Proceedings of the 36th Chinese Control Conference
July 26-28, 2017, Dalian, China

Dynamically feasible trajectory generation method for quadrotor unmanned vehicles with state constraints

Shupeng Lai[1], Kangli Wang[1], Ben M. Chen[1]

1. Department of Electrical Computer Engineering, National University of Singapore.
E-mail: elelais@nus.edu.sg, wangkangli@u.nus.edu, elebm@nus.edu.sg

Abstract: The recent development on micro aerial vehicles (MAV), especially the unmanned quadrotor proposes increased requirements on reference generation. This paper proposed a dynamically feasible, state constrained and smooth trajectory generation method using B-spline. The reference can be shown to satisfy the dynamics of the quadrotor and accurate tracking performance is guaranteed. The algorithm can be applied for filming trajectory design, 3D environment reconstruction and entertainment events. The algorithm has been tested on real vehicles with actual flight experiment with satisfying results.

Key Words: Quadrotor, Trajectory generation, B-spline, Optimization

1 Introduction

Traditionally, an aerial vehicle adopts a waypoint or line-segment based strategy as its basic mission elements. That is, the vehicle is to fly towards a certain point or over a certain flight path. Methods such as pure pursuit [1], L1 nonlinear guidance law [2] and vector field path following [3] are commonly used to guide different types of vehicles on these mission. However, with the growing popularity of quadrotors and their application in area such as filming, entertaining, industrial monitoring and other low attitude mission, it is more difficult to assign tasks with only points and line segments.

A solution introduced from the robotics community is to find a dynamically feasible reference trajectory to suit user's need. The flying trajectory could be reviewed before execution and a wide range of controller is capable of tracking the reference with bounded error. The reviewing process could either be offline or online, with human operator or automatically. Such an approach has the capability of handling complicated flying missions with high tracking precision. It allows quadrotor to perform tasks like flying through narrow gap [4], to intercept fast moving object [5], to cooperate and construct buildings [6], to perform multi-UAV light show or even writing calligraphy [7].

Three major types of the trajectory generation algorithms are:

1) Numerical optimization based: the trajectory of a dynamic system can be expressed either directly through discretization or indirectly using base functions [8]. With these expressions, an optimization problem can be formulated (usually on minimization of total energy or time) and solved numerically. Furthermore, the quadrotor is shown to be differentially flat [4] and the dynamic feasibility depends on the derivatives of the position and heading angle. In [4], minimum snap trajectories are first adopted by the quadrotors for aggressive flight. In [9], the algorithm is improved for guiding a quadrotor in obstacle-strewn environments.
2) Path re-parametrization based: a pure geometric path is first designed using primitives such as line segments [10], polynomials [11] and splines [12]. An augmented velocity profile along the geometric path guarantees its dynamic feasibility. Such methods are usually seen in the field of CNC machining where the milling path needs to be followed precisely.
3) Boundary value problem (BVP) based: some applications must be conducted in dynamic and partially known environment while considering the computational and economical limitations. A reliable and real-time trajectory planning become necessary. BVP solvers with superior realtime capability is usually considered for such an purpose. The entire trajectory planning problem is decomposed into a series of BVPs which are solved sequentially. It has been utilized on quadrotor for navigating the unknown environment and intercepting of moving objects.

With the increased popularity in quadrotor aero filming, reconstruction and entertaining performance where a high-quality preplanned trajectory is needed. The numerical optimization based methods have received extra notice. However, the rapid design of dynamically feasible and derivative-constrained trajectory still remains a problem. In [4] and [9] a polynomical spline based trajectory generation algorithm is proposed, but it requires multiple quadratic programming to fulfill the derivative constraints. While in [13] and [6], BVP based methods handle the constraints but only consider a two-point BVP case (only the initial and final states).

In this paper, we present a B-spline based trajectory generations algorithm that is capable of designing complex flying trajectory while satisfying the derivative constraints. Similar to [4], the problem is to search for the smooth trajectory crossing several waypoints. The quartic B-spline is used as the base function. The optimization formulation is similar to [15] but with the clamped (instead of open) normalized uniform B-spline (CNUBS) considered. A time sequence estimation and optimization procedure is also covered. With it a trajectory can be generated from pure positional input.

The organization of the paper is as follows. Section 2 introduces the quadrotor model while Section 3 discusses the optimization problem and its solution with simulation results. In Section 4, real flight experiment data in windy environment using micro aerial vehicle is given. Finally in Section 5, a conclusion is made on the presented approach.

2 Quadrotor Dynamic Model

To express both the translational and rotational movement, the quadrotor is usually expressed in a body frame $\mathbf{B}$ and a global frame $\mathbf{G}$ as shown in Figure 1. Here, the $[\mathbf{x}_\mathrm{B}, \mathbf{y}_\mathrm{B}, \mathbf{z}_\mathrm{B}]$

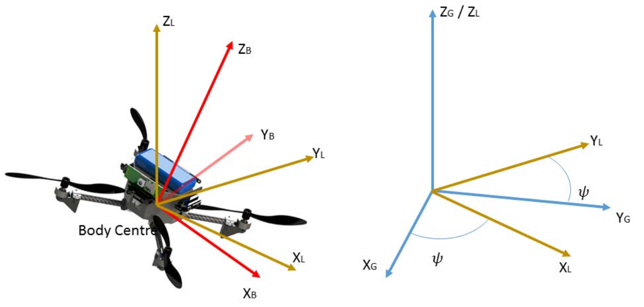

Fig. 1: The coordinate systems

and $[\mathbf{x}_\mathrm{G}, \mathbf{y}_\mathrm{G}, \mathbf{z}_\mathrm{G}]$ are the base vectors of the corresponding frame. According to [4], the quadrotor model can now be expressed as:

$$m\ddot{r} = -mg\mathbf{z}_\mathrm{G} + u_1\mathbf{z}_\mathrm{B} \tag{1}$$

$$\dot{\omega} = I^{-1}\left[-\omega \times I\omega + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}\right] \tag{2}$$

where m is the mass of the vehicle, r denotes the position of its center of gravity, ω is its angular velocity and I is its moment of inertia, and u_1, u_2, u_3, u_4 are the inputs to the system. From Eq. 1, u_1 can be expressed as a function of $\ddot{r}$. By taking the first derivative of Eq. 1, ω can be expressed as a function of $\dddot{r}$. Whereas by the taking second derivative of Eq. 1, $\dot{\omega}$ can be expressed as a function of $\ddddot{r}$. Therefore, the input u_2, u_3, u_4 can also be expressed as a function of $\ddddot{r}$. To suppress the energy consumption of the system and achieve smooth flying experience, a choice is to design trajectory that minimizes $\int w_\mathrm{a}\ddot{r}^2 + w_\mathrm{j}\dddot{r}^2 + w_\mathrm{s}\ddddot{r}^2$ with $w_\mathrm{a}, w_\mathrm{j}, w_\mathrm{s}$ being the weightings.

3 Optimal Trajectory Design

3.1 Quartic CNUBS

The task of the trajectory designing process is to generate user-desired and dynamically feasible references for the quadrotor. With the presented model, the inputs appear as functions of the forth derivative of the positional trajectory. Therefore, we choose to deisgn trajectory with at least C_3 continuity. Further, for the dynamically feasibility and smooth flight experience, the derivatives of the trajectory are to be constrained, and the integral of the square of their norms are to be minimized. Similar problem formulation with successful implementation can also be found in [4, 9, 17].

A quartic CNUBS is chosen in our problem formulation for its capability of prducing C_3 continous trajectory. It is similar to [16] but with a clamped base. The clamped spline is considered so that the flying path is tangent to the first and last legs formed by the spline control points to match human intuition.

A kth-order CNUBS can be writen in the followin form:

$$S_\mathrm{k}(s) = \sum_{i=0}^{M-1} c_i N_{i,k}(s), \quad c_i \in \mathbb{R}, \tag{3}$$

where $N_{i,k}$ is called the base function and c_i is the control points of the B-spline. With a predefined knot vector $S = [s_0, s_1 ... s_m]$ a path parameter s, the base function of B-spline is recursively defined as:

$$\begin{aligned} N_{i,0}(s) &= \begin{cases} 1, & \text{if } s_i \le s < s_{i+1}, \\ 0, & \text{otherwise.} \end{cases} \\ N_{i,k}(s) &= N_A(s,i,k)N_{i,k-1}(s) + N_B(s,i,k)N_{i+1,k-1}(s) \end{aligned} \tag{4}$$

where

$$\begin{aligned} N_A(s,i,k) &= \begin{cases} 0, & \text{if } s_i = s_{i+k}, \\ \frac{s-s_i}{s_{i+k}-s_i}, & \text{otherwise.} \end{cases} \\ N_B(s,i,k) &= \begin{cases} 0, & \text{if } s_{i+1} = s_{i+k+1}, \\ \frac{s_{i+k+1}-s}{s_{i+k+1}-s_{i+1}}, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

for a quartic CNUBS with M control points, the knot vector is

$$S = [0, 0, 0, 0, 0, 1, 2..., M-4, M-4, M-4, M-4, M-4]. \tag{6}$$

The base function with $M = 4$ is shown in Figure 2. Further, following [14], a linear relation between the path paramenter s and the time t can be assigned as

$$\frac{s}{t} = \alpha \tag{7}$$

thus the time derivative of the trajectory can be assessed.

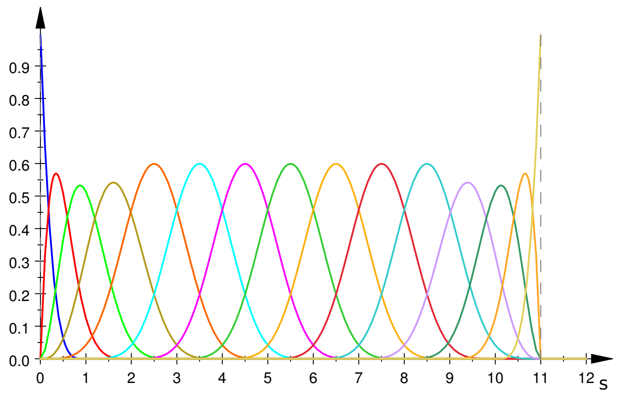

Fig. 2: Base function of quartic clamped B-spline

3.2 Optimal Trajectory

During trajectory design, the input usually consists of

1) A data vector $D_\mathrm{s} = [d_0, d_1, ..., d_I]^\mathrm{T}, d_i \in \mathbb{R}$.
2) A time vector indicating when the corresponding data is sampled $D_\mathrm{time} = [t_0, t_1, ..., t_I]^\mathrm{T}, t_i \in \mathbb{R}$.

For the 3D flight trajectory, there are three data vectors, one for each dimension: $\{D_{\mathrm{s}x}, D_{\mathrm{s}y}, D_{\mathrm{s}z}\}$. However, in many situations, the time vector is not given and needs to be optimized individually. In this paper, if the time vector is provided, it is refered as an interpolation problem. Otherwise, it is called a designing problem.

3.2.1 Trajectory Interpolation

We first focus on the problem with a time vector being provided. As shown in [15], for optimal smoothing and interpolation, the optimization target for each dimension can be expressed as

$$J_\mathrm{s} = \sum_n \int_{-\infty}^{\infty} w_n J_{\mathrm{k,n}}^2(t)dt + \sum_i (S_k(\alpha t_i) - d_i)^2, \quad (8)$$

where the $J_{\mathrm{k,n}}$ is defined as

$$J_{\mathrm{k,n}}(t) = \frac{d^n}{dt^n} S_k(s). \quad (9)$$

The first part of the cost function denotes the integral of the square of the norm of the derivatives. It produces smooth the trajectories. The second part represents the cost for deviating from the input data points D_s.

Following [14], $\int_{-\infty}^{\infty} w_n J_{\mathrm{k,n}}^2(t)dt$ can be rewritten in a quadratic form as:

$$\int_{-\infty}^{\infty} w_n J_{\mathrm{k,n}}^2(t)dt = {C_\mathrm{s}^k}^T G_{\mathrm{k,n}} C_\mathrm{s}^k \quad (10)$$

where

$$C_\mathrm{s}^k = [c_0^k, c_1^k, ...c_{M-1}^k] \quad (11)$$

is the control point vector and $G_{\mathrm{k,n}}$ is a square matrix with its element defined as

$$G_{\mathrm{k,n}}(i,j) = \alpha^{2n-1} \int_{-\infty}^{\infty} \frac{d^n N_{i,k}(s)}{ds^n} \frac{d^n N_{j,k}(s)}{ds^n} ds. \quad (12)$$

The deviation from data points $\sum_i (S_k(\alpha t_i) - d_i)^2$ can be expressed as

$$\sum_{i=0}^{I} (S_k(\alpha t_i) - d_i)^2 = (HC_\mathrm{s}^k - D_\mathrm{s})^\mathrm{T}(HC_\mathrm{s}^k - D_\mathrm{s}) \quad (13)$$

with

$$H = \begin{bmatrix} N_{0,k}(\alpha t_0) & N_{1,k}(\alpha t_0) & ... & N_{M-1,k}(\alpha t_0) \\ N_{0,k}(\alpha t_1) & N_{1,k}(\alpha t_1) & ... & N_{M-1,k}(\alpha t_1) \\ \vdots & \vdots & \ddots & \vdots \\ N_{0,k}(\alpha t_I) & N_{1,k}(\alpha t_I) & ... & N_{M-1,k}(\alpha t_I) \end{bmatrix} \quad (14)$$

Substitute Eq. 10 and 13 into Eq. 8, it gives

$$\begin{aligned} J_{\mathrm{s}\min} &= \min_{C_\mathrm{s}^k} \Big\{ {C_\mathrm{s}^k}^\mathrm{T} (\textstyle\sum_n w_\mathrm{n} G_{\mathrm{k,n}} \\ &+ H^\mathrm{T} H) C_\mathrm{s}^k - 2D_\mathrm{s}^\mathrm{T} H C_\mathrm{s}^k + D_\mathrm{s}^\mathrm{T} D_\mathrm{s} \Big\} \end{aligned} \quad (15)$$

which is a quadratic programming problem. Following the analysis in [14], since Eq. 8 consists of only square terms, the quadratic programming problem in Eq. 15 is convex. Following the analysis in [6], to guarantee the dynamic feasibility, the trajectory's derivatives such as velocity and acceleration shall also be limited. In [16], derivative constraints of the open spline has been covered. Here, we discuss the case of the clamped spline. To limit the CNUBS's derivatives, the following properties [18] are utilized.

1) The base function of B-spline is always non-negative.
2) For a kth order CNUBS $S_k(s)$ in Eq. 3 , its derivative is still a CNUBS as

$$\frac{dS_k(s)}{ds} = \sum_{i=0}^{M-2} c_i^{k-1} N_{i+1,k-1}(s) \quad (16)$$

where c_i^{k-1} are the new control points of the reduced order B-spline and it is given as

$$c_i^{k-1} = \frac{k}{s_{i+k+1} - s_{i+1}} (c_{i+1}^k - c_i^k) \quad (17)$$

The reduced order CNUBS shall be evaluated on a modified knot vector which is obtained by removing the first and last items from the original knot vector.
3) The CNUBS passes through its first and last control points.

$$S_k(0) = c_0^k S_k(s_m) = c_M^k \quad (18)$$

For a CNUBS S_k with

$$c_0^k = c_1^k = c_2^k = \cdots = c_{M_\mathrm{e}-1}^k = c_\mathrm{C}^k, \quad (19)$$

according to Eq. 17 there is

$$c_0^{k-1} = c_1^{k-1} = c_2^{k-1} = \cdots = c_{M-2}^{k-1} = 0, \quad (20)$$

which with Eq. 16 gives

$$\frac{dS_k(s)}{ds} = 0. \quad (21)$$

Therefore the value of $S_k(s)$ is a constant:

$$\begin{aligned} &S_k(s) = \sum_{i=0}^{M-1} c_i^k N_{i,k}(s) = c_\mathrm{C}^k \\ &\text{if} \quad c_0^k = c_1^k = c_2^k = \cdots = c_{M-1}^k = c_\mathrm{C}^k \end{aligned} \quad (22)$$

Combining Eq. 7 16, the nth time derivative of $S_k(s)$ can be expressed as:

$$\frac{d^n S_k(s)}{dt^n} = \alpha^n \sum_{i=0}^{M-n-1} c_i^{k-n} N_{i+n,k-n}(s) \quad (23)$$

Since the base functions of the B-spline are always non-negative, it gives:

$$\begin{aligned} &\tfrac{d^n S_k(s)}{dt^n} = \alpha^n \sum_{i=0}^{M-n-1} c_i^{k-n} N_{i+n,k-n}(s) \\ &\geq \alpha^n \sum_{i=0}^{M-n-1} c_\mathrm{C} N_{i+n,k-n}(s) \\ &\text{if} \quad c_i^{k-n} \geq c_\mathrm{C}, \quad \forall i \in [0, M-n-1] \end{aligned} \quad (24)$$

Thus, the sufficient condition to limit the nth derivative of CNUBS $S_k(s)$ between $C_{\min}$ and $C_{\max}$ is

$$C_{\min} \leq \alpha^n c_i^{k-n} \leq C_{\max}, \quad \forall i \in [0, M-n-1] \quad (25)$$

And from Eq. 17, the control points of the derivative spline is a linear combination of the original control point. Substitute Eq. 6 into Eq. 17 and set $k = 4$ for the quartic CNUBS, we can get

$$C_\mathrm{s}^3 = K_\mathrm{s} C_\mathrm{s}^4 \quad (26)$$

where $C_{\mathrm{s}}^3 = [c_0^3, c_1^3, \ldots c_{M-2}^3]$ is the control point vector for $\frac{dS_4(s)}{ds}$ and K_{s} is obtained through Eq. 17 as:

$$K_{\mathrm{s}} = \alpha \begin{bmatrix} -4 & 4 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & -2 & 2 & 0 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 0 & -\frac{4}{3} & \frac{4}{3} & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & \ldots & 0 \\ \vdots & & & & \ddots & & & & \vdots \\ 0 & 0 & \ldots & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \ldots & -\frac{4}{3} & \frac{4}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & -2 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & -4 & 4 \end{bmatrix}.$$

Similarly there is

$$C_{\mathrm{s}}^2 = Q_{\mathrm{s}} C_{\mathrm{s}}^3 \tag{27}$$

where $C_{\mathrm{s}}^2 = [c_0^2, c_1^2, \ldots c_{M-2}^2]$ is the control point vector for $\frac{d^2S_4(s)}{ds^2}$. It is simple to verify that:

$$C_{\mathrm{s}}^2 = Q_{\mathrm{s}} K_{\mathrm{s}} C_{\mathrm{s}}^4 \tag{28}$$

Following the same logic, we can find the control points of the nth order derivative splines as

$$C_{\mathrm{s}}^{k-n} = T_{\mathrm{n}} C_{\mathrm{s}}^k \tag{29}$$

Combine it with the sufficient condition in Eq. 25, the constraints on the n-th order derivative of the trajectory can be written as the following linear form

$$[c_{\min\ \times(M-n-1)}^{k-n}] \leq T_{\mathrm{n}} C_{\mathrm{s}}^k \leq [c_{\max\ \times(M-n-1)}^{k-n}] \tag{30}$$

where $c_{\min}^{k-n}$ and $c_{\max}^{k-n}$ are the minimum and maximum of the corresponding derivative. Further, according to Eq. 18, the boundary conditions can be satisfied by introducing equality constraints on the first and last element of C_{s}^{k-n} as

$$\begin{array}{rcl} C_{\mathrm{s}}^{k-n}[0] & = & B_{ini}^n \\ C_{\mathrm{s}}^{k-n}[M-n-1] & = & B_{\mathrm{end}}^n \end{array} \tag{31}$$

where $B_{\mathrm{ini}}^n, B_{\mathrm{end}}^n$ denotes the boundary conditions of the n-th derivative. Combining Eq. 15, 30 and 31 a quadratic optimization problem with linear constraints is formulated. Efficient off-the-shelf solvers such as *quadprog* [19] and *CPLEX* [20] can be used. By interpolating the data from each dimension $\{D_{\mathrm{s}x}, D_{\mathrm{s}y}, D_{\mathrm{s}z}\}$, 3D trajectory could be generated efficiently with this approach.

3.2.2 Trajectory Design

As mentioned in the previous section, the time vector is not always given such as during the design of camera trajectory. The time vector D_{time} needs to be optimized individually. Let $l_{\mathrm{i}} = t_{\mathrm{i+1}} - t_{\mathrm{i}}, \forall i \in [0, I-1]$ and define the time difference vector

$$L = [l_0, l_1, \ldots, l_{n-1}]. \tag{32}$$

For a given L, it is easy to compute the minimum value of J_{s} in Eq. 15 with constraints from Eq. 30 and 31 by solving the qudratic programming problem. Therefore, we define a function to map from a given L to the minimum cost J_{s} as

$$J_{\mathrm{s}} = f(L) \tag{33}$$

And for the trajectory designing problem, the task is to

$$\min_L f(L) \tag{34}$$

Since the optimization problem is no longer convex, a gradient descent backtrack line search is performed. To initialize the gradient descent procedure, a reasonable guess needs to be provided for desired convergence. Note l_i can be interpreted as the time used for travelling between data point i to $i+1$. A reasonable guess for a holonomic vehicle is to calculate the time consumption for a point-to-point maneuver. That is, assume the vehicle fly towards and stop at each data point, then calculate the duration t_{d}^i of flying from data point i to $i+1$ and use it to initialize the vector L.

A double integrator model with velocity and acceleration constraints is adopted to given a guess on $t_{\mathrm{d}}^{\mathrm{i}}$. Since the model enables fast calculation and reasonably accurate estimation according to our experience. In this paper, the time-optimal maneuver is chosen as its provide an analytical solution to the consumed time. For each axis, the problem of estimation the time consumption has been reduced to

$$\begin{aligned} & \min_{a(t), t\in[0,T]} \left\{ J = \int_{t=0}^{T} dt \right\} \ s.t. \\ & p(0) = p_0, \quad p(T) = p_{\mathrm{ref}} \\ & v(0) = v_0, \quad v(T) = 0 \\ & \dot{p}(t) = v(t) \\ & \dot{v}(t) = a(t) \\ & -v_{\max} \leq v(t) \leq v_{\max}, \quad \forall t \in [0, T] \\ & -a_{\max} \leq a(t) \leq a_{\max}, \quad \forall t \in [0, T] \end{aligned} \tag{35}$$

Here, $p_0 \in \mathbb{R}$, $v_0 \in \mathbb{R}$ are the initial position and velocity respectively, $v_{\max} \in \mathbb{R}$, , $a_{\max} \in \mathbb{R}$ are the maximum velocity and acceleration and $p_{\mathrm{ref}} \in \mathbb{R}$ is the target position.

According to , the system input $a(t)$ can only be $\{-a_{\max}, a_{\max}, 0\}$ which is called a bang-zero-bang control. With given $v_{\max}$ and $a_{\max}$, the analytical solution to the time optimal trajectory can be found in [21]. The pseudo code of the solution is given as follows in Algorithm 1.

The output t_{op} denotes the optimal time for the corresponding axis. For synchronizing the axes in the 3D space, we adopt the method in [22]. According to [22], the overall optimal time in the 3D space, t_{op}^*, is the maximum of each axis's t_{op}. Now, we can use t_{op}^* to initialize the time difference vector L and to start the gradient descent search on $\min\limits_L f(L)$. The gradient of $f(L)$ can be found through small perturbation method:

$$\nabla f = \frac{f(L + hg) - f(L)}{h} \tag{36}$$

where h is a small scalar in the order of 10^{-6} and g is a perturbation vector.

Next, let us define $\mathbf{D_s} = [D_{\mathrm{sx}}, D_{\mathrm{sy}}, D_{\mathrm{sz}}]$ as the data points in the 3D space and $\mathbf{C_s^4} = [C_{\mathrm{sx}}{}^4, C_{\mathrm{sy}}{}^4, C_{\mathrm{sz}}{}^4]$ as the control points of the quartic CNUBS in 3D. The algorithm for performing time vector optimization is given as Algorithm 2. The function $\mathrm{QuadConvexOptimize}()$ solves for the sin-

Algorithm 1 Optimal time solver

```
1: Input: $p_0$, $v_0$ ,$v_{\max}$ ,$a_{\max}$, $p_{\text{ref}}$
2: Output: $t_{\text{op}}$
3: $v_{\text{end}} = p_0 + \frac{v_0|v_0|}{2a_{\max}}$
4: $d_{\text{a}} = \text{sign}(p_{\text{ref}} - v_{\text{end}})$
5: $a_{\text{cruise}} = d_{\text{a}} \cdot v_{\max}$
6: $\Delta t_1 \leftarrow abs(a_{\text{cruise}} - v_0)/a_{\max}$
7: $u_{\text{j}_{\text{acc}}} \leftarrow a_{\max} \cdot \text{sign}(a_{\text{cruise}} - v_0)$
8: $v_1 \leftarrow p_0 + v_0 \cdot \Delta t_1 + 0.5 \cdot u_{\text{j}_{\text{acc}}} \cdot \Delta t_1^2$
9: $\Delta t_3 \leftarrow abs(-a_{\text{cruise}})/a_{\max}$
10: $u_{\text{j}_{\text{dec}}} \leftarrow a_{\max} \cdot \text{sign}(-a_{\text{cruise}})$
11: $\bar{v_3} \leftarrow a_{\text{cruise}} \cdot \Delta t_3 + 0.5 \cdot u_{\text{j}_{\text{dec}}} \cdot \Delta t_3^2$
12: $\bar{v_2} \leftarrow v_{\text{ref}} - v_1 - \bar{v_3}$
13: if $d_{\text{a}} = 0$ then
14:     $\Delta t_2 \leftarrow 0$
15: else
16:     $\Delta t_2 \leftarrow \bar{v_2}/a_{\text{cruise}}$
17: if $\Delta t_2 < 0$ then
18:     $a_{\text{cruise}} \leftarrow d_{\text{a}} \cdot \sqrt{d_{\text{a}} \cdot a_{\max} \cdot (v_{\text{ref}} - p_0) + 0.5 \cdot v_0^2}$
19:     $\Delta t_1 \leftarrow abs(a_{\text{cruise}} - p_0)/a_{\max}$
20:     $\Delta t_2 \leftarrow 0$
21:     $\Delta t_3 \leftarrow abs(-a_{\text{cruise}})/a_{\max}$
22: $t_{\text{op}} \leftarrow \Delta t_1 + \Delta t_2 + \Delta t_3$
```

gle axis optimization problem with given time vector as discussed previously. Its outputs are the minimum cost values (f_x, f_y, f_z) and the corresponding control point vectors ($C_{\text{sx}}^4, C_{\text{sy}}^4, C_{\text{sz}}^4$). Function Perturbe is an implementation of Equation 36 which gives the gradient. The initial guess of L is generated through Algorithm 1 while the time sector optimization further smooths down the trajectory through a typical gradient descent with backtrack line search (Line 4 – 16). An example of 3D trajectory passing through predefined waypoints are shown in Figure 3. The velocity and acceleration reference are both continuous and smooth. Their magnitudes are also limited which fulfills the dynamic feasibility of the quadrotor. When the reference is tracked by a simulated quadrotor, satisfying result is achieved. The position tracking performance can be seen in Figure 3 while the velocity error is shown in Figure 4.

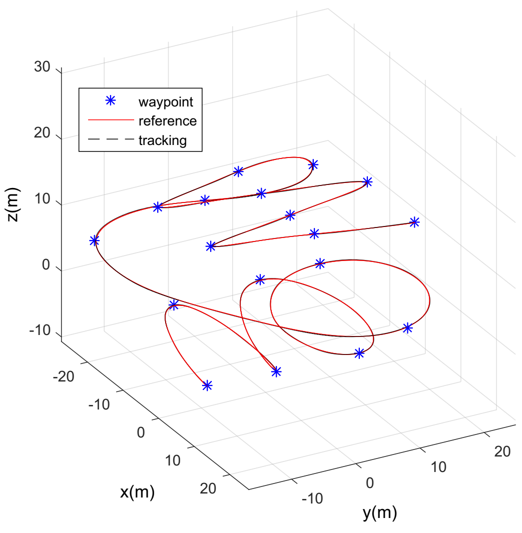

Fig. 3: 3D flight path and tracking performance

Algorithm 2 Optimization for time segmentation

```
1: Input: $L$, $\mathbf{D_s}$,$S_{\text{ini}}$
2: Output: $\mathbf{C_s^4}^*$, $L^*$
3: $(f_s, \mathbf{C_s^4}) = \text{QOptimize}(L, \mathbf{D_s})$
4: while Terminal condition not satisfied do
5:     $grad = \text{Perturbe}(L, \mathbf{C_s^4})$
6:     $StepLength = S_{\text{ini}}$
7:     for $i$ from 1 to $K$ do
8:         $L^{\text{new}} = L + StepLength \cdot grad$
9:         $(f_s^{\text{new}}, \mathbf{C_s^4}^{\text{new}}) = \text{QOptimize}(L^{\text{new}}, \mathbf{D_s})$
10:        if $f_s^{\text{new}} \leq f_s - StepLength \cdot grad \cdot grad'$ then
11:            $f_s = f_s^{\text{new}}$
12:            $L \leftarrow L^{\text{new}}$
13:            $\mathbf{C_s^4} \leftarrow \mathbf{C_s^4}^{\text{new}}$
14:            BREAK
15:        else
16:            $StepLength = StepLength \cdot 0.9$
17: $L^* \leftarrow L$
18: $\mathbf{C_s^4}^* \leftarrow \mathbf{C_s^4}$
19:
20: Function $(f_s, \mathbf{C_s^4}) = \text{QOptimize}(L, \mathbf{D_s})$
21: $(f_x, C_{\text{sx}}^4) = \text{QuadConvexOptimize}(L, D_{\text{s}x}, Constraint_x)$
22: $(f_y, C_{\text{sy}}^4) = \text{QuadConvexOptimize}(L, D_{\text{s}y}, Constraint_y)$
23: $(f_z, C_{\text{sz}}^4) = \text{QuadConvexOptimize}(L, D_{\text{s}z}, Constraint_z)$
24: $f_s = f_x + f_y + f_z + K_{\text{time}} \sum L$
25: $\mathbf{C_s^4} = [C_{\text{sx}}^4, C_{\text{sy}}^4, C_{\text{sz}}^4]$
```

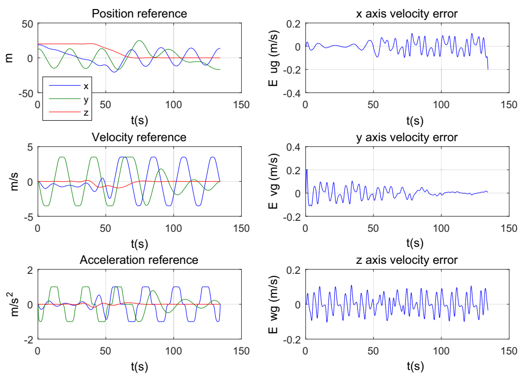

Fig. 4: Reference and velocity tracking error

4 Experiment

Real flight experiment has been conducted with an actual platform in windy environment and achieved satisfying results. The testing platform is a micro quadrotor with a weight of 700g. The trajectory is designed using method in previous section and the trajectory tracking results in shown in Figure 5. Accurate tracking has been achieved. Specially, a comparison between tracking of the optimal B-spline trajectory with the minimum time trajectory is given in Figure 6. While an accurate velocity tracking of the optimal B-spline is achieved, the tracking of minimum time trajectory shows a clear delay. The tracking of two different trajectories is done using the same platform, in the same environment and with the same control law. Our method of trajectory design could both constraint and minimize the derivatives of the position reference. Hence, it can be made to fulfill the dynamic limitations of real vehicles and achieve accurate tracking performance.

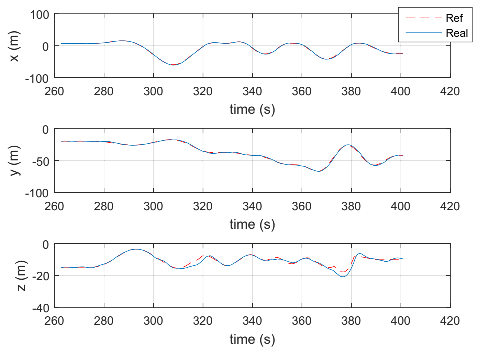

Fig. 5: Position tracking results

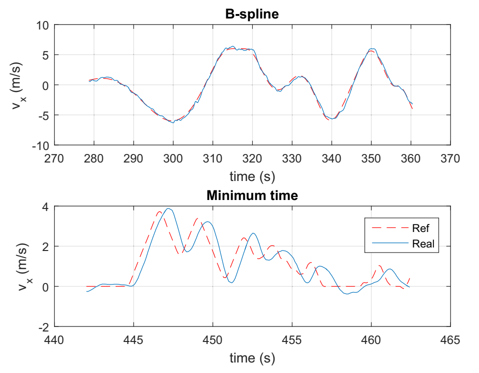

Fig. 6: Tracking of B-spline and time optimal trajectory

5 Conclusion

In this paper, we presented a method for generating dynamically feasible trajectory for quadrotor unmanned vehicles with state constraints. A clamped B-spline is chosen as the base function to formulate a numerical optimization problem. Pure positional input to the algorithm is made possible through the time vector estimation and optimization. The resulted trajectory is shown to be suitable for quadrotor through real experiment. The method can be further improved by introducing obstacle avoidance and online optimization.

References

[1] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," tech. rep., The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, United States, 1992.

[2] S. Park, J. Deyst, and J. How, "A new nonlinear guidance logic for trajectory tracking," in Proceedings of the *AIAA Guidance, Navigation, and Control Conference and Exhibit*, Rhode Island, United States, Aug 2004.

[3] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 519–529, Jun 2007.

[4] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in Proceedings of the *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, pp. 2520–2525, May 2011.

[5] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient algorithm for state-to-state quadrocopter trajectory generation and feasibility verification," in Proceedings of the *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, pp. 3480–3486, Nov 2013.

[6] M. Hehn and R. D'Andrea, "Real-Time Trajectory Generation for Quadrocopters," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 877–892, Aug 2015.

[7] S. K. Phang, S. Lai, F. Wang, M. Lan, and B. M. Chen, "Systems design and implementation with jerk-optimized trajectory generation for UAV calligraphy," *Mechatronics*, vol. 30, pp. 65–75, September 2015.

[8] O. von Stryk and R. Bulirsch, "Direct and indirect methods for trajectory optimization,"*Annals of Operations Research*, vol. 37, pp. 357–373, 1992.

[9] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in Proceedings of the *2016 IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, pp. 1476–1483, May 2016.

[10] G. M. Hoffmann, S. L. Waslander and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in Proceedings of the *AIAA Guidance, Navigation and Control Conference and Exhibit*, Hawaii, United States, Aug 2008.

[11] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, "A prototype of an autonomous controller for a quadrotor UAV," in Proceedings of the *2007 European Control Conference*, Kos, Greek, pp. 4001–4008, July 2007.

[12] Y. Bouktir, M. Haddad, and T. Chettibi, "Trajectory planning for a quadrotor helicopter," in Proceedings of the *16th Mediterranean Conference on Control and Automation*, Ajaccio, France, pp. 1258–1263, June 2008.

[13] M. W. Mueller, M. Hehn, and R. D'Andrea, "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation," *IEEE Transactions on Robotics*, vol. 31, pp. 1294–1310, Dec 2015.

[14] H. Kano, H. Nakata and C. F. Martin, "Optimal curve fitting and smoothing using normalized uniform B-splines: a tool for studying complex systems", *Applied Mathematics and Computation*, vol. 159, no. 1, pp. 96-128, 2005.

[15] H. Kano, H. Fujioka and C. F. Martin, "Optimal smoothing and interpolating splines with constraints", *Applied Mathematics and Computation*, vol. 218, no. 5, pp. 1831-1844, 2011

[16] H. Fujioka and H. Kano, " Control theoretic B-spline smoothing with constraints on derivatives", *IEEE 52nd Annual Conference on Decision and Control*, Firenze, Italy, 2013.

[17] J. Chen, K. Su, and S. Shen, "Real-time safe trajectory generation for quadrotor flight in cluttered environments," in Proceedings of the *2015 IEEE International Conference on Robotics and Biomimetics*, Zhuhai, China, pp. 1678–1685, Dec 2015.

[18] C. K. Shene, Lecture notes on *Introduction to computing with Geometry*, Michigan Technological University, 2011.

[19] "Matlab Mathworks, Inc." http://www.mathworks.com/

[20] "IBM ILOG CPLEX Optimizer IBM, Inc." https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[21] R. Haschke, E. Weitnauer, H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nice, France, pp. 3248 – 3253, 2008.

[22] T. Kroger and F. M. Wahl, "Online trajectory generation: Basic concepts for instantaneous reactions to unforeseen events," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 94–111, Feb 2010.