EGO-Planner: An ESDF-Free Gradient-Based Local Planner for Quadrotors

Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao

***Abstract*—Gradient-based planners are widely used for quadrotor local planning, in which a Euclidean Signed Distance Field (ESDF) is crucial for evaluating gradient magnitude and direction. Nevertheless, computing such a field has much redundancy since the trajectory optimization procedure only covers a very limited subspace of the ESDF updating range. In this letter, an ESDF-free gradient-based planning framework is proposed, which significantly reduces computation time. The main improvement is that the collision term in penalty function is formulated by comparing the colliding trajectory with a collision-free guiding path. The resulting obstacle information will be stored only if the trajectory hits new obstacles, making the planner only extract necessary obstacle information. Then, we lengthen the time allocation if dynamical feasibility is violated. An anisotropic curve fitting algorithm is introduced to adjust higher order derivatives of the trajectory while maintaining the original shape. Benchmark comparisons and real-world experiments verify its robustness and high-performance. The source code is released as ros packages.**

***Index Terms*—Motion and path planning, autonomous vehicle navigation, aerial systems, applications.**

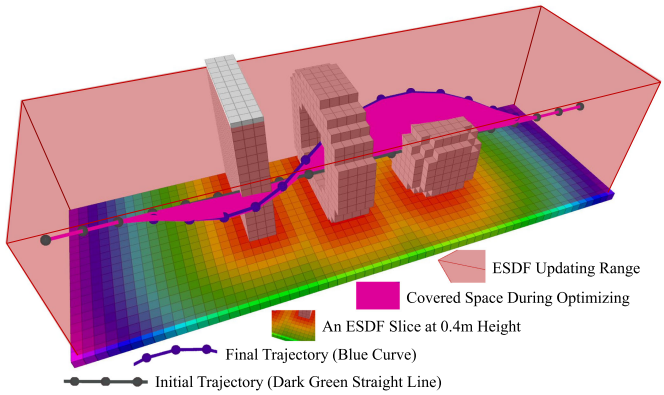


Fig. 1. Trajectory during optimizing just covers a very limited space of the ESDF updating range.

I. Introduction

In RECENT years, the emergence of quadrotor online planning methods has greatly pushed the boundary of aerial autonomy, making drones fly out of laboratories and appear in numerous real-world applications. Among these methods, gradient-based ones, which smooth a trajectory and utilize the gradient information to improve its clearance, have shown great potential and gain more and more popularity [1].

Traditionally, gradient-based planners rely on a pre-built ESDF map to evaluate the gradient magnitude and direction, and use numerical optimization to generate a local optimal solution. Although the optimization programs enjoy fast convergence, they suffer a lot from constructing the required ESDF beforehand. As the statistics (EWOK [2]'s Table II) states, the ESDF computation takes up to about 70% of total processing time for conducting local planning. Therefore, we can safely claim that, building ESDF has become the bottleneck of gradient-based planners, preventing the method from being applied to resource-limited platforms.

Though ESDF is widely used, few works analyze its necessity. Typically, there are two ways to build an ESDF. As detailed in Section II, methods can be categorized as the incremental global updating [3] ones, and the batch local calculation [4] ones. However, neither of them focuses on the trajectory itself. Consequently, too much computation is spent on calculating ESDF values that make no contribution to the planning. In other words, current ESDF-based methods do not serve the trajectory optimization solely and directly. As shown in Fig. 1, for a general autonomous navigation scenario where the drone is expected to avoid collisions locally, the trajectory covers only a limited space of the ESDF updating range. In practice, although some hand-crafted rules can decide a slim ESDF range, they lack theoretical rationality and still induce unnecessary computations.

In this letter, we design an **E**SDF-free **G**radient-based l**O**cal planning framework called **EGO**, and we incorporate careful engineering considerations to make it lightweight and robust. The proposed algorithm is composed of a gradient-based spline optimizer and a post-refinement procedure. Firstly, we optimize the trajectory with smoothness, collision, and dynamical feasibility terms. Unlike traditional approaches that query pre-computed ESDF, we model the collision cost by comparing the trajectory inside obstacles with a guiding collision-free path. We then project the forces onto the colliding trajectory and generate estimated gradient to wrap the trajectory out of obstacles. During the optimization, the trajectory will rebound a few times between nearby obstacles and finally terminate in a safe region. In this

Manuscript received August 19, 2020; accepted October 21, 2020. Date of publication December 28, 2020; date of current version January 12, 2021. This letter was recommended for publication by Associate Editor L. Tapia and Editor N. Amato upon evaluation of the reviewers' comments. This work was supported by the Fundamental Research Funds for the Central Universities under Grant 2020QNA5013. *(Corresponding author: Fei Gao.)*

The authors are with the State Key Laboratory of Industrial Control Technology, and the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: iszhouxin@zju.edu.cn; wangzhepei@zju.edu.cn; hkye@zju.edu.cn; cxu@zju.edu.cn; fgaoaa@zju.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/LRA.2020.3047728.

Digital Object Identifier 10.1109/LRA.2020.3047728

way, we only calculate the gradient when necessary, and avoid computing ESDF in regions irrelevant to the local trajectory. If the resulted trajectory violates dynamical limits, which is usually caused by unreasonable time allocation, the refinement process is activated. During the refinement, trajectory time is reallocated when the limits are exceeded. With the enlarged time allocation, a new B-spline that fits the previous dynamical infeasible one while balancing the feasibility and fitting accuracy is generated. To improve robustness, the fitting accuracy is modeled anisotropically with different penalties on axial and radial directions.

To the best knowledge of us, this method is the first to achieve gradient-based local planning without an ESDF. Compared to existing state-of-the-art works, the proposed method generates safe trajectories with comparable smoothness and aggressiveness, but lower computation time of over an order of magnitude by omitting the ESDF maintenance. We perform comprehensive tests in simulation and real-world to validate our method. Contributions of this letter are:

1) We propose a novel and robust gradient-based quadrotor local planning method, which evaluates and projects gradient information directly from obstacles instead of a pre-built ESDF.
2) We propose a lightweight yet effective trajectory refinement algorithm, which generates smoother trajectories by formulating the trajectory fitting problem with anisotropic error penalization.
3) We integrate the proposed method into a fully autonomous quadrotor system, and release our software for the reference of the community.[1]

II. Related Work

A. Gradient-Based Motion Planning

Gradient-based motion planning is the mainstream for UAV local trajectory generation, which formulates the problem as unconstrained nonlinear optimization. ESDF is first introduced in robotic motion planning by Ratliff *et al.* [5]. Utilizing its abundant gradient information, many planning frameworks directly optimize trajectories in the configuration space. Nevertheless, optimizing the trajectory in discrete-time [5], [6] is not suitable for drones, because it is much more sensitive to dynamical constraints. Thereby, [7] proposes a continuous-time polynomial trajectory optimization method for UAV planning. However, the involved integral of the potential function causes a heavy computation burden. Besides, the success rate of this method is around 70%, even with random restarts. For these drawbacks, [2] introduces a B-spline parameterization of the trajectory which takes good advantage of the convex hull property. In [8], the success rate is significantly increased by finding a collision-free initial path as the front-end. Moreover, the performance is further improved when the generation of the initial collision-free path takes into account kinodynamic constraints [9], [10]. Zhou *et al.* [11] incorporate perception awareness to make the system more robust. Among the above approaches, ESDF plays a vital role in evaluating distance with gradient magnitude and direction to nearby obstacles.

[1][Online]. Available: https://github.com/ZJU-FAST-Lab/ego-planner

Algorithm 1: CheckAndAddObstacleInfo.

Notation: Environment $\mathcal{E}$, Control Points Struct $\mathbf{Q}$, Anchor Points $\mathbf{p}$, Repulsive Direction Vector $\mathbf{v}$, Colliding Segments $\mathbf{S}$
Input: $\mathcal{E}$, $\mathbf{Q}$
for $\mathbf{Q}_i$ in $\mathbf{Q}$ **do**
  if FindConsecutiveCollidingSegment($\mathbf{Q}_i$) **then**
    $\mathbf{S}$.push_back(GetCollisionSegment())
  end if
end for
for $\mathbf{S}_i$ in $\mathbf{S}$ **do**
  $\mathbf{\Gamma} \leftarrow$ PathSearch($\mathcal{E}$, $\mathbf{S_i}$)
  for $\mathbf{S}_i$.begin $\leq j \leq$ $\mathbf{S}_i$.end **do**
    $\{\mathbf{p}, \mathbf{v}\} \leftarrow$ Find_p_v_Pairs($\mathbf{Q}_j, \mathbf{\Gamma}$)
    $\mathbf{Q}_j$.push_back($\{\mathbf{p}, \mathbf{v}\}$)
  end for
end for

B. Euclidean Signed Distance Field (ESDF)

ESDF has long been used to construct objects from noisy sensor data for over two decades [12], and revive interests in robotics motion planning since [5]. Felzenszwalb *et al.* [4] propose an envelope algorithm that reduces the time complexity of ESDF construction to $O(n)$ with n denoted as voxel numbers. This algorithm is not suitable for incremental building of ESDF, while dynamic updating of the field is often needed during quadrotor flight. To solve this problem, Oleynikova [13] and Han [3] propose incremental ESDF generation methods, namely *Voxblox* and *FIESTA*. Although these methods are highly efficient in dynamic updating cases, the generated ESDF almost always contains redundant information that may not be used in the planning procedure at all. As is shown in Fig. 1, this trajectory only sweeps over a very limited subspace of the whole ESDF updating range. Therefore, it is valuable to design a more intelligent and lightweight method, instead of maintaining the whole field.

III. Collision Avoidance Force Estimation

In this letter, the decision variables are control points $\mathbf{Q}$ of a B-spline curve. Each $\mathbf{Q}$ possesses its own environment information independently. Initially, a naive B-spline curve $\mathbf{\Phi}$ satisfying terminal constraints is given, regardless of collision. Then, the optimization procedure starts. For each colliding segment detected in an iteration, a collision-free path $\mathbf{\Gamma}$ is generated. Each control point $\mathbf{Q}_i$ of the colliding segment, after that, will be assigned an anchor point $\mathbf{p}_{ij}$ at the obstacle surface with a corresponding repulsive direction vector $\mathbf{v}_{ij}$, as shown in Fig. 3(a). Denote by $i \in \mathbb{N}_+$ the index of control points, and $j \in \mathbb{N}$ the index of $\{\mathbf{p}, \mathbf{v}\}$ pair. Note that each $\{\mathbf{p}, \mathbf{v}\}$ pair only belongs to one specific control point. For brevity, we omit the subscript ij without causing ambiguity. The detailed $\{\mathbf{p}, \mathbf{v}\}$ pair generation procedure in this letter is summarized in Algorithm 1 and is illustrated in Fig. 3(b). Then the obstacle distance from $\mathbf{Q}_i$ to the j^{th} obstacle is defined as

$$d_{ij} = (\mathbf{Q}_i - \mathbf{p}_{ij}) \cdot \mathbf{v}_{ij}. \tag{1}$$

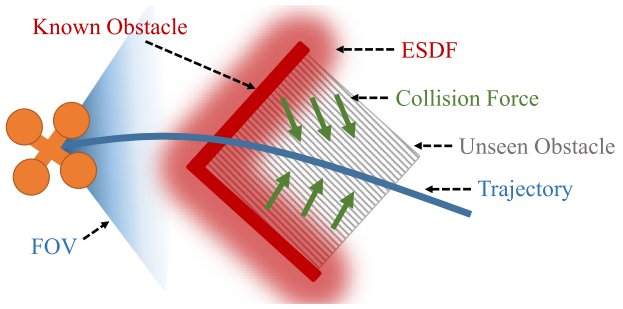


Fig. 2. The trajectory gets stuck into a local minimum, which is very common since the camera has no vision of the back of the obstacle.

In order to avoid duplicative $\{\mathbf{p}, \mathbf{v}\}$ pair generation before the trajectory escapes from the current obstacle during the first several iterations, we adopt a criterion that considers an obstacle which the control point $\mathbf{Q}_i$ lies in as newly discovered, only if the current $\mathbf{Q}_i$ satisfies $d_{ij} > 0$ for all valid j. Besides, this criterion allows only necessary obstacles that contribute to the final trajectory to be taken into optimization. Thus, the operation time is significantly reduced.

To incorporate necessary environmental awareness into the local planner, we need to explicitly construct an objective function that keeps the trajectory away from obstacles. ESDF provides this vital collision information but with the price of a heavy computation burden. In addition, as shown in Fig. 2, ESDF-based planners can easily fall into a local minimum and fail to escape from obstacles, due to the insufficient or even wrong information from ESDF. To avoid such situations, an additional front-end is always needed to provide a collision-free initial trajectory. The above methodology outperforms ESDF in providing the vital information for collision avoidance, since the explicitly designed repulsive force can be fairly effective regarding various missions and environments. Moreover, the proposed method has no requirement for collision-free initialization.

IV. Gradient-Based Trajectory Optimization

A. Problem Formulation

In this letter, the trajectory is parameterized by a uniform B-spline curve $\mathbf{\Phi}$, which is uniquely determined by its degree p_b, N_c control points $\{\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_{N_c}\}$, and a knot vector $\{t_1, t_2, \ldots, t_M\}$, where $\mathbf{Q}_i \in \mathbb{R}^3$, $t_m \in \mathbb{R}$ and $M = N_c + p_b$. For simplicity and efficiency of trajectory evaluation, the B-spline used in our method is uniform, which means each knot is separated by the same time interval $\triangle t = t_{m+1} - t_m$ from its predecessor. The problem formulation in this letter is based on the current state-of-the-art quadrotor local planning framework Fast-Planner [14].

B-spline enjoys convex hull property, as shown in Fig. 4. This property indicates that a single span of a B-spline curve is merely controlled by $p_b + 1$ successive control points and lies within the convex hull of these points. For example, a span within (t_i, t_{i+1}) lies inside the convex hull formed by $\{\mathbf{Q}_{i-p_b}, \mathbf{Q}_{i-p_b+1}, \ldots, \mathbf{Q}_i\}$. Another property is that the k^{th} derivative of a B-spline is still a B-spline with order $p_{b,k} = p_b - k$. Since $\triangle t$ is identical alone $\mathbf{\Phi}$, the control points of the velocity $\mathbf{V}_i$, acceleration $\mathbf{A}_i$, and jerk $\mathbf{J}_i$ curves are obtained by

$$\mathbf{V}_i = \frac{\mathbf{Q}_{i+1} - \mathbf{Q}_i}{\triangle t}, \ \mathbf{A}_i = \frac{\mathbf{V}_{i+1} - \mathbf{V}_i}{\triangle t}, \ \mathbf{J}_i = \frac{\mathbf{A}_{i+1} - \mathbf{A}_i}{\triangle t}. \tag{2}$$

We follow the work of [15] to plan the control points $\mathbf{Q} \in \mathbb{R}^3$ in a reduced space of *differentially flat outputs*. The optimization problem is then formulated as follows:

$$\min_{\mathbf{Q}} J = \lambda_s J_s + \lambda_c J_c + \lambda_d J_d, \tag{3}$$

where J_s is the smoothness penalty, J_c is for collision, and J_d indicates feasibility. $\lambda_s, \lambda_c, \lambda_d$ are weights for each penalty terms.

1) Smoothness Penalty: In [2], the smoothness penalty is formulized as the time integral over square derivatives of the trajectory (acceleration, jerk, etc.). In [10], only geometric information of the trajectory is taken regardless of time allocation. In this letter, we combine both methods to penalize squared acceleration and jerk without time integration.

Benefiting from the convex hull property, minimizing the control points of second and third order derivatives of the B-spline trajectory is sufficient to reduce these derivatives along the whole curve. Therefore, the smoothness penalty function is formulated as

$$J_s = \sum_{i=1}^{N_c-1} \|\mathbf{A}_i\|_2^2 + \sum_{i=1}^{N_c-2} \|\mathbf{J}_i\|_2^2, \tag{4}$$

which minimizes high order derivatives, making the whole trajectory smooth.

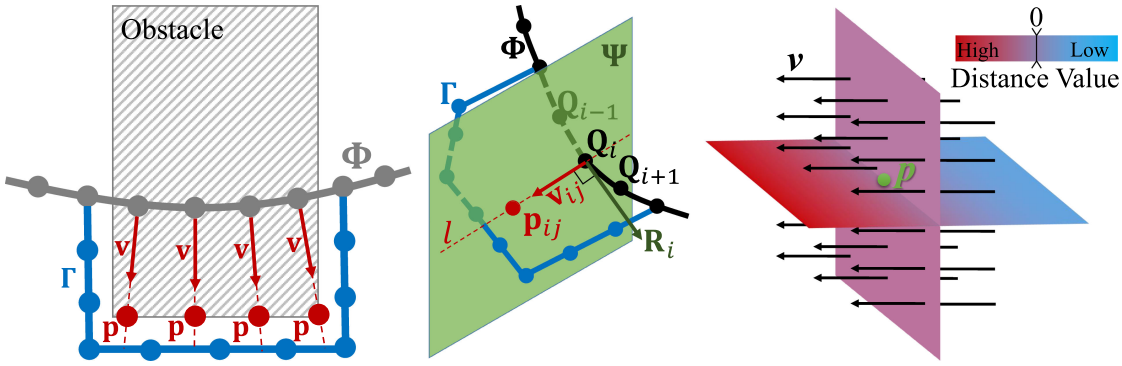
2) Collision Penalty: Collision penalty pushes control points away from obstacles. This is achieved by adopting a safety clearance s_f and punishing control points with $d_{ij} < s_f$. In order to further facilitate optimization, we construct a twice continuously differentiable penalty function j_c and suppress its slope as d_{ij} decreases, which yields the piecewise function

$$j_c(i,j) = \begin{cases} 0 & (c_{ij} \le 0) \\ c_{ij}^3 & (0 < c_{ij} \le s_f) \\ 3s_f c_{ij}^2 - 3s_f^2 c_{ij} + s_f^3 & (c_{ij} > s_f) \end{cases}, \\ c_{ij} = s_f - d_{ij}, \tag{5}$$

where $j_c(i,j)$ is the cost value produced by $\{\mathbf{p}, \mathbf{v}\}_j$ pairs on $\mathbf{Q}_i$. The cost on each $\mathbf{Q}_i$ is evaluated independently and accumulated from all corresponding $\{\mathbf{p}, \mathbf{v}\}_j$ pairs. Thus, a control point obtains a higher trajectory deformation weight if it discovers more obstacles. Specifically, the cost value added to the i^{th} control point is $j_c(\mathbf{Q}_i) = \sum_{j=1}^{N_p} j_c(i,j)$, N_p is the number of $\{\mathbf{p}, \mathbf{v}\}_j$ pairs belonging to $\mathbf{Q}_i$. Combining costs on all $\mathbf{Q}_i$ yields the total cost J_c, i.e.,

$$J_c = \sum_{i=1}^{N_c} j_c(\mathbf{Q}_i). \tag{6}$$

Unlike traditional ESDF-based methods [2], [10], which compute gradient by trilinear interpolation on the field, we obtain gradient by directly computing the derivative of J_c with respect



(a) $\{\mathbf{p}, \mathbf{v}\}$ Pairs (b) $\mathbf{p}, \mathbf{v}$ Generation (c) Distance Field of A $\{\mathbf{p}, \mathbf{v}\}$ Pair

Fig. 3. (a) A trajectory $\mathbf{\Phi}$ passing through an obstacle generates several $\{\mathbf{p}, \mathbf{v}\}$ pairs for control points. $\mathbf{p}$ are the points at the obstacle surface and $\mathbf{v}$ are unit vectors pointing from control points to $\mathbf{p}$. (b) A plane $\mathbf{\Psi}$ which is perpendicular to a tangent vector $\mathbf{R}_i$ intersects $\mathbf{\Gamma}$ forming a line l, from which a $\{\mathbf{p}, \mathbf{v}\}$ pair is determined. (c) Slice visualization of distance field definition $d_{ij} = (\mathbf{Q}_i - \mathbf{p}_{ij}) \cdot \mathbf{v}_{ij}$. The color indicates the distance and the arrows are identical gradients equal to $\mathbf{v}$. $\mathbf{p}$ is at the zero distance plane.

to $\mathbf{Q}_i$, which gives

$$\frac{\partial J_c}{\partial \mathbf{Q}_i} = \sum_{i=1}^{N_c} \sum_{j=1}^{N_p} \mathbf{v}_{ij} \begin{cases} 0 & (c_{ij} \leq 0) \\ -3c_{ij}^2 & (0 < c_{ij} \leq s_f) \\ -6s_f c_{ij} + 3s_f^2 & (c_{ij} > s_f) \end{cases}. \quad (7)$$

3) Feasibility Penalty: Feasibility is ensured by restricting the higher order derivatives of the trajectory on every single dimension, i.e., applying $|\mathbf{\Phi}_r^{(k)}(t)| < \mathbf{\Phi}_{r,\max}^{(k)}$ for all t, where $r \in \{x, y, z\}$ indicates each dimension. Thanks to the convex hull property, constraining derivatives of the control points is sufficient for constraining the whole B-spline. Therefore, the penalty function is formulated as

$$J_d = \sum_{i=1}^{N_c} w_v F(\mathbf{V}_i) + \sum_{i=1}^{N_c-1} w_a F(\mathbf{A}_i) + \sum_{i=1}^{N_c-2} w_j F(\mathbf{J}_i), \quad (8)$$

where w_v, w_a, w_j are weights for each terms and $F(\cdot)$ is a twice continuously differentiable metric function of higher order derivatives of control points.

$$F(\mathbf{C}) = \sum_{r=x,y,z} f(c_r), \quad (9)$$

$$f(c_r) = \begin{cases} a_1 c_r^2 + b_1 c_r + c_1 & (c_r \leq -c_j) \\ (-\lambda c_m - c_r)^3 & (-c_j < c_r < -\lambda c_m) \\ 0 & (-\lambda c_m \leq c_r \leq \lambda c_m) \\ (c_r - \lambda c_m)^3 & (\lambda c_m < c_r < c_j) \\ a_2 c_r^2 + b_2 c_r + c_2 & (c_r \geq c_j) \end{cases}, \quad (10)$$

where $c_r \in \mathbf{C} \in \{\mathbf{V}_i, \mathbf{A}_i, \mathbf{J}_i\}$, $a_1, b_1, c_1, a_2, b_2, c_2$ are chosen to meet the second-order continuity, c_m is the derivative limit, c_j is the splitting points of the quadratic interval and the cubic interval. $\lambda < 1 - \epsilon$ is an elastic coefficient with $0 < \epsilon \ll 1$ to make the final results meet the constraints, since the cost function is a tradeoff of all weighted terms.

B. Numerical Optimization

The formulated problem in this letter features in two aspects. Firstly, the objective function J alters adaptively according to the newly found obstacles. It requires the solver to be able to restart fast. Secondly, quadratic terms dominate the formulation of the objective function, making J approximate quadratic. It means that the utilization of Hessian information can significantly accelerate the convergence. However, obtaining the exact inverse Hessian is prohibitive in real-time applications since it consumes nonnegligible massive computation. To circumvent this, quasi-Newton methods that approximate the inverse Hessian from gradient information are adopted.

Since the performance of a solver is problem dependent, we compare three algorithms belonging to quasi-Newton methods. They are Barzilai-Borwein method [16] which is capable of fast restart with most crude Hessian estimation, truncated Newton method [17] which estimates Hessian by adding multiple tiny perturbations to a given state, L-BFGS method [18] which approximates Hessian from previous objective function evaluations but requires a serial of iterations to reach a relatively accurate estimation. Comparison in Section VI-B states that L-BFGS outperforms the other two algorithms with appropriately selected memory size, balancing the loss of restart and the accuracy of inverse Hessian estimation. This algorithm is briefly explained as follows. For an unconstrained optimization problem $\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$, the updating for $\mathbf{x}$ follows the approximated Newton step

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{H}_k \nabla \mathbf{f}_k, \quad (11)$$

where α_k is the step length and $\mathbf{H}_k$ is updated at every iteration by means of the formula

$$\mathbf{H}_{k+1} = \mathbf{V}_k^T \mathbf{H}_k \mathbf{V}_k + \rho_k \mathbf{s}_k \mathbf{s}_k^T, \quad (12)$$

where $\rho_k = (\mathbf{y}_k^T \mathbf{s}_k)^{-1}, \mathbf{V}_k = \mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T, \mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \nabla \mathbf{f}_{k+1} - \nabla \mathbf{f}_k$.

Here $\mathbf{H}_k$ is not calculated explicitly. The algorithm right multiplies $\nabla \mathbf{f}_k$ to Equ. (12) and recursively expands for m steps and then yields the efficient two-loop recursion updating method [16], resulting in linear time/space complexity. The weight of Barzilai-Borwein step is used as the initial inverse

Hessian $\mathbf{H}_k^0$ for L-BFGS updating, which is

$$\mathbf{H}_k^0 = \frac{\mathbf{s}_{k-1}^T \mathbf{y}_{k-1}}{\mathbf{y}_{k-1}^T \mathbf{y}_{k-1}} \mathbf{I} \text{ or } \frac{\mathbf{s}_{k-1}^T \mathbf{s}_{k-1}}{\mathbf{s}_{k-1}^T \mathbf{y}_{k-1}} \mathbf{I}. \tag{13}$$

A monotone line search under strong Wolfe condition is used to enforce convergence.

V. Time Re-Allocation and Trajectory Refinement

Allocating an accurate time profile before the optimization is unreasonable, since the planner knows no information about the final trajectory then. Therefore, an additional time re-allocation procedure is vital to ensure dynamical feasibility. Previous works [10], [19] parameterize the trajectory as a non-uniform B-spline and iteratively lengthen a subset of knot spans when some segments exceed derivative limits.

However, one knot span $\triangle t_n$ influences multiple control points and vice versa, leading to high-order discontinuity to the previous trajectory when adjusting knot spans near the start state. In this section, a uniform B-spline trajectory $\mathbf{\Phi}_f$ is re-generated with reasonable time re-allocation according to the safe trajectory $\mathbf{\Phi}_s$ from IV. Then, an anisotropic curve fitting method is proposed to make $\mathbf{\Phi}_f$ freely optimize its control points to meet higher order derivative constraints while maintaining a nearly identical shape to $\mathbf{\Phi}_s$.

Firstly, as Fast-Planner [14] does, we compute the limits exceeding ratio,

$$r_e = max\{|\mathbf{V}_{i,r}/v_m|, \sqrt{|\mathbf{A}_{j,r}/a_m|}, \sqrt[3]{|\mathbf{J}_{k,r}/j_m|}, 1\}, \tag{14}$$

where $i \in \{1, \ldots, N_c - 1\}$, $j \in \{1, \ldots, N_c - 2\}$, $k \in \{1, \ldots, N_c - 3\}$ and $r \in \{x, y, z\}$ axis. A notion with subscript m represents the limitation of a derivative. r_e indicates how much we should lengthen the time allocation for $\mathbf{\Phi}_f$ relative to $\mathbf{\Phi}_s$. Note that $\mathbf{V}_i$, $\mathbf{A}_j$ and $\mathbf{J}_k$ are inversely proportional to $\triangle t$, the square of $\triangle t$ and the cubic of $\triangle t$, respectively, from Equ.2. Then we obtain the new time span of $\mathbf{\Phi}_f$

$$\triangle t' = r_e \triangle t. \tag{15}$$

$\mathbf{\Phi}_f$ of time span $\triangle t'$ is initially generated under boundary constraints while maintaining the identical shape and control points number to $\mathbf{\Phi}_s$, by solving a closed-form min-least square problem. The smoothness and feasibility are then refined by optimization. The penalty function J' formulated by linear combinations of smoothness (Section IV-A1), feasibility (Section IV-A3) and curve fitting (introduced later) is

$$\min_{\mathbf{Q}} J' = \lambda_s J_s + \lambda_d J_d + \lambda_f J_f, \tag{16}$$

where λ_f is the weight of fitness term.

The fitting penalty function J_f is formulated as the integral of anisotropic displacements from points $\mathbf{\Phi}_f(\alpha T')$ to the corresponding $\mathbf{\Phi}_s(\alpha T)$, where T and T' are the trajectory duration of $\mathbf{\Phi}_s$ and $\mathbf{\Phi}_f$, $\alpha \in [0, 1]$. Since the fitted curve $\mathbf{\Phi}_s$ is already collision-free, we assign the axial displacement of two curves with low penalty weight to relax smoothness adjustment restriction, and radial displacement with high penalty weight to avoid collision. To achieve this, we use the spheroidal metric, shown in Fig. 5, such that displacements at the same spheroid surface produce identical penalties. The spheroid we use for $\mathbf{\Phi}_f(\alpha T')$

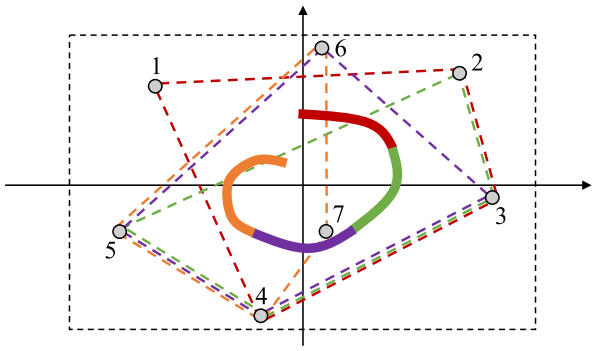


Fig. 4. Convex hull property of the B-spline curve. Gray points represent control points. The whole curve stays inside the feasibility bounding box(black dotted box) as long as all the control points are in that box. Without loss of generality, each convex hull consists of four vertexes.

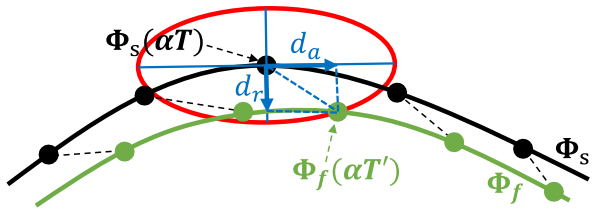


Fig. 5. Optimizing trajectory $\mathbf{\Phi}_f$ to fit trajectory $\mathbf{\Phi}_s$ while adjusting smoothness and feasibility. Black and green dots are sample points on the trajectory. The displacement between $\mathbf{\Phi}_f(\alpha T')$ and $\mathbf{\Phi}_s(\alpha T)$ breaks down into d_a and d_r along two ellipse principal axes. Points at the red ellipse surface produce identical penalties.

is obtained by rotating an ellipse centering at $\mathbf{\Phi}_s(\alpha T)$ about one of its principal axes, the tangent line $\dot{\mathbf{\Phi}}_s(\alpha T)$. So the axial displacement d_a and radial displacement d_r can be calculated by

$$\begin{aligned} d_a &= (\mathbf{\Phi}_f - \mathbf{\Phi}_s) \cdot \frac{\dot{\mathbf{\Phi}}_s}{\|\dot{\mathbf{\Phi}}_s\|}, \\ d_r &= \left\| (\mathbf{\Phi}_f - \mathbf{\Phi}_s) \times \frac{\dot{\mathbf{\Phi}}_s}{\|\dot{\mathbf{\Phi}}_s\|} \right\|. \end{aligned} \tag{17}$$

The fitness penalty function is

$$J_f = \int_0^1 \left[\frac{d_a(\alpha T')^2}{a^2} + \frac{d_r(\alpha T')^2}{b^2} \right] \mathrm{d}\alpha, \tag{18}$$

where a and b are semi-major and semi-minor axis of the ellipse, respectively. The problem is solved by L-BFGS.

VI. Experiment Results

A. Implementation Details

The planning framework is summarized in Algorithm 2. We set the B-spline order as $p_b = 3$. The number of control points N_c alters around 25, which is determined by the planning horizon (about 7 m) and the initial distance interval (about 0.3 m) of adjacent points. These are empirical parameters that balance the complexity of the problem with degrees of freedom. The time complexity is $O(N_c)$, since one control point only affects nearby

Algorithm 2: Rebound Planning.

Notation: Goal $\mathcal{G}$, Environment $\mathcal{E}$, Control Point Struct $\mathbf{Q}$, Penalty J, Gradient $\mathbf{G}$
Initialize: $\mathbf{Q} \leftarrow$ FindInit($\mathbf{Q}_{last}$, $\mathcal{G}$)
while $\neg$ IsCollisionFree($\mathcal{E}$, $\mathbf{Q}$) **do**
  CheckAndAddObstacleInfo($\mathcal{E}$, $\mathbf{Q}$)
  $(J, \mathbf{G}) \leftarrow$ EvaluatePenalty($\mathbf{Q}$)
  $\mathbf{Q} \leftarrow$ OneStepOptimize(J, $\mathbf{G}$)
end while
if $\neg$ IsFeasible($\mathbf{Q}$) **then**
  $\mathbf{Q} \leftarrow$ ReAllocateTime($\mathbf{Q}$)
  $\mathbf{Q} \leftarrow$ CurveFittingOptimize($\mathbf{Q}$)
end if
return $\mathbf{Q}$

segments according to the local support property of B-spline. The complexity of L-BFGS is also linear on the same relative tolerance. For collision-free path searching, we adopt A*, which has a good advantage that the path $\mathbf{\Gamma}$ always tends to be close to the obstacle surface naturally. Therefore, we can directly select $\mathbf{p}$ at $\mathbf{\Gamma}$ without obstacle surface searching. For vector $\mathbf{R}_i$ defined in Fig. 3(b), it can be deduced by the property of uniform B-spline parameterization, that the $\mathbf{R}_i$ satisfies

$$\mathbf{R}_i = \frac{\mathbf{Q}_{i+1} - \mathbf{Q}_{i-1}}{2\triangle t}, \tag{19}$$

which can be efficiently computed. Equ.18 is discretized to a finite number of points $\mathbf{\Phi}_f(k\triangle t')$ and $\mathbf{\Phi}_s(k\triangle t)$, where $k \in \mathbb{N}, 0 \leq k \leq \lfloor T/\triangle t \rfloor$. To further enforce safety, a collision check of a circular pipe with a fixed radius around the final trajectory is performed to provide enough obstacle clearance. The optimizer stops when no collision is detected. Real-world experiments are presented on the same flight platform of [19] with depth acquired by Intel RealSense D435.[2] Furthermore, we modify the ROS driver of Intel RealSense to enable the laser emitter strobe every other frame. This allows the device to output high quality depth images with the help of the emitter, and along with binocular images free from laser interference. The modified driver is open-sourced as well.

B. Optimization Algorithms Comparison

In this section, three different optimization algorithms, including Barzilai-Borwein (BB) method, limited-memory BFGS (L-BFGS) and truncated Newton (T-NEWTON) method [17], are discussed. Specifically, each algorithm runs for 100 times independently in random maps. All relevant parameters including boundary constraints, time allocation, decision variables initialization, and random seeds, are set identical for different algorithms. The data about success rate, computation time and numbers of objective function evaluations are recorded. Only the successful cases are counted due to the data in failed cases is meaningless. The associated results are shown in Table I, which states that L-BFGS significantly outperforms the other two algorithms. L-BFGS characterizes a type of approximation by means of second order Taylor expansions, which is suitable for optimizing the objective function described in Section IV-B. Truncated Newton method approximates the second order optimization direction $\mathbf{H}^{-1}\nabla\mathbf{f}_k$ as well. However, too many objective function evaluations increase the optimization time. BB-method estimates the Hessian as a scalar λ times $\mathbf{I}$. Nevertheless, the insufficient estimation of Hessian still leads to a low convergence rate.

[2][Online]. Available: https://www.intelrealsense.com/depth-camera-d435/

TABLE I
OPTIMIZATION ALGORITHMS COMPARISON

Algorithm	Success Rate	Time(ms)			Function Evaluations		
		Min	Avg	Max	Min	Avg	Max
BB	0.86	0.21	0.50	1.14	108	268.2	508
T-NEWTON	0.62	0.3	0.79	3.59	109	344.29	702
L-BFGS	**0.89**	**0.17**	**0.37**	**0.80**	**22**	**79.04**	**182**

TABLE II
ESDF/ESDF-FREE METHODS COMPARISON

Method	Success Rate	Energy	Velocity(m/s)		Time(ms)		
			Avg	Max	Optimize	ESDF	Total
EGO	**0.89**	49.92	2.12	**2.24**	**0.37**	**/**	**0.37**
ENI	0.69	35.55	2.09	2.23	0.43	5.03	5.46
EI	**0.89**	**42.27**	**2.11**	2.36	0.48	5.07	5.55

C. Trajectory Generation With & Without ESDF

We use the same setting as Section VI-B to perform this comparison. On account of the low success rate explained in [14] when using straight line initialization for an ESDF-based trajectory generator, we adopt a collision-free initialization. Comparison results are in Table II.

For clarity, ESDF-based methods with and without collision-free initialization are abbreviated as *EI* and *ENI*. This comparison gives that the proposed EGO algorithm achieves a comparable success rate to ESDF-based methods with collision-free initialization. However, trajectory energy (jerk integral) produced by EGO is slightly higher. This happens because the control points of EGO which contain more than one $\{\mathbf{p}, \mathbf{v}\}$ pair produce stronger trajectory deformation force than EI does, as described in Section IV-A2. On the other hand, stronger force accelerates the convergence procedure, resulting in shorter optimization time. Some statistics of ENI (shown in gray) can be less convincing because ENI tests can only succeed in fewer challenge cases where the resulting trajectories are naturally smoother with less energy cost and lower velocity, compared to EI and EGO. Something noteworthy is that although the ESDF updating size is reduced to $10 \times 4 \times 2$ m^3 with 0.1 m resolution for a 9 m trajectory, the ESDF updating still takes up a majority of the computation time.

D. Multiple Planners Comparison

We compare the proposed planner with two state-of-the-art methods, Fast-Planner [14] and EWOK [2], which utilize ESDF to evaluate obstacle distance and gradient. Each planner runs for ten times of different obstacle densities from the same starts to ends. The average performance statistics and the ESDF computation time are shown in Table III and Fig. 7. Trajectories

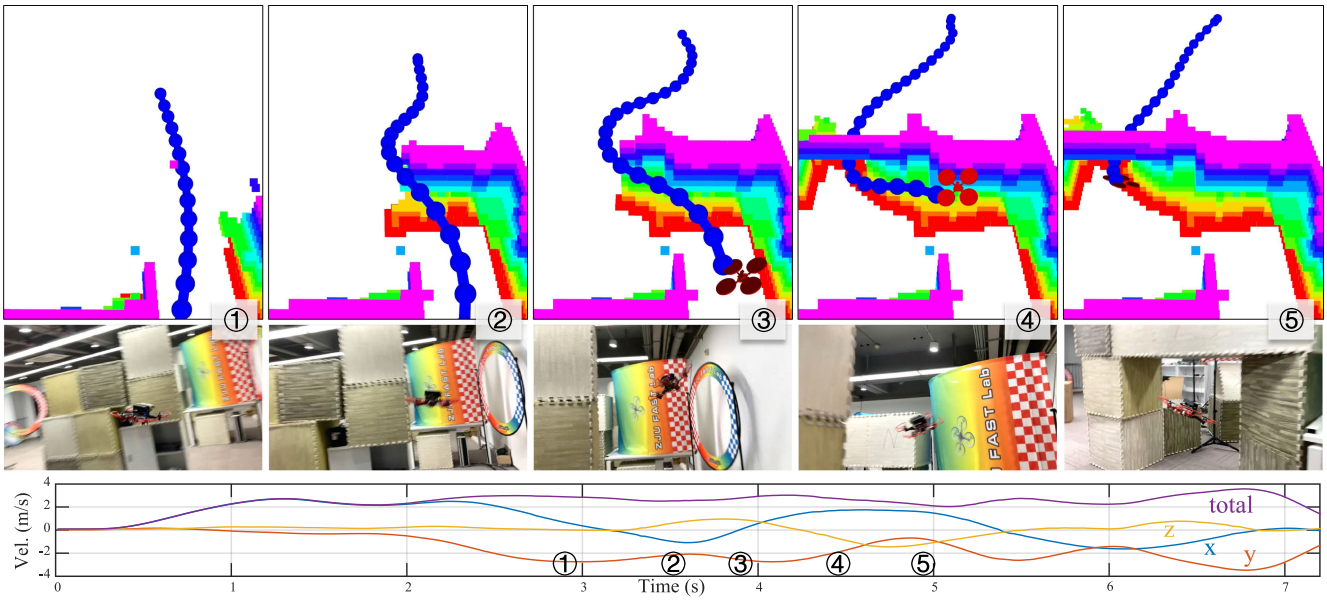


Fig. 6. Visualization of local trajectory planning over a short period of time with velocity profile.

TABLE III
PLANNERS COMPARISON

Planner	t(s)	Length	Energy	t_{ESDF}(ms)	t_{plan}(ms)
EWOK	31.00	59.05	246.12	6.43	1.39
Fast-Planner	30.76	45.18	**135.21**	4.01	3.29
EGO-Planner	**24.38**	**42.24**	196.64	/	**0.81**

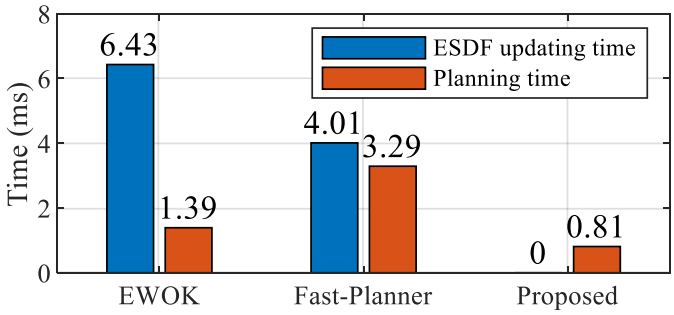


Fig. 7. Comparison of the proposed EGO-Planner against two SOTA planners with default parameters.

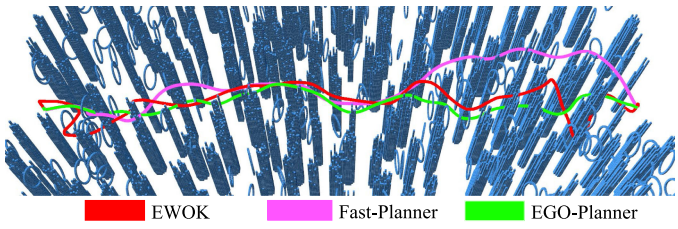


Fig. 8. Trajectory visualization in simulation.

generated by three methods on a map of 0.5 obstacles/m^2 are illustrated in Fig. 8.

From Table III we conclude that the proposed method achieves shorter flight time and trajectory length but ends up in higher energy cost compared to Fast-Planner. This is mainly caused by the front-end kinodynamic path searching in [14]. EWOK suffers twisty trajectories in dense environments, since the objective function contains exponential terms, which leads to unstable convergence in optimization. Furthermore, we conclude that a lot of computation time without ESDF updating is saved by the proposed method.

E. Real-World Experiments

We present several experiments in cluttered unknown environments with limited camera FOV. One experiment is to fly by waypoints given in advance. In this experiment, the drone starts from a small office room, passes through the door, flies around in a big cluttered room, and then returns to the office, as illustrated in Fig. 10(a) and Fig. 11. The narrowest passage of indoor experiments is less than one meter as shown in Fig. 6. By contrast, the drone reaches 3.56 m/s in such a cluttered environment.

Another indoor experiment is to chase goals arbitrarily and abruptly given during the flight, as shown in Fig. 10(c). In this test, limited FOV puts greater challenges that a feasible trajectory must be generated immediately once a new goal is received or collision threat is detected. Thus, this experiment validates that the proposed planner is capable of performing aggressive flight on the premise of feasibility.

In the outdoor experiments, the drone flies through a forest of massive trees and low bushes, as shown in Fig. 10(b) and Fig. 9. Although the wild airflow around the drone causes swinging of the branches and leaves, making the map less reliable, the drone still reaches a speed above 3 m/s. Therefore, the proposed planner can tackle both experimental and field environments. We refer readers to the video[3] for more information.

[3][Online]. Available: https://youtu.be/UKoaGW7t7Dk

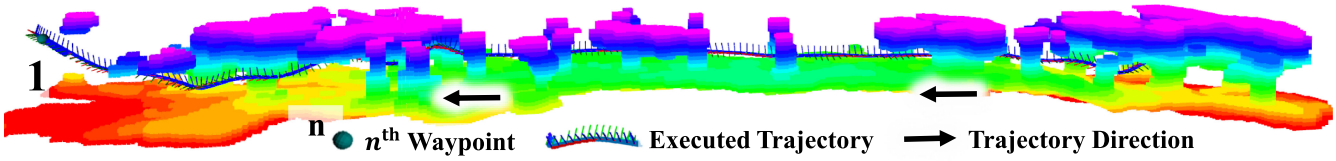


Fig. 9. Trajectory of an outdoor experiment in a forest.

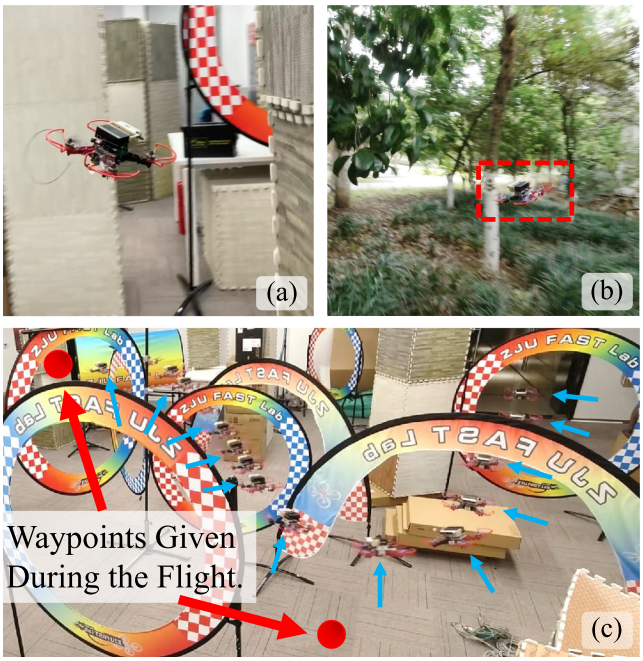


Fig. 10. Real-world experiments. (a) An indoor test. (b) An outdoor test. (c) Composite snapshots of indoor flights.

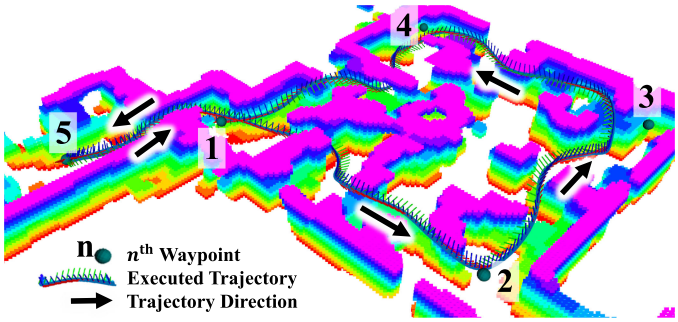


Fig. 11. Trajectory of an indoor experiment.

VII. Conclusion and Future Work

In this letter, we investigate the necessity of ESDF for gradient-based trajectory planning and propose an ESDF-free local planner. It achieves comparable performance to some state-of-the-art ESDF-based planners but reduces computation time for over an order of magnitude. Benchmark comparisons and real-world experiments validate that it is robust and highly efficient.

The proposed method still has some flaws, which are the local minimum introduced by A* search and the conservative trajectories introduced by unified time re-allocation. Therefore, we will work on performing topological planning to escape the local minimum and re-formulating the problem to generate near-optimal trajectories. The planner is designed for static environments and can tackle slowly moving obstacles (below 0.5 m/s) without any modification. We will work on dynamic environment navigation by moving object detection and topological planning in the future.

References

[1] L. Quan, L. Han, B. Zhou, S. Shen, and F. Gao, "Survey of uav motion planning," *IET Cyber-Systems Robot.*, vol. 2, no. 1, pp. 14–21, 2020.
[2] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for mavs using uniform b-splines and a 3d circular buffer," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 215–222.
[3] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4423–4430.
[4] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory Comput.*, vol. 8, no. 1, pp. 415–428, 2012.
[5] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 489–494.
[6] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 4569–4574.
[7] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Daejeon, Korea, Oct. 2016, pp. 5332–5339.
[8] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3681–3688.
[9] W. Ding, W. Gao, K. Wang, and S. Shen, "An efficient b-spline-based kinodynamic replanning framework for quadrotors," *IEEE Trans. Robot.*, vol. 35, no. 6, pp. 1287–1306, Dec. 2019.
[10] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3529–3536, Oct. 2019.
[11] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," 2020, *arXiv:2007.03465*.
[12] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. 23rd Annu. Conf. Comput. Graph. Interactive Techn.*, 1996, pp. 303–312.
[13] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.
[14] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3529–3536, Oct. 2019.
[15] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 2520–2525.
[16] J. Barzilai and J. M. Borwein, "Two-point step size gradient methods," *Ima J. Numer. Anal.*, vol. 8, no. 1, pp. 141–148, 1988.
[17] R. D. T. Steiha, "Truncatednewton algorithmsforlarge-scale optimization," *Math. Program.*, vol. 26, pp. 190–212, 1983.
[18] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Math. Program.*, vol. 45, no. 1-3, pp. 503–528, 1989.
[19] 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 Trans. Robot.*, vol. 36, no. 5, pp. 1526–1545, Oct. 2020.