Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Dynamic Obstacle Avoidance for UAVs Using a Fast Trajectory Planning Approach

Han Chen* and Peng Lu†

Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong, China
stark.chen@connect.polyu.hk
peng.lu@polyu.edu.hk

Chenxi Xiao*

Purdue University
West Lafayette, IN 47907, U.S.A

xiao237@purdue.edu

***Abstract* – Real-time path planning is crucial to the dexterity of UAVs when traversing through environments with unknown obstacles. In this paper, we proposed such a real-time planning algorithm that can be used in cluttered environments. The real-time computation is achieved by using dimensionality reduction as well as the rolling optimization. We test the algorithm both in the Gazebo simulation and in two real experiments. The results show that our algorithm can maintain a relatively high speed flight while avoiding obstacles successfully.**

Index Terms - obstacle avoidance, real-time path planning, UAV.

I. INTRODUCTION

In the past few years, autonomous navigation of drones in unknown environments has received intensive attention. The rationale behind is the difficulty of building an accurate map for many real-world scenarios. For example, walking pedestrians on the street cannot be modeled in a statistical map. It is also intractable to know the map in advance in search and rescue applications within complex environments such as a collapsed house.

Another motivation for developing real-time planners is the difficulty for current off-line planning approaches to be applied to scenarios with insufficient map information. The off-line planners must know some prior information for predicting the correct action. Besides, it is also difficult for these algorithms to satisfy the time requirement of real-time planning, because the computational complexity of those off-line planning approaches is still too high. Many airplane applications that fit in such a paradigm are often limited to a low-speed flight[1]. This computation cost is incurred by the inherent non-convexity of path planning optimization problems, the high mapping rates as well as the high planning rates.

This work proposes a novel method that allows the drone to avoid obstacles with a minor computational budget. By running our planner at a high frequency (50hz), we can use relatively high speed for obstacle avoidance.

In order to avoid excessive calculation, the current path planning methods often use a less accurate global path planner and a higher precision local planner. In the global planning process, the path calculated by the local planner is optimized (local planner gets the minimum value of the Cost-To-Go(CTG) function). However, when the drone detects obstacles, the map changes at the same time. It is necessary to calculate a path that takes into account the minimum CTG and is easy to be followed by the drone[3].

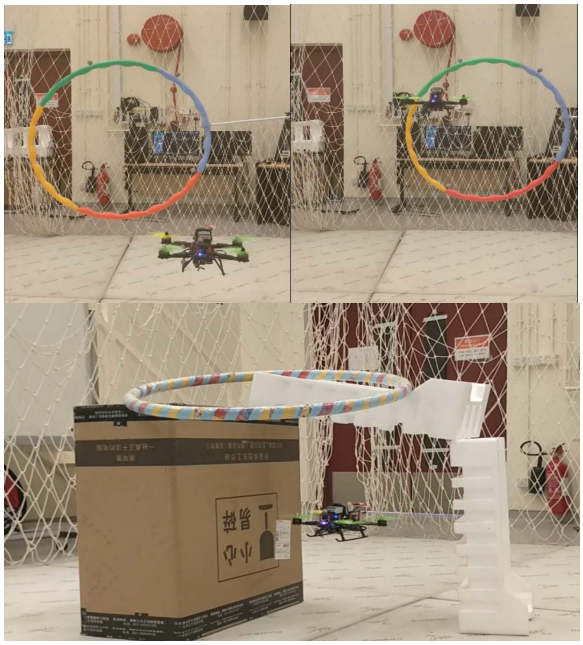


Fig. 1 Our experiment drone is crossing the moving circle and static boxes

Therefore, an accurate CTG calculation method capable of simultaneously capturing the global environment and kinetic feasibility is required while maintaining a relatively low calculation time. In addition, the way the local path planner performs path planning in the 3D map after updating the obstacle has a significant impact on the computational cost. Most of the current aircraft systems are unable to maintain the same rate as the sensor frame rate (~30 Hz). Usually complete 3D path planning is performed at a slower rate relative to the sensor. So one of the challenges is to find a way to significantly improve the efficiency of calculating local paths, and convert the local path into an executable control that

*These authors contributed equally
† Corresponding author

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

satisfies the dynamic constraints. In addition, we need a more efficient combination of global path planning and partial path planning while ensuring the optimal partial path and further reducing the amount of calculation[4].

This work addresses these challenges with the following contributions:

- A lightweight local path planning method in a 3D environment decomposes the 3D planning problem into two steps: finding a finite number of feasible planes and planning a path on the local map.
- An integration of the global planning and local planning: we mesh the map into grids of two scales, the global map has larger grids while the local map in front of the drone has small grids. We utilize the A* method to obtain a rough path on a global map and an exact path on a local map. A drone is always flying along the local path after optimal regeneration, the global path is used for setting the destination on the local map.
- The minimum snap was utilized to regenerate the path to making the UAVs flight more smooth and energy-saving. Simulation and hardware experiments showing fast and fluent flights in the environment with detected obstacles, with mean velocities about 1 m/s.

II. RELATED WORK

At present, there are many algorithms used in UAV route planning, such as static route planning algorithms based on known threats: the Voronoi diagram method, Dijkstra algorithm, A* algorithm, A rapidly exploring random tree method (RRT) and evolutionary algorithm(EA). Dynamic route search algorithms based on unknown threats, such as particle swarm optimization (PSO), D* algorithm, vector field histogram (VFH) algorithm, dynamic window method (DWA) and artificial potential field method (APF) [5].

Among these traditional algorithms, few of them or their variants can be applied to the real aircraft while taking into account the optimal path, effectively avoiding obstacles and quick passing at the same time. Therefore, the related research work in recent years is to combine the static programming algorithm applicable to global path planning with the dynamic path search algorithm applicable to local path planning, with the kinetic or dynamic constraints of the aircraft to make the path feasible for a drone to track under given performance conditions.

Dynamic path planners often use a memory-less representation of the environment, and closed-form primitives are usually chosen for planning in B. T. Lopez and S. LaValle's previous work[6][7]. These approaches often fail in complex cluttered scenarios. On the other hand, map-based static planners usually use occupancy grids or distance fields to represent the environment. These planners either plan all the trajectory at once, or implement a Receding Horizon Planning Framework optimizing trajectories locally and based on a global planner.

Moreover, when unknown space is also taken into consideration, there are several approaches: Some researchers[8][9] used optimistic planners (considering unknown space as free), while in Oleynikova's work [11], an optimistic global planner is used combined with a conservative local planner.

Jesus et al proposed a planning framework in which multi-fidelity models are used to reduce the discrepancy between the local and global planners. They address the interaction between a fast planner and a slower mapper by considering the sensor data not yet fused into the map during the collision check[1].In their later work, They propose FASTER (Fast and Safe Trajectory Planner) to obtain agile flights in unknown cluttered environments with fast velocities. It's safety guarantees are ensured by always having a feasible, safe back-up trajectory in the free-known space at the start of each replanning step. In addition, they use the Mixed-Integer Quadratic Program formulation in which the solver can choose the appropriate trajectory interval allocation for saving time.

Gao et al, utilize the property of the fast nearest neighbor search in KD-tree and adopt the sampling-based pathfinding method based on the incrementally built point cloud map to generate a flight corridor with a safety guarantee in 3-D space. They also use a trajectory generation method formulated in quadratically constrained quadratic programming (QCQP)to generate trajectories that constrained entirely within the corridor. In their later work, they adopted a fast marching-based path searching method to find a path on a velocity field induced by the Euclidean signed distance field (ESDF) of the map, to achieve better time allocation. After that, they represent the trajectory as piecewise Bezier curves by using the Bernstein polynomial basis to bound positions and higher-order dynamics of the trajectory entirely within safe regions[4].

III. REAL-TIME PLANNING

Planning on the original 3D world is time-consuming. To cope with this challenge, we introduce multi-scale grids and a dimension reduction trick. The whole algorithm is given as **Algorithm 1**. We describe line 1 in section A and describe lines 2,3 in Section B, C separately. Lines 4-6 are described in section C.

As a whole, the outer loop is executed at 50 Hz. So that our algorithm can be fully real-time.

Algorithm 1: our proposed real-time planner

1 while ***true***:
2 find the feasible plane, project the obstacle points on it
3 get map $\boldsymbol{M}$ using 2D map construction
4 $\boldsymbol{P}$ = find the path by A* on map $\boldsymbol{M}$
5 truncate the beginning k points $\boldsymbol{P_k}$ from $\boldsymbol{P}$
6 local trajectory regeneration using minimum snap by current speed $\boldsymbol{v}$ and $\boldsymbol{P_k}$
7 tracking local trajectory

A. Dimensionality reduction of the 3D path planning problem

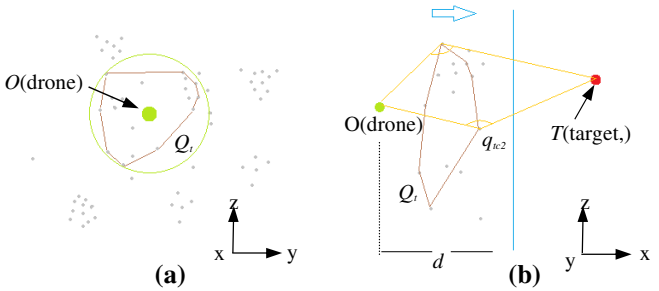


Fig. 2 A schematic diagram of the drone and a point cloud from different perspectives, (a) is viewed from the x-axis negative section of the drone to the positive direction, (b) is viewed along the y-axis direction. *x,y,z* are three coordinate axes of the body coordinate. Brown line connect all the points in Q_t

An employed trick in this part is dimension reduction. As the time and space complexity for searching in 3D plane is exponential to the problem dimension, significant time can be saved if we only use a 2D plane for path searching. The 2D plane is generated in real-time by the following rule: we find a "feasible plane", in which the path to target point may take the least cost, then we project the obstacle points close enough to this plane to make the path safe. The obstacle's corresponding grids in the 2D map mentioned above will be occupied.

As shown in Fig. 2, the green dots represent the drone, the gray dots represent the point cloud corresponding to the obstacle in front of the drone (along with the head), and the green circle represents the range of sensor detection on the drone. First, the point cloud in the detection range is projected on a plane perpendicular to the *x*-axis of the body coordinate system, and the distance *d* of the plane from the centroid position is the maximum detection distance of the sensor on the drone. Name the point projected on the plane within the detection range as $Q=(p_1,p_2...)$ and perform convex hull operations on these points, that is, find a polygon with the point in Q as the vertex, and wrap the polygon in the plane[14]. In this paper, the GRAHAM-SCAN algorithm is used to perform convex hull scanning on Q. The schematic diagram of the algorithm's pipeline is shown in Fig. 3

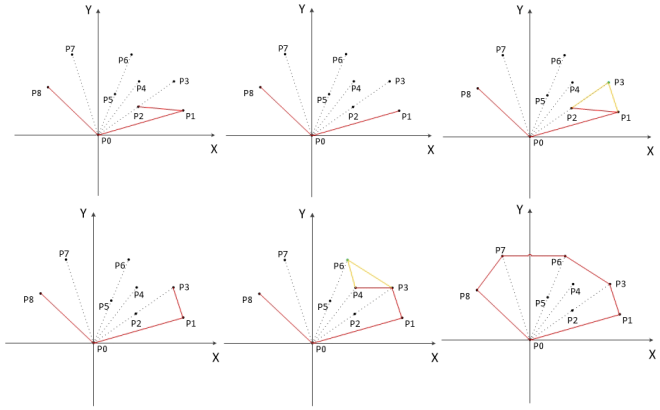


Fig. 3 GRAHAM-SCAN algorithm flow diagram

Algorithm 2:GRAHAM-SCAN(Q)

Let be p_0 be the point in Q with the minimum y-coordinate, or the leftmost such point in case of a tie
Let be $<p_1,p_2,...,p_m>$ be the remaining point in Q, sorted by the polar angle in counter-clockwise order around p_0(if more than one point has the same angle, remove all but the one that is farthest from p_0)
PUSH(p_0,S)
PUSH(p_1,S)
PUSH(p_2,S)
for $I \leftarrow 3$ **to** m
  do while the angle formed by points NEXT-TO- TOPS(S), TOPS, and p_i makes a nonleft turn
  do POP(S)
PUSH(p_i,S)
return S

After obtaining the vertex $Q_t=(q_{t1},q_{t2}...)\subset Q$ on the convex hull, we calculate the value of $\angle Oq_tT$ in turn, and select a point of Q_t to get the maximum value of $\angle Oq_tT$, then the point $q_{tc}\subset Q_t$ farthest from the centroid of the drone is selected. Now we can determine a plane $Oq_{tc}T$, which is the mentioned "feasible plane". After that, all the points in the detected 3D point cloud whose vertical distance of the plane is smaller than the maximum feature size d_s of the drone are projected onto the feasible plane, and the obtained point set Q_{map} is the plane obstacle information required for the 2D path planning. This projection step is to ensure the safety of the aircraft. Because the aircraft has a certain volume, it is necessary to take the point close to the feasible plane as an obstacle.

B. *Map Construction with Multi-scale grids*

A grid map is a regular choice to represent the world. For a conventional grid map, the grid size is inverse proportional to the number of cells. With small grids, we can achieve an accurate representation of obstacles, but the computational cost would be high. For a large grid size, the computational budget can be saved but the positional representation of the obstacle can be inaccurate.

We use a multi-scale grid map for achieving a balance between computational cost and accuracy. The map is divided into two layers, one layer is a global map with a relatively large grid size; The other layer is a local map around the drone with smaller grid size. With such a design, we can maintain a fast path generation speed while also maintaining fine-grained planning result around the drone. A visualization of this approach is given in Fig. 4:

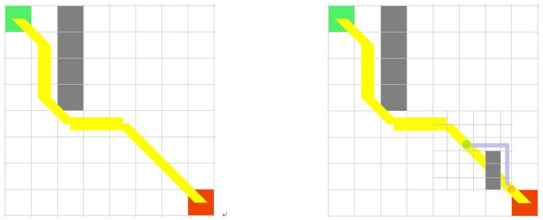


Fig. 4 A conventional grid map as the left image shows. A multi-scale map that has better representation for small obstacles, as shown in the right image

C. *Path Finding*

We choose a vanilla A* algorithm for path generation. A diagram of the A* algorithm is given as Fig. 5. A* algorithm is fast enough for path searching in real time. Another character of A* that we prefer is that the pattern of the results can be a consensus between multiple searches because no random factors were involved. Accordingly, we can avoid the oscillation in real motions.

We use a very simple heuristic function $h(p_i, p_j) = |x_i - x_j| + |y_i - y_j|$ because we can avoid square root calculation, and therefore the overall path searching could be faster.

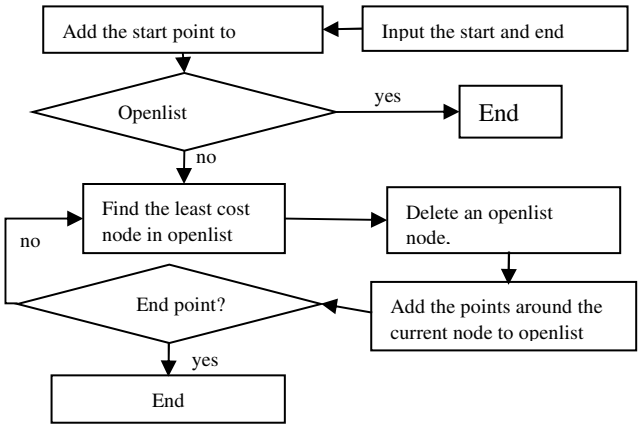


Fig. 5 Flow chart of A* algorithm.

D. local trajectory regeneration

By using the minimum snap method, the kinetic feasibility of the UAV was taken into account to make it accessible to test the path planning algorithm on a real drone, instead of only in the numeral simulation environment. Thus it makes our path planning algorithm more practical and reliable based on the real drone experiment.

Only the trajectory segment that is closer to the drone is meaningful because the drone only can make small motions between two planning sessions. After that, the old path will be replaced by a new path. In this paper, we implement a local trajectory regeneration using Minimum Snap(MS) [10, 13] only by the first 4 points in a path. this idea is similar to the rolling optimization in Model Predictive Control (MPC).

In the minimum snap algorithm, a complex trajectory is composed of multiple polynomial segments:

$$p(t) = \begin{cases} [1, t, t_2, ..., t_n] \cdot p_1 \quad (t_0 \le t < t_1) \\ [1, t, t_2, ..., t_n] \cdot p_2 \quad (t_1 \le t < t_2) \\ ... \\ [1, t, t_2, ..., t_n] \cdot p_k \quad (t_{k-1} \le t < t_k) \end{cases} \tag{1}$$

Where p_i is the parameter vector of a polynomial:

$$p_i = [p_{i0}, p_{i1}, ..., p_{in}]^T$$

and t_i is the time for starting the motion described in i^{th} polynomial[15]. k is the number of segments in this trajectory.

For minimizing the snap (the 4th derivation of position), the following target function is used as the cost function[12]:

$$\begin{aligned} &\min \int_0^T (p^{(4)}(t))^2 dt \\ &= \min \sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} (p^{(4)}(t))^2 dt \\ &= \min \sum_{i=1}^{k} \int_{t_{i-1}}^{t_i} [0,0,0,0,24,...,\frac{n!}{(n-4!)} t^{n-4}] \cdot p)^T [0,0,0,0,24,...,\frac{n!}{(n-4!)} t^{n-4}] \cdot p dt \\ &= \min \sum_{i=1}^{k} p^T \begin{bmatrix} 0_{4\times4} & 0_{4\times(n-3)} \\ 0_{(n-3)\times4} & \frac{r!}{(r-4)!}\frac{c!}{(c-4)!}\frac{1}{(r-4)+(c-4)+1}(t_i^{(r+c-7)} - t_{i-1}^{(r+c-7)}) \end{bmatrix} p \\ &s.t \ \ A_{eqp} = b_{eq}, \\ &A_{ieqp} \le b_{ieq} \end{aligned} \tag{2}$$

where *r, c* is the row index and column index of the matrix, the index starts from 0, that is, for the first row r=0. A_{eqp} , b_{eq} are the constrains for curved motion, A_{ieqp} is the inequation constrain. For our algorithm, we constrain the velocity of the first point as the current measured velocity of the quadrotor and also constrain the last point in the minimum snap trajectory as a given velocity value with a direction towards the next point. In this way, we can get the speed command in several future time steps, by which we can implement trajectory tracking using the quadrotor speed controllers[13].

Ⅳ. EXPERIMENTAL RESULTS

A. Experimental Configuration

We conduct our experiment on a self-assembled quadrotor with an QAV 250 frame, the diagonal length of which is 25cm. We use a PX4 module as the flight controller, and the diameter of the circle we use as an obstacle is 0.8m.

For a hole traversing task with a relatively high speed, the accuracy of the quadrotor and ring's position is crucial. We use our Vicon Mocap system for the position and velocity feedback.

The planning framework is supported by the Robot Operation System (ROS). **Mavros** package is deployed for establishing the communication between our planner node and the PX4 control module. The speed controller for tracking is provided by the PX4 module by default. The **vicon** package is used to interface with the Vicon server for retrieving the data package published by the Vicon system, but when we make a simulation with Gazebo, we use a Python script to replace the real Vicon system.

B. Simulation test result

We first tested our algorithm in the Gazebo simulation environment controlled by a PX4 controller firmware. By publishing packages that simulate the vicon_bridge, we can obtain an initial estimation of our algorithm performance. As shown in Fig. 6, the quadrotor flies along the green trajectory to cross the ring, which is shown in RVIZ as a cylinder for convenience. The red arrow heads to the positive x-direction of the body coordinate. In the simulation, we verified the correctness of the program code and the approximate effect of the algorithm.

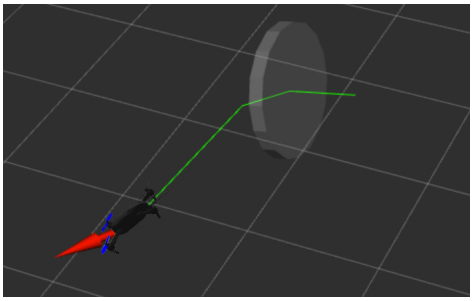


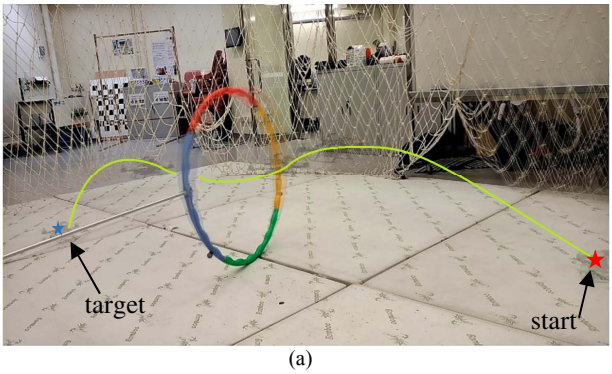
Fig. 6 The simulation progress viewed in RVIZ

B. *Real drone result*

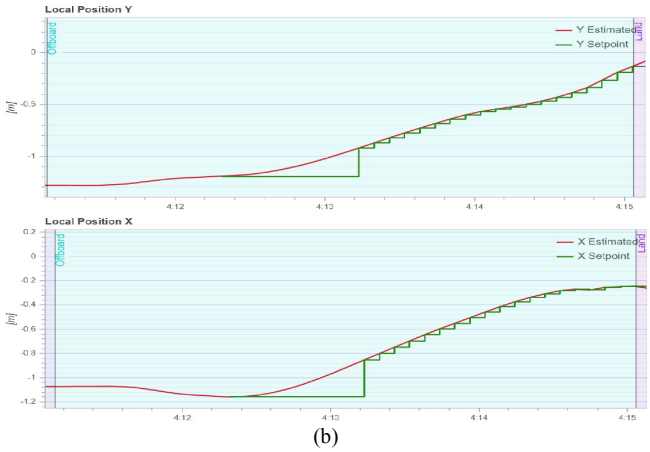
For real-world experiments, we conduct two experiments: (1) traversing a static hole, (2) traversing a moving hole.

(1) static obstacles

Since the drone does not have a sensor such as a depth camera that detects obstacles yet, we send the obstacle information to the drone in real-time through the VICON system to replace the UAV's detection of obstacles. When the distance of the drone from the obstacle reaches the set sensor detection range d(in this test d=0.5m), it means that the obstacle is detected by the drone sensor, and the real-time obstacle avoidance mode is triggered. The start and target points of the drone have been marked. We took the flying drone every 1 s with a fixed camera position and superimposed the pictures together. The trajectory is shown in Fig. 7:

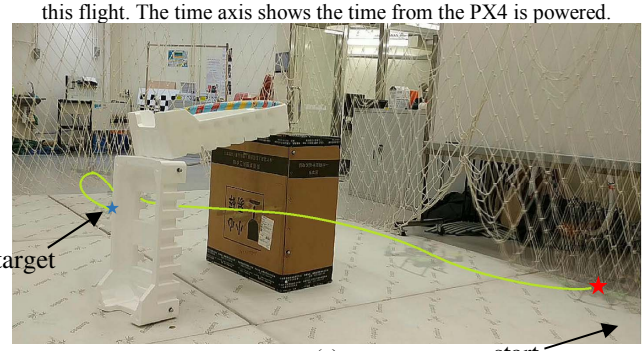


(a)

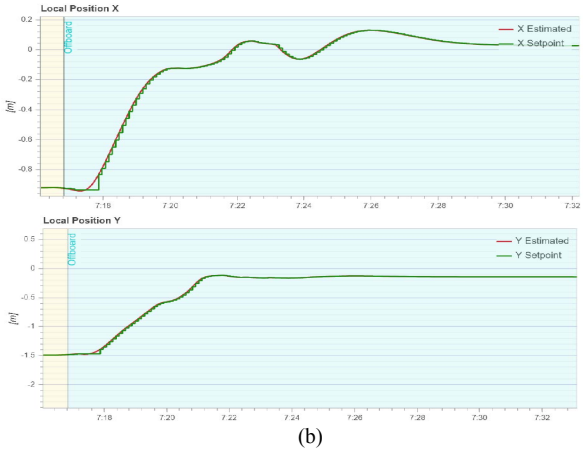


(b)

Fig. 7 (a) The superimposed image of a fixed camera timing shot of a drone passing through a fixed ring. (b) The X, Y coordinate record of the drone in this flight. The time axis shows the time from the PX4 is powered.



(a)



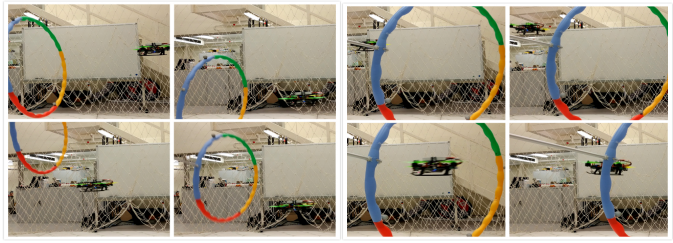
(b)

Fig. 8 (a) is the superimposed image of a fixed camera timing shot of a drone passing through some static boxes. (b) is the X, Y coordinate record of the drone in this flight. The time axis shows the time from the PX4 is powered.

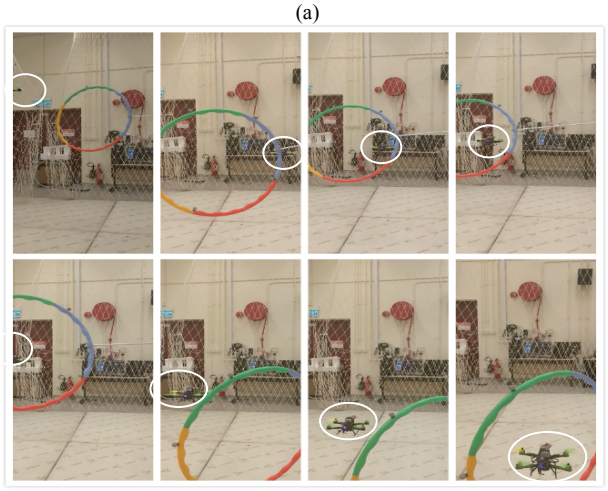
In the two experiments of static obstacles, the UAV can fly to the target point quickly and smoothly and can adjust the flight attitude sensitively when approaching the obstacle, while keeping a certain distance from the obstacle during flight. From the curve of the X and Y coordinates of the aircraft position versus time, we can see that the setpoint command issued to the aircraft by the algorithm proposed in this paper is constantly changing, because the path provided to the px4 controller in each iteration is only the first sampling point on the path.

(2) moving obstacles

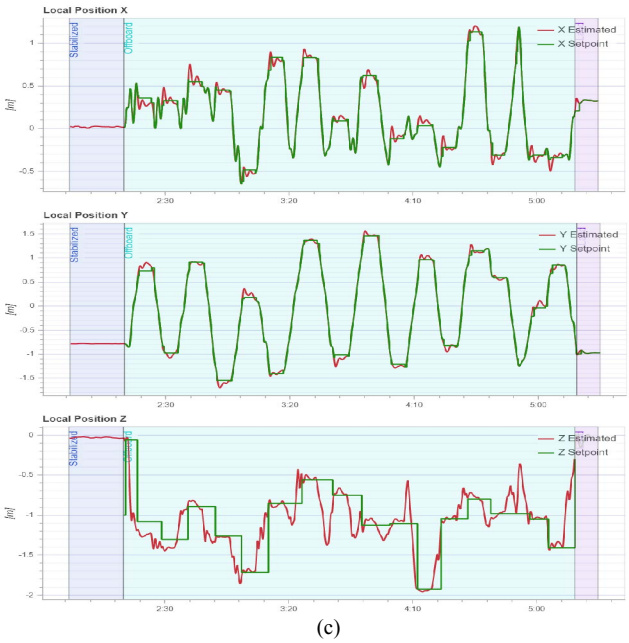
In order to test the real-time computing ability of the proposed method, we use a moving ring for testing, and the target point of the aircraft is also set to a point on the ring axis opposite to the drone, and the target point moves along with the ring. Obviously, in order to reach the target point, the shortest path is to pass through the ring.



(a)



(b)



(c)

Fig. 9 (a) The images of a fixed camera timing shot of a drone passing through a moving ring. (b) Pictures from a camera held in hand in a different view. (c) The *X, Y, Z* coordinate record of the drone in this flight. The time axis shows the time from the PX4 is powered.

In the experimental results of the moving ring shown in Fig. 9, we can see from the curve that the setpoint has a rapid response with the movement of the ring, and because the drone is flying faster, there is no change in the setpoint. The position of the machine fluctuates, which is unavoidable considering the limitations of aerodynamics. In the dozens of consecutive piercing tests, the aircraft can accurately and quickly track the position of the ring, which proves that the running time and accuracy of the proposed algorithm can meet the requirements.

VI. Conclusions

This paper proposed an algorithm that can control the autonomous flight of a drone to a target point in an environment where there are unknown obstacles. It includes path planning on known global maps, real-time path update after the occurrence of obstacles is detected, and the final optimization of the path which makes the path smooth. Before flight experiments, the algorithm's feasibility is verified in the ROS/px4 software package under Linux computer OS. Verified by multiple flight experiments on a small drone, our algorithm is fast and effective, and the average speed can be maintained above 1 m/s during flight.

Acknowledgment

We would like to thank Ran Duan and Anting Wang for their enthusiasm to help and hard work for the experiments in this paper.

References

[1] J. Tordesillas, B. T. Lopez, J. Carter, J. Ware, and J. P. How, "Real-Time Planning with Multi-Fidelity Models for Agile Flights in Unknown Environments," 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 2019, pp. 725-731.

[2] C. Goerzen, Z. Kong, and B. Mettler, "A Survey of Motion Planning Algorithms from the Perspective of Autonomous UAV Guidance," Journal of Intelligent & Robotic Systems, vol. 57, no. 1-4, p. 65, 2010.

[3] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *Int. J. Robot. Res.*, vol. 30, pp. 846–894,2011.

[4] F. Gao, W. Wu, Y. Lin and S. Shen, "Online Safe Trajectory Generation for Quadrotors Using Fast Marching Method and Bernstein Basis Polynomial," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, 2018, pp. 344-351.

[5] František Duchoň, Andrej Babinec, Martin Kajan et al. Path Planning with Modified a Star Algorithm for a Mobile Robot[J]. Procedia Engineering, 2014, 96:59-69.

[6] B. T. Lopez and J. P. How, "Aggressive 3-D collision avoidance for high-speed navigation," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 5759-5765.

[7] S. LaValle and J. Kuffner, "Randomized Kinodynamic Motion Planning," *Int. J. Rob. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[8] M. Pivtoraiko, D. Mellinger and V. Kumar, "Incremental micro-UAV motion replanning for exploring unknown environments," 2013 IEEE International Conference on Robotics and Automation, Karlsruhe, 2013, pp. 2452-2458.

[9] Jing Chen, Tianbo Liu, and Shaojie Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," 2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, 2016, pp. 1476-1483.

[10] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion
planning in high dimensions," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 2640–2645, 2011.

[11] H. Oleynikova, Z. J. Taylor, R. Siegwart, and J. Nieto, "Safe Local Exploration for Replanning in Cluttered Unknown Environments for Micro-Aerial Vehicles," IEEE Robotics & Automation Letters, vol. 3, no. 3, pp. 1474-1481, 2018.

[12] X. Zhang and Y. Fang, "Set-oriented optimal path planning of mobile robots by a polynomial rootfinder," 2013 IEEE International Conference on Control Applications (CCA), Hyderabad, 2013, pp. 778-783.

[13] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 2520-2525.

[14] J. Schulman et al., "Motion planning with sequential convex optimization and convex collision checking," International Journal of Robotics Research, vol. 33, no. 9, pp. 1251-1270, 2014.

[15] W. F. Ji, H. F. Xu, G. Y. Wang, and X. H. Yang, "Path Planning under Dynamic Threat Environment," Applied Mechanics & Materials, vol. 543-547, p. 5, 2014.