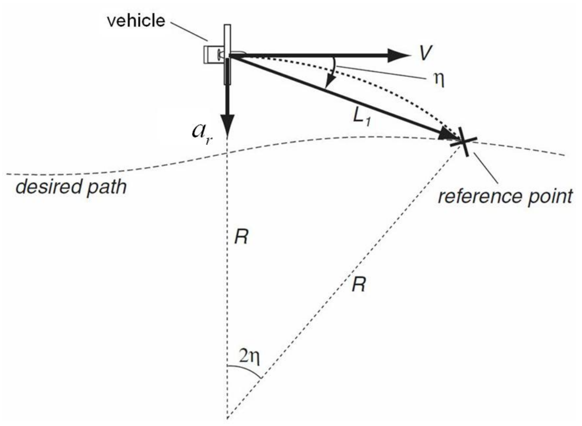


Fig. 3. Diagram for the guidance law [14].

Remark 1: If L_p is the wavelength of the highest frequency content in the desired path, then the look ahead distance L_1 must be chosen to be less than about $L_p/4.4$ if the vehicle is to accurately follow the desired path [14].

VI. Simulation Results

Simulations were run for the course with seven waypoints as illustrated in Figure 4. The initial and goal orientation are given by $\psi_0 = \psi_f = 0$. For path following, the constant longitudinal velocity $v = 10\ [m/s]$ is used. The magnitude of a_r is bounded within $|a_r^{max}| = 26.18\ [m/s^2]$. Subsequently, the magnitude of ω is bounded as follows:

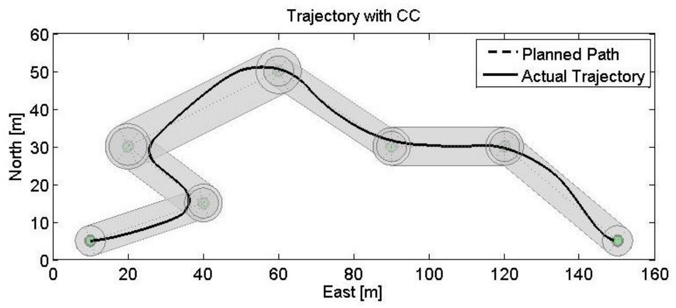
$$|\omega| = \sqrt{\kappa a_r} \le \frac{|a_r^{max}|}{v} = 2.618\ [rad/s].$$

In Figure 4(a), the reference path planned by applying the proposed algorithm is placed. The path is obtained by solving (15) with

$$J_i = \int_0^1 \sqrt{\dot{x}_i^2(t) + \dot{y}_i^2(t)}dt.$$

Since only arc lengths of ${}^i\mathbf{P}(t) = \begin{bmatrix}x_i(t) & y_i(t)\end{bmatrix}^T$ are penalized, the cost function leads to paths with minimum arc length subject to the corridor and the curvature constraint. To demonstrate the importance of the curvature constraint (CC), another path is planned by minimized the same cost, but without CC, in Figure 4(b). Both of the paths are illustrated as dashed curves.

The actual trajectories (solid curves) are generated by using the tracking method discussed in Section V. By Remark 1, L_1 is given by 1 $[m]$. With CC expressed as (13), the resulting path has large enough radii of curvature on every point for the vehicle to converge to the path. So the vehicle tracks the path accurately in every area. On the other hand, without CC, the vehicle overshoots at the points where the curvature exceeds the limit a_r^{max}/v^2, and hence resulting in large position error. (See areas around the second and the third waypoints in Figure 4(b).) We can demonstrate that more tangibly in cross track errors plot provided in Figure 5(a). Also, taking a look at the commanded steering angle rate in Figure 5(b), with CC, the vehicle complies feasible range of the rate thanks to the bounded curvature. However, without CC, the vehicle undergoes rapid changes and is constrained by the rate limit. This will lead to large torque in lateral motion.



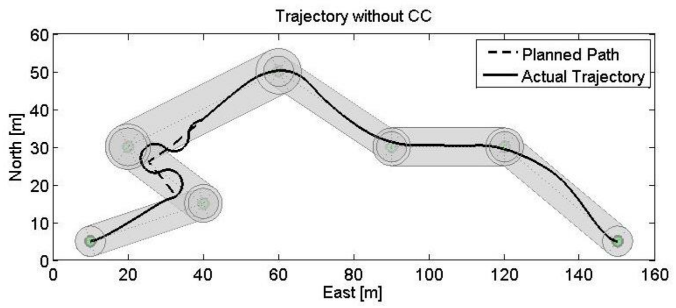
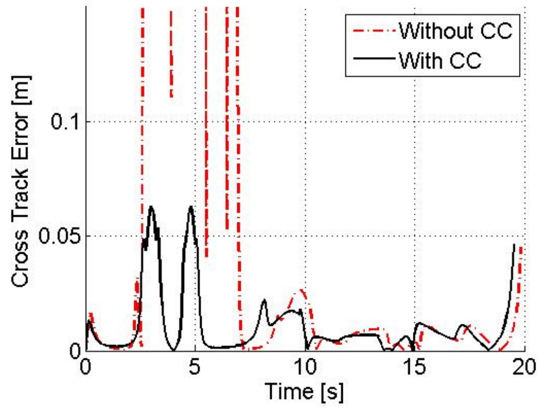


Fig. 4. The planned path and tracking result by the proposed algorithm.



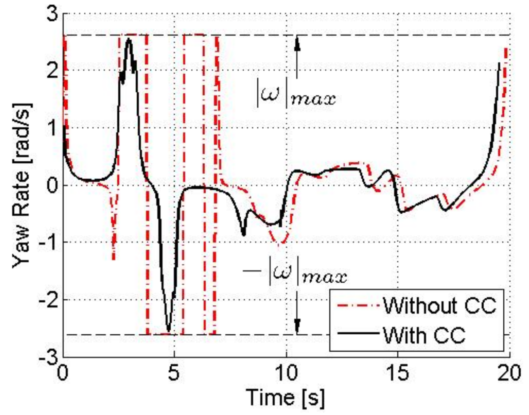


Fig. 5. The tracking results by the path following algorithm.

VII. Conclusions and Future Work

This paper presents a practical path planning algorithms based on Bézier curves for autonomous vehicles with waypoints and corridor constraints. Bézier curves provide an efficient way to generate the optimized path and satisfy the constraints at the same time. The simulation results also show that the trajectory of the vehicle follows the planned path within the constraints.

These path planning algorithms will be implemented on the Overbot, the autonomous ground vehicle at Autonomous Systems Lab at UCSC. Enabling autonomous vehicles to detect unknown obstacles and safely avoid them is essential to future operations. Future work will employ receding horizon control methods to generate real-time Bézier-based optimal trajectories while avoiding obstacles.

References

[1] J. H. Reif. Complexity of the mover's problem and generalizations. In *20th IEEE Symp. on Foundations of Computer Science (FOG'S)*, 1979.

[2] N. J. Nilsson. A mobile automation: An application of artificial intelligence techniques. In *The 1st International Joint Conference on Artificial Intelligence*, 1969.

[3] T. Lozano-Pérez. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems*, 11(10):681–698, 1981.

[4] N. H. Sleumer and N. Tschichold-Grman. Exact cell decomposition of arrangements used for path planning in robotics. Technical report, Institute of Theoretical Computer Science Swiss Federal Institute of Technology Zurich, Switzerland, 1999.

[5] A. Nash, K. Daniel, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. pages 1177–1183, 2007.

[6] L. E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *Amer. J. Math*, 1957.

[7] F. Lamiraux and J.-P. Laumond. Smooth motion planning for car-like vehicles. *Amer. J. Math*, 17(4):498–502, 2001.

[8] E. P. Anderson, R. W. Beard, and T. W. McLain. Smooth motion planning for car-like vehicles. *IEEE Transactions on Control System Technology*, 13(3), 2005.

[9] I. Miller, S. Lupashin, N. Zych, P. Moran, B. Schimpf, A. Nathan, and E. Garcia. *Cornell University's 2005 DARPA Grand Challenge Entry*, volume 36, chapter 12, pages 363–405. Springer Berlin / Heidelberg, 2007.

[10] I. Skrjanc and G. Klancar. Cooperative collision avoidance between multiple robots based on bézier curves. In *Proceedings of the Information Technology Interfaces, 2007 (ITI 2007)*, pages 451–456, 2007.

[11] K. G. Jolly, K. R. Sreerama, and R. Vijayakumar. A bézier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robotics and Autonomous Systems*, 57:23–33, 2009.

[12] DARPA Grand Challenge 2005, Route Data Definition File. http://www.darpa.mil/grandchallenge05/RDDF_Document.pdf.

[13] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[14] S. Park, J. Deyst, and J. P. How. Performance and lyapunov stability of a nonlinear path following guidance method. *Journal of Guidance, Control, and Dynamics*, 6, 2007.