Whole-Body Real-Time Motion Planning for Multicopters

Shaohui Yang[1,2], Botao He[1,3], Zhepei Wang[1], Chao Xu[1], and Fei Gao[1]

***Abstract*— Multicopters are able to perform high maneuverability yet their potential have not been fully achieved. In this work, we propose a full-body, optimization-based motion planning framework that takes shape and attitude of aerial robot into consideration such that the aggressiveness of drone maneuvering improves significantly in cluttered environment. Our method accepts series of intersecting polyhedrons describing any 3D free spaces and outputs a time-indexed trajectory in real time with a full-body collision-free fashion. We model the drone as tilted cuboid, yet we argue that our framework can be freely adjusted to fit aerial vehicles of all shapes. Guaranteeing dynamic feasibility and safety conditions, our framework transforms the original constrained nonlinear programming problem to an unconstrained one in higher dimensions which is further solved by quasi-Newton methods. Benchmark has shown that our method improves the state-of-art with orders of magnitude in terms of computation time and memory usage. Simulations and onboard experiments are carried out as validation.**

I. Introduction

As multicopters are endowed with increasingly diversified tasks such as searching over highly complicated unstructured indoor environment within bounded and narrow free spaces, crossing over dense and small-scaled irregularly shaped gaps and planning on-the-fly like birds to handle unexpected circumstances, unprecedented controllability over every single point on the drone in real time is urgently required from a realism perspective. The general statement is: carrying out milliseconds-level motion planning tasks under tight and rigorous geometrical constraints on entire body of the multicopters.

Nevertheless, the aforementioned demand is still far from fulfilled. As been pointed out by [1], kinodynamic motion planning considering attitude of multicopters and obstacle avoidance at the same time is a challenging task. A vital reason is that the contour of an quadrotor along a trajectory is non-convex as shown in Fig 1.

The main stream work-around is to ignore the orientation of drones completely by dilating obstacles radically according to its largest axis length, which leads to conservative performance that does not fully exploit free spaces. For existing works that indeed consider the attitude, optimization-based methods either formulate the problem on manifolds residing in high dimensions or make strong but inaccessible assumptions on the environments while search-based methods are applicable to certain resolution and optimality is only guaranteed in the discretized space. Both methods return unsatisfying results even at the cost of long computation time and heavy memory usage.

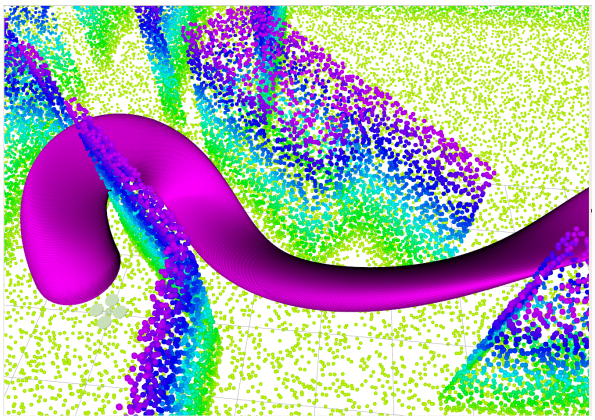


Fig. 1. Our method is able to generate highly aggressive trajectory under hard geometrical constraints such as gap crossing. The purple ellipsoid disks are simulated drone models at discrete time instants. It is trivial that the contours of quadrotor along trajectory form a non-convex set.

We are dedicated to constructing a trajectory optimizer that considers the dynamics and body shape of multicopters to achieve passable and aggressive maneuvers in complex environments. The rotation and translation of the rigid body are deeply coupled given that drones are under-actuated platforms. This work starts from our earlier optimization-based trajectory planning approach that takes in a series of polyhedrons as description of 3D free space and outputs a dynamically feasible trajectory inside them. We extend it by explicitly calculating the robot attitude along the trajectory and constructing a penalty term as part of objective function.

Building on our previous work [2], this work contributes to the following points:

- A *milliseconds*-level full-body optimization-based trajectory planning algorithm is proposed with collision-free and dynamic feasibility guarantee. To the best of our knowledge, this is the *first* method that generate trajectory satisfying all aforementioned constraints in real-time.
- A study and analysis on the comparison between our method and state-of-art.

II. Related Work

The front-end of motion planning is about properly finding and describing free spaces. Several works have concluded series of convex polyhedrons suitable for the task. In [3], polyhedrons are generated after taught by humans. Recent work [4] builds polytopes directly from points clouds in

[1] State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, 310027, China.
[2] School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden.
[3] School of Automation, Nanjing Institute of Technology, Nanjing, China
Email:shaohuiy@kth.se, botao.he@njit.edu.cn, {wangzhepei,cxu,fgaoaa}@zju.edu.cn

milliseconds. Safe flight corridors are created by inflating the input path first to an ellipsoid then to a convex polyhedron in [5]. The last one will be used in this work.

Back-end of full-body motion planning is typically achieved in two distinctive ways: either *dynamically* adjusting platform shape according to the environment, or optimizing / searching a passable trajectory under constraints imposed by obstacles and *fixed* model shape.

Foldable drone by Falanga et al. [6] represents the first type. Extra degrees of freedom provided by servos enables morphology modification for irregular space passablity. However, tilting a *fixed* drone like a bird rather than exploring hard-to-maintain mechanisms is our main focus.

Some optimization-based methods start with strong assumptions on geometrical constraints. Aggressive gap crossing is achieved in rectangle ones with negligible thickness and absolute knowledge of them in [7]. The assumption is eased by [8] such that onboard gap detection is possible. In [9], similar assumption still holds yet the optimizer is replaced by reinforcement learning technique. All these works are designed for certain gap crossing *only*. Lacking of generality is their greatest drawback.

Other optimization methods happen directly on manifolds. Watterson et al. [10] proposes a parameterization invariant manifold trajectory optimization algorithm respecting constraints by safe corridor on manifolds. It demonstrates application on $SO(3)$ yet concludes nothing about resource usage and comparison with other works. Meanwhile, manifolds are parameter-heavy thus when computing Hessian, the curse of dimensionality befalls.

Search-based motion planning framework considering the drone's attitude and shape is proposed by Liu et al. [11]. Constant control inputs are applied for fixed duration Δt to generate motion primitives, followed by a feasibility checker to filter the safe ones that have no intersections with the given point cloud. In practical implementation, adaptive dimensionality scheme is used to accelerate motion planning. However, the framework has following problems: 1. No smart way of generating motion primitives, causing numerous useless explorations; 2. Range and step of control inputs require fine-tuning; 3. Fixed control resolution makes smooth and flexible intractable.

III. Preliminaries

A. Generation of Body Attitude from Differential Flatness

The most straightforward way to consider the drone's attitude $\boldsymbol{R}_b$ in trajectory optimization is figuring out its functional form. Fortunately, leveraging results by Mellinger et al. [12], the orientation $\boldsymbol{R}_b = [\boldsymbol{r}_{1b}, \boldsymbol{r}_{2b}, \boldsymbol{r}_{3b}] \in SO(3)$ can be written an algebraic function of four differentially flat outputs, namely $\boldsymbol{\sigma} = [p_x, p_y, p_z, \psi]^{\mathrm{T}}$ where $\boldsymbol{p} = [p_x, p_y, p_z]^{\mathrm{T}}$ represents the coordinate of drone center of mass(CoM) and ψ is the yaw angle. We will get the explicit form of $\boldsymbol{R}_b(\boldsymbol{\sigma})$ in this subsection and show later how this derivation can be plugged into our optimization framework.

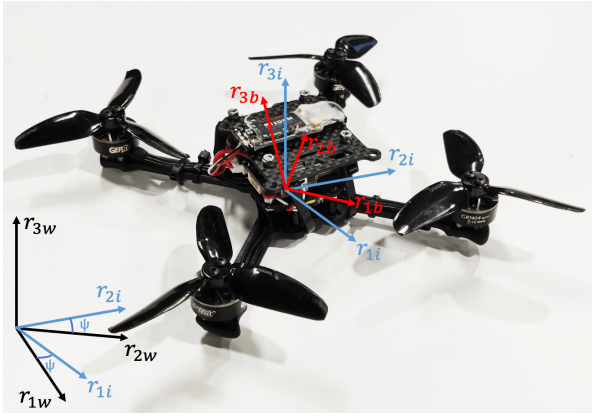


Fig. 2. World frame $\boldsymbol{r}_w$, intermediate frame $\boldsymbol{r}_i$ and body frame $\boldsymbol{r}_b$.

Ignoring air resistance and other disturbing forces, the force analysis of a quadrotor under Newton's law is:

$$-mg\boldsymbol{r}_{3w} + f_s\boldsymbol{r}_{3b} = m\ddot{\boldsymbol{p}} \tag{1}$$

where $\boldsymbol{r}_{3w} = [0, 0, 1]^{\mathrm{T}}$ is the world z-axis, $\boldsymbol{r}_{3b}$ the body z-axis, f_s the sum of forces provided by four propellers and g the gravitational acceleration, typically $g = 9.81m/s$.

A direct consequence of (1) is as follows:

$$\boldsymbol{r}_{3b} \coloneqq \frac{\boldsymbol{t}}{A} \tag{2a}$$

$$A \coloneqq \sqrt{\ddot{p_x}^2 + \ddot{p_y}^2 + (\ddot{p_z} + g)^2}, \boldsymbol{t} \coloneqq [\ddot{p_x}, \ddot{p_y}, \ddot{p_z} + g]^{\mathrm{T}} \tag{2b}$$

Now we introduce an intermediate frame denoted by $\boldsymbol{r}_i$ as shown in Fig. 2 with $\boldsymbol{r}_{3i} = \boldsymbol{r}_{3w}$. The x and y axis of are $\boldsymbol{r}_i$ rotated by angle ψ anti-clockwise:

$$\boldsymbol{r}_{1i} = [\cos\psi, \sin\psi, 0]^{\mathrm{T}}, \boldsymbol{r}_{2i} = [-\sin\psi, \cos\psi, 0]^{\mathrm{T}} \tag{3}$$

Leveraging the coordinate $\boldsymbol{r}_i$, $\boldsymbol{r}_{2b}$ can be written as:

$$\boldsymbol{r}_{2b} = \frac{\boldsymbol{r}_{3b} \times \boldsymbol{r}_{1i}}{\|\boldsymbol{r}_{3b} \times \boldsymbol{r}_{1i}\|} \coloneqq \frac{\boldsymbol{k}}{B} \tag{4a}$$

$$\boldsymbol{k} \coloneqq [-(\ddot{p_z} + g)\sin\psi, (\ddot{p_z} + g)\cos\psi, \ddot{p_x}\sin\psi - \ddot{p_y}\cos\psi]^{\mathrm{T}} \tag{4b}$$

$$B \coloneqq \sqrt{(\ddot{p_z} + g)^2 + (\ddot{p_x}\sin\psi - \ddot{p_y}\cos\psi)^2} \tag{4c}$$

Since $\boldsymbol{r}_{2b}$ and $\boldsymbol{r}_{3b}$ are uni-length vectors, the remaining $\boldsymbol{r}_{1b}$ is straightforward to obtain without any normalizing term:

$$\boldsymbol{r}_{1b} = \boldsymbol{r}_{2b} \times \boldsymbol{r}_{3b} \coloneqq \frac{\boldsymbol{s}}{AB} \tag{5a}$$

$$\boldsymbol{s} \coloneqq \begin{bmatrix} (\ddot{p_z} + g)^2 \cos\psi + \ddot{p_y}^2 \cos\psi - \ddot{p_x}\ddot{p_y}\sin\psi \\ (\ddot{p_z} + g)^2 \sin\psi + \ddot{p_x}^2 \sin\psi - \ddot{p_x}\ddot{p_y}\cos\psi \\ -(\ddot{p_z} + g)(\ddot{p_x}\cos\psi + \ddot{p_y}\sin\psi) \end{bmatrix} \tag{5b}$$

We summarize this subsection with the following notation:

$$\boldsymbol{R}_b(\boldsymbol{\sigma}) = \begin{bmatrix} \boldsymbol{r}_{1b}(\boldsymbol{\sigma}) & \boldsymbol{r}_{2b}(\boldsymbol{\sigma}) & \boldsymbol{r}_{3b}(\boldsymbol{\sigma}) \end{bmatrix} \tag{6}$$

and that the three axis are defined in (5),(4),(2) respectively.

B. Geometrically Constrained Trajectory Optimization for Multicopers Framework Revisit

1) Optimality Condition for Unconstrained Problem: The revisit starts from a multi-segment minimum control effort problem for a chain of s-integrators:

$$\min_{\boldsymbol{p}(t)} \quad \int_{t_0}^{t_M} \boldsymbol{u}(t)^{\mathrm{T}} \boldsymbol{W} \boldsymbol{u}(t) dt \tag{7a}$$

$$\text{s.t.} \quad \boldsymbol{u}(t) = \boldsymbol{p}^{(s)}(t), \forall t \in [t_0, t_M] \tag{7b}$$

$$\boldsymbol{p}^{[s-1]}(t_0) = \bar{\boldsymbol{p}}_o, \boldsymbol{p}^{[s-1]}(t_M) = \bar{\boldsymbol{p}}_f \tag{7c}$$

$$\boldsymbol{p}^{[d_i-1]}(t_i) = \bar{\boldsymbol{p}}_i, 1 \leq i < M \tag{7d}$$

with the time duration $[t_0, t_M]$ split into M stages by *given* timestamps $t_0 < t_1 < \cdots < t_M$. The variable to be optimized $\boldsymbol{p}(t) : [t_0, t_M] \to \mathbb{R}^m$ is the flat output of system, $\boldsymbol{u}$ is s-order derivative of $\boldsymbol{p}$ acting as control effort, $\boldsymbol{W}$ is the *given* penalty matrix for control variable. $\boldsymbol{p}^{[s-1]} = [\boldsymbol{p}^{\mathrm{T}}, \dot{\boldsymbol{p}}^{\mathrm{T}}, \ldots, (\boldsymbol{p}^{(s-1)})^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{ms}$ in (7c) represents the *given* initial and final conditions up to order $s-1$, $\boldsymbol{p}^{[d_i-1]} \in \mathbb{R}^{md_i}$ in (7d) are the *given* derivatives of flat output at intermediate timestamp t_i up to order $d_i - 1$.

To solve problem (7) with the *given* quantities, we first concatenate time allocation $\boldsymbol{T} = [T_1, \ldots, T_M]^{\mathrm{T}}$ with $T_i \coloneqq t_i - t_{i-1}$ and intermediate points $\boldsymbol{q} = [\bar{\boldsymbol{p}}_o, \bar{\boldsymbol{p}}_1, \ldots, \bar{\boldsymbol{p}}_{M-1}, \bar{\boldsymbol{p}}_f]$. Second, we construct matrices $\boldsymbol{M}(\boldsymbol{T}) \in \mathbb{R}^{2Ms \times 2Ms}$ and $\boldsymbol{b}(\boldsymbol{q}) \in \mathbb{R}^{2Ms \times m}$ following the pattern introduced in [2]. Finally, we solve the matrix equation:

$$\boldsymbol{M}(\boldsymbol{T})\boldsymbol{c} = \boldsymbol{b}(\boldsymbol{q}) \tag{8}$$

and extract it under the format $\boldsymbol{c} = [\boldsymbol{c}_1^{\mathrm{T}}, \ldots, \boldsymbol{c}_M^{\mathrm{T}}]^{\mathrm{T}}$, $\boldsymbol{c} \in \mathbb{R}^{2Ms \times m}$, $\boldsymbol{c}_i \in \mathbb{R}^{2s \times m}$. The $\boldsymbol{c}_i$ matrices are coefficients of polynomials.

By Theorem 2 (Optimality Condition) in [2], we guarantee that the optimal solution to problem (7) is given by:

$$\boldsymbol{p}(t) = \boldsymbol{p}_i(t - t_{i-1}), \forall t \in [t_{i-1}, t_i) \tag{9a}$$

$$\boldsymbol{p}_i(t) \coloneqq \boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{\beta}_0(t), t \in [0, T_i] \tag{9b}$$

$$\boldsymbol{\beta}_0(t) \coloneqq [1, t, t^2, \cdots, t^N]^{\mathrm{T}} \tag{9c}$$

with $\boldsymbol{\beta}_0(t)$ to be the basis time vector, $N = 2s - 1$ is the degree of polynomial. We also derive the velocity, acceleration and jerk of i-th trajectory for later usage:

$$\boldsymbol{v}_i(t) = \boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{\beta}_1(t) \tag{10a}$$

$$\boldsymbol{a}_i(t) = \boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{\beta}_2(t) = \begin{bmatrix} \ddot{p_x} & \ddot{p_y} & \ddot{p_z} \end{bmatrix}^{\mathrm{T}} \tag{10b}$$

$$\boldsymbol{j}_i(t) = \boldsymbol{c}_i^{\mathrm{T}} \boldsymbol{\beta}_3(t) \tag{10c}$$

$$\boldsymbol{\beta}_j(t) = \boldsymbol{\beta}_0^{(j)}(t), j \in \{1, 2, 3\} \tag{10d}$$

The above procedure describes how to find the optimal M-stage trajectory parameterized by $\boldsymbol{c}$ *given* $\boldsymbol{q}$ and $\boldsymbol{T}$. However, our interest lies in finding the optimal intermediate points and time allocation that leads to the best $\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{T})$:

$$\min_{\boldsymbol{q}, \boldsymbol{T}} \quad H(\boldsymbol{q}, \boldsymbol{T}) \tag{11a}$$

$$\text{s.t.} \quad \text{certain constraints} \tag{11b}$$

where $H(\boldsymbol{q}, \boldsymbol{T}) \coloneqq F(\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{T}), \boldsymbol{T})$ and $F(\boldsymbol{c}, \boldsymbol{T})$ is user-defined the control effort of a piecewise polynomial with parameters $\boldsymbol{c}$ and $\boldsymbol{T}$. [2] also shows that given $\frac{\partial F}{\partial \boldsymbol{q}}$, $\frac{\partial F}{\partial \boldsymbol{c}}$, both $\frac{\partial H}{\partial \boldsymbol{q}}$ and $\frac{\partial H}{\partial \boldsymbol{T}}$ can be obtained within $O(M)$ time and space complexity. Later chapter will show usage of this property.

2) Geometrically Constrained Optimization: The general form of problem that [2] aims at is:

$$\min_{\boldsymbol{p}(t), T} \quad \int_0^T \boldsymbol{u}(t)^{\mathrm{T}} \boldsymbol{W} \boldsymbol{u}(t) dt + \rho(T) \tag{12a}$$

$$\text{s.t.} \quad \boldsymbol{u}(t) = \boldsymbol{p}^{(s)}(t), \forall t \in [0, T] \tag{12b}$$

$$\mathcal{G}(\boldsymbol{p}(t), \ldots, \boldsymbol{p}^{(s)}(t)) \preceq \boldsymbol{0}, \forall t \in [0, T] \tag{12c}$$

$$\boldsymbol{p}(t) \in \mathcal{F}, \forall t \in [0, T] \tag{12d}$$

$$\boldsymbol{p}^{[s-1]}(0) = \bar{\boldsymbol{p}}_o, \boldsymbol{p}^{[s-1]}(T) = \bar{\boldsymbol{p}}_f \tag{12e}$$

where T is the total time, $\boldsymbol{p}(t)$, $\boldsymbol{u}$, $\boldsymbol{W}$, $\boldsymbol{p}^{[s-1]}$ are defined similarly as (7), $\rho(\cdot)$ is the time regularization function, $\mathcal{G}$ represents nonlinear inequalities, $\mathcal{F}$ represents the obstacle-free region in the configuration space.

In our settings, $\boldsymbol{p}(t) \in \mathbb{R}^3$ (thus $m = 3$) is part of $\boldsymbol{\sigma}$ and we are ignoring the yaw angle from now on by setting $\psi = 0$. The obstacle-free region(also known as flight corridor) is approximated with M given polyhedrons in $\mathcal{H}$-representation and they are assumed to be consecutively intersected for convenience:

$$\mathcal{F} = \bigcup_{i=1}^{M} \mathcal{P}_i^{\mathcal{H}} \subset \mathbb{R}^3 \tag{13a}$$

$$\mathcal{P}_i^{\mathcal{H}} = \{\boldsymbol{x} \in \mathbb{R}^3 | \boldsymbol{A_i x} \preceq \boldsymbol{b_i}\} \tag{13b}$$

$$\begin{cases} \mathcal{P}_i^{\mathcal{H}} \cap \mathcal{P}_j^{\mathcal{H}} = \emptyset & \text{if} |i - j| > 1 \\ \mathcal{P}_i^{\mathcal{H}} \cap \mathcal{P}_j^{\mathcal{H}} \neq \emptyset & \text{if} |i - j| \leq 1 \end{cases} \tag{13c}$$

Under deduction in III-B.1, problem (12) is equivalent to the following:

$$\min_{\boldsymbol{q}, \boldsymbol{T}} \quad J_q(\boldsymbol{q}, \boldsymbol{T}) + \rho(\|\boldsymbol{T}\|_1) \tag{14a}$$

$$\text{s.t.} \quad \boldsymbol{T} \succeq 0 \tag{14b}$$

$$\boldsymbol{p}(t) \in \mathcal{F}, \forall t \in [t_0, t_M] \tag{14c}$$

$$\mathcal{G}(\boldsymbol{p}(t), \ldots, \boldsymbol{p}^{(s)}(t)) \preceq \boldsymbol{0}, \forall t \in [t_0, t_M] \tag{14d}$$

where $\boldsymbol{p}(t)$ comes from (9) and can thus be regarded as $\boldsymbol{p}(\boldsymbol{q}, \boldsymbol{T})(t)$ and $J_q(\boldsymbol{q}, \boldsymbol{T}) \coloneqq J_c(\boldsymbol{c}(\boldsymbol{q}, \boldsymbol{T}), \boldsymbol{T})$ corresponding to the integral of control efforts.

The general thoughts of solving (14) is to bypass the constraints and do unconstrained optimization using quasi-Newton method. For temporal and spatial constraints in (14b) and (14c), we propose diffeomorphism based method to eliminate them in [2]. We write $\boldsymbol{T} = \boldsymbol{T}(\boldsymbol{\tau})$ and $\boldsymbol{q} = \boldsymbol{q}(\boldsymbol{\xi})$ such that $\boldsymbol{\tau}$ and $\boldsymbol{\xi}$ are unconstrained variables with higher dimension. For general nonlinear constraints in (14d), we add its discretized version to the objective function in (14a) as penalizing term.

However, it is critical to note that by enforcing (14c), we only guarantee that drone CoM is within the obstacle-free area along the trajectory rather than the entire body of drone because we do not take the drone's attitude ans shape into

account. To overcome that, we formulate a series of nonlinear constraints $\mathcal{G}_{att}(\boldsymbol{p}(t), \ddot{\boldsymbol{p}}(t)) \preceq \mathbf{0}$ in the next section.

IV. Full-body Motion Planning

In this section, we first show an efficient way of modeling the shape of drone and specify the full-body collision-free conditions explicitly with the given free spaces (13). Then, we further formulate it into nonlinear constraints $\mathcal{G}_{att}$ and soften it with time integral penalty with fixed relative resolution. Finally, we derive the derivatives of penalty term to add it up to the overall optimization framework.

A. Quadrotor Modeling and Trajectory within Polyhedrons

The $\mathcal{H}$-representation of M closed convex polyhedrons in (13b) can be alternatively written as: $\forall i \in \{1, 2, \dots, M\}$

$$\mathcal{P}_i^{\mathcal{H}} = \{\boldsymbol{q} \in \mathbb{R}^3 | (\boldsymbol{n}_i^k)^{\mathrm{T}}(\boldsymbol{q} - \boldsymbol{o}_i^k) \leq 0, k = 1, \dots, K_i\} \quad (15)$$

with each polyhedron $\mathcal{P}_i^{\mathcal{H}}$ composed of K_i hyperplanes and each hyperplane characterized with normal vector $\boldsymbol{n}_i^k$ pointing inwards and one point $\boldsymbol{o}_i^k$ on it.

To achieve full-body obstacle avoidance, we would like the inequality (15) always holding for arbitrary point $\boldsymbol{q} \in \mathcal{Q}_i(t)$

$$\begin{aligned} \mathcal{Q}_i(t) = \{\boldsymbol{q} | \boldsymbol{q} = \boldsymbol{q}_i^o(t) + \boldsymbol{p}_i(t)\} \\ \forall t \in [0, T_i], \forall i \in \{1, 2, \dots, M\} \end{aligned} \quad (16)$$

where $\boldsymbol{p}_i(t)$ is the drone CoM position along the i-th trajectory and $\boldsymbol{q}_i^o(t)$ is the offset from any point on the drone from CoM, both of which are in world coordinate. Non-convex set $\mathcal{Q}_i(t)$ can be interpreted as an union of all points on the drone moving along the i-th trajectory, or an inflation of i-th trajectory according to the drone model.

The general form of $\boldsymbol{q}_i^o(t)$ is:

$$\begin{aligned} \boldsymbol{q}_i^o(t) = \begin{bmatrix} \boldsymbol{r}_{1b}^i & \boldsymbol{r}_{2b}^i & \boldsymbol{r}_{3b}^i \end{bmatrix} \begin{bmatrix} \tilde{q}_x & \tilde{q}_y & \tilde{q}_z \end{bmatrix}^{\mathrm{T}} = \boldsymbol{R}_b^i(t)\tilde{\boldsymbol{q}} \\ \forall t \in [0, T_i], \quad \forall i \in \{1, 2, \dots, M\} \end{aligned} \quad (17)$$

with carefully selected $\tilde{\boldsymbol{q}} = [\tilde{q}_x, \tilde{q}_y, \tilde{q}_z]^{\mathrm{T}} \in \tilde{\mathcal{Q}}$. Modeling of the drone determines $\tilde{\mathcal{Q}}$. Here we list two examples but we argue that this constant set $\tilde{\mathcal{Q}}$ may vary in a great range and only depends on the shape of certain multicopters.

In [11], the drone is modeled as an ellipsoid (See Fig. 3) with radius r, height h and diagonal matrix $\boldsymbol{E}$:

$$\tilde{\mathcal{Q}}_{ellip} = \{\boldsymbol{E}\tilde{\boldsymbol{q}}_n | \|\tilde{\boldsymbol{q}}_n\| \leq 1\} \quad \boldsymbol{E} \coloneqq \mathrm{diag}(r, r, h) \in \mathbb{R}^{3\times 3} \quad (18)$$

Though the above ellipsoid description is closer to the actual shape of a drone, it suffers from the drawback that there are infinitely many points to check. As a workaround, the drone is modeled as cuboid with half length and width $= r$ and half height $= h$ as shown in Fig. 3 such that only eight vertices are to be considered:

$$\tilde{\mathcal{Q}}_{cub} = \{\tilde{\boldsymbol{q}}_v = \begin{bmatrix} \pm r & \pm r & \pm h \end{bmatrix}^{\mathrm{T}}, v = 1, 2, \dots, 8\} \quad (19)$$

Thus, one way of constructing nonlinear constraints to achieve full-body collision-free is to combine (13b), (16) and (19):

$$\mathcal{G}_{att}^v(\boldsymbol{p}_i(t), \ddot{\boldsymbol{p}}_i(t)) = \boldsymbol{A}_i(\boldsymbol{p}_i(t) + \boldsymbol{R}_b^i(t)\tilde{\boldsymbol{q}}_{\boldsymbol{v}}) - \boldsymbol{b}_i \in \mathbb{R}^{K_i} \quad (20a)$$

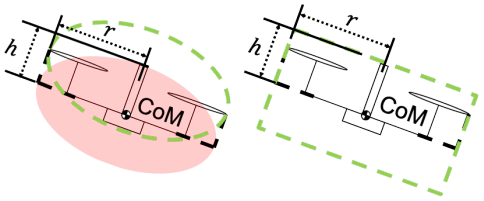


Fig. 3. Ellipsoid and cuboid model of drone. If height h is measured from CoM that does not lie on the plane formed by propellers, ellipsoid in red might not fully contain the drone. Cuboid model does not have this concern.

$$\mathcal{G}_{att}(\boldsymbol{p}_i(t), \ddot{\boldsymbol{p}}_i(t)) = \left[[\mathcal{G}_{att}^v(\boldsymbol{p}_i(t), \ddot{\boldsymbol{p}}_i(t))^{\mathrm{T}}]_{v=1}^8 \right]^{\mathrm{T}} \in \mathbb{R}^{8K_i} \quad (20b)$$

Note that in (20), $\mathcal{G}_{att}(\boldsymbol{p}_i(t), \ddot{\boldsymbol{p}}_i(t)) = \mathcal{G}_{att}(\boldsymbol{c}_i, T_i)$ is constructed such that the inflation of i-th trajectory is completely within the i-th polyhedron, or simply $\mathcal{Q}_i(t) \subset \mathcal{P}_i^{\mathcal{H}}, \forall t \in [0, T_i]$. This is one reasonable way of construction, yet it might lead to conservative optimization results since in intersection of two polyhedrons, the inflated drone trajectory set does not necessarily stay within either one. But as will be shown in the experiments part, this does not matter much.

B. Construction of Constraint Violation Function

Since it is troublesome and intractable to directly handle the nonlinear constraints $\mathcal{G}_{att}(\boldsymbol{p}_i(t), \ddot{\boldsymbol{p}}_i(t))$, as suggested by [2], we only consider the potential violation happens at normalized timestamps and construct the constraint violation function $\mathcal{G}_{att} : \mathbb{R}^{2s\times 3} \times \mathbb{R}_+ \times [0, 1] \to \mathbb{R}^{8K_i}$ at the normalized timestamp $\hat{t} \in [0, 1]$ as:

$$\mathcal{G}_{att}^k(\boldsymbol{c}_i, T_i, \hat{t}) = \left[(\boldsymbol{n}_i^k)^{\mathrm{T}}(\boldsymbol{p}_i(\hat{t}\cdot T_i) + \boldsymbol{R}_b^i(\hat{t}\cdot T_i)\tilde{\boldsymbol{q}}_{\boldsymbol{v}} - \boldsymbol{o}_i^k) \right]_{v=1}^8 \in \mathbb{R}^8 \quad (21a)$$

$$\mathcal{G}_{att}(\boldsymbol{c}_i, T_i, \hat{t}) = \left[[\mathcal{G}_{att}^k(\boldsymbol{c}_i, T_i, \hat{t})^{\mathrm{T}}]_{k=1}^{K_i} \right]^{\mathrm{T}} \in \mathbb{R}^{8K_i} \quad (21b)$$

Note that (20) and (21) have same content but different appearance. The reason of writing the function in (21a) form is for easier gradient derivation in later subsection.

Provided with a constant weight vector $\chi \in \mathbb{R}^{8K_i}$, the time integral penalty function over $\boldsymbol{p}_i(t)$, denoted by $I_{att} : \mathbb{R}^{2s\times 3} \times \mathbb{R}_+ \times \mathbb{Z}_{\geq} \to \mathbb{R}_+$ is given by the quadrature of cubic penalty over $[0, T_i]$:

$$I_{att}(\boldsymbol{c}_i, T_i, \kappa_i) = \frac{T_i}{\kappa_i} \sum_{j=1}^{\kappa_i} \omega_j \chi^{\mathrm{T}} \max[\mathcal{G}_{att}(\boldsymbol{c}_i, T_i, \frac{j}{\kappa_i}), \mathbf{0}]^3 \quad (22)$$

where $\max[\cdot, \mathbf{0}]^3$ is a composite function of entry-wise maximum and entry-wise cubic function. The pre-given constant $\frac{1}{\kappa_i}$ is the relative resolution of the quadrature. The constant scalar ω_j is the j-th quadrature coefficient.

The penalty term along the entire trajectory is given by

$$I_{\Sigma att}(\boldsymbol{c}, \boldsymbol{T}) = \sum_{i=1}^{M} I_{att}(\boldsymbol{c}_i, T_i, \kappa_i) \quad (23)$$

and as mentioned in III-B.2. It is added directly to the objective function. Thus the problem defined in (14) has no constraints and any quasi-Newton method can be deployed.

C. Derivation of Derivatives of Attitude Penalty

Now that the function $I_{att}(\boldsymbol{c}_i, T_i, \kappa_i)$ is a part of the overall objective function and $\boldsymbol{c}_i$ and T_i are the variables to be optimized, it is necessary to compute the derivative $\frac{\partial I_{att}}{\partial \boldsymbol{c}_i}$ and $\frac{\partial I_{att}}{\partial T_i}$. $\mathcal{G}_{att}^{k,v}$ is the v-th component of vector $\mathcal{G}_{att}^{k}$. Due to the limitation on pages, we leave the hardest parts of deriving $\frac{\partial \mathcal{G}_{att}^{k,v}}{\partial \boldsymbol{c}_i}$ and $\frac{\partial \mathcal{G}_{att}^{k,v}}{\partial T_i}$ with fixed $\tilde{\boldsymbol{q}}_{\boldsymbol{v}} \in \tilde{\mathcal{Q}}_{cub}$ in [13].

It is easy to concatenate the derivative of $\mathcal{G}_{att}^{k,v}$ with different k, v together to attain $\frac{\partial I_{\Sigma att}}{\partial \boldsymbol{c}}$ and $\frac{\partial I_{\Sigma att}}{\partial \boldsymbol{T}}$ and further use them in quasi-Newton optimization methods.

D. Unifying the Framework

In earlier sessions, we finish constructing the substitution $I_{\Sigma att}(\boldsymbol{c}, \boldsymbol{T})$ and its derivatives for collision-free nonlinear constraints $\mathcal{G}_{att}$ in problem (14). However, a drone is not able to fly as aggressively as we desire due to the limitations on propeller thrust and difficulties on controllers. This is typically known as the *dynamic feasibility* constraints which can be equivalently transferred to maximum velocity and acceleration bounds in general. Since highly tiltedness of drone is expected in our method, including a jerk bound is also necessary. All these nonlinear constraints $\mathcal{G}_{dyn}$ are approximated by $I_{\Sigma dyn}(\boldsymbol{c}, \boldsymbol{T})$ in a similar way.

We must point out that an approximation has been made by replacing $\mathcal{G}_{att}, \mathcal{G}_{dyn}$ with $I_{\Sigma att}, I_{\Sigma dyn}$. Violation of nonlinear constraints may happen because the approximation has only finite resolution. However, later experiments proves that this does not affect the overall quality of the final trajectory.

After adding the discretized penalty terms $I_{\Sigma att}(\boldsymbol{c}, \boldsymbol{T})$ and $I_{\Sigma dyn}(\boldsymbol{c}, \boldsymbol{T})$ into the objective function (14a) and bypassing temporal and spatial constraints via diffeomorphism based methods, the optimization problem in (14) is purely unconstrained. During the process, we use cddlib [14] to transform polyhedrons between $\mathcal{H}$-representation and $\mathcal{V}$-representation [15]. We use a customized LBFGS [16] optimizer [1] to solve the problem in a quasi-Newton fashion.

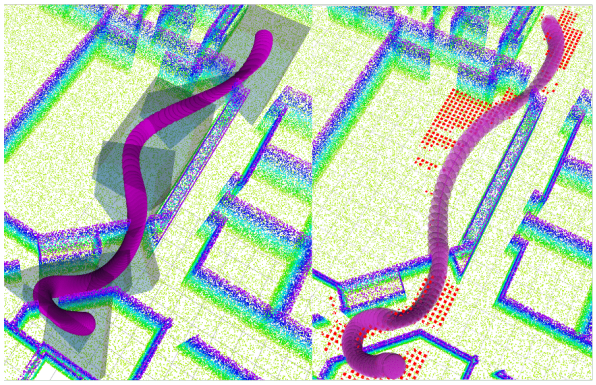
V. Results

A. Benchmark for Full-Body Motion Planning

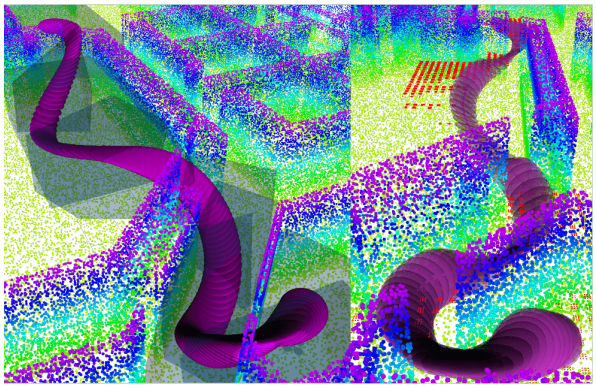
Since there is no available optimization-based full-body motion planning framework, we benchmark our solution with the state-of-the-art search-based method by [11][2]. Both methods accept point cloud, start and goal states as well as dynamic constraints. Theoretically speaking, constant control should be applied in all three dimensions for primitive generation, yet Liu's method [11] suffers from the curse of dimensionality heavily so it only generates motion primitives on x-y plane with fixed z-axis value. Such distinction is straightforward to see in Fig. 4(b). It also needs the user-defined searching range, which is manually adjusted to fit for respective starters and endings in latter benchmarks. Constant control resolution makes no guarantee that the desired final state to be reached *exactly* so a tolerance must be set. Fig. 4(c) exposes this drawback.

[1] https://github.com/ZJU-FAST-Lab/LBFGS-Lite
[2] https://github.com/sikang/mpl_ros

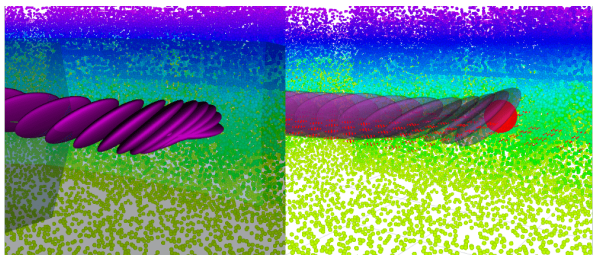
Our method generates series of convex polyhedrons along a given path and solves the unconstrained optimization problem accordingly as described in III and IV. An overall graphical comparison is shown in Fig. 4(a).



(a) Overview Comparison.



(b) 3D v.s. 2D.



(c) Ending Status.

Fig. 4. Left: Our method. Right: Liu's method [11]. Fig. 4(a) shows our smooth solution is contained in polyhedrons. Meanwhile, there are unused expanded states shown as red squares in Liu's work [11]. Fig. 4(b) shows that our solution fully exploits the 3D free spaces while Liu's solution is only limited to fixed height in 2D. Fig. 4(c) shows that drone reaches steady state eventually in our method while Liu's method [11] makes no guarantee about the final state, i.e., it might not be hovering.

As for detailed benchmarking, the same point cloud map is used and drone all have radius $r = 0.5\,\mathrm{m}$, height $h = 0.1\,\mathrm{m}$. Table I summarizes the parameters for both methods. ρ is the time penalizing term. Different ρ are used to achieve the best collision avoidance result in two methods. $\bar{v}$, $\bar{a}$ and $\bar{j}$ are the bounding of respective physical quantity. For Liu's method [11], $\bar{u}$ is the maximum jerk input, du is the stepping of input and τ is the constant input duration for single primitive. All of them are identical as in [11]. For our method, χ_{att} and κ_{att} appear in (22).χ_{att} is the penalty weight. κ_{att} is the relative

resolution or number of samples for penalty per pieces. Due to limited pages, only parameters related to full-body motion planning are listed. All comparisons are conducted under Linux environment on an Intel Core i7-10750H CPU.

TABLE I

1ST ROW: OUR METHOD, 2ND ROW: LIU'S METHOD [11]

ρ	χ_{att}	$\bar{v}$	$\bar{a}$	$\bar{j}$	κ_{att}
1024	60000	$10\,\mathrm{m/s}$	$10\,\mathrm{m/s^2}$	$60\,\mathrm{m/s^3}$	16
ρ	τ	$\bar{v}$	$\bar{a}$	$\bar{u}$	du
10000	$0.2\,\mathrm{s}$	$10\,\mathrm{m/s}$	$10\,\mathrm{m/s^2}$	$60\,\mathrm{m/s^3}$	$30\,\mathrm{m/s^3}$

The benchmark results are shown in Table II. Bold quantities stands for better physical meaning, i.e., shorter time, less memory or faster speed. N_{poly} is the number of polyhedron corridors, which roughly reflects the flight distance. T_{cpu} is the CPU time for executing respective programs. Memory usage comes from System Monitor. T_{all} is the trajectory execution time. v_{max} and a_{max} are the maximum velocity and acceleration along the trajectory. Our a_{max} may violate the bound a little bit as been explained in IV-D. Our method has demonstrated superior advantages in time consumption and memory usage over Liu's method [11] by *orders of magnitude* while returning trajectories that fully leverage the dynamical feasibility bounds such that highly aggressive maneuvers are possible. This superiority increases as the problem size becomes larger because the LBFGS optimizer usually occupies same amount of memory and consumes linearly increasing time. In Liu's method [11] however, both numbers grow exponentially. Since there is no universal rule of quantifying how successful a certain trajectory is regarding to obstacle avoidance, we do not include such comparisons and leave the readers to see graphically whether the full-body motion planning is of success.

TABLE II

1ST ROW: OUR METHOD, 2ND ROW: LIU'S METHOD [11]

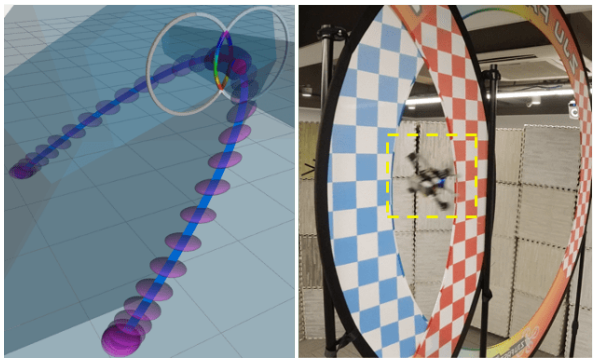
N_{poly}	T_{cpu}	Memory	T_{all}	v_{max}	a_{max}
7	**45.35 ms**	**52.7 MB**	4.41 s	5.67 m/s	**10.02 m/s²**
	2000 ms	194.7 MB	**4.40 s**	**6.46 m/s**	8.49 m/s²
10	**62.58 ms**	**52.6 MB**	**5.78 s**	**7.59 m/s**	**10.02 m/s²**
	1774 ms	173.0 MB	5.80 s	7.23 m/s	8.49 m/s²
16	**89.39 ms**	**52.6 MB**	**8.81 s**	**8.96 m/s**	**10.03 m/s²**
	38 667 ms	986.5 MB	9.60 s	8.65 m/s	8.49 m/s²
20	**64.01 ms**	**52.7 MB**	**10.07 s**	**8.98 m/s**	**10.25 m/s²**
	119 912 ms	3276.8 MB	10.80 s	8.10 m/s	8.49 m/s²

B. Aggressive Flight Experiments

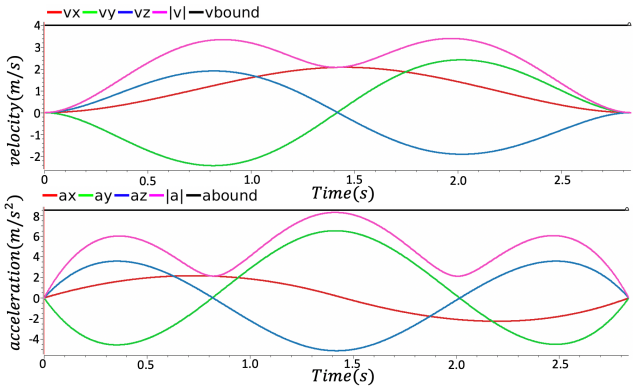
Real world experiments have been conducted as executablity proof for the generated trajectory on drones. We self-assemble a lightweight platform with diameter $214\,\mathrm{mm}$, height $62\,\mathrm{mm}$ and weight $190\,\mathrm{g}$ to demonstrate high aggressiveness. External VICON motion capture system is used to obtain position and orientation of the drone. Convex polyhedron series that form the corridor are generated in advance. PixRacer flight controller is deployed onboard. We set up a ground station receiving the attitude information from VICON, generating trajectory in milliseconds and sending the desired command to the PixRacer via Wi-Fi.

The experiment is about crossing a narrow gap with largest width $l = 170\,\mathrm{mm}$ as shown in Fig. 5. We set $\bar{v} = 4.0\,\mathrm{m/s}$, $\bar{a} = 8.5\,\mathrm{m/s^2}$, $\rho = 1024$. More details can be found in the attached video.

(a) Sequential Instants.



(b) Left: Trajectory Generated; Right: Crossing Instant.



(c) Velocity and Acceleration along Time.

Fig. 5. Details of aggressive gap crossing. Fig. 5(a) and Fig. 5(b) shows the environment setup and drone status. Fig. 5(c) provides velocity and acceleration profiles. The trajectory fully exploits the dynamic feasibility constraints with no violation.

VI. CONCLUSION

In this work, we build on our previous geometrically constrained motion planning framework to achieve online full-body optimization-based trajectory generation within series of polyhedrons. It is thus theoretically possible to integrate the entire pipeline – generation of polyhedron from depth sensor [4], optimization in full-body fashion with onboard computer (this work) and feedback with state estimation results [17] on flight controller – to a uniformed compact platform. This will be our future work. Meanwhile, we are also working on optimization under non-convex or manifold-typed geometric constraints while preserving the advantage of little resource usage.

REFERENCES

[1] J. Canny, B. R. Donald, J. Reif, and P. G. Xavier, "On the complexity of kinodynamic planning," Cornell University, Tech. Rep., 1988.

[2] Z. Wang, S. Yang, C. Xu, and F. Gao. Geometrically constrained trajectory optimization for multicopters. [Online]. Available: https://zhepeiwang.github.io/pubs/gctofm.pdf

[3] 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*, 2020.

[4] X. Zhong, Y. Wu, D. Wang, Q. Wang, C. Xu, and F. Gao, "Generating large convex polytopes directly on point clouds," *arXiv preprint arXiv:2010.08744*, 2020.

[5] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[6] D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza, "The foldable drone: A morphing quadrotor that can squeeze and fly," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 209–216, 2018.

[7] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 404–411, 2016.

[8] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 5774–5781.

[9] J. Lin, L. Wang, F. Gao, S. Shen, and F. Zhang, "Flying through a narrow gap using neural network: an end-to-end planning and control approach," *arXiv preprint arXiv:1903.09088*, 2019.

[10] M. Watterson, S. Liu, K. Sun, T. Smith, and V. Kumar, "Trajectory optimization on manifolds with applications to quadrotor systems," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 303–320, 2020.

[11] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se (3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, 2018.

[12] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 2520–2525.

[13] S. Yang, B. He, C. Xu, and F. Gao. Detailed derivations of whole-body motion planning for micro aerial vehicles. [Online]. Available: https://syangav.github.io/publication/ICRA2021_addon

[14] K. Fukuda, "Cddlib reference manual," *Report version 093a, McGill University, Montréal, Quebec, Canada*, 2003.

[15] K. Fukuda *et al.*, "Frequently asked questions in polyhedral computation."

[16] 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.

[17] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.