2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Gradient-Based Online Safe Trajectory Generation for Quadrotor Flight in Complex Environments

Fei Gao, Yi Lin and Shaojie Shen

Abstract— **In this paper, we propose a trajectory generation framework for quadrotor autonomous navigation in unknown 3-D complex environments using gradient information. We decouple the trajectory generation problem as front-end path searching and back-end trajectory refinement. Based on the map that is incrementally built onboard, we adopt a sampling-based informed path searching method to find a safe path passing through obstacles. We convert the path consists of line segments to an initial safe trajectory. An optimization-based method which minimizes the penalty of collision cost, smoothness and dynamical feasibility is used to refine the trajectory. Our method shows the ability to online generate smooth and dynamical feasible trajectories with safety guarantee. We integrate the state estimation, dense mapping and motion planning module into a customized light-weight quadrotor platform. We validate our proposed method by presenting fully autonomous navigation in unknown cluttered indoor and outdoor environments.**

I. Introduction

In recent years, there has been increasing interest in equipping micro aerial vehicles (MAVs) with autonomous navigation capabilities such that they can rapidly and safely fly through complex environments. Given the vehicle state estimation and environment perception, the motion planning module online generates smooth and safe trajectories from the current state to a target state in the configuration space, while avoiding unexpected obstacles during the navigation. In this paper, we propose an online collision-free trajectory generation method utilizing gradient information in the configuration space. The proposed method can guarantee the safety of the vehicle, while at the same time considering the smoothness and dynamical feasibility of the trajectory. We integrate the proposed planning module into a fully autonomous vision-based quadrotor system and present online experiments in unknown complex environments.

Given measurements from onboard sensors, a description of the environment can be established and a configuration space for the high-level motion planning task is built upon it. In our previous work [1], we proposed a trajectory generation method which directly operates on unordered point cloud data from range measurements, without the building of post-processed maps such as occupancy grid or octo-map. This bypasses the costly map building and maintenance and achieves the best adaptivity for different environments. Then the trajectory generation problem was formulated as a convex program with hard constraints. We have also shown the implementation of the method on a monocular dense mapping system [2]. However trajectories generated with hard constraints may suffer from being close to obstacles, making the quadrotor flying with hazard when there is uncertainty in control and perception, since in the objective only energy saving was considered. In this paper, we propose a novel method to improve the trajectory's clearance by utilizing the gradient information in the environments. Compared to benchmarked gradient-based trajectory generation methods, the proposed method is superior in generating feasible and safe trajectories in a high success rate.

As presented by Mellinger et al. [3], a quadrotor system enjoys the differential flatness property, making it possible to reduce the full state space to the 3-D positions and yaw

(a) Autonomous flight in an unknown indoor environment

(b) Autonomous flight in an unknown outdoor environment

Fig. 1. Our quadrotor platform equipped with a monocular fish-eye camera and an IMU. Onboard computation resources include an Intel i7-5500U CPU running at 3.00 GHz and an Nvidia TX1. We demonstrate fully autonomous flights through complex indoor and outdoor environments. Video is available at `https://www.youtube.com/watch?v=armfr_UiXBI&t=20s`

This work was supported by HKUST project R9341 and HKUST institutional studentship. All authors are with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong, China. `fgaoaa@connect.ust.hk`, `ylinax@connect.ust.hk`, `eeshaojie@ust.hk`

angle and their derivatives. Smooth piecewise polynomial trajectories of 3-D position with bounded derivatives can therefore be followed by a properly designed geometric controller [4]. We utilize this property and design a novel method for generating collision-free smooth trajectories directly on the output of our mapping module. Our method employs a randomized geometric graph expanded with heuristic in 3-D space, utilizing fast nearest neighbor search on a Kd-tree. An initial straight-line safe trajectory is searched on the graph, followed by an optimization method to smooth the trajectory and constrain the dynamical feasibility, while at the same time also guaranteeing the safety of the trajectory. We summarize our contributions as follows:

1) A sampling-based, informed rapidly-exploring random graph (RRG) method, for finding a collision-free piecewise line segment path with asymptotic optimality.
2) A trajectory optimization framework, for generating a safe, smooth and dynamically feasible trajectory based on the piecewise line segment initial path.
3) Implementation of the proposed method in an incremental re-planning scheme, and integration with visual-inertial state estimation and monocular fish-eye dense mapping modules in a complete quadrotor platform. Autonomous navigation in unknown indoor and outdoor complex environments are presented.

We discuss relevant literature in Sect. II, and introduce the overall system architecture and the motivation for this work in Sect. III. Our path finding method and trajectory optimization method are detailed in Sect. IV and in Sect. V, respectively. The implementation details about the proposed method, and a brief introduction to visual-inertial state estimation and monocular dense mapping, which build the prerequisite for the navigation, are presented in Sect. VI. In Sect. VII, benchmark results are given. Indoor and outdoor experimental results which show fully autonomous flight in unknown environments are also presented. The paper is concluded in Sect. VIII.

II. Related Work

For robotic motion planning, many methods, ranging from sampling-based [5] to optimization-based [3], have been proposed and applied. Here we provide an overview of representative approaches that are especially relevant to motion/trajectory planning for quadrotor.

Rapidly-exploring random tree (RRT) was developed by LaValle [5]. In this method, samples are drawn randomly from the configuration space and guide the tree to grow towards the target; however standard RRT algorithms are not asymptotically optimal, Karaman and Frazzoli [6] summarized and proposed sampling-based motion planning methods which have asymptotic optimality, including RRT*, probabilistic road maps* (PRM*) and rapidly-exploring random graph (RRG). RRG is an extension of the RRT algorithm, as it connects new samples not only to the nearest node but also all other nodes within a ball. Under the same category, the approach in [7] combines a fixed final state and free final time controller with the RRT* method to ensure asymptotic optimality. In this way, closed-form solutions for optimal trajectories can be derived. Another method combining RRG and the belief roadmap was proposed by Bry and Roy et al. [8], in which a partial ordering is introduced to trade-off belief and distance.

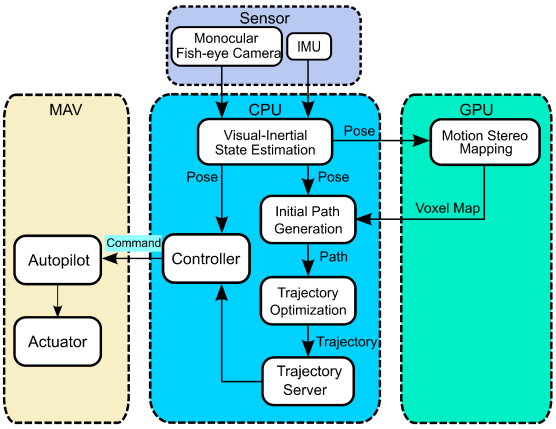
Although sampling-based method is good for finding safe path, the path is often not smooth enough for flying robots to track. Mellinger et al. [3] pioneered a minimum snap trajectory generation algorithm, where the trajectory is represented by piecewise polynomial functions, and the trajectory generation problem is formulated as a quadratic programming (QP) problem. Unconstrained QP with a closed-form solution was formulated in [9], which introduced a method to convert the constrained QP problem to an unconstrained QP problem, thus bypassing the procedure of solving the constrained QP. Both numerical stability and computational efficiency are improved in this way. Chen et al. [10] proposed an online method that utilizes efficient operations in the octomap data structure for online generation of a flight corridor. Then safe and dynamically feasible trajectories are generated using QP. Our previous work [1] proposed a method to carve a flight corridor consisting of ball-shaped safe regions directly on point clouds. Quadratically constrained quadratic programming (QCQP) is then used to generate trajectories that are constrained entirely within the corridor.

Gradient information in the map is also useful for motion planning. As presented by Ratliff et al. [11], the motion planning problem can be formulated as minimizing the cost of two terms: one is the penalty for the trajectory having collisions with the obstacles and the other is the smoothness of the trajectory itself. Another representative method was developed in [12], where a complete formulation of utilizing gradient of the environments to optimize the coefficients of piecewise polynomial trajectory was proposed. However, in [12], even with random restart, the success rate of generating a safe trajectory is around 60%, which is not acceptable in real-world navigation. Also the number of segments of the trajectory is pre-defined, making the method may fail to work in environments with variant obstacle density. In this paper, we follow [12]'s formulation and present a planning framework which has a 100% success rate and is adaptive to all kinds of environments.

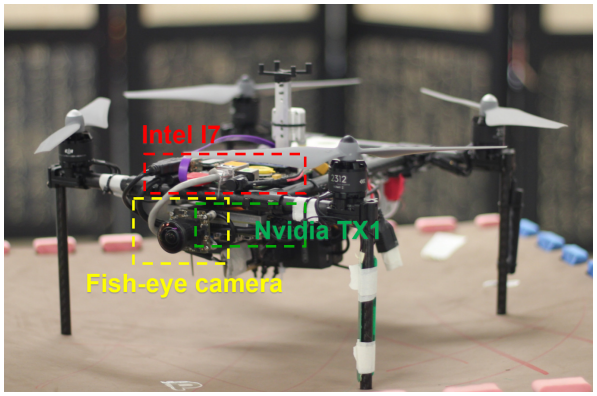
III. Overview

A. Motivation

As is illustrated in Sect. II, our previous work [1] followed and built upon the classical minimum snap trajectory generation framework, and has shown the capability of finding a collision-free, safe and dynamically feasible trajectory in cluttered environments. However, our previous method had an inherent nature that all the free space in the flight corridor is treated equally, making the generated trajectories may be very close to obstacles since energy saving is the only cost in the objective function and we ignored all the gradient information. This drawback motivates us to utilize the gradient information in the environment to add a penalty based on the distance to obstacles, therefore push the optimized trajectory



(a) The complete system architecture.



(b) Our developed vision-based quadrotor test-bed in hover.

Fig. 2. The system diagram and a snapshot of our quadrotor platform are shown in Fig.2(a) and 2(b). We use a monocular fish-eye camera and an IMU for sensing. The mapping module is running on the Nvidia TX1, while state estimation, control and the entire motion planning pipeline are running on the Intel i7 CPU. The high-level controller is used to track the generated trajectory. The quadrotor is stabilized by a DJI A3 autopilot.

away from obstacles. We argue that given a safe initial trajectory, with properly designed optimization procedure, a local optimal trajectory which is smooth, safe and at the same time dynamically feasible, can be obtained.

B. System Architecture

The system architecture is shown in Fig. 2. The vision-based quadrotor platform we develop estimates the states and builds the dense map online with a monocular camera. And we use a fish-eye lens to provide a large field of view (235°). We adopt the state-of-the-art visual inertial system (VINS) [13] to get the 6 degrees of freedom (DoF) state estimation using vision and IMU data stream. And the dense map is built by motion stereo [2] which is running at 10 Hz using the high-accuracy pose estimation. The output local voxel map is fed into the path finding module where an initial safe path from the current state to the target state is found directly on the voxels. Then the trajectory optimization module optimizes the trajectory for smoothness, safety and dynamical feasibility. We use a geometric controller [4] to track the generated trajectory.

IV. Informed Sampling-based Path Generation

We adopt the method proposed in our previous work [1] [2] to generate an initial path directly on the unordered voxels from the mapping module. Having the voxels which represent the obstacles in the environment, a random-exploring graph is generated, and meanwhile a K-d tree is used to evaluate the safe volume of a given point by fast nearest neighbor search. After the generation of the graph, A* is used to search a minimum distance path on it. In our proposed path finding method, by controlling the minimum radius of the ball-shaped safe regions, which will be added into the graph as nodes, the minimum clearance of the path can be adjusted easily.

In this paper, we utilize the informed sampling scheme which is introduced in [14] to improve the efficiency of the sampling and the quality of the path. After the random graph reaches the target, which means the newly generated safe ball contains the target location, we can determine that one path has been found, and we get this path by back tracking through nodes in the graph. Having the total distance of the path, we can use it to define a heuristic domain for subsequent sampling. We generate samples in a hyper-ellipsoid heuristic domain, with the start position and target position on its focal points. The transverse diameter of the hyper-ellipsoid is the distance of the best path so far has been found, and the conjugate diameter of the hyper-ellipsoid is the direct straight distance between the start and target position, as is shown in Fig. 3. After enough samples are generated in the informed domain, A* search will be used again to find a better path and update the minimum distance of the path as well as the heuristic domain. The procedure enjoys asymptotic optimality, which was proved in [14]. Thus if the random sampling and informed domain update are done iteratively, the path will finally converge to the global optimal path given infinite random samples.

To achieve real-time high-speed navigation, the path finding procedure should be terminated when a time limit is reached. In our implementation, after finding the first feasible path, once time exceeds the limit, no more samples will be generated and the best solution so far will be returned. Otherwise A^* will be utilized to search a minimum distance path after a batch of samples and the heuristic sampling domain will be updated based on the current best result. The output path will be initialized to an initial safe piecewise polynomial trajectory for the following optimization, with all the derivatives at waypoints set as 0 and time for each segment is allocated using an average velocity. Details about the formulation of the trajectory can be found at Sect. V-A.

V. Piecewise Polynomial Trajectory Optimization

A. Problem Formulation

The trajectory consisting of piecewise polynomials is parametrized to the time variable t in each dimension μ out of x, y, z. The N^{th} order M-segment trajectory of one

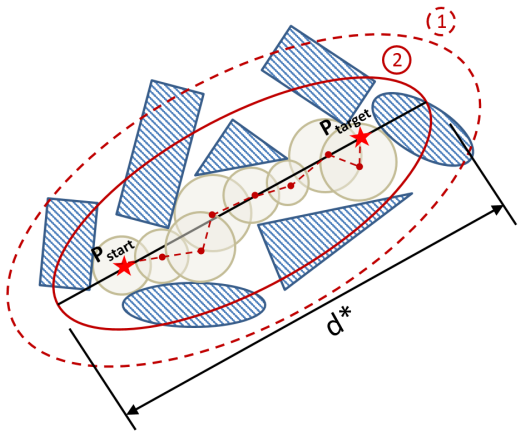


Fig. 3. Heuristic sampling domain updated in path finding. Red dashed ellipsoid with marker 1 is the last hyper-ellipsoid domain for generating samples. After a new better solution d^* has been found, the sampling domain shrinks to the red solid ellipsoid with marker 2. Red stars indicate the start point and the target point for the path finding mission. Gray spheres are safe regions found by sampling and nearest neighbor query, and the red dashed poly-line is the current best path searched by A^* in the graph.

dimension can be written as follows:

$$p_\mu(t) = \begin{cases} \sum_{j=0}^{N} \eta_{1j}(t-T_0)^j & T_0 \le t \le T_1 \\ \sum_{j=0}^{N} \eta_{2j}(t-T_1)^j & T_1 \le t \le T_2 \\ \vdots & \vdots \\ \sum_{j=0}^{N} \eta_{Mj}(t-T_{M-1})^j & T_{M-1} \le t \le T_M, \end{cases} \tag{1}$$

where η_{ij} is the j^{th} order polynomial coefficient of the i^{th} segment of the trajectory. $T_1, T_2, ..., T_M$ are the end times of each segment, with a total time $\tau = T_M - T_0$.

We follow [12] to optimize directly on the coefficients of the piecewise polynomial trajectory. The formulation of the complete objective function is written as

$$\min \quad \lambda_1 f_s + \lambda_2 f_o + \lambda_3 (f_v + f_a), \tag{2}$$

where f_s is the regularized smoothness term to keep the trajectory smooth, f_o is the cost of the clearance of the trajectory, f_v and f_a are the penalties of the velocity and acceleration exceeding the dynamical feasibility. λ_1, λ_2 and λ_3 are weight parameters to trade off the smoothness, trajectory clearance and dynamical feasibility. The smoothness term is the integral of the square of the ϕ^{th} derivative. In this paper, we minimize the snap (4^{th} derivative of the position) of the quadrotor to obtain a smooth trajectory, which is good for our vision-based localization and mapping system, so ϕ is 4, and the order N of the polynomial we use is 8.

We adopt the method proposed in [9] and introduce a mapping matrix $\mathbf{M}$ and a selection matrix $\mathbf{C}$ to map the original variables, the polynomial coefficients, to the derivatives at each segment points of the piecewise polynomial:

$$\boldsymbol{\eta} = \mathbf{M}^{-1}\mathbf{C}\begin{bmatrix}\mathbf{d}_F \\ \mathbf{d}_P\end{bmatrix}. \tag{3}$$

Here the matrix $\mathbf{M}$ maps the polynomial coefficients $\boldsymbol{\eta}$ to $\mathbf{d}$, and matrix $\mathbf{C}$ separates the derivatives $\mathbf{d}$ into fixed derivatives $\mathbf{d}_F$ (which are pre-defined before the trajectory generation) and free derivatives $\mathbf{d}_P$ (which are the optimized variables). In this paper, all derivatives at middle waypoints, including the position, velocity and acceleration, are free variables, and derivatives at the start and target points are fixed variables. The cost of the smoothness term can be written as

$$f_s = \begin{bmatrix}\mathbf{d}_F \\ \mathbf{d}_P\end{bmatrix}^T \mathbf{C}^T\mathbf{M}^{-T}\mathbf{Q}\mathbf{M}^{-1}\mathbf{C}\begin{bmatrix}\mathbf{d}_F \\ \mathbf{d}_P\end{bmatrix}. \tag{4}$$

Denote $\mathbf{C}^T\mathbf{M}^{-T}\mathbf{Q}\mathbf{M}^{-1}\mathbf{C}$ as matrix $\mathbf{R}$, then the cost function can be written in a partitioned form as

$$f_s = \begin{bmatrix}\mathbf{d}_F \\ \mathbf{d}_P\end{bmatrix}^T \begin{bmatrix}\mathbf{R}_{FF} & \mathbf{R}_{FP} \\ \mathbf{R}_{PF} & \mathbf{R}_{PP}\end{bmatrix}\begin{bmatrix}\mathbf{d}_F \\ \mathbf{d}_P\end{bmatrix}. \tag{5}$$

The Jacobian and Hessian of f_s with respect to $\mathbf{d}_p$ can be written as follows, in where $\mu \in (x, y, z)$:

$$\mathbf{J}_s = \left[\frac{\partial f_s}{\partial \mathbf{d}_{P_x}}, \frac{\partial f_s}{\partial \mathbf{d}_{P_y}}, \frac{\partial f_s}{\partial \mathbf{d}_{P_z}}\right], \mathbf{H}_s = \left[\frac{\partial^2 f_s}{\partial \mathbf{d}_{P_x}^2}, \frac{\partial^2 f_s}{\partial \mathbf{d}_{P_y}^2}, \frac{\partial^2 f_s}{\partial \mathbf{d}_{P_z}^2}\right],$$
$$\frac{\partial f_s}{\partial \mathbf{d}_{P_\mu}} = 2\mathbf{d}_F^T\mathbf{R}_{FP} + 2\mathbf{d}_P^T\mathbf{R}_{PP}, \quad \frac{\partial^2 f_s}{\partial \mathbf{d}_{P\mu}^2} = 2\mathbf{R}_{PP}^T, \tag{6}$$

For the cost of collision on the trajectory, we need a differentiable cost function to penalize the distance value, and we want the cost to rapidly grow up to infinity at where near the obstacles and to be flat at where away from the obstacles. If we provide a proper collision-free initial trajectory, the collision cost at where near obstacles will dominate the objective function and prevent the trajectory from colliding with obstacles. The cost function we select is an exponential function. At a position in the map with distance value d, the cost $c(d)$ is written as

$$c(d) = \alpha \cdot \exp(-(d - d_0)/r), \tag{7}$$

where α is the magnitude of the cost function, d_0 is the threshold where the cost starts to rapidly rise, and r controls the rate of the function's rise. The cost is negatively correlated to the distance value. We follow [12] to formulate the collision cost as the line integral of the distance value over the arc length along the trajectory. For numerical calculation, we can discretize the integral and formulate it as a summation of costs on different time stamps:

$$\begin{aligned} f_o &= \int_{T_0}^{T_M} c(p(t))ds \\ &= \int_{T_0}^{T_M} c(p(t))\|v(t)\|dt = \sum_{k=0}^{\tau/\delta t} c(p(\mathcal{T}_k))\|v(t)\|\delta t, \end{aligned} \tag{8}$$

where $\mathcal{T}_k = T_0 + k\delta t$, and $v(t)$ is the velocity at $p(t)$.

The Jacobian of the collision term in a discrete form is

$$\mathbf{J}_o = \left[\frac{\partial f_o}{\partial \mathbf{d}_{P_x}}, \frac{\partial f_o}{\partial \mathbf{d}_{P_y}}, \frac{\partial f_o}{\partial \mathbf{d}_{P_z}}\right]$$
$$\frac{\partial f_o}{\partial \mathbf{d}_{P_\mu}} = \sum_{k=0}^{\tau/\delta t}\left\{\nabla_\mu c(p(\mathcal{T}_k))\|v\|\mathbf{F} + c(p(\mathcal{T}_k))\frac{v_\mu}{\|v\|}\mathbf{G}\right\}\delta t, \tag{9}$$

where $\mu \in (x, y, z)$, $\mathbf{L}_{dp}$ is the right block of matrix $\mathbf{M}^{-1}\mathbf{C}$ which corresponds to the free derivatives on the μ axis $\mathbf{d}_{p_\mu}$. $\mathbf{F} = \mathbf{T}\mathbf{L}_{dp}$, $\mathbf{G} = \mathbf{T}\mathbf{V}_m\mathbf{L}_{dp}$. $\nabla_\mu c(\cdot)$ is the gradient in the μ axis of the collision cost. $\mathbf{V}_m$ is the mapping matrix which maps the polynomial coefficients of the position to the polynomial coefficients of velocity, and $\mathbf{T} = [\mathcal{T}_k^0, \mathcal{T}_k^1, ..., \mathcal{T}_k^n]$ is the vector of a given time t_0. Furthermore, we get the Hessian matrix as follows:

$$\mathbf{H}_o = \left[\frac{\partial^2 f_o}{\partial \mathbf{d}_{P_x}^2}, \frac{\partial^2 f_o}{\partial \mathbf{d}_{P_y}^2}, \frac{\partial^2 f_o}{\partial \mathbf{d}_{P_z}^2}\right],$$

$$\frac{\partial^2 f_o}{\partial \mathbf{d}_{P\mu}^2} = \sum_{k=0}^{\tau/\delta t} \Big\{ \mathbf{F}^T \nabla_\mu c(p(\mathcal{T}_k)) \frac{v_\mu}{\|v\|} \mathbf{G} + \mathbf{F}^T \nabla_\mu^2 c(p(\mathcal{T}_k)) \|v\| \mathbf{F}$$

$$+ \mathbf{G}^T \nabla_\mu c(p(\mathcal{T}_k)) \frac{v_\mu}{\|v\|} \mathbf{F} + \mathbf{G}^T c(p(\mathcal{T}_k)) \frac{v_\mu^2}{\|v\|^3} \mathbf{G} \Big\} \delta t, \qquad (10)$$

where $\nabla_\mu^2 c(\cdot)$ is the 2^{nd} derivative of the collision cost in μ.

For formulating the cost on the dynamical feasibility, we generate an artificial cost field on velocity between the maximum velocity and minus maximum velocity in the x, y and z axis. And we write f_v as the sum of the line integral of the velocity in x, y and z axis. The cost function of the high-order constraints is also an exponential function, since it is good at penalizing when close to or beyond the limit of bounds and staying flat when away from the bounds, $c_v(v)$ is the cost function applied on the velocity, which has the same form as in Eq.(7). The cost of velocity feasibility is:

$$\begin{aligned} f_v &= \sum_{\mu \in \{x,y,z\}} \int_{T_0}^{T_M} c_v(v_\mu(t)) ds \\ &= \sum_{\mu \in \{x,y,z\}} \int_{T_0}^{T_M} c_v(v_\mu(t)) \|a_\mu(t)\| dt. \end{aligned} \qquad (11)$$

The Jacobian and Hessian of $\mathbf{J}_v$ has similar formulation to Eq.(9) and Eq.(10). We omit the formulation of the acceleration feasibility cost f_a for brevity.

Having the total Jacobian $\mathbf{J} = \lambda_1 \mathbf{J}_s + \lambda_2 \mathbf{J}_o + \lambda_3(\mathbf{J}_v + \mathbf{J}_a)$ and Hessian $\mathbf{H} = \lambda_1 \mathbf{H}_s + \lambda_2 \mathbf{H}_o + \lambda_3(\mathbf{H}_v + \mathbf{H}_a)$, we use the Newton trust region method [15] to optimize the objective. The update equation is:

$$(\mathbf{H} + \lambda \mathbf{I}) \Delta \mathbf{d}_p = -\mathbf{J}^T, \ \mathbf{d}_p \leftarrow \mathbf{d}_p + \Delta \mathbf{d}_p, \qquad (12)$$

where λ is the factor determined by a heuristic and is initialized with a large value to ensure convergence.

B. Locally Voxel Caching for Collision Cost Evaluation

In order to evaluate the cost of the collision penalty and get the gradient information along the trajectory, the most commonly used method is to maintain a distance map. But even for a local map with a short range, the maintenance and updating of the complete distance field is costly. Furthermore, maintaining a distance field map and getting the gradient by taking the difference on the map in x, y and z directions only provides an approximated gradient and can be easily undifferentiable at places where the distance value is indeed not continuous. Thus we propose a new light-weight method to help us evaluate the collision cost.

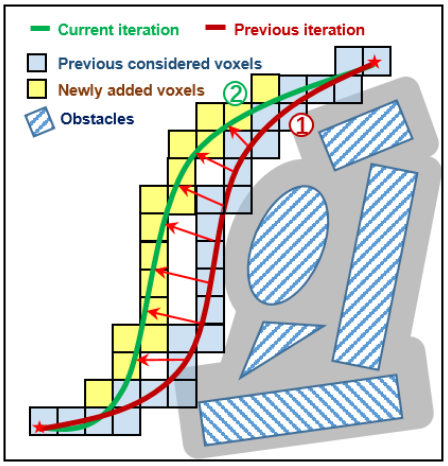
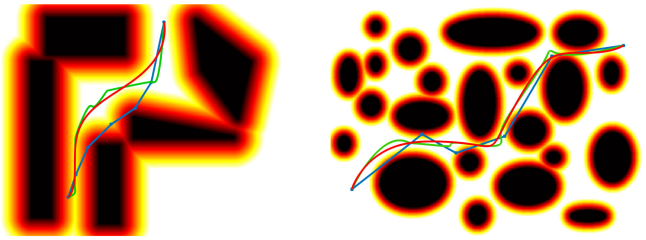


Fig. 4. Illustration of the voxel hashing. Shapes with shading are obstacles. The red curve and green curve are trajectories before and after one iteration, respectively. Blue voxels have already been recorded, while yellow voxels are newly added. Gray areas are collision margins near to obstacles but still considered as safe. Red arrows roughly show the gradient direction.

Having the initial trajectory generated in Sect. IV, only a small part of the space will be evaluated during the optimization. Thanks to the sparsity of the distribution of the evaluated voxel in the map, we can accelerate the procedure by voxel hashing. We evaluate the cost and gradient of a point on the trajectory and simply hash the voxel it belongs to according to a pre-defined resolution (voxel size). For each point, we do nearest neighbour search using the K-d tree. Since the tree has been built in path finding, there is no extra cost on building the tree. The nearest neighbour query only takes $O(logN)$ computation time, where N is the number of voxels. In this way, the Euclidean distance in 3-D space of a point to its nearest neighbour is obtained, and at the same time we get the position of the nearest neighbour, which directly indicates the direction of the gradient. During the optimization, we cached all voxels which have been queried with the cost and gradient information for further iterations.

C. Two-Steps Optimization Framework

Nonlinear optimization is applied to smooth the collision-free initial trajectory, and at the same time ensure the safety and dynamical feasibility. The initial trajectory which geometrically follows the collision-free path (see Sect. IV) is used as the initial value. We apply a two-steps optimization strategy which can be summarized as follows: 1) Optimize the collision cost only. All the free variables including positions of middle waypoints on the initial trajectory will be pushed to the basin of the distance field from the neighbourhood of obstacles. 2) Re-allocate the time and re-parametrize the trajectory to time according to current waypoints' positions. Then optimize the objective with a smoothness term and dynamical penalty term added. The optimizer smooths the safe trajectories and squeezes the



(a) Trajectory generated in a sparse environments (b) Trajectory generated in a dense environments

Fig. 5. Trajectory generated in 2-D sparse and dense map. The colour code shows the value of the distance to the nearest obstacles, with darker colors corresponding to a higher cost of collision. Orange areas are buffer areas near obstacles but still safe, and black areas are obstacles. The initial trajectory shown with the blue line is a collision-free but hard-to-follow straight-line trajectory. After the first step in the optimization, the trajectory and middle waypoints are pushed away from the near obstacles, as the green line shows in Fig. 5(b). The final trajectory is shown with the red curve.

dynamic infeasibility, and at the same time prevents the trajectories from moving near to obstacles again.

Since path finder may place waypoints close to obstacles, optimizing only on the collision cost in the first step can lead the trajectory quickly converge to a much more safe trajectory. After that, the re-allocation of time in each segments of the trajectory according to current waypoints' positions makes the time allocation more appropriate for following optimization. The entire optimization procedure finishes when the termination condition is met or the pre-defined time limit for optimization is reached. Some results of the two-steps optimization are shown in Fig. 5.

VI. Implementation Details

A. System Settings

Our trajectory generation module is implemented in C++11. The flight experiments are done on a self-developed quadrotor platform (Fig. 2(b)), and the computing resources include a mini computer, which has a dual-core Intel i7-5500U processor running at 3.00 GHz with 8 GB RAM and 256 GB SSD, and an Nvidia TX1. Autonomous flight experiments are done in unknown environments using only online sensor measurements. The number of samples for path finding in the experiments is set as 2000. The maximum number of iterations in trajectory optimization is set as 50, and the weights of the smoothness, collision and dynamical cost are 2.5, 1.0 and 1.0. The range of the local map is set as $5\,m$ to ensure the mapping module works online at a high frequency (10 Hz). The time limits of the front-end path finding and back-end trajectory optimization for both indoor and outdoor tests are 20 ms and 30 ms, respectively.

B. State Estimation and Dense Mapping

Referring to our previous work [2], the 6-degrees of freedom state estimation results, which are provided by a tightly-coupled visual-inertial fusion framework, are fed into the feedback control as well as the motion stereo mapping. The mapping module fuses each keyframe's depth map into a global map by means of truncated signed distance fusion (TSDF). A frontward-looking fish-eye camera with a large field of view of 235° is used to ensure wide surrounding environment perception. The state estimation module runs at 20 Hz and the dense mapping module outputs a local map with a range of $5\,m$ at 10 Hz.

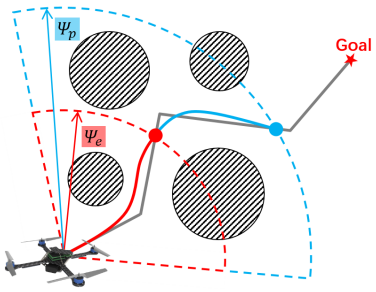


Fig. 6. Illustration of the receding horizon incremental planning strategy. The global guiding path is shown with the gray line. The red curve and blue curve are the planning trajectory and the execution trajectory, respectively. See Sect. VI-C for details.

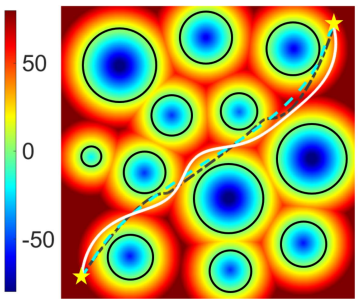
C. Incremental Planning Scheme

Due to the limited sensing range of our mapping module, we use a receding horizon incremental planning strategy [16]. Before the flight starts, a target point is set and a global path finder is utilized to find a global path to reach the target. During the navigation, a local planner is used to find the path and optimize the trajectory within the planning horizon Ψ_p. The trajectory will only be executed in an execution horizon Ψ_e, after which the local planner will be called again, as is shown in Fig. 6. And if the newly updated map has a collision with the trajectory which is going to be executed, the local planner will be called immediately. The planning target for the local planner is selected firstly on the global path, and if the target is occupied (e.g. has obstacles), the planner will automatically adjust the target and search for a free point in a nearby region. In the motion planning module, both the front-end path finding and back-end trajectory optimization part have pre-defined time limits, which control the termination conditions of the whole pipeline. Note that all unknown space (outside the perception range) is treated as free in the global path finder and is treated as occupied in the local planner.

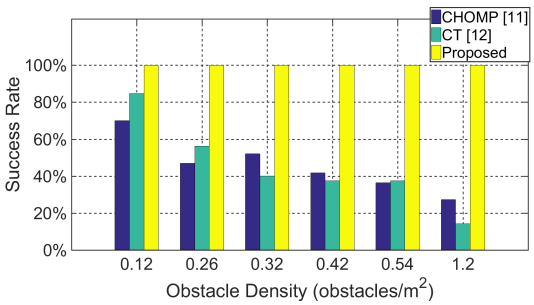
VII. Results

A. Benchmark Results

We compare our proposed method with continuous-time (CT) trajectory generation [12] and CHOMP [11] using obstacles randomly deployed 2-D maps. For CHOMP, we set the parameters for balancing the computing time and success rate and set the number of points as 100 and number of iterations as 200. For CT, we use the best parameters claimed in [12]. We set the number of segments of the trajectory as 3 and minimize the jerk of the trajectory. The results are given in Fig. 7. As the obstacle density increases, the success rate of CHOMP and CT both decrease rapidly, while at the same time, our method still generates feasible and safe trajectories. For 3-D environments with randomly deployed obstacles, the map is $20m \times 20m \times 10m$ and the planning target is selected

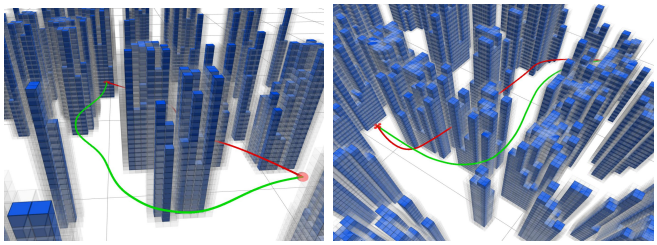


(a) Trajectories generated by our method, CT and CHOMP.

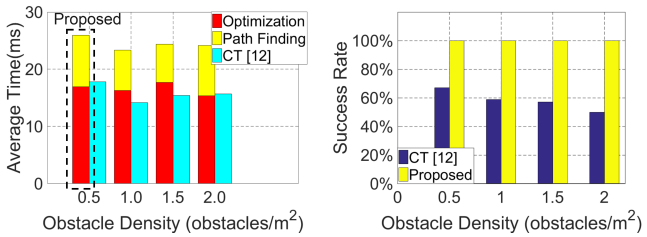


(b) Comparison of success rate. As the obstacle density increases, the success rate of CHOMP and CT both decrease rapidly.

Fig. 7. Benchmark results. In Fig. 7(a), the white curve, brown dashed curve and blue dashed curve are trajectories generated by our method, CT and CHOMP, respectively. The color code indicates the distance value in the field. Black circles are obstacles randomly deployed.



(a) On average 1.5 obstacles/m^2 (b) On average 2.0 obstacles/m^2



(c) Comparison of the computing time and success rate

Fig. 8. Comparison in random 3-D maps. The green curve indicates the trajectory generated by our method, and the red curve is the trajectory generated by continuous-time trajectory generation [12]. Comparing our method with CT, the time consumption of the optimization is roughly equal, but there is extra time consumed for path finding.

randomly. The results are shown in Fig. 8. The computing time for the optimization of our method is roughly equal to that of CT, while we obtain a 100% success rate with the cost of finding the initial safe path. Note that since we use our own implementation of CHOMP and CT, the comparison may not be completely fair.

B. Indoor Autonomous Flight

We present several autonomous flights in our laboratory with randomly placed obstacles to validate our proposed method. A snapshot is shown in Fig. 1(a). In the experiment, the quadrotor is commanded to fly to the targets. State estimation, mapping, and trajectory generation are all performed online without any prior information about the environments. The local planning horizon is set as $5\,m$ and the execution horizon is $2\ m$. Computing time limits for path finding and trajectory optimization are set as $15\ ms$ and $25\,ms$, respectively. In the mapping module, the size of the local map is $5m \times 5m \times 5m$, and the resolution of TSDF in mapping and local free voxel hashing in planning are both set as $0.15\,m^3$. Representative figures which record the indoor flights are shown in Fig. 9. The quadrotor generated safe and smooth trajectories and travelled through complex environments autonomously. In real implementation, when observations are not sufficient, unknown area may block the way for finding a path by the local planner, especially when there is no significant base-line for motion stereo. Under this situation, the quadrotor will generate safe but short trajectories ($< 1\ m$) in the free space, and explore the surrounding environments until the map is fully observed and a feasible path is found. Details about the autonomous flights and the exploration strategy are shown in the video.

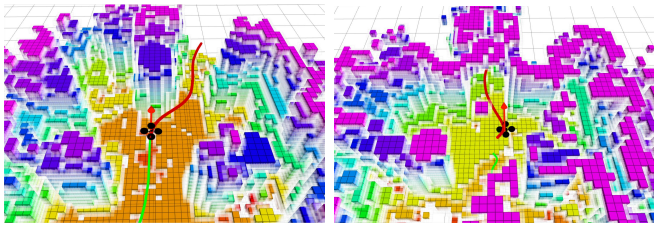
C. Outdoor Autonomous Flight

The outdoor experiments are done in two unknown complex environments, both contain a slope and obstacles in various shapes. Figures record the outdoor flight are given in Figs. 1(b) and Fig. 10. A complete mesh map recording the flight is given in Fig. 11. The parameters of the planning module are set as the same as in the indoor flights and our quadrotor autonomously navigates in the cluttered environments. Note that although our planning module can run at a high frequency (25 Hz), since we adopt the planning and execution horizon, the planer would not be called everytime the map updated. This property saves the computing resources significantly. We also employ the exploration strategy (Sect. VII-B) in the outdoor flights.

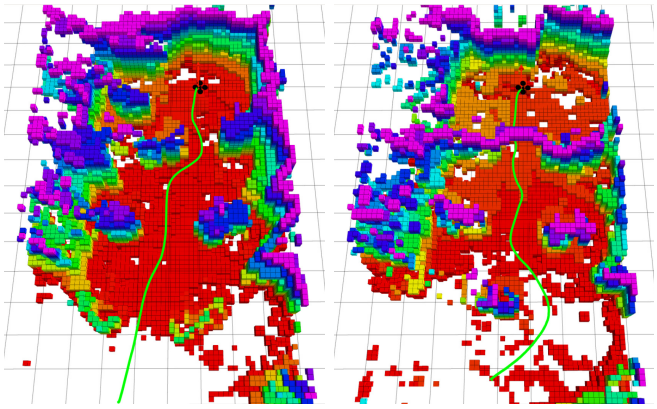
VIII. Conclusion and Future Work

In this paper, we propose a novel online motion planning framework for quadrotor autonomous navigation. The proposed method combines a sampling-based path finding method to find a collision-free path, and an optimization-based method to refine the trajectory and improve the dynamical feasibility. We integrate the planning, state estimation and dense mapping module onboard our self-developed vision-based quadrotor platform and present fully autonomous flights in both indoor and outdoor unknown environments to validate our method.

For quadrotor motion planning, path finding and trajectory optimization work as front-end solution searching and back-end solution refinement, respectively. Compared to our



(a) Test 1: Passing through obstacles (b) Test 2: Passing below an arch



(c) Complete map and path in test 1 (d) Complete map and path in test 2

Fig. 9. Autonomous indoor flight in unknown environments. Two tests in different environments are given in Figs. 9(a) and 9(b). The colour code indicates the height of the obstacles. The white transparent area is the safe margin in the trajectory planning module. The red curve is the latest generated trajectory and the green curve is the trajectory that has been executed. The delay of the executed path in Fig. 9(b) is caused by an I/O jam onboard the quadrotor. The complete map and trajectory are shown in Figs. 9(c) and 9(d), real scene of the flight can be refereed to Fig. 1(a). More indoor trials of our proposed method are presented in the video.

previous work, which formulates the problem as a convex program with hard constrains, using unconstrained nonlinear optimization has the natural advantages of modeling costs for several different considerations and the ability to be terminated at any-time to meet the high-speed requirement and limited computing resources. In the future, we plan to do more analysis and comparison of these two back-end optimization methods. We also intend to challenge our quadrotor system in more extreme situations, including large-scale or dynamic environments. Furthermore, we plan to add a time adjustment term in the objective.

REFERENCES

[1] F. Gao and S. Shen, "Online quadrotor trajectory generation and autonomous navigation," in *Proc. of the IEEE Intl. Symp. on Safety, Security, and Rescue Robot.*, Lausanne, Switzerland, Oct. 2016.

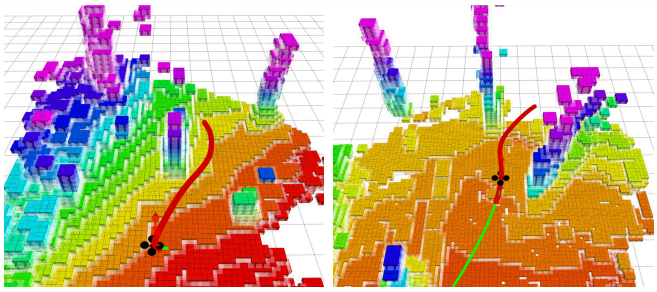
[2] Z. Yang, F. Gao, and S. Shen, "Real-time monocular dense mapping on aerial robots using visual-inertial fusion," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, 2017.

[3] 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, pp. 2520–2525.

[4] T. Lee, M. Leoky, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *Proc. of the IEEE Control and Decision Conf.*, Atlanta, GA, Dec. 2010, pp. 5420–5425.

[5] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, pp. 846–894, 2011.



(a) Test 1: Pass through obstacles (b) Test 2: Pass below the arch

Fig. 10. Autonomous outdoor flight in unknown environments. Two tests in different environments are shown in Figs. 10(a) and 10(b). Markers can be interpreted in the same way as Fig. 9. More outdoor trials of our proposed method are presented in the video.

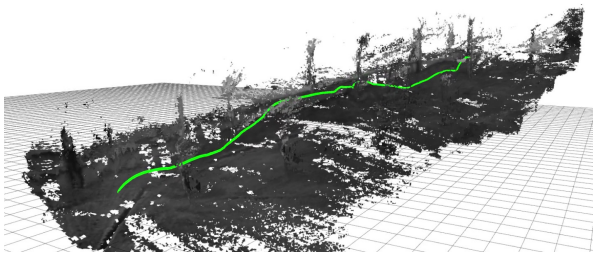


Fig. 11. Complete map built and trajectory executed in outdoor flight. Note that the mesh map is generated offline for visualization, since only the occupancy voxels are essential in assisting autonomous flight. The green curve is the complete trajectory which has been executed. One can take Fig. 1(b) as reference for the real scene of the flight.

[7] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Germany, May 2013.

[8] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011, pp. 723–730.

[9] 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.

[10] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016.

[11] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Kobe, Japan, May 2009.

[12] 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, Oct. 2016.

[13] Z. Yang and S. Shen, "Monocular visual–inertial state estimation with online initialization and camera–imu extrinsic calibration," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 39–51, 2017.

[14] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Chicago, IL, Sept 2014.

[15] D. C. Sorensen, "Newtons method with a model trust region modification," *SIAM Journal on Numerical Analysis*, vol. 19, no. 2, pp. 409–426, 1982.

[16] 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*, 2017.