robot was also used to palpate a clinical prostate model. The generated surface-stiffness map matched the known pathological conditions of the model. These experimental results suggested a strong potential of using the continuum robots for surgical applications that require force feedback in confined spaces.

References

[1] L. N. Verner and A. M. Okamura, "Sensor/actuator asymmetries in telemanipulators: Implications of partial force feedback," in *Proc. Symp. Haptic Interfaces Virtual Environ. Teleoperator Syst.*, Alexandria, VA, 2006, pp. 309–314.

[2] M. C. Cavusoglu, A. Sherman, and F. Tendick, "Design of bilateral teleoperation controllers for haptic exploration and telemanipulation of soft environments," *IEEE Trans. Robot. Autom.*, vol. 18, no. 4, pp. 641–647, Aug. 2002.

[3] C. R. Wagner, N. Stylopoulos, and R. D. Howe, "The role of force feedback in surgery: Analysis of blunt dissection," in *Proc. Symp. Haptic Interfaces Virtual Environ. Teleoperator Syst.*, Orlando, FL, 2002, pp. 68–74.

[4] K. Ikuta, S. Daifu, T. Hasegawa, and H. Higashikawa, "Hyper-finger for remote minimally invasive surgery in deep area," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Interv.*, Tokyo, Japan, 2002, pp. 173–181.

[5] M. Tavakoli, R. V. Patel, and M. Moallem, "A force reflective master-slave system for minimally invasive surgery," in *Proc. Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, 2003, pp. 3077–3082.

[6] S. Shimachi, Y. Fujiwara, and Y. Hakozaki, "New sensing method of force acting on instrument for laparoscopic robot surgery," in *Proc. Comput. Assist. Radiol. Surg.*, Chicago, IL, 2004, pp. 775–780.

[7] U. Seibold, B. Kubler, and G. Hirzinger, "Prototype of instrument for minimally invasive surgery with 6-axis force sensing capability," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, 2005, pp. 496–501.

[8] K. Tadano and K. Kawashima, "Development of 4-DoFs forceps with force sensing using pneumatic servo system," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, 2006, pp. 2250–2255.

[9] M. Mitsuishi, N. Sugita, and P. Pitakwatchara, "Force feedback augmentation modes in the laparoscopic minimal invasive telesurgical system," *IEEE/ASME Trans. Mechatronics*, vol. 12, no. 4, pp. 447–454, Aug. 2007.

[10] D. T. Wallace, G. Stahler, A. Goldenberg, G. Reis, R. G. Younge, M. Clopp, D. Camarillo, and T. J. King St., "Method of sensing forces on a working instrument," U.S. Patent 0 197 9 39, Mountain View, CA: Hansen Medical, 2007.

[11] C. Sultan and R. Skelton, "A force and torque tensegrity sensor," *Sens. Actuators A: Phys.*, vol. 112, pp. 220–231, 2004.

[12] K. Xu and N. Simaan, "An investigation of the intrinsic force sensing capabilities of continuum robots," *IEEE Trans. Robot.*, vol. 24, no. 3, pp. 576–587, Jun. 2008.

[13] R. Ranganath, P. S. Nair, T. S. Mruthyunjaya, and A. Ghosal, "A force-torque sensor based on a stewart platform in a near-singular configuration," *J. Mech. Mach. Theory*, vol. 39, no. 9, Sep. 2004.

[14] M. Uchiyama, E. Bayo, and E. Palma-Villalon, "A systematic design procedure to minimize a performance index for robot force sensors," *J. Dyn. Syst., Meas., Control*, vol. 113, pp. 388–394, Sep. 1991.

[15] A. Bicchi, "A criterion for the optimal design of multi-axis force sensors," *Robot. Auton. Syst.*, vol. 10, pp. 269–286, 1992.

[16] M. M. Svinin and M. Uchiyama, "Optimal geometric structures of force/torque sensors," *Int. J. Robot. Res.*, vol. 14, no. 6, pp. 560–573, 1995.

[17] S. Hirose, *Biologically Inspired Robots, Snake-Like Locomotors and Manipulators*. Oxford, U.K.: Oxford Univ. Press, 1993.

[18] I. A. Gravagne and I. D. Walker, "Kinematic transformations for remotely-actuated planar continuum robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, pp. 19–26.

[19] I. A. Gravagne and I. D. Walker, "Manipulability, force and compliance analysis for planar continuum manipulators," *IEEE Trans. Robot. Autom.*, vol. 18, no. 3, pp. 263–273, Jun. 2002.

[20] I. A. Gravagne and I. D. Walker, "On the kinematics of remotely-actuated continuum robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, 2000, pp. 2544–2550.

[21] N. Simaan, R. H. Taylor, and P. Flint, "A dexterous system for laryngeal surgery," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, 2004, pp. 351–357.

[22] B. A. Jones and I. D. Walker, "Practical kinematics for real-time implementation of continuum robots," *IEEE Trans. Robot. Autom.*, vol. 22, no. 6, pp. 1087–1099, Dec. 2006.

[23] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.

An Analytical Continuous-Curvature Path-Smoothing Algorithm

Kwangjin Yang and Salah Sukkarieh

***Abstract*—An efficient and analytical continuous-curvature path-smoothing algorithm, which fits an ordered sequence of waypoints generated by an obstacle-avoidance path planner, is proposed. The algorithm is based upon parametric cubic Bézier curves; thus, it is inherently closed-form in its expression, and the algorithm only requires the maximum curvature to be defined. The algorithm is, thus, computational efficient and easy to implement. Results show the effectiveness of the analytical algorithm in generating a continuous-curvature path, which satisfies an upper bound-curvature constraint, and that the path generated requires less control effort to track and minimizes control-input variability.**

***Index Terms*—Bézier curves, continuous-curvature path, path smoothing, upper bounded curvature constraint.**

I. Introduction

Nonholonomic motion planning in an obstacle-strewed environment is a difficult problem because the planner must simultaneously consider collision avoidance and nonholonomic constraints. Typical approaches involve generating a piecewise linear obstacle-free path using techniques such as A*, Voronoi diagrams, and probabilistic roadmaps [1]–[3] and then applying a secondary-smoothing algorithm over this linear path in order to generate a continuous path for the vehicle to follow.

The construction of smooth paths has been actively investigated by the mobile robot community because nonsmooth motions can cause slippage and overactuation [4]. Several methods have been applied, the most popular being Dubins curves [5]–[7]. This algorithm computes the shortest path between two postures in the plane via the concatenation of line segments and arcs of circles, taking into account the vehicles maximum rate of change of turn [8]. The Dubins method has been extended to other more-complex vehicle models but is still limited to line segments and arcs of circles. Discontinuities still arise at the junction of the line and arc in a Dubins curve, and these cause tracking errors. To overcome this problem, two categories of curves are used to generate a continuous-curvature path [9]. The first categories are those where the curvature is parameterized by the curve's arc length, such as Clothoids [9], [10]. Clothoid pairs provide smooth transitions with

Manuscript received October 27, 2009; revised February 3, 2010; accepted February 3, 2010. Date of publication March 11, 2010; date of current version June 9, 2010. This paper was recommended for publication by Associate Editor F. Lamiraux and Editor G. Oriolo upon evaluation of the reviewers' comments. This work was supported in part by the Australian Research Council (ARC) Centre of Excellence Program, funded by the ARC and the New South Wales State Government. This paper was presented in part at the IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, September 22–26, 2008.

The authors are with the Australian Centre for Field Robotics, The University of Sydney, Sydney, N.S.W. 2006, Australia (e-mail: k.yang@acfr.usyd.edu.au; salah@acfr.usyd.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2010.2042990

continuous curvature, have the advantage of providing the minimum-length curves for a given limit on jerk [11], and allow for optimization [12]. However, there is no closed-form expression for position along the clothoid path, and therefore, approximations and look-up tables are required.

The second categories are curves that do have a closed-form expression of position such as Bézier curves [13] and, more generally, B-splines [14]. These curves have been used for the path planning of autonomous ground vehicles [15] and unmanned aerial vehicles (UAVs) [3]. A significant disadvantage of this type of parametric curve is that its curvature is a complicated function of its parameters. Works on the use of B-splines [15] and Bézier curves [16] to generate continuous-curvature paths have not considered maximum curvature constraints. Although an optimization method that minimizes the path length by satisfying the maximum curvature constraint was proposed in [17], the work failed to solve this optimization. In [16], the maximum curvature constraint is used to check the feasibility of the path but only after path has been generated. Generally, it is not easy to use a parametric curve to design curvature-controlled curves [18]. Continuous-curvature path generation using composite Bézier curves is shown in [19], and results on the number and location of curvature extrema of a planar-parametric cubic curve are shown in [18]. However, it is not at all straightforward to generate a parametric curve that satisfies both curvature continuity and maximum curvature requirements simultaneously. In this paper, we present an algorithm that generates a cubic Bézier curve, which satisfies both the curvature continuity and maximum curvature constraints, is efficient, and has an analytical solution.

This paper is organized as follows: Section II presents the problem statement, and in Section III, we present the first-order geometric continuous path-smoothing approach that is popularly used so that we can compare it with our algorithm. We present the second-order geometric continuous path-smoothing algorithm in Section IV and the three-dimensional (3-D) second-order geometric continuous path smoothing in Section V. Section VI describes the collision-check method for Bézier curves. We present the simulation results in Section VII and the conclusion in Section VIII.

II. Problem Statement and Definitions

Barsky and DeRose [20] define the order of smoothness by parametric and geometric continuity. We follow the same definition as in [20].

Definition 1 (C^n and regularity): A scalar function $g(x)$ belongs to the class C^n on an interval I if it is n times continuously differentiable on I. It is regular on I if

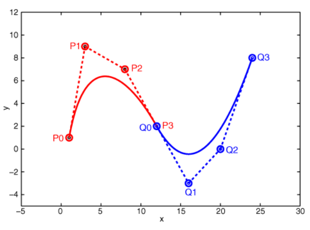
$$\frac{dg}{dx} \neq 0 \quad \forall x \in I.$$

Definition 2 (C^n continuity): Let $P(s_0, s_1 : s)$ and $Q(t_0, t_1 : t)$ be regular C^n parameterizations such that $P(s_1) = Q(t_0) = J$. That is, the right endpoint of P agrees with the left endpoint of Q. They meet with nth order parametric continuity (C^n) at J if

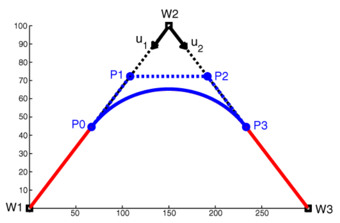
$$\left.\frac{d^k P}{ds^k}\right|_{s_1} = \left.\frac{d^k Q}{dt^k}\right|_{t_0}, k = 1, \ldots, n.$$

Definition 3 (G^n continuity): Let $P(s_0, s_1 : s)$ and $Q(t_0, t_1 : t)$ be regular C^n parametrizations such that $P(s_1) = Q(t_0) = J$, where J is a simple point of $C_P \cup C_Q$. They meet with nth-order geometric continuity (G^n) at J if the natural parameterizations of P and Q meet with C^n continuity at J.

In [20], Barsky and DeRose mention that C^n continuity will not allow flexible parametrizations because of the strict constraints of the



(a)



(b)

Fig. 1. G1CBC path smoothing. (a) G^1 continuous curve between two curves is obtained if the natural parametrizations of two curves meet with C^1 continuity at the joint. (b) If the first two control points P_0, P_1 are located between W_1 and W_2 and the last two control points P_2, P_3 are located between W_2 and W_3, a G1CBC path can be generated.

derivatives. By contrast, the constraints imposed by geometric continuity accommodate the differences between parameterizations of adjacent curve segments, and we use this to smooth the piecewise linear path. G^1 continuity is the first-order geometric continuity, which refers to continuity of the unit tangent, and G^2 continuity is the second-order geometric continuity, which refers to continuity of the curvature vectors at the joint.

Let WPS $= \{\mathrm{W}_1, \mathrm{W}_2, \ldots, \mathrm{W}_n\} = \{(x_1, y_1, z_1), (x_2, y_2, z_2), \ldots, (x_n, y_n, z_n)\}$ be a sequence of objective points in a 3-D space offered by the high-level planner. The problem is to fit these waypoints smoothly satisfying the curvature continuity and maximum curvature constraints, while simultaneously avoiding collision. Before we proceed, some preliminary ideas are in order. To stitch the piecewise linear path together, we use a Bézier curve. Let the degree n Bézier curve with $n + 1$ control points $(P_0, P_1, \ldots, P_n)$ be defined as [21]

$$P(s) = \sum_{i=0}^{n} P_i B_{n,i}(s), \quad B_{n,i}(s) = \binom{n}{i} s^i (1-s)^{n-i} \tag{1}$$

where s is $0 \leq s \leq 1$, and $B_{i,n}(s)$ are Berstein polynomials.

Since the computational cost increases with degree n, a lower order Bézier curve is preferable. A cubic Bézier curve is the minimum-degree curve in which a continuous-curvature locus can be generated.

III. G^1 Continuous Path Smoothing

Our concern is to connect two curves smoothly, as can be seen in Fig. 1(a). By using Definition 3, a G^1 continuous curve between two curves is obtained if the natural parametrizations of two curves meet with C^1 continuity at the joint. Therefore, to achieve a G^1 continuous path smoothing of the linear path, as shown in Fig. 1(b), the first two control points P_0, P_1 must be located between W_1 and W_2, and the last two control points P_2, P_3 must be located between W_2 and W_3, respectively.

From this, a G^1 continuous cubic Bézier curve (G1CBC) path can be obtained by placing four control points as follows:

$$P_0 = W_2 + d_1 u_1, \quad P_1 = W_2 + \frac{d_1 u_1}{2}$$
$$P_2 = W_2 + \frac{d_2 u_2}{2}, \quad P_3 = W_2 + d_2 u_2 \tag{2}$$

where u_1 is a unit vector along the line $\overline{W_2 W_1}$, u_2 is for $\overline{W_2 W_3}$, and d_1 is the length between W_2 and P_0, while d_2 is between that of W_2 and P_3.

Although the G^1 continuous curve has tangent continuity, there is a discontinuity of curvature at each joint. To achieve a G^2 continuous curve, the curvature must be identical at the joint.

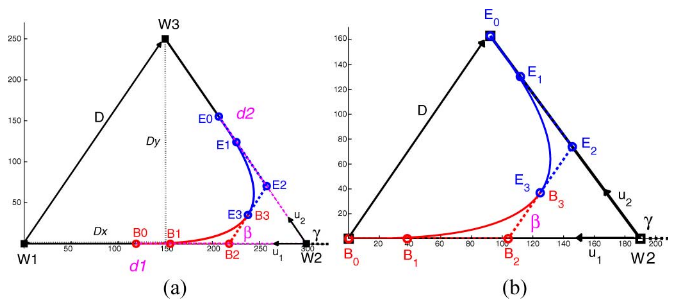


Fig. 2. (a) Conditions for G2CBS path smoothing. There are three variables (d_1, d_2, β) to generate a continuous-curvature curve. To solve this problem, a numerical algorithm must be applied to find β in (3) after deciding d_1 and d_2. (b) Analytical solution for G2CBS path smoothing. By applying $d = d_1 = d_2$ and $\beta = \gamma/2$, only one design variable (d) remains. d can also be decided analytically if the upper bound curvature is specified.

IV. G^2 Continuous Path Smoothing

A. *G^2 Continuity Condition*

Generating a G^2 continuous curve is not as simple as making a G^1 continuous curve. In [22], a planar G^2 continuous cubic Bézier spiral (G2CBS) path-smoothing algorithm is suggested, which is composed of cubic Bézier spiral segments

$$\begin{aligned} F(\beta) = {} & \cos^2(\gamma-\beta)\sin\beta[(c_1+6\cos^2\beta)D_y - 6D_x\cos\beta\sin\beta] \\ & + \cos^2\beta\sin(\gamma-\beta)[c_1(D_y\cos\gamma - D_x\sin\gamma) \\ & + 6\cos(\gamma-\beta)(D_y\cos\beta - D_x\sin\beta)] = 0 \end{aligned} \quad (3)$$

where $c_1 = 7.2364$, β is the angle between the vector $\overrightarrow{B_2W_2}$ and $\overrightarrow{B_2E_2}$, γ is the angle between the vector $\overrightarrow{W_1W_2}$ and $\overrightarrow{W_2W_3}$, and $\mathbf{D} = (D_x, D_y) = \overrightarrow{W_1W_3}$ is a vector that connects W_1 and W_3, as shown in Fig. 2(a).

To find β that satisfies the condition (3), d_1 and d_2 have to be determined first. If we decide d_1 and d_2, the length between W_2 and B_0, and between W_2 and E_0, the solution of (3) can be obtained through numerical methods. Having obtained β, the eight control points can be determined by constructing two cubic Bézier spiral curves. The first curve consists of the following four control points:

$$\begin{aligned} B_0 &= W_2 + d_1u_1, & B_1 &= B_0 - g_bu_1 \\ B_2 &= B_1 - h_bu_1, & B_3 &= B_2 + k_bu_d \end{aligned} \quad (4)$$

and the second curve consists of the following four control points:

$$\begin{aligned} E_0 &= W_2 + d_2u_2, & E_1 &= E_0 - g_eu_2 \\ E_2 &= E_1 - h_eu_2, & E_3 &= E_2 - k_eu_d \end{aligned} \quad (5)$$

where u_d is a unit vector along the line $\overline{B_2E_2}$, and

$$\begin{aligned} h_b &= \frac{(c_2+4)D_y\cos^2(\gamma-\beta)\sin\beta}{[(c_1-6)\cos\beta\sin(\gamma-\beta)+6\sin\gamma]\cos\beta\sin\gamma} \\ h_e &= \frac{h_b\cos^2\beta\sin(\gamma-\beta)}{\sin\beta\cos^2(\gamma-\beta)} \\ g_b &= c_2h_b, \quad g_e = c_2h_e \\ k_b &= \frac{6}{c_2+4}h_b\cos\beta, \quad k_e = \frac{6}{c_2+4}h_e\cos(\gamma-\beta) \\ c_2 &= \frac{2}{5}(\sqrt{6}-1). \end{aligned} \quad (6)$$

There is one known variable (γ) and three unknown variables d_1(4), d_2(5), β(6). There is no rule to decide on what values d_1 and d_2 should take, and even if there was, a numerical algorithm must be applied to find β in (3).

B. *Analytical Solution of the G^2 Continuity*

Our goal is to find the solution of (3) without using numerical methods. Originally, $\mathbf{D}$ was defined as a vector that connects W_1 and W_3. If we use $\mathbf{D}$ as a vector that connects B_0 and E_0 and assign the length of d_1 to be the same as d_2 ($d = d_1 = d_2$), which is the length between W_2 and B_0 or the length between W_2 and E_0, an analytical solution of (3) can be obtained.

Theorem 1: If $d_1 = d_2$ and $\beta = \gamma/2$, then there exists a continuous curvature curve that connects two straight lines.

Proof: If one line is aligned to the x-axis and the same design variable d is applied, D_x and D_y can be determined as follows [see Fig. 2(b)]:

$$D_x = d + d\cos\gamma, \quad D_y = d\sin\gamma. \quad (7)$$

By substituting (7) into (3), the following equation is obtained:

$$\begin{aligned} F(\beta) = {} & \cos^2(\gamma-\beta)\sin\beta\Big[(c_1+6\cos^2\beta)d\sin\gamma \\ & - 6(d+d\cos\gamma)\cos\beta\sin\beta\Big] + \cos^2\beta\sin(\gamma-\beta) \\ & \times \big[c_1\big(d\sin\gamma\cos\gamma - (d+d\cos\gamma)\sin\gamma\big) \\ & + 6\cos(\gamma-\beta)\big(d\sin\gamma\cos\beta - (d+d\cos\gamma)\sin\beta\big)\big]. \end{aligned} \quad (8)$$

Using the trignometrical properties, (8) can be arranged as follows:

$$\begin{aligned} F(\beta) = {} & c_1d\sin\gamma\Big[\cos^2(\gamma-\beta)\sin\beta - \cos^2\beta\sin(\gamma-\beta)\Big] \\ & + 6d\cos^2(\gamma-\beta)\sin\beta\Big[\cos^2\beta\sin\gamma - \cos^2\frac{\gamma}{2}\sin 2\beta\Big] \\ & + 6d\cos^2\beta\sin(\gamma-\beta)\cos(\gamma-\beta)[\sin(\gamma-\beta) - \sin\beta]. \end{aligned} \quad (9)$$

Since $F(\beta)$ is zero in (9), if β is chosen as a half of γ ($\beta = \gamma/2$), the spiral condition of (3) is always satisfied. □

If we use these conditions, all eight-control points can be easily determined. For example, h_b and h_e values can be obtained as follows:

$$\begin{aligned} h_b &= \frac{(c_2+4)d\sin 2\beta\cos^2\beta\sin\beta}{[(c_1-6)\cos\beta\sin\beta + 6\sin 2\beta]\cos\beta\sin 2\beta} = c_3d \\ h_e &= \frac{h_b\cos^2\beta\sin\beta}{\sin\beta\cos^2\beta} = h_b \end{aligned} \quad (10)$$

where $c_3 = (c_2+4)/(c_1+6)$.

Therefore, (6) can be reduced as follows:

$$\begin{aligned} h_b &= h_e = c_3d \\ g_b &= g_e = c_2c_3d \\ k_b &= k_e = \frac{6c_3\cos\beta}{c_2+4}d. \end{aligned} \quad (11)$$

From (11), we can see that the G2CBS path can be generated once we determine d because β has already been decided as half of γ.

C. Upper Bound Curvature Constraint of the G^2 Continuous Cubic Bézier Spiral Path

Although three variables (d_1, d_2, β) are normally needed to generate a continuous-curvature path, from Theorem 1, we reduce these to just one variable (d). Here, we want to decide what value d should take, which satisfies the maximum curvature constraint. This can be achieved using the following Theorem.

Theorem 2: If the maximum curvature of the vehicle is given by $\kappa_{\max}$, then the design variable d, which satisfies this maximum curvature constraint, can be determined as follows:

$$d_\kappa = \left(\frac{(c_2+4)^2}{54c_3}\right)\frac{\sin\beta}{\kappa_{\max}\cos^2\beta}.$$

Proof: The maximum curvature of the G2CBS is located at the end point of the first curve or the starting point of the second curve since its curvature monotonically increases or decreases from the starting point to the ending point.

The first curve is used to calculate the maximum curvature of the curves. At the end point of the first curve, the first and second derivative of a cubic Bézier curve are obtained as follows:

$$\dot{P}(1) = 3\overrightarrow{k_b}, \quad \ddot{P}(1) = 6\overrightarrow{k_b} - 6\overrightarrow{h_b}. \tag{12}$$

Therefore, the maximum curvature of the cubic Bézier spiral can be obtained as follows:

$$\kappa_{\max} = \frac{|\dot{P}(1)\times\ddot{P}(1)|}{|\dot{P}(1)|^3} = \frac{|3\overrightarrow{k_b}\times(6\overrightarrow{k_b}-6\overrightarrow{h_b})|}{|3\overrightarrow{k_b}|^3} = \frac{2h_b\sin\beta}{3k_b^2} \tag{13}$$

where "$\times$" is a cross product between two vectors, and $|A|$ is the norm of vector A.

Since $k_b = (6dc_3\cos\beta)/(c_2+4)$, from (11), the maximum curvature is

$$\kappa_{\max} = \frac{2(c_3 d)\sin\beta}{3\left[(\frac{6}{c_2+4})(c_3 d)\cos\beta\right]^2} = \left(\frac{(c_2+4)^2}{54c_3}\right)\frac{\sin\beta}{d\cos^2\beta}. \tag{14}$$

Therefore, the design variable d that satisfies the maximum curvature constraint can be obtained as follows:

$$d_\kappa = \frac{c_4\sin\beta}{\kappa_{\max}\cos^2\beta} \tag{15}$$

where $c_4 = (c_2+4)^2/54c_3$, and d_κ denotes the path-smoothing length satisfying the maximum curvature constraint. □

From (11) and (15), we can see that the smoothing algorithm generates a continuous-curvature path satisfying the maximum curvature constraint directly without more processing. Only $\kappa_{\max}$ is required, which specifies the upper bounded curvature value.

V. Three-Dimensional G^2 Continuous Cubic Bézier Spiral Path Smoothing

Continuous-curvature path smoothing in 3-D is a complex problem, and it is not easy to find an analytical solution. Instead of finding the solution directly in 3-D, we utilize the result of the planar case. Since the path-smoothing algorithm is applied on consecutive triplets of waypoints that forms a plane, we can apply planar G2CBS path smoothing to these three points after mapping the 3-D waypoints to the 2-D space. If we properly map these values back again into 3-D, then we can achieve 3-D G2CBS path smoothing. To do this, we need a coordinate transformation, which maps 3-D to 2-D, and *vice versa*.

The 3-D G2CBS path-smoothing algorithm is achieved using the following steps.

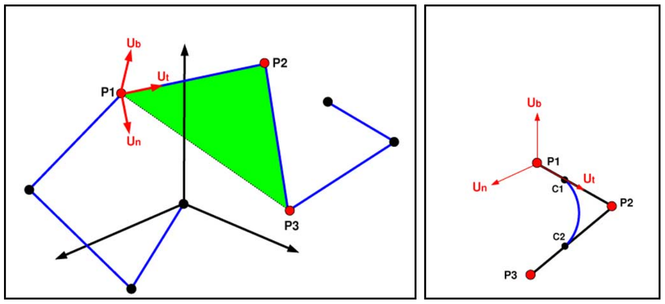


Fig. 3. Three-dimensional G2CBS path smoothing. (Left) Three consecutive points in space from a unique 2-D plane. (Right) Planar G2CBS path smoothing is applied to these consecutive three points after mapping 3-D waypoints to 2-D.

A. Construction of Orthonormal Vectors

Different three points in the space make a plane. To specify this plane, we define the local coordinate system for this plane. The Frenet frame has been used in differential geometry to analyze a curve. There are three vectors in the Frenet frame, which represent forward direction of the curve (unit tangent vector: u_t), the direction in which the curve is turning (unit normal vector: u_n), and the vector in the direction perpendicular to this plane (unit binormal vector: u_b) [23]. Although the Frenet frame is used for the arc-length parameterized curve in differential geometry, we can use this frame to specify the plane in a 3-D space. Since three waypoints are required to specify a plane, there exists an $n-2$ Frenet frame for n waypoints.

To construct the unit vector, we consider three consecutive points ($P_1P_2P_3$) as given on the left-hand side of Fig. 3.

$$P_1 = \begin{bmatrix} P_1(x)\\ P_1(y)\\ P_1(z)\end{bmatrix}, \quad P_2 = \begin{bmatrix} P_2(x)\\ P_2(y)\\ P_2(z)\end{bmatrix}, \quad P_3 = \begin{bmatrix} P_3(x)\\ P_3(y)\\ P_3(z)\end{bmatrix}. \tag{16}$$

The unit tangent vector u_t can be obtained by calculating the unit vector along the line (P_1, P_2)

$$u_t = \frac{P_2 - P_1}{|P_2 - P_1|} \tag{17}$$

Since the unit binormal vector u_b is perpendicular to the $P_1P_2P_3$ plane, it can be obtained by the cross product of the two vectors in this plane

$$u_b = u_t \times u_p \tag{18}$$

where u_p is the unit vector along the line (P_3, P_2)

$$u_p = \frac{P_2 - P_3}{|P_2 - P_3|}. \tag{19}$$

Finally, the unit normal vector u_n can be obtained from the cross product between u_t and u_b

$$u_n = u_b \times u_t. \tag{20}$$

These three vectors are mutually orthogonal unit vectors, which defines the local coordinate system consisting of three waypoints. Fig. 4 shows the orthonormal vectors of a collection of the 3-D waypoints. There exist seven sets of orthonomal vectors since there are nine waypoints.

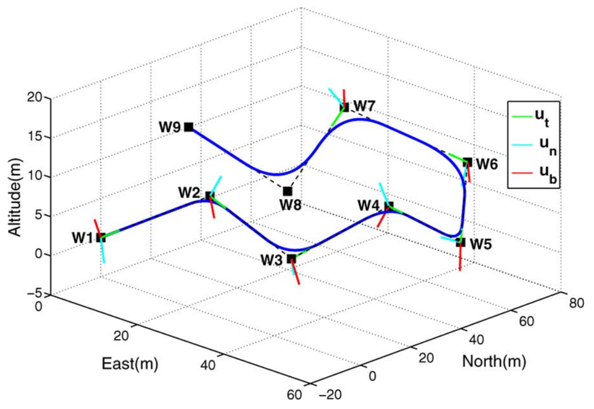


Fig. 4. Three-dimensional G2CBS path-smoothing example. Since there are nine waypoints, seven local coordinate frames are constructed. The computational complexity of 3-D G2CBS path smoothing compared with planar G2CBS path smoothing is negligible, since the 3-D G2CBS path-smoothing process only requires the calculation of the orthonormal vectors.

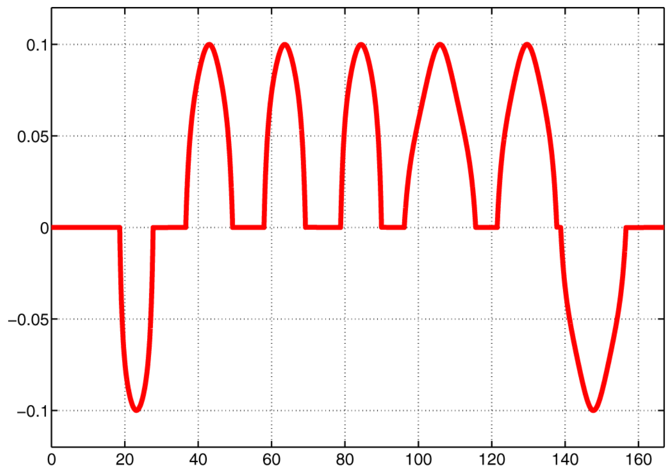


Fig. 5. Curvature of the 3-D G2CBS path. It satisfies both the curvature continuity and the upper bound limit of the curvature.

B. Mapping 3-D Waypoints to 2-D

The next step is to map the 3-D waypoints to the local 2-D space. The center of the local coordinate system is the first waypoint. To achieve this, we construct the following transformation matrix:

$$\mathbf{TM} = \begin{bmatrix} u_t & u_n & u_b & P_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{21}$$

Using the inverse of this transformation matrix, the 3-D waypoints can be transformed to the 2-D plane by using

$$P_{2D} = \mathbf{TM}^{-1} \cdot P_{3\text{-D}} \tag{22}$$

where $\mathbf{TM}^{-1}$ is an inverse matrix of the transformation matrix.

C. Applying Planar G2CBS Path Smoothing

After obtaining the three waypoints in 2-D, the planar G2CBS path-smoothing algorithm will be applied to these waypoints. The planar G2CBS path can be obtained using (4), (5), and (11), which generates the following matrix:

$$\mathbf{S}_{2\text{-D}} = \begin{bmatrix} B_0(x) & \cdots & B_3(x) & E_3(x) & \cdots & E_0(x) \\ B_0(y) & \cdots & B_3(y) & E_3(y) & \cdots & E_0(y) \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ 1 & \cdots & 1 & 1 & \cdots & 1 \end{bmatrix}. \tag{23}$$

D. Mapping 2-D Waypoints to 3-D

The transformation of 2-D cubic Bézier control points to 3-D is the inverse process of (22)

$$\mathbf{S}_{3\text{-D}} = \mathbf{TM} \cdot \mathbf{S}_{2\text{-D}}. \tag{24}$$

The transformation from 3-D to 2-D in (22) is applied for the three waypoints, but the transformation from 2-D to 3-D in (24) is applied for eight control points, which is the result of the planar G2CBS path-smoothing algorithm.

E. Three-Dimensional G^2 Continuous Cubic Bézier Spiral Path Smoothing

Since we obtained the control points in a 3-D space, the 3-D cubic Bézier curve can be obtained using (1).

C_1 and C_2 points in Fig. 3 (right-hand side) represent the first and the last control points in (23)

$$C_1 = \begin{bmatrix} B_0(x) \\ B_0(y) \\ 0 \end{bmatrix} C_2 = \begin{bmatrix} E_0(x) \\ E_0(y) \\ 0 \end{bmatrix}. \tag{25}$$

The 3-D G2CBS path-smoothing algorithm always guarantees the continuity of the curvature since the curvature at C_1 and C_2 points are zero.

Fig. 4 shows the 3-D G2CBS path-smoothing result. The computational cost of the 3-D G2CBS path-smoothing process compared with the 2-D G2CBS path-smoothing process is almost the same since the 3-D G2CBS path-smoothing process requires only the calculation of the orthonormal vectors. Fig. 5 shows the curvature of the path after the 3-D G2CBS path smoothing. We set the maximum curvature value as 0.1 for this case. It satisfies both the curvature continuity and the upper bound limit of the curvature.

VI. Collision Check

Arc-length parameterization of the path is required to detect collisions along the path with equidistance interval. The arc length for a parametric curve is a geometric integration given two parameters s_1 and s_2

$$L(s) = \int_{s_1}^{s_2} \sqrt{\dot{x}(s)^2 + \dot{y}(s)^2 + \dot{z}(s)^2} ds \tag{26}$$

where $0 \le s_1 \le s_2 \le 1$.

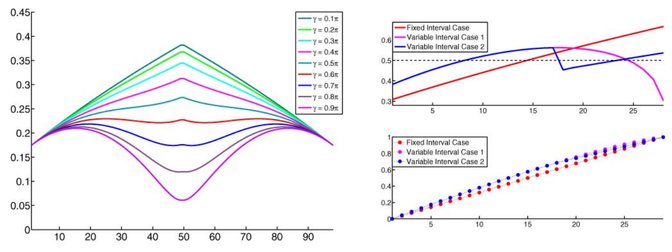
There is no analytical solution for this integral; therefore, in general numerical methods are used to calculate the length of the path [13]. It is difficult to bound the distance between two values of the parameters to check for collision since the curve cannot be parameterized by the arc-length, and the parameter variables are not linearly related to the curve length. Since an accurate fixed distance between two parameters is not a strict requirement for collision check, the problem can be relaxed to parameterize the G2CBS path which has an approximately similar distance between two parameters. To achieve this, we have to answer the following two questions.

1) How many knots are required?
2) How can we distribute these knots to get approximately equidistance spacing between knots?

The knot represents the specific parameter s value, which is defined as $s \in [0, 1]$.

Algorithm 1 Knot Spacing for the Collision Check

```
1: $n_c = \mathsf{Ceil}\left(\frac{L_{bc}}{d_c} + 1\right)$
2: if $\gamma < 0.5\pi$ then
3:     $s = \left|\mathsf{LS}(1, 0, n_c)^{(1+k_1 \cos\gamma)} - 1\right|$
4:     $n_r = \mathsf{Round}(0.65 \times n_c)$
5:     $s(n_r : end) = \mathsf{LS}\Big(s(n_r), 1, (n_c+1-n_r)\Big)^{(1+k_2 \cos\gamma)}$
6: else if $\gamma \geq 0.5\pi$ then
7:     $s = \left|\mathsf{LS}(1, 0, n_c)^{(1-k_3(\cos\gamma)^4)} - 1\right|$
8: end if
```



(a) (b)

Fig. 6. Relationship between the ALBK and the angle. (a) ALBK is monotonically increased if the angle (γ) is smaller than 0.5π, and it is decreased if the angle is closer to π. (b) (Top) Arc length between knots and (bottom) the distribution of knots for $\gamma = 0.1\pi$ case.

To distribute knots to achieve similar curve distances between them, we devise Algorithm 1 for the cubic Bézier curve.

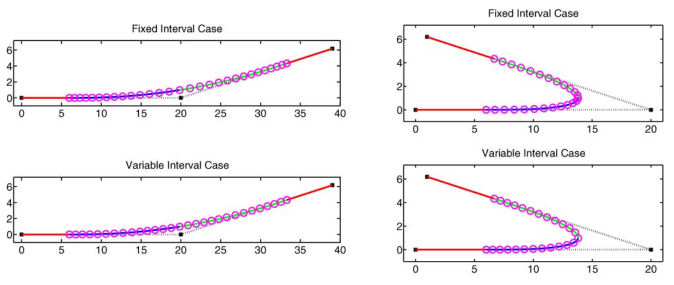
In Algorithm 1, $\mathsf{Ceil}(A)$ rounds the elements of A to the nearest integers greater than or equal to A, $\mathsf{Round}(A)$ rounds the elements of A to the nearest integers, and $\mathsf{LS}(a, b, n)$ generates n point vectors, which are linearly spaced between a and b [24]. L_{bc} is a length of the Bézier curve, d_c is a collision check interval, and γ is an angle between waypoints as in Fig. 2. k_1, k_2, and k_3 are the design parameters to control the distribution of knots, which are $k_1 = 0.25$, $k_2 = 0.03$, and $k_3 = 0.3$ in our application.

First, the number of knots (n_c) is calculated by dividing the length of Bézier curve (L_{bc}) by the collision check interval (d_c) (line 1). We add 1 to the $\mathsf{Ceil}\,(L_{bc}/d_c)$ since we need $n+1$ knots to divide an n distance interval. Gravesen suggested an algorithm that can approximate the length of the Bézier curves efficiently without using a numerical method in [25]

$$L_{bc} = \frac{2L_c + (n-1)L_p}{n+1} = \frac{L_c + g + h + k}{2} \tag{27}$$

where L_c denotes the length of the chord, L_p is the sum of all polygon lengths, and n is the order of the Bézier curve. In our application, $n = 3$, since a cubic Bézier curve is used, and $L_p = g + h + k$. Therefore, the length of the cubic Bézier curves is obtained by the right part of (27). Using (27), the number of knots (n_c) for collision check can be decided by line 1 in Algorithm 1.

After obtaining a required knot number, the next task is to distribute knots to achieve roughly equidistance spacing between them since the knot and the curve length are not linearly related. The relationship between the arc length between knots (ALBK) and angle (γ) between waypoints is shown in Fig. 6(a). The 50 knots are linearly distributed in each curve, which constitute the G2CBS curve (one is a monotonically increasing curvature path, and the other is a monotonically decreasing curvature path). The ALBK monotonically increases if γ is smaller than 0.5π, and the increasing rate decreases if the angle approaches 0.5π. If γ is larger than 0.5π, the ALBK initially increases, then it



(a) (b)

Fig. 7. Comparison of the knot spacing performance. (a) $\gamma = 0.1\pi$. (b) $\gamma = 0.9\pi$. The ALBK in the fixed-interval case monotonically increases when γ is 0.1π, and it monotonically decreases when γ is 0.9π, but the proposed method maintains a similar ALBK.

decreases after passing some value, and the decreasing rate is higher if the angle is closer to π.

To accommodate the effect of angle γ on knot spacing, we first use γ in a power index of LS (line 3, 5, 7 in Algorithm 1), and second, we split the knot spacing rule for the $\gamma \leq 0.5\pi$ case, as in lines 2–5 and for the $\gamma \geq 0.5\pi$ case, as in lines 6 and 7 in Algorithm 1. The reason for the inclusion of lines 4 and 5 is that the decreasing rate of the ALBK is too high when s is close to 1. Fig. 6(b) shows three different ALBK and knot spacings for $\gamma = 0.1\pi$. The collision-check interval is $d_c = 0.5$ m in this example. The fixed-interval case distributes knots as $\mathsf{LS}(0, 1, n_c)$, which has the same space between knots. The ALBK monotonically increases from 0.31 to 0.67 m. The variable interval case 1 distributes knots by considering the effect of the angle γ (line 3 in Algorithm 1). The ALBK of variable-interval case 1 monotonically increases until 65% of the knot and begins to decrease after this knot point. It becomes very small as the knot gets closer to 1, as can be seen in the top of Fig. 6(b). To solve this problem, we redistribute knots when 65% of the knot number using lines 4 and 5 in Algorithm 1 (variable interval case 2) is reached. The mean of the ALBK is 0.496 m in all the three cases, but the standard deviation of the ALBK of the variable interval case 2 (0.048) is below half of the fixed-interval case (0.108).

Fig. 7 shows the collision-check point on the path of the fixed-interval method and the proposed method, respectively. The ALBK in the fixed-interval method monotonically increases when γ is 0.1π, and it monotonically decreases when γ is 0.9π, but the proposed method maintains a similar ALBK.

VII. Simulation Experiments

We now compare the performance of the G1CBC and G2CBS path smoothing algorithms. A rapidly exploring random tree (RRT) algorithm is used to generate a collision-free path among obstacles, which in turn generates a piecewise linear path. First, we show the performance of the 3-D G2CBS path smoothing satisfying the upper bound curvature constraint. Second, the 3-D G1CBC path-smoothing and the G2CBS path-smoothing results are compared. Finally, a fixed-wing UAV (FUAV) model is use to analyze the path-tracking performance to see the effect of the curvature continuity in different path scenarios.

A. Three-Dimensional G^2 Continuous Cubic Bézier Spiral Path Smoothing

Fig. 8(a) shows a 3-D collision-free path generated by the RRT path planner. Unnecessary waypoints are removed as can be seen in Fig. 8(b), which result in a piecewise linear path (this is done within

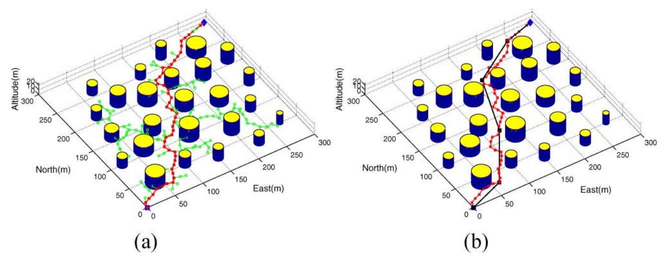


Fig. 8. Path planning using RRT algorithm. (a) RRT path planner generates a 3-D collision-free path among obstacles. (b) Most extraneous nodes are eliminated by the path-pruning algorithm.

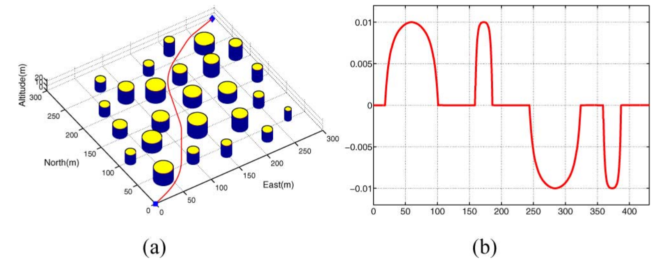


Fig. 9. 3-D G2CBS path-smoothing algorithm. (a) G2CBS path-smoothing algorithm generates a smooth path satisfying the upper bounded curvature constraint. (b) Curvature of the G2CBS path is continuous over the whole path, and the curvature is smaller than the specified maximum curvature (0.01).

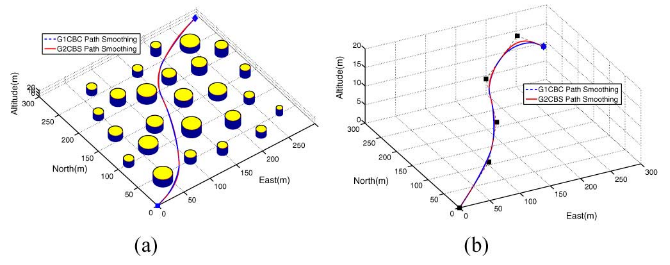


Fig. 10. Comparison between the G1CBC and the G2CBS path smoothing. (a) Blue dotted line is the G1CBC path, and the red line is the G2CBS path. (b) Paths without obstacles are plotted to clearly see the 3-D path-smoothing results.

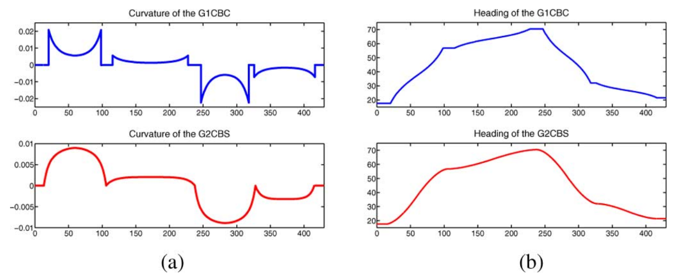


Fig. 11. Comparison of the curvature and heading of the G1CBC and G2CBS paths. (a) Curvature of the G1CBS path is discontinuous at the junction points, while the curvature of G2CBS path is continuous over the whole path. (b) Change of headings are very sharp at the joints in the G1CBC path, but it is smooth in the G2CBS path.

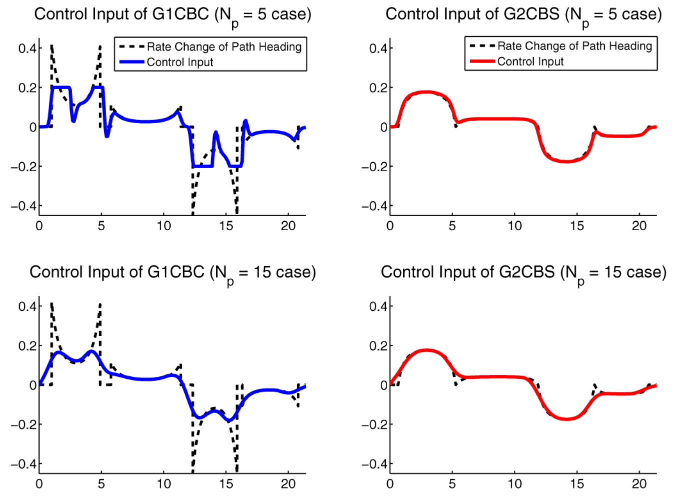


Fig. 12. Rate change of path heading ($\dot{\psi}_p$) and the control input of the G1CBC and G2CBS path. Saturations occur and the variation of the control input is severe when a small control horizon ($N_p = 5$) is applied. By increasing the control horizon ($N_p = 15$), the saturation of the G1CBC path is removed, but it increases the tracking error and the computational cost required for optimization. However, the control input variation and the amount of control energy of the G2CBS path is smaller than the G1CBC path, and the control input of the G2CBS path is almost same as the rate change of heading.

the path-planning algorithm, the details of which are not needed for this study. (See [26] for this planning.) The maximum curvature of this path is set as 0.01. Fig. 9(a) shows the result of the 3-D G2CBS path-smoothing algorithm. The 3-D G2CBS path-smoothing algorithm generates a continuous curvature path that satisfies the upper bounded curvature constraint, as can be seen in Fig. 9(b).

B. Comparison Between G1CBC and G2CBS Path Smoothing

The G1CBC and the G2CBS path-smoothing algorithms are applied to the polygonal path in Fig. 8(b), which utilize the remaining length margin [27]. Fig. 10(a) shows the paths generated in cluttered environment, and Fig. 10(b) shows only paths without obstacles to clearly see the 3-D path-smoothing results. There is seemingly little difference between the two paths, but curvature of these paths is very different, as can be seen in Fig. 11(a). The curvature of the G1CBS path is discontinuous at the junction points, while the curvature of the G2CBS path is continuous over the whole path. Fig. 11(b) shows the heading angle of the two paths. The change of headings is very sharp at the joints in the G1CBC path but are smooth in the G2CBS path.

C. FUAV Path Tracking

A nonlinear model predictive control (NMPC) process is used for the FUAV path tracking. The paths in Fig. 10 are used as reference paths, but altitudes are fixed since we apply a planar FUAV kinematic model for the tracking. All simulation conditions and the FUAV model are the same as in [28]. The FUAV flies 20 m/s, and the control frequency is 10 Hz. The yaw angular rate is restricted to ± 0.2 rad/s. Since the FUAV flies 20 m/s, and the maximum yaw angular rate is 0.2 rad/s, the maximum curvature that the FUAV can follow is $\kappa_{\max} = \dot{\psi}_{\max}/V = 0.01$.

Fig. 12 shows the rate change of path heading ($\dot{\psi}_p$) and the control input of the G1CBC and the G2CBS path. The rate change of path heading is obtained by $\dot{\psi}_p = \kappa \times V$. Because the G1CBC path violates the maximum curvature constraints, control input saturations occur, and the variation of the control input is severe [using a small control horizon ($N_p = 5$)] (see the top-left of Fig. 12). However, the control input variation and the amount of control energy of the G2CBS path is smaller than the G1CBC path (see the top-right of Fig. 12).

A larger control horizon gives more damping to the tracking controller reducing the overshoot of the FUAV response. By increasing the control horizon ($N_p = 15$), the saturation of the G1CBC path is removed (see the bottom-left of Fig. 12), but still requires more control input and has a large variation of the control input compared with the G2CBS path (see the bottom-right of Fig. 12). Moreover, if a larger control horizon is applied to the path, the tracking error is increased due to the corners cut, and the computational cost required for the optimization grows with the size of the control horizon. Although the G1CBC path needs a larger control horizon to avoid the control-input saturation, which results in the increased tracking error and computational time, the G2CBS path can achieve a accurate path tracking with smaller control horizon.

VIII. Conclusion

This paper proposes a continuous-curvature path smoothing algorithm, which fits an ordered sequence of waypoints generated by the obstacle-avoidance path planner satisfying maximum curvature constraints. The maximum curvature is the only required information for the generation of the G2CBS path. If this value is specified, the algorithm generates a collision-free continuous curvature path satisfying the upper bound curvature directly and provides a complete analytical solution. Simulation results show the proposed algorithm combined with the high-level path planner can generate an upper bound continuous-curvature path in a cluttered environment. The path generated by the G2CBS path requires much less control effort to track the path and minimizes the variability of the control input. For future work, we want to combine the analytical solution of the G2CBS path smoothing into the path-planning process. Currently, the G2CBS path-smoothing algorithm is applied only after the path planner generates a path. Since the proposed algorithm can calculate the required length for the continuous and upper bound curvature path, the planner can generate a path that satisfies these two constraints.

References

[1] Y. Qu, Q. Pan, and J. Yan, "Flight path planning of UAV based on heuristically search and genetic algorithms," in *Proc. 31st Annu. Conf. IEEE Ind. Electron. Soc.*, Nov. 2005, p. 5.

[2] R. Beard, T. McLain, M. Goodrich, and E. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Trans. Robot. Autom.*, vol. 18, no. 6, pp. 911–922, Dec. 2002.

[3] P. Pettersson and P. Doherty, "Probabilistic roadmap based path planning for an autonomous unmanned helicopter," *J. Intell. Fuzzy Syst.*, vol. 17, no. 4, pp. 395–405, 2006.

[4] E. Magid, D. Keren, E. Rivlin, and I. Yavneh, "Spline-based robot navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, Oct. 2006, pp. 2296–2301.

[5] P. Jacobs and J. Canny, "Planning smooth paths for mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Scottsdale, AZ, May 1989, pp. 2–7.

[6] T. Fraichard, "Smooth trajectory planning for a car in a structured world," in *Proc. IEEE Int. Conf. Robot. Autom.*, Sacramento, CA, Apr. 1991, pp. 218–323.

[7] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Proc. IEEE Conf. Decis. Control*, Las Vegas, NV, 2002, pp. 1301–1306.

[8] 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.*, vol. 79, pp. 497–516, 1957.

[9] T. Fraichard and A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *IEEE Trans. Robot.*, vol. 20, no. 6, pp. 1025–1035, Dec. 2004.

[10] T. Kito, J. Ota, R. Katsuiu, T. Mizuta, T. Arai, T. Ueyama, and T. Nishiyama, "Smooth path planning by using visibility graph-like method," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, Sep. 2003, pp. 3770–3775.

[11] W. Nelson, "Continuous-curvature paths for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Scottsdale, AZ, 1989, pp. 1260–1264.

[12] S. Fleury, P. Soueres, J.-P. Laumond, and R. Chatila, "Primitives for smoothing mobile robot trajectories," *IEEE Trans. Robot. Autom.*, vol. 11, no. 3, pp. 441–448, Jun. 1995.

[13] J. Hwang, R. Arkin, and D. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, Oct. 2003, pp. 1444–1449.

[14] K. Komoriya and K. Tanie, "Trajectory design and control of a wheel-type mobile robot using B-spline curve," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Tsukuba, Japan, Sep. 1989, pp. 398–405.

[15] J. Connors and G. Elkaim, "Analysis of a spline based, obstacle avoiding path planning algorithm," presented at the IEEE 65th Veh. Technol. Conf., Apr. 2007, Tiger, Dublin, Ireland.

[16] K. Nagatani, Y. Iwai, and Y. Tanaka, "Sensor-based navigation for car-like mobile robots based on a generalized Voronoi graph," *Adv. Robot.*, vol. 17, no. 5, pp. 385–401, 2003.

[17] M. Khatib, H. Jaouni, R. Chatila, and J. P. Laumond, "Dynamic path modification for car-like nonholonomic mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Albuquerque, New Mexico, Apr. 1997, pp. 2920–2925.

[18] D. J. Walton and D. S. Meek, "Curvature extrema of planar parametric polynomial cubic curves," *J. Comput. Appl. Math.*, vol. 134, no. 1–2, pp. 69–83, Sep. 2001.

[19] B. A. Barsky and T. D. DeRose, "Geometric continuity of parametric curves: constructions of geometrically continuous splines," *IEEE Comput. Graph. Appl.*, vol. 10, no. 1, pp. 60–68, Jan. 1990.

[20] B. A. Barsky and T. D. DeRose, "Geometric continuity of parametric curves," Univ. Calif., Berkeley, Tech. Rep. UCB/CSD 84/205, 1984.

[21] R. H. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for use in Computer Graphics and Geometric Modeling*. Los Altos, CA: Morgan Kaufmann, 1986.

[22] D. J. Walton, D. S. Meek, and J. M. Ali, "Planar G2 transition curves composed of cubic Bezier spiral segments," *J. Comput. Appl. Math.*, vol. 157, no. 2, pp. 453–476, Aug. 2003.

[23] R. L. Finney, M. D. Weir, and F. R. Giordano, "Thomas's Calculus," in *Computational Geometry*, Reading, MA: Addison-Wesley, 2001.

[24] (2009). [Online]. Available: http://www.mathworks.com/

[25] J. Gravesen, "Adaptive subdivision and the length and energy of Bezier curves," *Comput. Geometry*, vol. 8, no. 1, pp. 13–31, Jun. 1997.

[26] K. Yang and S. Sukkarieh, "3D smooth path planning for a UAV in cluttered natural environments," presented at the IEEE/RSJ Int. Conf. Intell. Robots Syst., Nice, France, Sep. 2008.

[27] K. Yang and S. Sukkarieh, "Planning continuous curvature paths for UAVs amongst obstacles," presented at the Australas. Conf. Robot. Autom., Canberra, Australia, Dec. 2008.

[28] K. Yang, S. Sukkarieh, and Y. Kang, "Adaptive nonlinear model predictive path tracking control for a fixed-wing unmanned aerial vehicle," presented at the AIAA Guid., Navigat. Control Conf., Chicago, IL, Aug. 2009.