Control Theory Tech, Vol. 17, No. 1, pp. 48–61, February 2019

Control Theory and Technology

http://link.springer.com/journal/11768

Axis-coupled trajectory generation for chains of integrators through smoothing splines

Shupeng LAI [1†], Menglu LAN [2], Kehong GONG [1], Ben M. CHEN [3,1]

1.*Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583;*

2.*Graduate School for Integrative Science & Engineering, National University of Singapore, Singapore 119077;*

3.*Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China*

Received 18 September 2018; revised 26 October 2018; accepted 29 October 2018

Abstract

Integrator based model is used to describe a wide range of systems in robotics. In this paper, we present an axis-coupled trajectory generation algorithm for chains of integrators with an arbitrary order. Special notice has been given to problems with pre-existing nominal plans, which are common in robotic applications. It also handles various type of constraints that can be satisfied on an entire time interval, including non-convex ones which can be transformed into a series of convex constraints through time segmentation. The proposed approach results in a linearly constrained quadratic programming problem, which can be solved effectively with off-the-shelf solvers. A closed-form solution is achievable with only the boundary constraints considered. Finally, the proposed method is tested in real experiments using quadrotors which represent high-order integrator systems.

Keywords: B-spline, trajectory generation, chains of integrators

DOI https://doi.org/10.1007/s11768-019-8201-y

1 Introduction

Chains of integrators are used in approaching and analyzing the dynamics of many systems in robotic applications, such as the electrical motor, robotic arms, and computer numerical control (CNC) machines. Especially, many of the holonomic vehicles, including the recently popular multi-copters, can be effectively simplified into chains of integrator because of their differential flat properties [1]. In many of these applications, it is desirable to generate a trajectory that satisfies certain constraints while minimizing the deviation from a pre-existing nominal plan [22].

. **Boundary conditions:** The trajectory is required to start and end in a specific state. For example, many methods rely on trajectory re-planning to handle the system and environment uncertainties, the initial state

†Corresponding author.
E-mail: elelais@nus.edu.sg.

in such a problem is usually the current or near future state of the system.

. **Input constraints:** Most robotic system has limited input, such as limited torque, current, etc. If a trajectory is to be tracked precisely and safely, the limits on inputs need to be satisfied.

. **Safety constraints:** The trajectory has to stay in certain safe regions. Moreover, in many cases, this leads to non-convex constraints due to the nature of the system or the environment.

. **Derivative constraints:** Many systems come with state limits that can be expressed as the derivative of the trajectory, such as the maximum angular speed of a motor, the limited acceleration of a multicopter, or the jerk constraint in a CNC machine.

Generally, the last two constraints can also be summarized as the state constraints. In robotic applications, due to the limited computational power and the requirement of real-time capability, they are usually dealt with vastly different methods.

Many systems adopt a multi-layer approach in motion planning to reduce the volume of the search space. Global planning is usually conducted with a vehicle model with highly reduced dynamics, and results in a nominal plan consisting of simple geometric primitives such as points and lines. A trajectory generation algorithm shall be able to minimize the derivative to the nominal plans while satisfying all the above mentioned constraints. Finally, a smooth reference trajectory is usually desirable. It implies a minimized energy consumption and lesser wear and tear, which are beneficial in general.

There is a rich literature on trajectory generation. A common approach is first to generate multiple keyframes which are then interpolated through polynomials. A recent example can be seen in [1], where the authors formulate a (QP) problem to interpolate a series of waypoints using polynomial splines. It can be considered as a multiple shooting method due to the usage of equality constraints to connect different polynomial segments. For chains of integrators, the underlying ordinary differential equations (ODEs) in a shooting method can be solved analytically. Therefore, it is possible to reformulate the problem with different variables and eliminate the unnecessary equality constraints [2]. However, due to the base functions used in [1] and [2], they only support point-wise constraints. One needs to enforce the constraints at multiple discrete samples [3] or solve the problems various times [4] to ensure the constraints are well satisfied on the entire trajectory. To address the issue, Bézier curve is adopted in [5] and [6] to enforce the constraints in a continuous time interval. However, it is not applied to the trajectory's derivatives, and the equality constraints for segment connection cannot be omitted.

On the other hand, the authors of [10] presents a method based on the pontryagin's maximum principle to generate a jerk limited and time-optimal trajectory in the form of cubic splines. The method utilizes a decision tree to differentiate between multiple shapes of acceleration profiles where each of them has an analytic solution. Therefore, it is highly efficient, and a trajectory could be generated in micro-seconds level. The author of [7] later extended it for multi-axis synchronization and applied to robotic arms [8]. In [9], the authors adopt a binary searching based approach to simplify the solution of the original problem and extend it to support higher order systems. But the time consumption grows exponentially with the system order and the time optimally is lost. The input reference generated by the methods mentioned above [7, 10] always switches between its maximum magnitude and zero. While it is less obvious for electric motors, it could trigger unnecessary aggressive maneuvers for other systems such as the multicopters. Moreover, these methods can only be used to solve two-point boundary value problems instead of a general interpolation. In [21], this method has been successfully implemented on a vertical-take-off-and-landing vehicle.

Furthermore, B-splines have also been used for generating trajectories for chains of integrators. By using the control points as variables, it guarantees the continuity of the trajectory implicitly, and the equality constraints for segment connection can be safely ignored. The formulation presented in [11] and [12] explores the non-negativity and partition-of-unity properties of the base function to satisfy linear boundaries over a time interval. However, the proposed piece-wise linear boundaries decomposition is only applied to 2-dimensional cases, and it is not sufficient to constrain the trajectory inside arbitrary non-convex regions. Moreover, it fits only point-wise or time-dependent nominal plans but not other geometric primitives.

In this paper, we propose a method based on the B-spline to solve the problems mentioned above with efficient QP formulation. Its major advantages include:

. Addressing the issues of axis-coupling and interval-wise effectiveness through the convex hull property of

the B-spline.

. Converting non-convex constraints into convex ones through segmentation.

. Resulting in a QP with adjustable problem size, which offers a trade-off between solution quality and computing time.

Throughout the paper, $\mathcal{X} \in \mathbb{R}^{d\times 1}$ denotes the d dimensional workspace, where $d \in \mathbb{N}$. For vectors or matrixes, the bracketed subscript denotes specific elements. For a vector $a \in \mathbb{R}^{m\times 1}$, then $a_{(i)}$ represents its ith element. Similarly, for a matrix $A \in \mathbb{R}^{m\times n}$, $A_{(i,j)}$ denotes its element in row i and column j, and $A_{(i,*)}$ represents the entire ith row, and $A_{(*,j)}$ represents the entire jth column. For the axis-coupled formulation, the Kronecker product is marked by the $\otimes$ operator, and I_d denotes the identity matrix $\mathbb{R}^{d\times d}$. Finally, for a matrix A, its vectorization $\hat{A}$ is defined as $\hat{A} = [A_{(1,*)}, A_{(2,*)}, \ldots, A_{(m,*)}]^{\mathrm{T}}$ where T denotes the transpose.

The rest of the paper is organized as follows. The introduction to the B-spline formulation is covered in Section 2; whereas, in Section 3, we show how to formulate the desired problems with quadratic cost functions and linear constraints. In Section 4, the resulting QP is presented, and its solution is discussed, including a closed-form solution. To demonstrate the proposed method, real-experiments are performed using quadrotors, the results and analysis are provided in Section 5. Finally, a conclusion is drawn in Section 6.

2 Background material on B-spline approximation

B-spline, or the basis spline is widely used in statistics for curve fitting and data differentiation. A kth order B-spline S_k can be expressed as

$$S_k(s) = \sum_{i=0}^{M-1} c_i N_i^k(s), \quad c_i, S \in \mathcal{X}, \quad N_i^k \in \mathbb{R}, \tag{1}$$

where N_i^k is the basis function and $c_i \in \mathbb{R}^{d\times 1}$ is called the control point. The basis functions are defined over an knot vector $\mathcal{K} = [s_0, s_1, \ldots, s_{M+k}]$ and a path parameter s:

$$\begin{cases} N_i^0(s) = \begin{cases} 1, & \text{if } s_i \leqslant s < s_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \\ N_i^j(s) = N_{\mathrm{A}}(s,i,j) N_i^{j-1}(s) + N_{\mathrm{B}}(s,i,j) N_{i+1}^{j-1}(s), \end{cases} \tag{2}$$

where

$$\begin{aligned} N_{\mathrm{A}}(s,i,j) &= \begin{cases} 0, & \text{if } s_i = s_{i+j}, \\ \dfrac{s - s_i}{s_{i+j} - s_i}, & \text{otherwise}. \end{cases} \\ N_{\mathrm{B}}(s,i,j) &= \begin{cases} 0, & \text{if } s_{i+1} = s_{i+j+1}, \\ \dfrac{s_{i+j+1} - s}{s_{i+j+1} - s_{i+1}}, & \text{otherwise}. \end{cases} \end{aligned} \tag{3}$$

In order to achieve an efficient QP problem, the cardinal B-spline is chosen where the knots have equal distances in between [13]. Since most robotic application have boundary conditions to be satisfied, the first and last knots are repeated k times, thus an closed-form solution for boundary value problems can be achieved. Therefore, the knots can be written as

$$\begin{aligned} \mathcal{K} &= [s_0, s_1, \ldots, s_{M+k}] \\ &= [\underbrace{0, \ldots, 0}_{k+1 \text{ times}}, 1, 2, \ldots, \underbrace{M-k, \ldots, M-k}_{k+1 \text{ times}}]. \end{aligned} \tag{4}$$

A linear mapping from path parameter s to time t is introduced:

$$\frac{s}{t} = \alpha. \tag{5}$$

It gives $S_k(s) = S_k(\alpha t)$ where $t \in [0, T_{\mathrm{end}}]$ and $T_{\mathrm{end}} = \dfrac{M-k}{\alpha}$. The basis of a 4th order cardinal B-spline can be seen in Fig. 1.

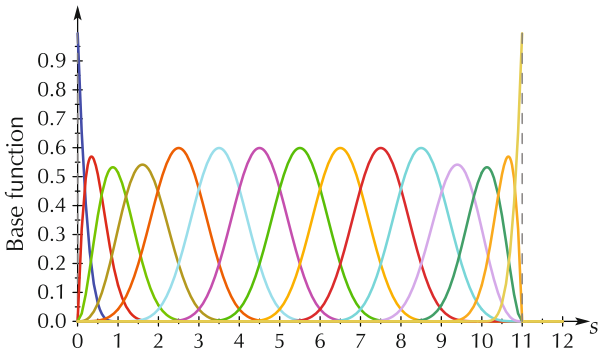


Fig. 1 Base function of a cardinal B-spline.

For the ease of analysis, we also define

$$C = [c_0, c_1, \ldots, c_{M-1}], \quad C \in \mathbb{R}^{d\times M} \tag{6}$$

as the control point matrix.

Due to the selected basis function, the B-spline has the following two important properties:

. Non-negativity: $N_i^k(s) \geqslant 0$ for all $i \in \{0, 1, \ldots, M-1\}$ and $s \in [s_0, s_{M+k}]$.

. Partition-of-unity: $\sum_{i=0}^{M-1} N_i^k(s) = 1$ for all $s \in [s_0, s_{M+k}]$.

The two property together define the convex-hull property and will be used through out the paper.

3 Axis-coupled smooth trajectory

In this section, we formulate trajectory generation as an optimization problem. The focus is on achieving an efficient QP to fit various geometric primitives, and transforming non-convex constraints into convex ones. With such a simplification, the proposed formulation can be solved with off-the-shelf QP solvers and satisfy the real-time requirement in robotic applications.

3.1 Method overview

The trajectory generation problem can be briefly considered as

$$\begin{aligned} \min\ & E + \omega\epsilon \\ \text{s.t.}\ & \text{boundary conditions,} \\ & \text{safety constraints,} \\ & \text{input constraints, } \textit{and} \\ & \text{derivative constraints,} \end{aligned} \tag{7}$$

where E stands for the integration of the square of the trajectory's derivatives; and by minimizing E the trajectory becomes more smooth. The term ϵ represents the deviation from the desired nominal plan. Previously, the description of nominal plan in such a fitting problem is limited to points or keyframes. In this paper, it is extended to cover general geometrical flats in the high-dimensional space. Both E and ϵ can be expressed as quadratic terms related to the control points. Moreover, ω is a weighting factor and all constraints are linear. Among them, the boundary conditions are a set of equality constraints, and the others are sets of inequality constraints. The rest of this section shows how the cost function and the constraints are constructed.

3.2 Smoothness cost

By minimizing the integration of the square of the trajectory's derivatives, abrupt changes on the trajectory is penalized, and a smoother trajectory is expected. Here the cost function E can be expressed as

$$E = \sum_{n=1}^{k} \sigma_{(n)} \int_{-\infty}^{\infty} \left\| \frac{\mathrm{d}^n S_k}{\mathrm{d}t^n} \right\|^2 \mathrm{d}t$$

$$= \sum_{n=1}^{k} \sum_{i=1}^{d} \sigma_{(n)} \int_{-\infty}^{\infty} \left(\frac{\mathrm{d}^n S_{k(i)}}{\mathrm{d}t^n} \right)^2 \mathrm{d}t, \tag{8}$$

where $\sigma_{(n)} \geqslant 0$, $\forall n \in \{1, \ldots, k\}$ are the weighting factors. By introducing $\hat{C}$, which is the vecorization of the control point matrix C, it can be expressed in quadratic form [11] as

$$E = \hat{C}^{\mathrm{T}} [\sum_{n=1}^{k} (\sigma_{(n)} V_n) \otimes I_d] \hat{C}, \tag{9}$$

where

$$V_{n(i,j)} = \alpha^{2n-1} \int_{-\infty}^{\infty} \frac{\mathrm{d}^n N_i^k(s)}{\mathrm{d}s^n} \frac{\mathrm{d}^n N_j^k(s)}{\mathrm{d}s^n} \mathrm{d}s. \tag{10}$$

From equation (8), it is guaranteed $E \geqslant 0$, therefore $\sigma_{(n)} V_n \otimes I_d$ is positive semi-definite.

3.3 Deviation cost

We show how to formulate quadratic cost functions to penalize the deviation from a given nominal plan which includes points, lines, and planes. The presented method can be applied to arbitrary geometric flats in high dimension.

3.3.1 Points

Point and keyframe based nominal plane are widely used in robotics. For a problem in a d dimensional space, a higher level planner would generate a list of N_{W} desired position-points, each of which is in $\mathbb{R}^d$ and to be reached at a specific time-point. Let the ith position-point and time-point be denoted by $p_i \in \mathbb{R}^d$ and $t_i \in \mathbb{R}$ with $i \in \{0, 1, \ldots, N_{\mathrm{W}} - 1\}$. The cost function shall penalize the distance from the trajectory to the position-points at the corresponding time-points. Let $D = [p_0, p_1, \ldots, p_{N_{\mathrm{W}}-1}]^{\mathrm{T}}$ and $T = [t_0, t_1, \ldots, t_{N_{\mathrm{W}}-1}]^{\mathrm{T}}$, $t_i \in \mathbb{R}$, the cost function can be written as

$$\begin{aligned} \epsilon_{\text{point}} &= \sum_{i=0}^{N_{\mathrm{W}}-1} \| S_k(\alpha t_i) - p_i \|^2 \\ &= \sum_{i=0}^{N_{\mathrm{W}}-1} \sum_{j=1}^{d} (S_{k(j)}(\alpha t_i) - p_{i_{(j)}})^2 \\ &= \sum_{j=1}^{d} (HC_{(j,*)} - D_{(j,*)})^{\mathrm{T}} (HC_{(j,*)} - D_{(j,*)}) \\ &= \hat{C}^{\mathrm{T}} (H^{\mathrm{T}} H \otimes I_d) \hat{C} - 2\hat{D}^{\mathrm{T}} (H \otimes I_d) \hat{C} + \hat{D}^{\mathrm{T}} \hat{D}, \end{aligned} \tag{11}$$

where $H \in \mathbb{R}^{N \times M}$ is constructed as

$$\begin{aligned} H_{(i+1,*)} = [&N_0^k(\alpha t_i), N_1^k(\alpha t_i), \ldots, N_{M-1}^k(\alpha t_i)], \\ & i \in \{0, 1, \ldots, N_{\mathrm{W}} - 1\}. \end{aligned} \tag{12}$$

Here $H^{\mathrm{T}}H \otimes I_d$ is also positive-semi definite. On the other hand, if the position-points need to be reached exactly, the following equality constraints shall be introduced:

$$S_k(\alpha t_i) = p_i,\ \forall i \in \{0, 1, \ldots, N_{\mathrm{W}} - 1\} \Leftrightarrow H \otimes I_d = \hat{D}. \tag{13}$$

3.3.2 Lines

For holonomic vehicles, the high-level planning algorithm usually produces nominal plans consist of straight lines, which is then sampled into multiple position-points in a trajectory generation problem. Here, we present a method to penalize the trajectory's deviation to straight lines without drawing position-point samples. Assume the high level planner produces N_{L} straight lines. Whereas the jth straight line passes through the position-points $v_j, w_j \in \mathbb{R}^d$, where $j \in \{0, 1, \ldots, N_{\mathrm{L}} - 1\}$.

For a point $p \in \mathbb{R}^d$, the square of its distance to the jth straight line can be written as

$$e_1^2 = p^{\mathrm{T}} Q_{1j} p + R_{1j} p + S_{1j}, \tag{14}$$

where

$$\begin{cases} Q_{1j} = \dfrac{(u_j^{\mathrm{T}} u_j) \otimes I_d - u_j u_j^{\mathrm{T}}}{u_j^{\mathrm{T}} u_j}, \\ R_{1j} = \dfrac{-2u_j^{\mathrm{T}} u_j w_j - w_j^{\mathrm{T}} u_j^{\mathrm{T}} u_j}{u_j^{\mathrm{T}} u_j}, \\ S_{1j} = \dfrac{u_j^{\mathrm{T}} u_j (w_j^{\mathrm{T}} w_j) - w_j^{\mathrm{T}} (u_j u_j^{\mathrm{T}}) w_j}{u_j^{\mathrm{T}} u_j}, \\ u_j = v_j - w_j. \end{cases} \tag{15}$$

Assume c_d is the furthest control point to the straight line, the following shows that the deviation from the trajectory to the straight line will be no further than c_d.

Proof Combining equations (1) and (5), a point p on the B-spline at time t_{s} is

$$p = \sum_{i=0}^{M-1} c_i N_i^k(\alpha t_{\mathrm{s}}). \tag{16}$$

For simplicity, we denote $N_i^k(\alpha t_{\mathrm{s}})$ as N_i and drop the subscripts for Q_{1j}, R_{1j} and S_{1j}. Then, there is

$$\begin{aligned} e_1^2 &= (\textstyle\sum c_i N_i)^{\mathrm{T}} Q (\sum c_i N_i) + R(\sum {c_i}^{\mathrm{T}} N_i) + S \\ &= \sum_i \sum_m N_i N_m {c_i}^{\mathrm{T}} Q c_m + \sum_i N_i R c_i + S, \end{aligned} \tag{17}$$

where Q is positive-semi definite (equation (15)), and there is

$$c_i^{\mathrm{T}} Q c_m \leqslant \frac{1}{2}(c_i^{\mathrm{T}} Q c_i + c_m^{\mathrm{T}} Q c_m). \tag{18}$$

Through the non-negativity and partition-of-unity property of the B-spline, there are

$$\sum_i N_i c_i = \frac{1}{2} \sum_i \sum_m N_i N_m (c_i + c_m) \tag{19}$$

and

$$\sum_i \sum_m N_i N_m = 1. \tag{20}$$

Since c_d is assumed to be the furthest control point from the straight line, there is

$$c_d^{\mathrm{T}} Q c_d + R c_d \geqslant c_i^{\mathrm{T}} Q c_i + R c_i,\ \ \forall i \in \{0, 1, \ldots, M - 1\}. \tag{21}$$

Substitute equations (18)–(21) into (17) produces

$$\begin{aligned} e_1^2 &\leqslant \frac{1}{2} \sum_i \sum_m N_i N_m (c_i^{\mathrm{T}} Q c_i + c_m^{\mathrm{T}} Q c_m + R c_i + R c_m) + S \\ &\leqslant \sum_i \sum_m N_i N_m (c_d^{\mathrm{T}} Q c_d + R c_d) + S \\ &= c_d^{\mathrm{T}} Q c_d + R c_d + S. \end{aligned} \tag{22}$$

□

It is then possible to penalize the trajectory's deviation by minimizing the distance between the control points and the straight line.

3.3.3 Planes

As an example to extend to other higher dimensional flats, we show how to minimize the distance between the trajectory and desired planes. Let the jth plane be defined by a normal vector ζ_j, an anchor point $g_j \in \mathbb{R}^d$, where $j \in \{0, 1, \ldots, N_{\mathrm{P}} - 1\}$. It indicates a plane contains g_j and perpendicular to ζ_j. Similar to the previous case, it is assumed that the jth plane shall be approached on time-interval $[\tau_j, \rho_j)$. Then, the square of the distance from a point p to a plane can be written as

$$e_p^2 = p^{\mathrm{T}} Q_{pj} p + R_{pj} p + S_{pj}, \tag{23}$$

where

$$Q_{pj} = \frac{\zeta_j \zeta_j^{\mathrm{T}}}{\zeta_j^{\mathrm{T}} \zeta_j},\ R_{pj} = \frac{-2g_j^{\mathrm{T}}(\zeta_j \zeta_j^{\mathrm{T}})}{\zeta_j^{\mathrm{T}} \zeta_j},\ S_{pj} = \frac{g_j^{\mathrm{T}}(\zeta_j \zeta_j^{\mathrm{T}}) g_j}{\zeta_j^{\mathrm{T}} \zeta_j}. \tag{24}$$

We could then formulate the cost function to penalize the deviation by minimizing the control points' distance to the plane.

3.4 Time segmentation

It is usually less meaningful to approach all the straight lines or planes simultaneously. Here, we separate them in time by penalizing the deviation to the jth straight line or plane only within a time interval $[\tau_j, \rho_j)$, where $j \in \{0, 1, \ldots, N_{\mathrm{L}} - 1\}$.

From equation (5), the corresponding path variable of a time point τ_j is $\alpha\tau_j$, which falls between the knot vector $[s_{\eta(\tau_j)}, s_{\eta(\tau_j)+1})$ with $\eta(\tau_j) = \lfloor \alpha\tau_j \rfloor + k$. From equation (3), there are at most $k+1$ non-zero basis functions on the knot span $[u_i, u_{i+1})$, specifically: $N^k_{i-k}, N^k_{i-k+1}, \ldots, N^k_i$. Correspondingly, given the time interval $[\tau_j, \rho_j)$, only the basis functions $N^k_i(s)$, $i \in \{\eta(\tau_j) - k, \ldots, \eta(\rho_j)\}$can be non-zero. Therefore, we can ignore the control points paired with zero basis functions.

For the N_{L} straight lines, the cost function to penalize the deviation on their individual time interval can be written as

$$\begin{aligned}\epsilon_{\mathrm{l}} &= \sum_{j=0}^{N_{\mathrm{L}}-1} \sum_{i=\lambda_{\tau_j}}^{\lambda_{\rho_j}} (c_i^{\mathrm{T}} Q_{\mathrm{l}j} c_i + R_{\mathrm{l}j} c_i) \\ &= \hat{C}^{\mathrm{T}} \Big(\sum_{j=0}^{N_{\mathrm{L}}-1} \sum_{i=\lambda_{\tau_j}}^{\lambda_{\rho_j}} \Lambda_i^{\mathrm{T}} Q_{\mathrm{l}j} \Lambda_i \Big) \hat{C} + R_{\mathrm{l}j} \Big(\sum_{j=0}^{N_{\mathrm{L}}-1} \sum_{i=\lambda_{\tau_j}}^{\lambda_{\rho_j}} \Lambda_i \Big) \hat{C}, \quad (25)\end{aligned}$$

where $\lambda_{\tau_j} = \eta(\tau_j) - k$, $\lambda_{\rho_j} = \eta(\rho_j)$, and Λ_i is a mapping matrix between the vectorization of the control point matrix and the ith control point:

$$c_i = \Lambda_i \hat{C}, \quad i \in \{0, 1, \ldots, M-1\}. \quad (26)$$

In Section 3.3.2, it shows $Q_{\mathrm{l}j}$ is positive-semi definite, so does the $\Lambda_i^{\mathrm{T}} Q_{\mathrm{l}j} \Lambda_i$ in equation (25).

Similarly, for the N_{P} planes, the cost function in equation (23) can be written as

$$\epsilon_p = \hat{C}^{\mathrm{T}} \Big(\sum_{j=0}^{N_{\mathrm{P}}-1} \sum_{i=\lambda_{\tau_j}}^{\lambda_{\rho_j}} \Lambda_i^{\mathrm{T}} Q_{p_j} \Lambda_i \Big) \hat{C} + R_{p_j} \Big(\sum_{j=0}^{N_{\mathrm{P}}-1} \sum_{i=\lambda_{\tau_j}}^{\lambda_{\rho_j}} \Lambda_i \Big) \hat{C}, \quad (27)$$

which is also quadratic and convex.

3.5 Deviation from line segments

By combining the time segmentation method with the cost function for the basic geometric flats, it is possible to write quadratic cost to penalize the deviation to more complex geometric primitives. Here, we consider penalizing the deviation to a series of interconnected line segments. Many sampling-based planners would produce such a line-segment based nominal plan. Assume the ith line-segment is defined by two waypoints Wp_i and Wp_{i+1}, an example then can be illustrated as in Fig. 2.

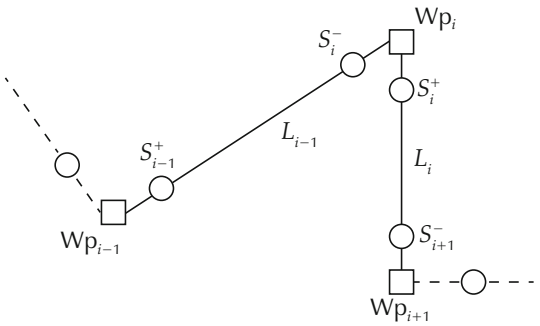


Fig. 2 Problem construction of the proposed method.

Due to the reduced system dynamics during higher-level planning, the waypoints usually are not associated with a time-point. Therefore, we first estimate these time-points using a single or double integrator model, where closed-form solution exists. Assume the time point for the ith waypoint Wp_i is t_i where $i \in \{0, 1, \ldots, N_{\mathrm{W}} - 1\}$. Here we adopt the double integrator estimation model, then $\Delta t_i = t_{i+1} - t_i$ is calculated as the minimum time to move from Wp_i to Wp_{i+1} along the line L_i, with zero boundary velocities, while satisfying the maximum speed $v_{\max i}$ and the maximum acceleration $a_{\max i}$. The time optimal trajectory and Δt_i can be calculated efficiently using the algorithm in [14]. Additionally, two intermediate waypoints S_i^+ and S_{i+1}^- are selected from the time optimal trajectory at moments $t_i + \kappa\Delta t_i$ and $t_i + (1-\kappa)\Delta t_i$ where κ is a design variable. These moments also serve as the time-points for S_i^+ and S_{i+1}^-. Since the time optimal trajectory always stays on the underlying straight line, so does S_i^+ and S_{i+1}^-. Then, to penalize the deviation from the line segments, we minimize:

. The distance to points $\mathrm{Wp}_i, S_i^-, S_i^+$ at their corresponding time-points.

. The deviation from the underlying straight line L_i over $[t_i + \kappa\Delta t_i,\ t_i + (1-\kappa)\Delta t_i)$ for $i \in \{0, 1, \ldots, N_{\mathrm{W}} - 2\}$.

3.6 Safety constraints

Many robotic applications require the trajectory to stay in certain safe regions which are usually non-convex such as in the case of obstacle avoidance. However, non-

convex optimization is generally difficult to solve, which is against requirements of reliability and efficiency in robotic trajectory generation problem. To address this issue, the non-convex safe region is decomposed into multiple interconnected convex regions through time segmentation. It can be illustrated using the following example.

In Fig. 3, the safe region $\mathcal{S}$ is originally non-convex. However, it can be decomposed into multiple convex regions $C = [C_0, C_1, C_2]$. Assume the trajectory is to stay inside $\mathcal{S}$ on the timer interval $[\tau_0, \tau_f)$, we can construct the following sufficient and convex conditions:

$$\begin{cases} S_k \in C_0, & \forall t \in [\tau_0, \tau_a), \\ S_k \in C_1, & \forall t \in [\tau_a, \tau_b), \\ S_k \in C_2, & \forall t \in [\tau_b, \tau_f), \end{cases} \tag{28}$$

which guarantees the trajectory to stay inside $\mathcal{S}$. The example can be extended into higher dimensions with an arbitrary number of decomposed convex regions. Assume the convex regions are all polytopes, where the interior of each can be represented by $\{p \in \mathbb{R}^d | A_j p \leqslant b_j\}$ with $j \in \{0, 1, \ldots, N_C - 1\}$. For the completeness of the paper, we now show that if all control points for a B-spline satisfy

$$Ac_i \leqslant b, \quad i \in \{0, 1, \ldots, M-1\}, \tag{29}$$

so does the entire trajectory which is commonly called the convex hull property of the B-spline.

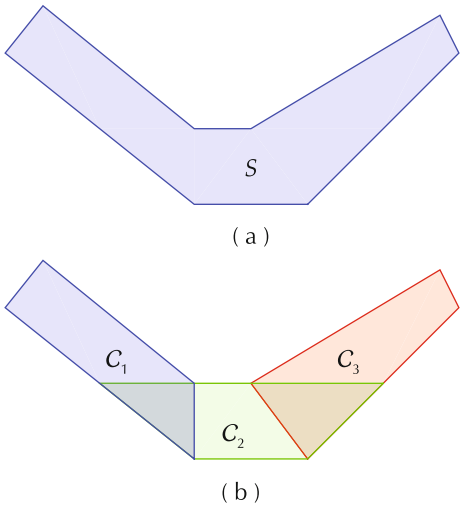


Fig. 3 Dcomposing of the non-convex region. (a) Non-convex region. (b) Convex subregions.

Proof Assume that a point p is select from the trajectory at time t_s. For simplicity, we use N_i to denote $N_i^k(t_s)$ then there is

$$Ap = A \sum_i N_i c_i = \sum_i N_i (Ac_i). \tag{30}$$

By the non-negativity and partition-of-unity of the basis functions, and equation (29),

$$\sum_i N_i (Ac_i) \leqslant \sum_i N_i b = b \Rightarrow Ap \leqslant b. \tag{31}$$

Through the time segmentation, we can assume the trajectory satisfies the jth convex region constraint on the time interval $[\tau_j, \tau_{j+1})$. Following the analysis in Section 3.4, it affects a finite subset of the control points, which can be written as $A_j c_i \leqslant b_j$, $\forall i \in \{\eta(\tau_j) - k, \ldots, \eta(\rho_j)\}$. Combining equation (26), we have the following linear inequality constraints for the safe regions:

$$\begin{aligned} A_j \Lambda_i \hat{C} < b_j, \forall j \in \{0, \ldots, N_C - 1\}, \\ \forall i \in \{\eta(\tau_j) - k, \ldots, \eta(\rho_j)\}. \end{aligned} \tag{32}$$

3.7 Derivative constraints

In many cases, it is also desirable to constrain the trajectory's derivatives. Several previous methods [10, 15, 16] propose to enforce the constraints along each axis, which spans an axis-aligned cuboid. By using the proposed method, it is possible to construct axis-coupled constraints that come with larger interior volume, with which the trajectory could achieve a lower cost. In Fig. 4, the desired acceleration constraint spans a cylinder. By enforcing constraints along each axis, it results in the blue cuboid. On the other hand, the axis-coupled method spans the red octagonal prism which covers more usable volume.

To enforce constraints on the trajectory's derivatives, it is first noticed that the derivative of the B-spline is a reduced-order B-spline. For example, the first derivative of a kth order B-spline $S_k^{(1)}$ is a $(k-1)$th order B-spline:

$$S_k^{(1)} = \frac{\mathrm{d}S^k(s)}{\mathrm{d}t} = \sum_{i=0}^{M-2} c_i^{(1)} N_{i+1}^{k-1}(s) \tag{33}$$

where $c_i^{(1)}$ is given as

$$c_i^{(1)} = \alpha \frac{k(c_{i+1} - c_i)}{s_{i+k+1} - s_{i+1}}. \tag{34}$$

$S_k^{(1)}$ now has only $M-1$ control points, therefore there

shall be $M+k-1$ knots which are given as

$$\begin{aligned}\mathcal{K}^{(1)} &= [s_0^{(1)}, s_1^{(1)}, \ldots, s_{M+k-2}^{(1)}] \\ &= [\underbrace{0, \ldots, 0}_{k \text{ times}}, 1, 2, \ldots, \underbrace{M-k, \ldots, M-k}_{k \text{ times}}].\end{aligned} \tag{35}$$

For simplicity, we write equation (34) in a matrix form

$$\hat{C}^{(1)} = \alpha \Gamma_1 \hat{C}, \tag{36}$$

where $C^{(1)} = [c_0^{(1)}, c_1^{(1)}, \ldots, c_{(1)}^{(M-2)}]$ is the control point matrix of $S_k^{(1)}$. By repeating the above process, the control points of the nth order derivative $S_k^{(n)}$ are

$$\hat{C}^{(n)} = \alpha^n \Gamma_n \hat{C}. \tag{37}$$

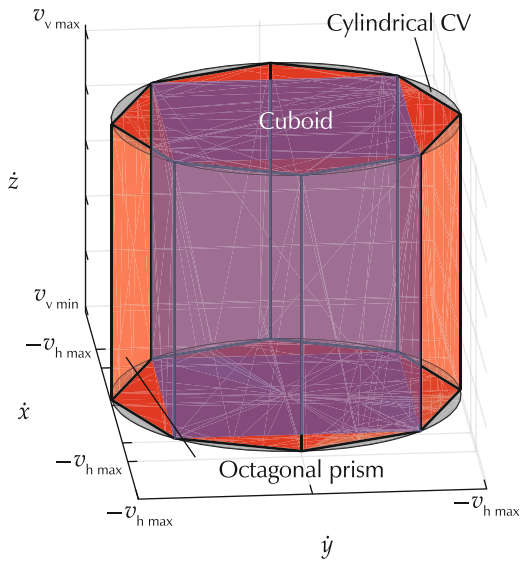


Fig. 4 Constrained volume for acceleration.Axis-decoupled methods select an axis-aligned cuboid (the blue cube). The axis-coupled method selects arbitrarily shaped convex polytopes (the octagonal prism).

Following equation (26), there exists an mapping matrix for the control points of $S_k^{(n)}$ as

$$c_i^{(n)} = \Lambda_i^{(n)} \hat{C}^{(n)}. \tag{38}$$

Using the results in Section 3.6, one could then construct convex safe regions for $S_k^{(n)}$. Assume the convex region is described by $\{p \in \mathbb{R}^d | A_n p \leqslant b_n\}$, the inequality constraints are

$$\begin{aligned}A_n c_i^{(n)} &= A_n \Lambda_i^{(n)} \hat{C}^{(n)} \\ &= \alpha^n A_n \Lambda_i^{(n)} \Gamma_n \hat{C} < b_n, \quad \forall i \in \{0, \ldots, M-n-1\}.\end{aligned} \tag{39}$$

Similarly, for non-convex constraints, the time segmentation method can be utilized.

3.8 Input constraints

For a qth order integrator, if its trajectory is represented by an kth order B-spline S_k, then its input is the qth order derivative of S_k.

$$u = S_k^{(q)}.$$

Therefore, the results in Section 3.7 can be applied to limit the input to the system as equation (39) holds true for arbitrary ordered derivatives.

3.9 Boundary conditions

Let S_{ini}, S_{end} denote the desired initial and final states of the trajectory S_k, respectively. $S_{\text{ini}}^{(n)}$, $S_{\text{end}}^{(n)}$ represent the desired initial and final states of the trajectory's nth derivative $S_k^{(n)}$, respectively. From Eqs. (1) and (3), the initial and end states of S_k are determined by the first and last control points:

$$\begin{cases} S_k(0) = c_0, \\ S_k(\alpha T_{\text{end}}) = c_{M-1}. \end{cases} \tag{40}$$

Since $S_k^{(n)}$ is a reduced-order B-spline, it also satisfies:

$$\begin{cases} S_k^{(n)}(0) = c_0^{(n)}, \\ S_k^{(n)}(\alpha T_{\text{end}}) = c_{M-n-1}^{(n)}. \end{cases} \tag{41}$$

From equations (37), (26) and (38), the boundary conditions can be written as

$$\begin{cases} \Lambda_0 \hat{C} = S_{\text{ini}}, \\ \Lambda_{M-1} \hat{C} = S_{\text{end}}, \\ \alpha^n \Lambda_0^{(n)} \Gamma_n \hat{C} = S_{\text{ini}}^{(n)}, \\ \alpha^n \Lambda_{M-n-1}^{(n)} \Gamma_n \hat{C} = S_{\text{end}}^{(n)}, \end{cases} \tag{42}$$

where $\hat{C}$ is the vectorization of the control point matrix.

4 Quadratic programming problem

So far, we have formulated convex and quadratic cost functions in equations (9), (11), (25) and (27), and linear constraints in equations (32), (39) and (42). Therefore, it is straightforward to construct a QP to describe the trajectory generation problem as

$$\min_{\hat{C}} \ \hat{C}^{\mathrm{T}} G \hat{C} + F \hat{C}$$

$$\text{s.t.} \quad A_{\text{eq}}\hat{C} = b_{\text{eq}}, \quad A_{\text{ieq}}\hat{C} \leqslant b_{\text{ieq}}. \tag{43}$$

The QP problem can be solved from various off-the-shelf solvers such as the MATLAB QUADPROG [19] and the IBM CPLEX [20]. Among the four types of constraints in Section 1, if the problem only has boundary conditions, then our formulation gives a closed form solution. In robotic applications, the other constraints can usually be omitted through various ad-hoc methods, but the boundary constraints are difficult to be replaced. From equation (42), the elements between the $(k+1)$th and $(M-k)$th column of A_{eq} are all zero when only considering the boundary constraint. To separate out the non-zero parts, we define

$$\begin{aligned} A_{\text{F}} = [&A_{\text{eq}_{(*,0)}}, A_{\text{eq}_{(*,1)}}, \ldots, A_{\text{eq}_{(*,k)}}, \\ &A_{\text{eq}_{(*,M-k+1)}}, \ldots, A_{\text{eq}_{(*,M)}}] \end{aligned} \tag{44}$$

and re-arrange the control point matrix C into

$$\begin{cases} C_F = [c_0, c_1, \ldots, c_{k-1}, c_{M-k}, \ldots, c_{M-1}], \\ C_M = [c_k, c_{k+1}, \ldots, c_{M-k-1}]. \end{cases} \tag{45}$$

Here, the "fixed" part C_F is fully determined by the boundary conditions:

$$C_F = A_F^{-1} b_{\text{eq}}. \tag{46}$$

In order to isolate the "fixed" part from the optimization problem, we define a mapping matrix Φ:

$$\hat{C} = \Phi \begin{bmatrix} \hat{C}_F \\ \hat{C}_M \end{bmatrix}. \tag{47}$$

Then, equation (43) can be written as

$$[\hat{C}_F^{\text{T}} \ \hat{C}_M^{\text{T}}]\Phi^{\text{T}} G \Phi \begin{bmatrix} \hat{C}_F \\ \hat{C}_M \end{bmatrix} + F\Phi \begin{bmatrix} \hat{C}_F \\ \hat{C}_M \end{bmatrix}. \tag{48}$$

By defining

$$\begin{cases} G_\Phi = \Phi^{\text{T}} G \Phi, \\ F_\Phi = F\Phi, \end{cases} \tag{49}$$

equation (48) can be written as

$$[\hat{C}_F^{\text{T}} \ \hat{C}_M^{\text{T}}] G_\Phi \begin{bmatrix} \hat{C}_F \\ \hat{C}_M \end{bmatrix} + F_\Phi \begin{bmatrix} \hat{C}_F \\ \hat{C}_M \end{bmatrix}, \tag{50}$$

and equals

$$[\hat{C}_F^{\text{T}} \ \hat{C}_M^{\text{T}}] \underbrace{\begin{bmatrix} G_\Phi^{FF} & G_\Phi^{FM} \\ G_\Phi^{MF} & G_\Phi^{MM} \end{bmatrix}}_{G_\Phi} \begin{bmatrix} \hat{C}_F \\ \hat{C}_M \end{bmatrix} + \underbrace{[F_\Phi^F \ F_\Phi^M]}_{F_\Phi} \begin{bmatrix} \hat{C}_F \\ \hat{C}_M \end{bmatrix}, \tag{51}$$

and can be further simplified as

$$\begin{aligned} &\hat{C}_F^{\text{T}} G_\Phi^{FF} \hat{C}_F + \hat{C}_F^{\text{T}} G_\Phi^{FM} \hat{C}_M + \hat{C}_M^{\text{T}} G_\Phi^{MF} \hat{C}_F \\ &+ \hat{C}_M^{\text{T}} G_\Phi^{MM} \hat{C}_M + F_\Phi^F \hat{C}_F + F_\Phi^M \hat{C}_M. \end{aligned} \tag{52}$$

Finally, by collecting the terms, there is

$$\begin{aligned} &\hat{C}_M^{\text{T}} G_\Phi^{MM} \hat{C}_M + (\hat{C}_F^{\text{T}} G_\Phi^{FM} + F_\Phi^M + \hat{C}_F^{\text{T}} (G_\Phi^{MF})^{\text{T}}) \hat{C}_M \\ &+ \hat{C}_F^{\text{T}} G_\Phi^{FF} \hat{C}_F + F_\Phi^F \hat{C}_F. \end{aligned} \tag{53}$$

Since the value of $\hat{C}_F^{\text{T}} G_\Phi^{FF} \hat{C}_F + F_\Phi^F \hat{C}_F$ is constant due to the pre-determined $\hat{C}_F$ from equation (46), the minimum of equation (53) can then be found at

$$\hat{C}_M = (G_\Phi^{MM})^{-1} (\hat{C}_F^{\text{T}} G_\Phi^{FM} + \hat{C}_F^{\text{T}} (G_\Phi^{MF})^{\text{T}} + F_\Phi^M). \tag{54}$$

5 Flight experiments and analysis

To demonstrate the performance of the proposed approach, real experiments are performed using a small quadrotor as shown in Fig. 5.

Fig. 5 The quadrotor used for experiment.

The quadrotor has been proven as a differential flat system and is commonly approximated as a triple or 4th ordered integrator system during trajectory generation. In this paper, we adopt the 4th order model and also utilize a 4th order B-spline ($k = 4$). The algorithm is implemented in Matlab except when calculating H, where a C application is linked through Mex. Our method is compared to the previous interpolation based trajectory

generation for quadrotors [1], [2], the implementation of which is from the open source project [17].

5.1 Point-type data fitting

In the first example, we demonstrate the algorithm's capability in fitting point-type nominal plans. The nominal plan is generated through direct user input through drawing. Since the sketching inputs come with a large amount of sampled data points, we formulate the trajectory generation as an approximation problem and achieves better results compared to the previous interpolation based methods [1, 2]. The major problem with the interpolation based methods is overfitting. Especially in the case of noisy or crudely estimated time-points. The inaccurate time-point would introduce unnecessary excursions. Through our formulation, the excursions can be reduced by tunning the weighting factors. With the introducing of the deviation cost, the numerical stability of the optimization is also increased. Fig. 6 shows the result of fitting a user sketch with more than 1000 data points. The time-points are estimated heuristically with a single integrator model. The proposed method generates a smooth trajectory that could be tracked precisely by the quadrotor while the interpolation method diverges.

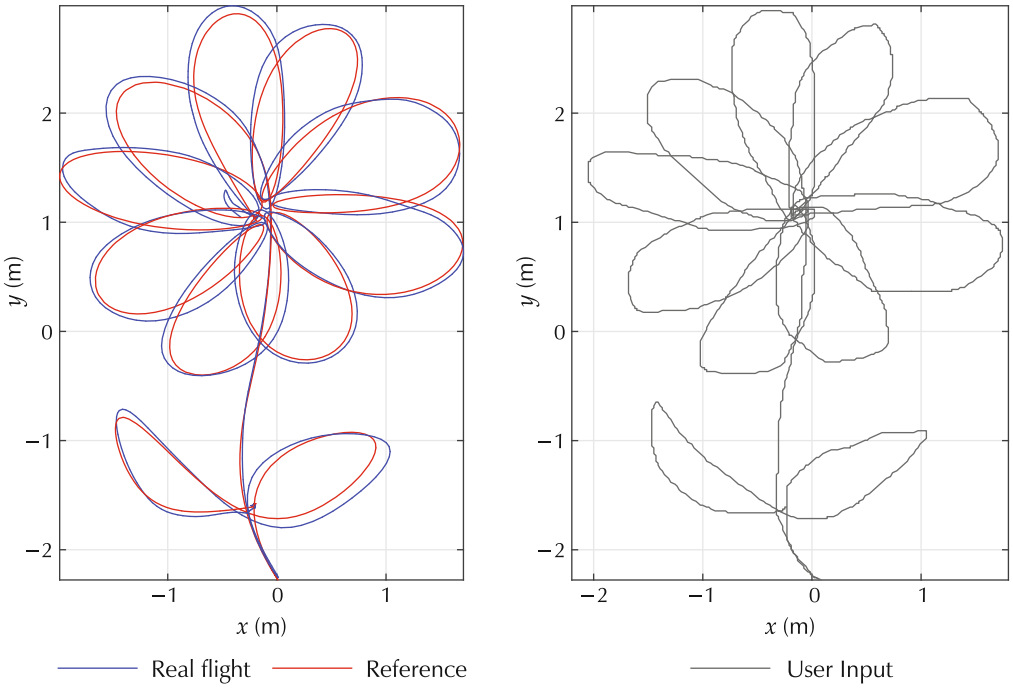


Fig. 6 Densely fitting of user sketching inputs. The blue curve shows the real vehicle tracking performance.

5.2 Nominal plan with line-segments

We also compare the performance at fitting a nominal plan consists of line-segments. As mentioned previously, poorly chosen time-points usually lead to large excursions [18]. A common practice is to insert intermediate position-points along the line-segments [2]. However, it often leads to more aggressive maneuvers of the vehicle. Here, we use the snap cost, which is the integration of the square of the snap along the trajectory, to measure the aggressiveness of the trajectory.

In Fig. 7, it shows a comparison between three methods. Methods 1 and 2 are the interpolation based methods using a 7th order polynomial spline. Method 1 only interpolates the original waypoints while Method 2 also inserts and interpolates intermediate position-points. Finally, Method 3 is the proposed technique for fitting line segments. The experiment shows that Methods 2 and 3 perform similarly at fitting the nominal plan with an average deviation of 0.05 m and 0.02 m, respectively. However, compared to Method 2 which gives a snap cost of 6923, Method 3 produces a much less aggressive trajectory with a snap cost of 348.

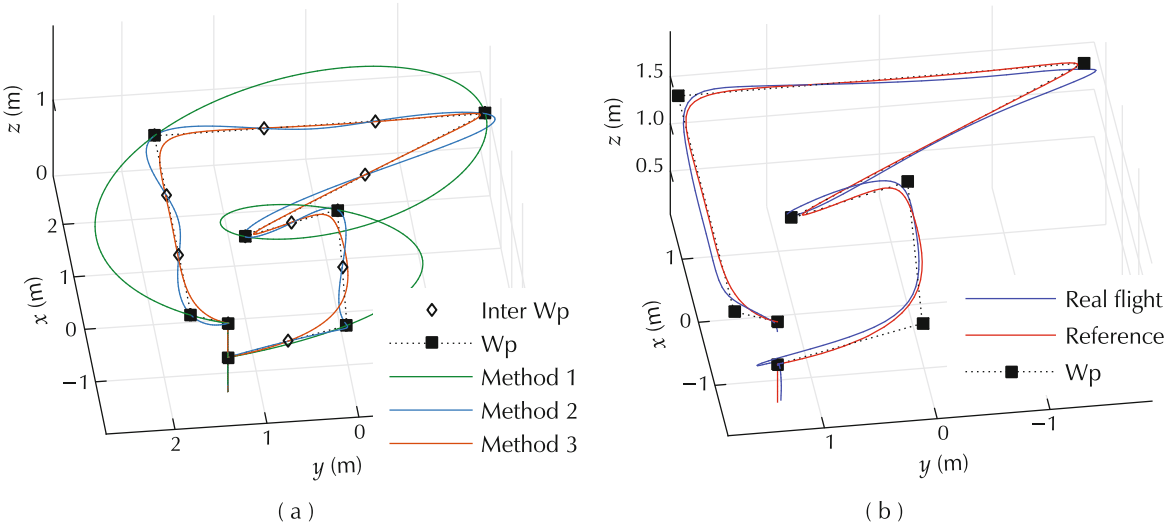


Fig. 7 Comparison for fitting line segments. The time-point allocation for the original waypoints are the same for each method.

5.3 Trajectories on desired planes

In a light paint event with quadrotors, it is desirable to have the trajectory stay close to a plane to prevent the shape from distorted when viewed from different angles. We apply the results in Section 3.3.3 to achieve the desired results. In Fig. 8, four position-points on the same surface defined an M shaped trajectory. However, if the trajectory now considers the velocity and acceleration constraints, it might cause an asynchronous movement and distort the desired shape, especially when viewed from aside. (see Fig. 8 (b)). On the other hand, the proposed approach could penalize the deviation to the desired plane and reduce the distortion.

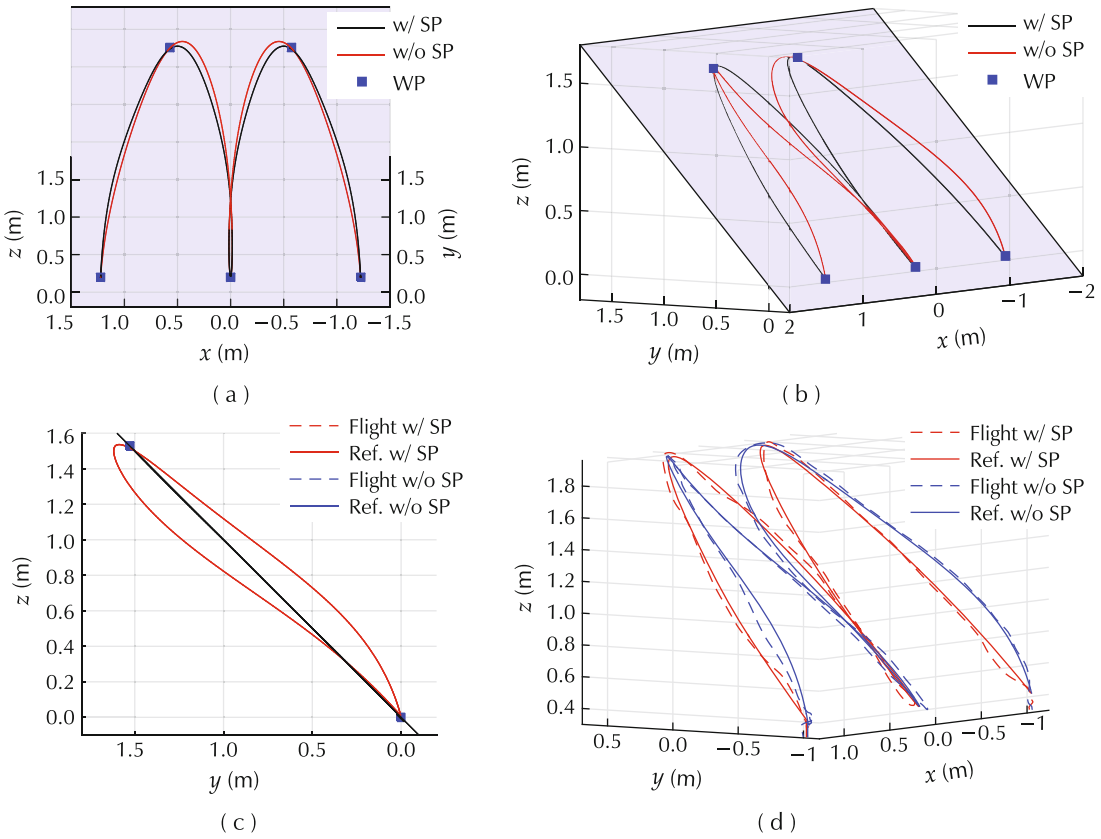


Fig. 8 Comparison between trajectory with surface penalty (w/SP) and without surface penalty (w/o SP). (a) 0^o view. (b) 55^o view. (c) 90^o view. (d) Real flight.

5.4 Safety and feasibility

Quadrotors are usually operated in obstacle strewn environments. The safety and feasibility of its trajectory can be guaranteed using the methods presented in Sections 3.6 and 3.7. In Fig. 9, the safe operation region is constructed using 7 pieces of oriented bounding boxes. Furthermore, the trajectory is also required to satisfy the following conditions:

$$\begin{cases} \sqrt{\dot{x}^2+\dot{y}^2} \leqslant 3.1, \quad -0.55 \leqslant \dot{z} \leqslant 2.2, \\ \sqrt{\ddot{x}^2+\ddot{y}^2} \leqslant 2.8, \quad -0.5 \leqslant \ddot{z} \leqslant 2, \\ \sqrt{\dddot{x}^2+\dddot{y}^2} \leqslant 7.1, \quad -5 \leqslant \dddot{z} \leqslant 5, \end{cases} \tag{55}$$

where x, y, z denotes the trajectory's component on the corresponding axis. These constraints on the velocity, acceleration, and jerk spans three separate cylinders. With the axis-decoupled constraints, the largest axis-aligned cuboid is adopted inside the cylinder. Using the proposed axis-coupled constraints, the largest hexagonal prism inside the cylinder is formed.

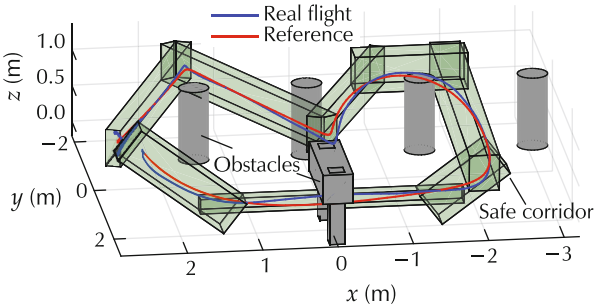
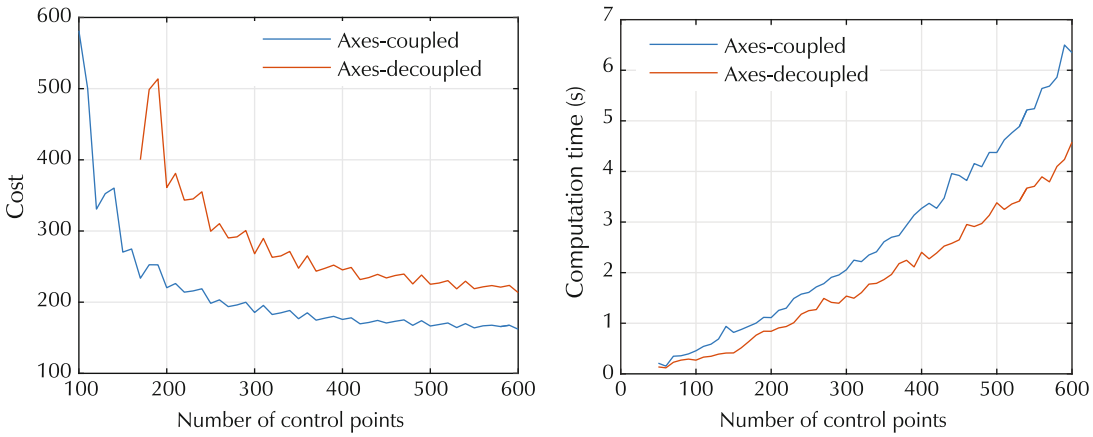


Fig. 9 Safe corridor.

We compare the resulting cost value and computational time against the number of control points used (see Fig. 10). The formulation with axis-coupled constraints generates trajectories with lower costs but slightly longer computational time. However, it also produces a feasible solution with fewer control points.



(a) (b)

Fig. 10 Comparison between axes-coupled and decoupled methods. (a) Cost. (b) Computing time.

6 Conclusions

In this paper, we have presented a method to generate trajectories for chains of integrators using the B-spline technique. It systematically studies the issue of axis-coupling and interval-wise effectiveness. The application includes penalizing the deviation from arbitrary geometric flats. Through the convex hull property, all convex constraints can be satisfied throughout the entire trajectory. We have shown that non-convex constraints and nominal plans can be converted into convex ones through time segmentation. To guarantee the real-time capability and reliability which are required in robotic applications, we have formulated the problem into a QP. A closed-form solution has been derived for solving boundary value problems efficiently. Finally, the overall approach has been successfully tested and verified in real experiments using quadrotors which is commonly considered as high-order integrators.

References

[1] D. Mellinger, V. Kumar. Minimum snap trajectory generation and control for quadrotors. *IEEE International Conference on Robotics and Automation*, Shanghai: IEEE, 2011: 2520 – 2525.

[2] C. Richter, A. Bry, N. Roy. Polynomial trajectory planning for quadrotor flight. *RSS Workshop on Resource-Efficient Integratiton*

of Perception, Control and Navigation for Micro Air Vehicles MAVs, 2013: 1 – 16.

[3] S. Liu, M. Watterson, K. Mohta, et al. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. *IEEE Robotics and Automation Letters*, 2017, 2(3): 1688 – 1695.

[4] J. Chen, T. Liu, S. Shen. Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments. *IEEE International Conference on Robotics and Automation*, Stockholm: IEEE, 2017: 1476 – 1483.

[5] J. S. Tang, V. Kumar. Safe and complete trajectory generation for robot teams with higher-order dynamics. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Deajeon: IEEE, 2016: 1894 – 1901.

[6] J. A. Preiss, W. Honig, N. Ayanian, et al. Downwash-aware trajectory planning for quadrotor swarms. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vancouver: IEEE, 2017: 250 – 257.

[7] T. Kröger. Opening the door to new sensor-based robot applications-The Reflexxes Motion Libraries. *IEEE International Conference on Robotics and Automation*, Shanghai: IEEE, 2011: 1 – 4.

[8] T. Kröger, K. Oslund, T. Jenkins, et al. JediBot – Experiments in human-robot sword-fighting. *Experimental Robotics: The 13th International Symposium on Experimental Robotics*, Heidelberg: Springer, 2013: 155 – 166.

[9] B. Ezair, T. Tassa, Z. Shiller. Planning high order trajectories with general initial and final conditions and asymmetric bounds. *The International Journal of Robotics Research*, 2014, 33(6): 898 – 916.

[10] 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: IEEE, 2008: 3248 – 3253.

[11] H. Kano, H. Fujioka. Velocity and acceleration constrained trajectory planning by smoothing splines. *IEEE International Symposium on Industrial Electronics*, Edinburgh: IEEE, 2017: 1167 – 1172.

[12] H. Kano, H. Fujioka, C. F. Martin. Optimal smoothing and interpolating splines with constraints. *Applied Mathematics and Computation*, 2011, 218(5): 1831 – 1844.

[13] H. Kano, M. Egerstedt, H. Nakata, et al. B-splines and control theory. *Applied Mathematics and Computation*, 2003, 145(2): 263 – 288.

[14] S. Lai, K. Wang, B. M. Chen. Dynamically feasible trajectory generation method for quadrotor unmanned vehicles with state constraints. *Proceedings of the 36th Chinese Control Conference*, Dalian: IEEE, 2017: 6252 – 6257.

[15] T. Kröger, F. M. Wahl. Online trajectory generation: basic concepts for instantaneous reactions to unforeseen events. *IEEE Transactions on Robotics*, 2010, 26(1): 94 – 111.

[16] J. Chen, S. Shen. Improving octree-based occupancy maps using environment sparsity with application to aerial robot navigation. *IEEE International Conference on Robotics and Automation*, Singapore: IEEE, 2017: 3656 – 3663.

[17] Y. Lu, M. Cai, W. Ling, et al. *Quadrotor Control, Path Planning and Trajectory Optimization*. 2018: https://github.com/stormmax/quadrotor.

[18] S. Tang, V. Kumar. Autonomous flight. *Annual Review of Control, Robotics, and Autonomous Systems*, 2018, 1: 29 – 52.

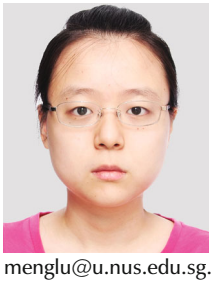
[19] Matlab Mathworks, Inc.: http://www.mathworks.com/.

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

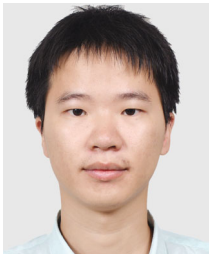
[21] K. Wang, Y. Ke, B. M. Chen. Autonomous reconfigurable hybrid tail-sitter UAV U-Lion. *Science China Information Sciences*, 2017, 60(3): DOI 10.1007/s11432-016-9002-x.

[22] K. Peng, F. Lin, B. M. Chen. Online schedule for autonomy of multiple unmanned aerial vehicles. *Science China Information Sciences*, 2017, 60(7): DOI 10.1007/s11432-016-9025-9.

Shupeng LAI received his B.Eng (1st) degree in Electrical and Electronics Engineering from Nanyang Technological University and his Ph.D. degree from the National University of Singapore. He is currently working as a research fellow in the National University of Singapore. His research interest is in mobile robots motion planning and control. E-mail: elelais@nus.edu.sg.



Menglu LAN received her B.Eng (1st) degree in Electrical and Computer Engineering from National University of Singapore (NUS) in 2015. She is currently pursuing her Ph.D. degree in NUS Graduate School for Integrative Sciences and Engineering, National University of Singapore, with research interest in task and motion planning for micro sized aerial vehicles (MAVs). E-mail: lanmenglu@u.nus.edu.sg.



Kehong GONG studied in Nation University of Singapore since 2012, and received the B.Eng. degree in Engineering Science Program in 2017. He started working with UAV in year 3 summer vocation, and continued working on it in his final year program. After graduation, he worked as a research engineer in the Unmanned Systems Research Group at the National University of Singapore. His research interests are in robotics and artificial intelligence. E-mail: gongkehong626@gmail.com.

Ben M. CHEN is currently a Professor in the Department of Mechanical and Automation Engineering, the Chinese University of Hong Kong. He was a Provost Chair Professor in the Department of Electrical and Computer Engineering, the National University of Singapore (NUS), where he was also serving as the Director of Control, Intelligent Systems and Robotics Area, and Head of Control Science Group, NUS Temasek Laboratories. His current

research interests are in unmanned systems, robust control, control applications, and financial market modeling. Dr. Chen has published more than 400 journal and conference articles, and a dozen research monographs in control theory and applications, unmanned systems as well as financial market modeling. He had served on the editorial boards of several international journals including IEEE Transactions on Automatic Control and Automatica. He currently serves as an Editor-in-Chief of Unmanned Systems. Dr. Chen has received a number of research awards nationally and internationally. His research team has actively participated in international UAV competitions, and won many championships in the contests. He is an IEEE Fellow. E-mail: bmchen@cuhk.edu.hk.