2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561585

Generating Large-Scale Trajectories Efficiently using Double Descriptions of Polynomials

Zhepei Wang[1,2,3], Hongkai Ye[1,2,3], Chao Xu[1,2], and Fei Gao[1,2]

***Abstract*— For quadrotor trajectory planning, describing a polynomial trajectory through coefficients and end-derivatives both enjoy their own convenience in energy minimization. We name them double descriptions of polynomial trajectories. The transformation between them, causing most of the inefficiency and instability, is formally analyzed in this paper. Leveraging its analytic structure, we design a linear-complexity scheme for both jerk/snap minimization and parameter gradient evaluation, which possesses efficiency, stability, flexibility, and scalability. With the help of our scheme, generating an energy optimal (minimum snap) trajectory only costs 1 μs per piece at the scale up to 1,000,000 pieces. Moreover, generating large-scale energy-time optimal trajectories is also accelerated by an order of magnitude against conventional methods.**

I. Introduction

Smooth polynomial trajectories generated from minimizing jerk/snap are widely used in the navigation of quadrotors [1]–[3]. Among these applications, the double descriptions of polynomial trajectory are frequently involved. One description, consisting of piece coefficients and piece times, is convenient for cost evaluation and trajectory configuration. Another description, consisting of piece end-derivatives and times, is convenient and stable for cost minimization [4].

Although these double descriptions offer an efficient and accurate way to obtain energy-optimal trajectories, the overhead and instability are often inevitable caused by numerical transformations between them [5]. Besides, piece times are coupled into transformations. Without knowing its structure, directly optimizing times becomes hard or quite inefficient. In this situation, many perturbed energy-optimal trajectories are often generated to obtain gradient information [6], thus ruining the convenience from descriptions.

To overcome these drawbacks, we study the transformation between double descriptions. Its concrete structure and analytic expression are clearly provided, which is indeed a diffeomorphism. Leveraging its analytic form, we first design a scheme for linear-complexity jerk/snap minimization. Unnecessary computation on transformation is eliminated from this scheme, making its speed faster than many known schemes by at least an order of magnitude. Utilizing the smoothness, we also derive an analytic gradient for waypoints and times, which also enjoys minimal complexity.

This work was supported by National Natural Science Foundation of China under Grant 62003299.

[1]State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China. {wangzhepei, hkye, cxu, fgaoaa}@zju.edu.cn

[2]Huzhou Institute, Zhejiang University, Huzhou 313000, China.

[3]National Engineering Research Center for Industrial Automation (Ningbo Institute), Ningbo 315000, China.

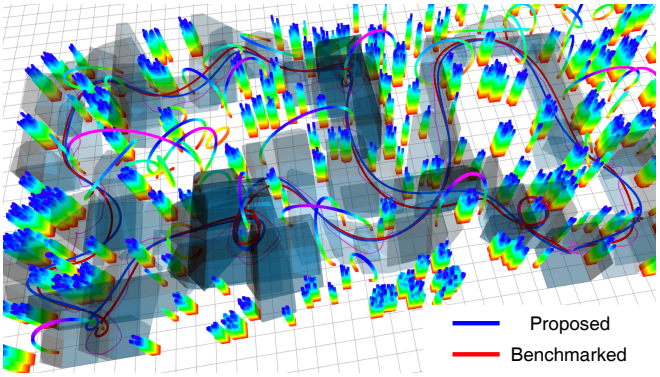

Fig. 1. Large-scale energy-time optimal trajectories generated by the proposed scheme and the benchmarked scheme which is the global trajectory optimization in Teach-Repeat-Replan [7]. The proposed one takes 0.43 seconds while the other takes 7.70 seconds. The energy-time cost is 2643.43 for the proposed one while it is 3269.76 for the other. The efficiency is improved by an order of magnitude.

The exact gradient information makes it possible to directly optimize all parameters under complex constraints.

In this paper, we propose a framework based on the above results. Provided with a collection of waypoints and piece times, the minimization of jerk/snap is taken as a black box with promising efficiency. Through the cheap exact gradient, our framework directly optimizes intermediate waypoints and piece times to meet the safety constraints and dynamic limits, respectively. Its flexibility and efficiency are validated by applications and benchmarks on classic problems. Summarizing our contributions in this work:

- An analytic transformation between double descriptions is derived.
- A linear-complexity minimum jerk/snap solution is designed with extreme efficiency.
- An analytic gradient for waypoints and piece times is provided with linear complexity.
- Applications and benchmarks on classic problems are provided to validate the superiority of our framework.
- High-performance implementation of solution and gradient computation are open-sourced for the reference of the community[1].

II. Related Work

Quadrotor trajectory planning using polynomials has been prosperous since Mellinger et al. [6]. They eliminate differential constraints from quadrotor dynamics via differential flatness. Then, enough smoothness of flat output trajectories

[1]Source code: https://github.com/ZJU-FAST-Lab/large_scale_traj_optimizer

guarantees the constraint satisfaction by default. It is thus quite different from common robotics trajectory generation where standard Nonlinear Programming (NLP) approaches must be employed to enforcing complicated dynamic constraints. Consequently, they conduct snap minimization for smooth flying trajectories with description of the first kind, where the coefficients are optimized through a Quadratic Programming (QP) with prescribed times and waypoints. The gradient for time is roughly estimated by solving perturbation problems. Richter et al. [1] propose the description of the second kind which eliminates equality constraints in the QP, thus forming a closed-form solution. For large problems, its efficiency deteriorates because of the involved sparse matrix inverse. Safety constraints and dynamic limits are heuristically enforced, which can cause low trajectory quality. The two methods above provide many insights in quadrotor trajectory planning, even though there are weaknesses.

Many schemes are proposed to improve the above two frameworks. To improve efficiency, Burke et al. [8] first propose a linear-complexity scheme to solve primal and dual variables of the QP, which generates trajectories with $500,000$ pieces in less than 3 minutes. However, its efficiency is still not satisfactory because of the redundant problem size and block inverses. Optimizing waypoints and times is still not considered. Burri et al. [2] and Oleynikova et al. [3] directly optimize all end-derivatives of inner pieces through NLP, where piece times are fixed. Almeida et al. [9] train a supervised neural network to learn time allocation offline. Thus relatively good piece times can be allocated online. Our previous work [10] proposes an efficient optimization scheme for piece times, while the handling for safety constraints lacks flexibility.

Due to the seeming inflexibility in raw polynomial splines, other methods describe trajectories via control points of B-Splines and Bézier curves [11]–[13]. The safety constraints and dynamic limits can be easily enforced via the convex hull property. Tordesillas et al. [14] utilize the property and optimize the interval allocation. A Mixed Integer Quadratic Programming (MIQP) is formulated to assign each trajectory piece into a convex region. However, the property can be conservative since the temporal profile of the trajectory is limited by the geometrical profile. To handle the issue, Gao et al. [7] propose a convex formulation to improve the dynamic performance of an already-known geometrical curve.

III. Preliminaries

Since the differential flatness of quadrotor has been validated in [6], polynomial trajectories are widely used in the continuous-time motion planning for quadrotors. Consider an $(N+1)$-order M-piece spline whose i-th piece is indeed an N-degree polynomial $p_i(t) = c_i^{\mathrm{T}}\beta(t),\ t \in [0, T_i]$, where $c_i \in \mathbb{R}^{(N+1)}$ is a coefficient vector of this piece and $\beta(t) = (1, t, t^2, \cdots, t^N)^{\mathrm{T}}$ a natural basis. The entire spline on the duration $[0, \tau_M]$ is defined by $p(t) := c_i^{\mathrm{T}}\beta(t - \tau_{i-1})$ where $t \in [\tau_{i-1}, \tau_i]$ and $\tau_i = \sum_{j=1}^{i} T_j$.

For any given differentially flat quadrotor, high-order continuity is required to satisfy the differential constraints induced by the dynamics. Besides, the smoothness is also ensured by penalizing the integral of square of the high-order derivative. Assuming the order of the penalized derivative to be s, the cost function can be written as

$$J(c, T) = \int_0^{\tau_M} p^{(s)}(t)^2 \mathrm{d}t, \tag{1}$$

where $c = (c_1^{\mathrm{T}}, \ldots, c_M^{\mathrm{T}})^{\mathrm{T}} \in \mathbb{R}^{M(N+1)}$ is a coefficient vector of the entire spline and $T = (T_1, \ldots, T_M)^{\mathrm{T}} \in \mathbb{R}^M$ is a piece time vector. The cost J implicitly decouples for each dimension [4], thus we only consider 1-dimensional splines. According to the Linear Quadratic Minimum-Time (LQMT) problem [15], we set the degree as $N = 2s - 1$ hereafter because it is the optimal degree for the minimizer of J.

A spline $p(t)$ can be naturally described by the collection $\{c, T\}$, i.e, the first description. Consider the minimization of $J(c, T)$ with fixed T and some derivatives specified at certain timestamps. It can be formulated as a QP [6] if c is taken as a vector of decision variables. The second description is denoted by the collection $\{d, T\}$ where $d \in \mathbb{R}^{2Ms}$ is an end-derivative vector $d = (d_1^{\mathrm{T}}, \ldots, d_M^{\mathrm{T}})^{\mathrm{T}}$ where

$$d_i = (p_i(0), \ldots, p_i^{(s-1)}(0), p_i(T_i), \ldots, p_i^{(s-1)}(T_i))^{\mathrm{T}}. \tag{2}$$

Bry et al. [4] leverage the convenience of $\{d, T\}$ to eliminate equality constraints in the QP so that a closed-form solution for the optimal d is constructed. Obviously, the double descriptions of polynomial trajectories provide a lot of convenience for energy minimization.

IV. Efficient Solution and Gradient Computation With Double Descriptions

In this section, we derive an explicit diffeomorphism between double descriptions of polynomial trajectories. Based on the transformation, we construct the solution for $J(c, T)$ with any T in $O(M)$ linear complexity. Utilizing its properties, we obtain the analytical gradient for problem parameters, i.e., specified derivatives and T, which offers much flexibility to further improve the trajectory quality.

A. Explicit Diffeomorphism Between Double Descriptions

First, we explore the relation between parameterization spaces of both descriptions. We denote by $\mathbb{P}_{\mathrm{c}} : \mathbb{R}^{2Ms} \times \mathbb{R}^M_{>0}$ the space for $\{c, T\}$ and by $\mathbb{P}_{\mathrm{d}} : \mathbb{R}^{2Ms} \times \mathbb{R}^M_{>0}$ the space for $\{d, T\}$. The relation between $\mathbb{P}_{\mathrm{c}}$ and $\mathbb{P}_{\mathrm{d}}$ is given as below.

Proposition 1. *For a polynomial trajectory, denote by $\{c, T\}$ its parameters in $\mathbb{P}_{\mathrm{c}}$ and by $\{d, T\}$ its parameters in $\mathbb{P}_{\mathrm{d}}$. The map from $\{c, T\}$ to $\{d, T\}$ is a C^∞-diffeomorphism between $\mathbb{P}_{\mathrm{c}}$ and $\mathbb{P}_{\mathrm{d}}$. An explicit diffeomorphism consists of an identity map on T and a smooth bijection between c and d:*

$$d = \mathbf{A}_{\mathrm{F}}(T)c,\ c = \mathbf{A}_{\mathrm{B}}(T)d, \tag{3}$$

where $\mathbf{A}_{\mathrm{F}}(T) = \oplus_{k=1}^{M}\mathbf{A}_f(T_k)$, $\mathbf{A}_{\mathrm{B}}(T) = \oplus_{k=1}^{M}\mathbf{A}_b(T_k)$, and $\oplus$ is the direct sum stacking all diagonal sub-matrices. The forward and backward sub-matrices are

$$\mathbf{A}_f(t) = \begin{pmatrix} \mathbf{E} & \mathbf{0} \\ \mathbf{F}(t) & \mathbf{G}(t) \end{pmatrix},\ \mathbf{A}_b(t) = \begin{pmatrix} \mathbf{U} & \mathbf{0} \\ \mathbf{V}(t) & \mathbf{W}(t) \end{pmatrix}, \tag{4}$$

which are partitioned into four blocks in $\mathbb{R}^{s\times s}$. The entry at the i-th row and j-th column of any block is defined by

$$\mathbf{E}_{ij} = \begin{cases} (i-1)! & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \tag{5}$$

$$\mathbf{F}_{ij}(t) = \begin{cases} (j-1)!/(j-i)! \cdot t^{j-i} & \text{if } i \leq j, \\ 0 & \text{if } i > j, \end{cases} \tag{6}$$

$$\mathbf{G}_{ij}(t) = \frac{(s+j-1)!}{(s+j-i)!} \cdot t^{s-i+j}, \tag{7}$$

$$\mathbf{U}_{ij} = \begin{cases} 1/(i-1)! & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \tag{8}$$

$$\mathbf{V}_{ij}(t) = \frac{\sum_{k=0}^{s-\max(i,j)} (-1)^k \binom{s}{i+k} \binom{2s-j-k-1}{s-1}}{(j-1)!\ (-1)^i \cdot t^{s+i-j}}, \tag{9}$$

$$\mathbf{W}_{ij}(t) = \frac{\sum_{k=0}^{s-\max(i,j)} \binom{s-k-1}{i-1} \binom{2s-j-k-1}{s-1}}{(j-1)!\ (-1)^{i+j} \cdot t^{s+i-j}}. \tag{10}$$

Proof. The forward sub-matrix $\mathbf{A}_f(t)$ given in (4), (5), (6), and (7), comes from the definition of d_i in (2). Actually, $\mathbf{A}_f(t)$ is a *general confluent Vandermonde Matrix* generated from two variables $\lambda_0 = 0$ and $\lambda_1 = t$, both with multiplicity s. According to *Spitzbart's Theorem* [16], the inverse of $\mathbf{A}_f(t)$ always exists for $\lambda_0 \neq \lambda_1$, whose entries are exactly coefficients of a set of polynomials constructed from λ_0 and λ_1. The backward sub-matrix $\mathbf{A}_b(t)$ given in (4), (8), (9), and (10), is derived following [16]. The process only involves lengthy but mechanical derivation thus is omitted here for brevity. Obviously, the map from c_i to d_i and the inverse are always smooth at any $T \in \mathbb{R}^M_{>0}$. The bijectivity and the smoothness imply a diffeomorphism. □

Proposition 1 gives analytic transformations between double descriptions, which have long been evaluated unwisely. In [1], $\mathbf{A}_b(t)$ is numerically computed by inverting $\mathbf{A}_f(t)$ for any given t. In [2], the structure of $\mathbf{A}_f(t)$ is exploited so that $\mathbf{A}_b(t)$ is evaluated efficiently and stably via *Schur complement*, where only the inverse of a sub-matrix is needed. Our previous work [10] has the fewest online computations, where coefficients in $\mathbf{A}_b(t)$ is offline numerically computed by setting $t = 1$. However, all of these schemes involve numerically unstable matrix inverse. Proposition 1 is free of these drawbacks for the exact analytic expression. The diffeomorphism structure can be further utilized to greatly improve the efficiency and quality of trajectory generation.

B. Linear-Complexity Trajectory Generation

Consider the following trajectory generation problem,

$$\min_{c,T}\ J(c,T) = \int_0^{\tau_M} p^{(s)}(t)^2 \mathrm{d}t, \tag{11}$$

$$s.t.\ \ p^{(j)}(0) = d_{0,j},\ 0 \leq j < s, \tag{12}$$

$$p^{(j)}(\tau_M) = d_{M,j},\ 0 \leq j < s, \tag{13}$$

$$p(\tau_i) = q_i,\ 0 \leq i < M. \tag{14}$$

The initial and terminal derivatives are specified by $d_0 = (d_{0,0}, \ldots, d_{0,s-1})^{\mathrm{T}}$ and $d_M = (d_{M,0}, \ldots, d_{M,s-1})^{\mathrm{T}}$, respectively. Each q_i is a specified intermediate waypoint at τ_i. The $s-1$ times continuous differentiability of $p(t)$ on $[0, \tau_M]$ is implicitly required by the definition of $J(c,T)$, which is not explicitly formulated as equality constraints here.

Although a closed-form solution of (11) is given by Bry et al. [4], its efficiency is limited by the numerical evaluation of $\mathbf{A}_b(t)$ and the sparse permutation matrix inverse. To attain the linear complexity, Burke et al. [8] leverage the problem structure to calculate both primal and dual variables through a block tridiagonal linear equation system. However, it still needs frequently inverting sub-blocks. The size of the system is also redundant because of the dual variables. Therefore, we give a linear-complexity scheme with minimal problem size, where the matrix inverse is totally eliminated.

Consider the $\{d, T\}$ description of $p(t)$. We only need to obtain all unspecified entries in d, which are indeed $p^{(j)}(\tau_i)$ for $1 \leq i < M$ and $1 \leq j < s$. Rewrite d as

$$d = \mathbf{P}(\bar{d} + \mathbf{B}\widetilde{d}) \tag{15}$$

where $\widetilde{d} \in \mathbb{R}^{(M-1)(s-1)}$ is a vector containing all unspecified entries. The constant vector $\bar{d} \in \mathbb{R}^{(M+1)s}$ is defined as

$$\bar{d} = (d_0^{\mathrm{T}}, q_1, 0_{s-1}, \ldots, q_{M-1}, 0_{s-1}, d_M^{\mathrm{T}})^{\mathrm{T}}. \tag{16}$$

where $0_{s-1} \in \mathbb{R}^{s-1}$ is a zero vector. The permutation matrix $\mathbf{P} = (\mathbf{P}_1^{\mathrm{T}}, \ldots, \mathbf{P}_M^{\mathrm{T}})^{\mathrm{T}}$ is defined as

$$\mathbf{P}_i = \left(\mathbf{0}_{2s\times(i-1)s}, \mathbf{I}_{2s}, \mathbf{0}_{2s\times(M-i)s}\right). \tag{17}$$

The matrix $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_{M-1})$ is defined as

$$\mathbf{B}_i = \left(\mathbf{0}_{(s-1)\times is}, \mathbf{D}, \mathbf{0}_{(s-1)\times(M-i)s}\right)^{\mathrm{T}}, \tag{18}$$

where $\mathbf{D} = (0_{s-1}, \mathbf{I}_{s-1})$. It should be noted that computing (15) only involves orderly accessing entries. We do not really need to compute the matrix product considering that $\mathbf{P}$ and $\mathbf{B}$ is highly structured.

The cost function $J(c,T)$ indeed takes a quadratic form

$$J(c,T) = c^{\mathrm{T}}\mathbf{Q}_\Sigma(T)c \tag{19}$$

in which $\mathbf{Q}_\Sigma(T) = \oplus_{i=1}^M \mathbf{Q}(T_i)$. Note that the symmetric matrix $\mathbf{Q}(t)$ has an analytical form whose entries are simple power functions of t. Its analytical form is provided by Bry et al. in the appendix of [4], to which we refer for details. Substituting (3) and (15) into (19) gives

$$J = (\bar{d} + \mathbf{B}\widetilde{d})^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{H}_\Sigma(T)\mathbf{P}(\bar{d} + \mathbf{B}\widetilde{d}) \tag{20}$$

in which $\mathbf{H}_\Sigma(T) = \oplus_{i=1}^M \mathbf{H}(T_i)$ is a symmetric matrix. All its diagonal blocks is fully determined by the matrix function

$$\mathbf{H}(t) = \mathbf{A}_b^{\mathrm{T}}(t)\mathbf{Q}(t)\mathbf{A}_b(t). \tag{21}$$

Obviously, the analytical form of $\mathbf{H}(t)$ can be easily derived for a fixed s by combining our Proposition 1 and the result from Bry et al. [4]. Therefore, we omit the parameter T and denote $\mathbf{H}_\Sigma = \oplus_{i=1}^M \mathbf{H}_i$ hereafter because it is trivial to obtain all diagonal blocks for any given T in linear time and space.

Differentiating J w.r.t. $\widetilde{d}$ gives

$$\mathrm{d}J/\mathrm{d}\widetilde{d} = 2\mathbf{B}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{H}_\Sigma\mathbf{P}(\mathbf{B}\widetilde{d} + \bar{d}). \tag{22}$$

The optimal $\widetilde{d}$ satisfies $\|\mathrm{d}J/\mathrm{d}\widetilde{d}\| = 0$, i.e.,

$$\mathbf{M}\widetilde{d} = \bar{b} \tag{23}$$

where $\mathbf{M} = \mathbf{B}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{H}_{\Sigma}\mathbf{P}\mathbf{B}$ and $\bar{b} = -\mathbf{B}^{\mathrm{T}}\mathbf{P}^{\mathrm{T}}\mathbf{H}_{\Sigma}\mathbf{P}\bar{d}$.

Actually, the computation for $\mathbf{M}$ and $\bar{b}$ is quite easy. We partition each $\mathbf{H}_i$ into four square blocks as

$$\mathbf{H}_i = \begin{pmatrix} \mathbf{\Gamma}_i & \mathbf{\Lambda}_i \\ \mathbf{\Phi}_i & \mathbf{\Omega}_i \end{pmatrix}. \tag{24}$$

Expanding $\mathbf{M}$ gives

$$\mathbf{M} = \begin{pmatrix} \boldsymbol{\alpha}_1 & \boldsymbol{\beta}_2 & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} \\ \boldsymbol{\gamma}_2 & \boldsymbol{\alpha}_2 & \boldsymbol{\beta}_3 & \cdots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\gamma}_3 & \boldsymbol{\alpha}_3 & \cdots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\alpha}_{M-2} & \boldsymbol{\beta}_{M-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{\gamma}_{M-1} & \boldsymbol{\alpha}_{M-1} \end{pmatrix} \tag{25}$$

where

$$\boldsymbol{\alpha}_i = \mathbf{D}(\mathbf{\Omega}_i + \mathbf{\Gamma}_{i+1})\mathbf{D}^{\mathrm{T}}, \tag{26}$$

$$\boldsymbol{\beta}_i = \mathbf{D}\mathbf{\Lambda}_i\mathbf{D}^{\mathrm{T}},\ \boldsymbol{\gamma}_i = \mathbf{D}\mathbf{\Phi}_i\mathbf{D}^{\mathrm{T}}. \tag{27}$$

Similarly, we partition $\bar{d}$ as

$$\bar{d} = (\kappa_0^{\mathrm{T}}, \kappa_1^{\mathrm{T}}, \ldots, \kappa_{M-1}^{\mathrm{T}}, \kappa_M^{\mathrm{T}})^{\mathrm{T}} \tag{28}$$

where each $\kappa_i \in \mathbb{R}^s$ is a constant vector. Expanding $\bar{b}$ gives

$$\bar{b} = (b_1^{\mathrm{T}}, b_2^{\mathrm{T}}, \ldots, b_{M-2}^{\mathrm{T}}, b_{M-1}^{\mathrm{T}})^{\mathrm{T}} \tag{29}$$

in which the i-th part can be computed as

$$b_i = -\mathbf{D}\left(\mathbf{\Phi}_i\kappa_{i-1} + (\mathbf{\Omega}_i + \mathbf{\Gamma}_{i+1})\kappa_i + \mathbf{\Lambda}_i\kappa_{i+1}\right). \tag{30}$$

Now we conclude the procedure to obtain the optimal trajectory. Firstly, compute each $\mathbf{H}_i$ for any given T using the analytical function $\mathbf{H}(t)$ defined in (21). Secondly, compute $\bar{d}$ according to its definition (16) for the specified derivatives. Thirdly, compute nonzero entries in $\mathbf{M}$ and $\bar{b}$ according to (26), (27) and (30). Fourthly, solve the linear equation system (23) using *Banded PLU Factorization* [17]. Finally, recover the optimal coefficient vector c through (15) and (3).

The above-concluded procedure only costs $O(M)$ linear time and space complexity. There is no numerical matrix inverse needed in the whole procedure because $\mathbf{A}_b$ is overcome by deriving its analytical form. Moreover, the linear equation system needed to be solved only involves necessary primal decision variables, thus achieving the minimal problem size.

C. Parameter Gradient Computation in Double Descriptions

Although the double descriptions make it possible to obtain the solution of (11) in a cheap way, the resultant trajectory quality is still determined by the parameter of the problem, i.e., intermediate waypoints $q = (q_1, \ldots, q_{M-1})^{\mathrm{T}}$ and piece times T. Further optimization on q and T is needed to improve the trajectory quality while maintaining feasibility. Therefore, we utilize the diffeomorphism between double descriptions of the trajectory to derive the analytical gradient w.r.t. q and T. The gradient helps to obtain optimal q and T, which bridges the gap in many traditional trajectory planning methods such as [18] and [19].

For any pair of q and T, denote by $\widetilde{d}(q,T)$ the corresponding optimal $\widetilde{d}$, which satisfies

$$\widetilde{d}(q,T) = \arg\min_{\widetilde{d}} J(\mathbf{A}_{\mathrm{B}}(T)\mathbf{P}(\bar{d} + \mathbf{B}\widetilde{d}), T). \tag{31}$$

Then, the optimal c, denoted by $c(q,T)$, is computed as

$$c(q,T) = \mathbf{A}_{\mathrm{B}}(T)\mathbf{P}(\bar{d} + \mathbf{B}\widetilde{d}(q,T)). \tag{32}$$

Define a new cost as $\hat{J}(q,T) := J(c(q,T),T)$. The gradient we concern is indeed for $\hat{J}(q,T)$, i.e., $\partial\hat{J}/\partial q$ and $\partial\hat{J}/\partial T$.

Without causing ambiguity, we temporarily omit the parameters in $\hat{J}(q,T)$, $J(c,T)$, $\widetilde{d}(q,T)$, $c(q,T)$, $\mathbf{A}_{\mathrm{B}}(T)$ and $\mathbf{A}_{\mathrm{F}}(T)$ for simplicity. As for the cost function in (31), $\widetilde{d}$ is a stationary point, which means its gradient w.r.t. $\widetilde{d}$ satisfies $\|(\partial J/\partial c)^{\mathrm{T}}\mathbf{A}_{\mathrm{B}}\mathbf{P}\mathbf{B}\| = 0$. Now we know that the gradient computation in either $\mathbb{P}_c$ or $\mathbb{P}_d$ possesses its convenience. In $\mathbb{P}_c$, the gradient $\partial J/\partial T_i$ and $\partial J/\partial c_i$ both have easy-to-derive analytic expressions. In $\mathbb{P}_d$, the gradient of the cost (31) by $\widetilde{d}$ is already zero. Thus, we first express $\partial\hat{J}/\partial T$ and $\partial\hat{J}/\partial q$ by $\partial J/\partial T_i$ and $\partial J/\partial c_i$. Then, the diffeomorphism in Proposition 1 is utilized to transform them into $\mathbb{P}_d$ such that terms relevant to $\widetilde{d}$ can be eliminated.

As for the gradient of $\hat{J}$ w.r.t. T_i, we have

$$\begin{aligned} \frac{\partial\hat{J}}{\partial T_i} &= \frac{\partial J}{\partial T_i} + \left(\frac{\partial J}{\partial c}\right)^{\mathrm{T}}\frac{\partial\mathbf{A}_{\mathrm{B}}}{\partial T_i}\mathbf{P}(\bar{d} + \mathbf{B}\widetilde{d}) \\ &\quad + \left(\frac{\partial J}{\partial c}\right)^{\mathrm{T}}\mathbf{A}_{\mathrm{B}}\mathbf{P}\mathbf{B}\frac{\partial\widetilde{d}}{\partial T_i} \\ &= \frac{\partial J}{\partial T_i} + \left(\frac{\partial J}{\partial c}\right)^{\mathrm{T}}\frac{\partial\mathbf{A}_{\mathrm{B}}}{\partial T_i}\mathbf{P}\mathbf{A}_{\mathrm{F}}c \\ &= \frac{\partial J}{\partial T_i} + \left(\frac{\partial J}{\partial c_i}\right)^{\mathrm{T}}\dot{\mathbf{A}}_b(T_i)\mathbf{A}_f(T_i)c_i. \end{aligned} \tag{33}$$

As for the gradient of $\hat{J}$ w.r.t. q_i, we have

$$\begin{aligned} \frac{\partial\hat{J}}{\partial q_i} &= \left(\frac{\partial J}{\partial c}\right)^{\mathrm{T}}\mathbf{A}_{\mathrm{B}}\mathbf{P}\left(\frac{\partial\bar{d}}{\partial q_i} + \mathbf{B}\frac{\partial\widetilde{d}}{\partial q_i}\right) \\ &= \left(\frac{\partial J}{\partial c}\right)^{\mathrm{T}}\mathbf{A}_{\mathrm{B}}\mathbf{P}\frac{\partial\bar{d}}{\partial q_i} \\ &= \sum_{k=0,1}\left(\frac{\partial J}{\partial c_{i+k}}\right)^{\mathrm{T}}\mathbf{A}_b(T_{i+k})e_{(1-k)s+1}. \end{aligned} \tag{34}$$

where e_j is the j-th column vector of $\mathbf{I}_{2s}$. As for $\partial J/\partial T_i$ and $\partial J/\partial c_i$, their analytic expressions are clearly derived in the appendix of [4].

It is obvious that the gradient computations given in (33) and (34) are both irrelevant to the piece number M. Thus, computing the gradient w.r.t. q and T only costs $O(M)$ linear time and space complexity. Besides, all involved formulas have smooth analytic forms for $T \in \mathbb{R}^M_{>0}$, which much increase the efficiency of gradient computation.

Aside from efficiency, our analytical gradient scheme enjoy advantages in numerical stability. Specifically, the major numerical issue comes from the structure of $\hat{J}(q,T)$ on T. As

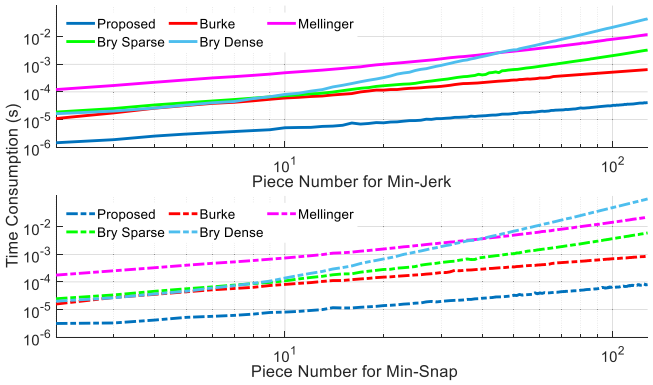

Fig. 2. Benchmark of small-scale trajectory generation. Pink lines are from Mellinger's scheme [6]. Lightblue lines are from Bry's scheme [4]. Green lines are from its sparse version. Red lines are from Burke's scheme [8]. Darkblue lines are from the proposed scheme. Solid lines are for jerk minimization. Dashed lines are for snap minimization.

analyzed in [10], if any piece time goes to zero, the trajectory energy goes to infinity, thus making $\hat{J}$ behave like a barrier function. Consequently, any scheme that evaluates gradient indirectly suffers instability from bad accuracy because this requires unrealistically small step size for finite difference [6] near the barrier. In comparison, ours is free from this issue because of the available accurate gradient.

V. Applications

A. Fast Minimum Jerk/Snap Trajectory Generation

One natural application of the previous linear-complexity solution scheme is that we can compute minimum-jerk/snap trajectory with extreme efficiency. Actually, there are already many reliable trajectory generation schemes for this topic. One thing that really matters is whether we can use a scheme as a black box without even worrying about its computation burden. We show that our scheme almost satisfies the requirements on this black box.

To demonstrate the efficiency of our scheme, we benchmark it with several conventional schemes, including the QP scheme by Mellinger et al. [6], the closed-form scheme by Bry et al. [4] and the linear-complexity scheme by Burke et al. [8]. As for Mellinger's scheme, we use OSQP [20] to solve the QP formed by linear conditions and quadratic cost. As for Bry's scheme, both the dense and sparse linear system solvers are implemented to calculate the closed-form solution. As for Burke's scheme, we implement an optimized version of our own for the sake of fairness. More specifically, it only costs several seconds to calculate a minimum-snap trajectory with $500,000$ pieces, while the one in their paper is reported to cost more than 2 minutes [8]. All schemes are implemented in C++11 without any explicit hardware acceleration.

We benchmark all these five schemes on 3-dimensional minimum jerk ($s = 3$) and minimum snap ($s = 4$) problems. All comparisons are conducted on an Intel Core i7-8700 CPU under Linux environment. For small-scale problems, the piece number ranges from 2 to 2^7. In each case, 100 sub-problems are randomly generated to be solved by these

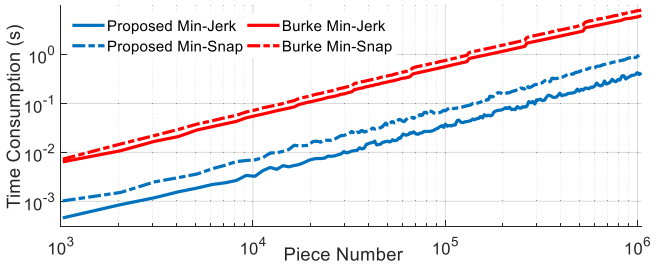

Fig. 3. Benchmark of large-scale trajectory generation. Red lines are from Burke's scheme [8]. Darkblue lines are from the proposed scheme. Solid lines are for jerk minimization. Dashed lines are for snap minimization.

schemes. The results are provided in Fig. 2. For large-scale problems, we only benchmark two linear-complexity schemes, where the piece number is sparsely sampled from 2^{10} to 2^{20}. The results are given in Fig. 3.

As shown in Fig. 2, Bry's closed-form solution and Burke's scheme are faster than Mellinger's scheme for small piece numbers. The dense version of Bry's scheme becomes the slowest since the cubic complexity for the dense solver dominates the time. Its sparse version still retains the efficiency as piece number grows. Due to the linear complexity, Burke's scheme and our scheme consume significantly less time than all other schemes. Moreover, ours is nearly an order of magnitude faster than Burke's at any problem scale in Fig. 2 or Fig. 3. Intuitively, ours is able to generate trajectories with 10^6 pieces in less than 1 second.

B. Fast Global Trajectory Optimization

Now we give another application for the linear-complexity solution and gradient. We provide a simple example here to significantly improve the efficiency of large-scale global trajectory optimization.

A drawback in traditional minimum snap based schemes is the lack of flexibility to adjust the piece times and waypoints. Such kind of scheme can only obtain gradient information unwisely by solving several perturbed problems [6]. To avoid this drawback, Gao et al. [7] propose a more flexible framework. It alternately optimizes the geometrical and temporal profile of a trajectory through two well-designed convex formulations. This framework generates high-quality large-scale trajectories within safe flight corridors while it cannot be done in real-time.

We assume that a polyhedron-shaped flight corridor has been generated as in [7]. The flight corridor $\mathcal{F}$ is defined as

$$\mathcal{F} = \bigcup_{i=1}^{M} \mathcal{P}_i \tag{35}$$

where each $\mathcal{P}_i$ is a finite convex polyhedron

$$\mathcal{P}_i = \left\{ x \in \mathbb{R}^3 \;\middle|\; \mathbf{A}_i x \preceq b_i \right\}. \tag{36}$$

Besides, locally sequential connection is also assumed

$$\begin{cases} \mathcal{P}_i \cap \mathcal{P}_j = \varnothing & \text{if } |i-j| = 2, \\ \mathcal{P}_i \cap \mathcal{P}_j \neq \varnothing & \text{if } |i-j| \leq 1. \end{cases} \tag{37}$$

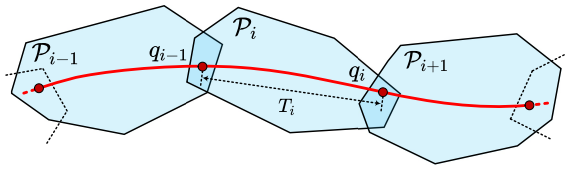

Fig. 4. Each intermediate waypoint q_i is confined within the intersection of two polyhedra $\mathcal{P}_i \cap \mathcal{P}_{i+1}$ using barrier functions. All piece times T_i and intermediate waypoints q_i are decision variables to be optimized.

For such a $\mathcal{F}$, the start and goal position is located in $\mathcal{P}_1$ and $\mathcal{P}_M$, respectively. As is shown in Fig. 4, we assign each 3-dimensional intermediate waypoint q_i in the intersection $\mathcal{P}_i \cap \mathcal{P}_{i+1}$, which roughly ensures the trajectory safety.

To obtain the spatial-temporal optimal trajectory within $\mathcal{F}$, we optimize the following cost function

$$J_\Sigma(q,T) = \hat{J}(q,T) + J_F(q) + J_D(q,T). \tag{38}$$

The cost term $\hat{J}(q,T)$ is also the 3-dimensional version. The cost term $J_F(q,T)$ is just a logarithmic barrier term to ensure that each q_i is confined within $\mathcal{P}_i \cap \mathcal{P}_{i+1}$, defined as

$$J_F(q) = -\kappa \sum_{i=1}^{M-1} \sum_{j=i}^{i+1} \mathbf{1}^{\mathrm{T}} \ln\left[b_j - \mathbf{A}_j q_i\right], \tag{39}$$

where κ is a constant barrier coefficient, $\mathbf{1}$ an all-ones vector with an appropriate length and $\ln[\cdot]$ the entry-wise natural logarithm. Actually, any C^2 clamped barrier function is an alternative to further eliminate the potential of the barrier in the interior of $\mathcal{P}_i \cap \mathcal{P}_{i+1}$, which can be easily constructed by following [21]. The cost term $J_D(q,T)$ is just a penalty to adjust the trajectory aggressiveness. It is defined as

$$\begin{aligned} J_D(q,T) = {} & \rho_t \sum_{i=1}^{M} T_i + \\ & \rho_v \sum_{i=1}^{M-1} g\left(\left\| \frac{q_{i+1}-q_{i-1}}{T_{i+1}+T_i} \right\|^2 - v_m^2\right) + \\ & \rho_a \sum_{i=1}^{M-1} g\left(\left\| \frac{(q_{i+1}-q_i)/T_{i+1} - (q_i-q_{i-1})/T_i}{(T_{i+1}+T_i)/2} \right\|^2 - a_m^2\right) \end{aligned} \tag{40}$$

where $g(x) = \max\{x,0\}^3$ is a C^2 penalty function, v_m the velocity limit and a_m acceleration limit. The constant ρ_t prevents the entire duration from growing too large. Constants ρ_v and ρ_a prevent the trajectory from being too aggressive. It also costs linear-complexity time for the value and gradient computation on $J_\Sigma(q,T)$. Then, we utilize the L-BFGS [22] with strong Wolfe conditions as an efficient quasi-Newton method to minimize the cost function.

As for constraints, it is possible that the interior part of a piece becomes unsafe or violates the limit too much. To handle such situations, we first utilize the feasibility checker proposed in [10] to locate such a piece. Then, the corresponding $\mathcal{P}_i$ is split into two intersecting parts, $\mathcal{P}_{i,0}$ and $\mathcal{P}_{i,1}$, as shown in Fig. 5. An intermediate waypoint $q_{i-1,i}$ is added as decision variables. Accordingly, we increase the corresponding penalty or barrier coefficient only for these two new pieces. In general, a larger penalty coefficient and

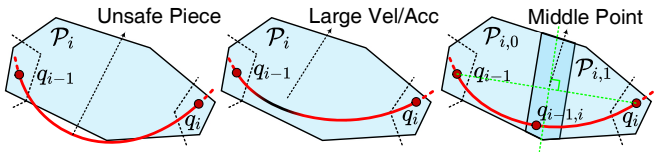

Fig. 5. If constraints are much violated on the piece between q_{i-1} and q_i, the corresponding polyhedron $\mathcal{P}_i$ is split into two intersecting polyhedra $\mathcal{P}_{i,0}$ and $\mathcal{P}_{i,1}$. where the barrier/penalty coefficients are also increased. Two perturbed planes near the perpendicular bisector of q_{i-1} and q_i are chosen as new facets. A new waypoint $q_{i-1,i}$ is also added in $\mathcal{P}_{i,0} \cap \mathcal{P}_{i,1}$.

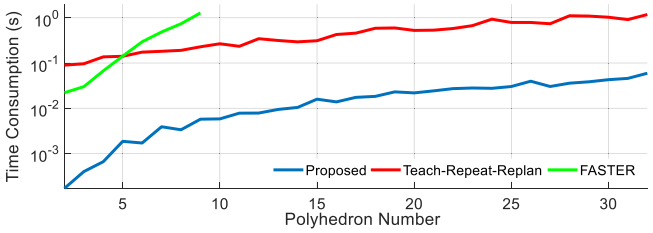

Fig. 6. Benchmark of spatial-temporal trajectory optimization in safe flight corridor. The red line is from Teach-Repeat-Replan [7]. The green line is from FASTER [14]. The blue line is from the proposed one. FASTER only supports small-scale corridors since the computation time of its MIQP grows approximately in an exponential way.

shorter piece length make the soft constraint $J_D(q,T)$ tighter. A larger barrier coefficient in $J_F(q)$ makes $q_{i-1,i}$ more likely to be in the interior of $\mathcal{P}_{i,0} \cap \mathcal{P}_{i,1}$, which helps to ensure the safety. Empirically, these operations are seldom needed according to our simulations.

We benchmark our simple scheme with the global trajectory optimizer in Teach-Repeat-Replan [7] which minimizes $J(c,T)+\rho_t \sum_{i=1}^{M} T_i$ under the same constraints. The interval allocation optimization in FASTER [14] is also compared here since it supports polyhedron-shaped corridor constraints. Due to the fact that it does not optimize time, we initialize it using total time from Teach-Repeat-Replan. We set $s = 3$, $\rho_t = 32.0$, $v_m = 4.0$ and $a_m = 5.0$ for all schemes and $\rho_v = \rho_a = 128.0$ for the proposed one. The relative cost tolerance is set as 10^{-4} for ours and the default value for the other two open-source implementations. The time out is set as $1.5s$. For each size, the computation time is averaged over 10 randomly generated corridors. The results from three methods is shown in Fig. 6. An intuitive comparison for large-scale trajectory generation is also provided in Fig. 1 where a spatial-temporal optimal trajectory is generated by our scheme using significantly less time.

VI. Conclusion

In this paper, we explore and exploit the relation between double descriptions of quadrotor polynomial trajectory. The resultant linear-complexity solution and gradient computation provide much flexibility and efficiency in classic trajectory generation problems. Simple applications are provided to demonstrate their promising performance. Our future work focus on incorporating our results into existing local planners that use polynomial trajectories. It remains to be validated whether our results can greatly improve the trajectory quality in these planners without sacrificing the efficiency.

REFERENCES

[1] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. of the Intl. Sym. of Robot. Research*, Singapore, Dec 2013.

[2] M. Burri, H. Oleynikova, M. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard MAVs in unknown environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Hamburg, Germany, Sep 2015.

[3] H. Oleynikova, C. Lanegger, Z. Taylor, M. Pantic, A. Millane, R. Siegwart, and J. Nieto, "An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments," *Journal of Field Robotics*, vol. 37, no. 4, pp. 642–666, 2020.

[4] A. Bry, C. Richter, A. Bachrach, and N. Roy, "Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments," *The International Journal of Robotics Research*, vol. 34, p. 1002, 2015.

[5] M. M. De Almeida and M. Akella, "New numerically stable solutions for minimum-snap quadcopter aggressive maneuvers," in *Proc. of the American Control Conference*, Seattle, USA, 2017.

[6] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011.

[7] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-Repeat-Replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[8] D. Burke, A. Chapman, and I. Shames, "Generating minimum-snap quadrotor trajectories really fast," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Las Vegas, USA, 2020.

[9] M. M. de Almeida, R. Moghe, and M. Akella, "Real-time minimum snap trajectory generation for quadcopters: Algorithm speed-up through machine learning," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Montreal, Canada, May 2019.

[10] Z. Wang, X. Zhou, C. Xu, J. Chu, and F. Gao, "Alternating minimization based trajectory generation for quadrotor aggressive flight," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4836–4843, 2020.

[11] J. A. Preiss, K. Hausman, G. S. Sukhatme, and S. Weiss, "Trajectory optimization for self-calibration and navigation." in *Robotics: Science and Systems*, 2017.

[12] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and Bernstein basis polynomial," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Brisbane, Australia, May 2018.

[13] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[14] J. Tordesillas, B. T. Lopez, and J. P. How, "FASTER: Fast and safe trajectory planner for flights in unknown environments," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Macau, China, 2019.

[15] E. Verriest and F. Lewis, "On the linear quadratic minimum-time problem," *IEEE Transactions on Automatic Control*, vol. 36, no. 7, pp. 859–863, 1991.

[16] R. Schappelle, "The inverse of the confluent vandermonde matrix," *IEEE Transactions on Automatic Control*, vol. 17, no. 5, pp. 724–725, 1972.

[17] G. H. Golub and F. V. Loan, *Matrix Computations*. The Johns Hopkins University Press, 2013.

[18] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Daejeon, Korea, 2016.

[19] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments," *Journal of Field Robotics*, vol. 36, no. 4, pp. 710–733, 2019.

[20] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, pp. 1–36, 2020.

[21] M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, and D. M. Kaufman, "Incremental potential contact: Intersection-and inversion-free, large-deformation dynamics," *ACM Transactions on Graphics*, vol. 39, no. 4, 2020.

[22] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.