Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots

Fanze Kong[1*], Wei Xu[1*], Yixi Cai[1], Fu Zhang[1]

***Abstract*—In practical applications, autonomous quadrotors are still facing significant challenges, such as the detection and avoidance of very small and even dynamic obstacles (e.g., tree branches, power lines). In this paper, we propose a compact, integrated, and fully autonomous quadrotor system, which can fly safely in cluttered environments while avoiding dynamic small obstacles. Our quadrotor platform is equipped with a forward-looking three-dimensional (3D) light detection and ranging (lidar) sensor to perceive the environment and an onboard embedded computer to perform all the estimation, mapping, and planning tasks. Specifically, the computer estimates the current pose of the UAV, maintains a local map (time-accumulated point clouds KD-Trees), and computes a safe trajectory using kinodynamic A* search to the goal point. The whole perception and planning system can run onboard at 50Hz. Various indoor and outdoor experiments show that the system can avoid dynamic small obstacles (down to 9mm diameter bar) while flying at 2m/s in cluttered environments. High-speed experiments are also carried out, with a maximum speed of 5.5m/s. Our codes are open-sourced on Github[2].**

***Index Terms*—Aerial Systems: Perception and Autonomy, Motion and Path Planning, Collision Avoidance.**

I. Introduction

UNMANNED aerial vehicles (UAVs) have shown increasing potentials in a variety of applications, such as delivery, inspection, mapping, and search-and-rescue missions. To enable these widespread applications, a fully autonomous UAV that is able to fly in cluttered environments is crucial. One major challenge in achieving this goal is the detection and avoidance of very small and/or dynamic objects such as tree branches, power lines, small pipes, and stair rails that are common in real-world environments.

Despite the plenty of works on autonomous UAVs recently developed, there are still two critical unsolved issues. Most existing works cannot handle dynamic obstacles with only onboard UAV resources due to the limited rate of sensing and/or planning. Avoiding small obstacles is another challenge to UAVs due to the low resolution of mapping in existing works.

These two issues are rooted in the limited sensing capabilities available for existing UAVs. The popularly used RGB-D sensors have limited sensing range (a few meters) and resolution, preventing the timely and reliable detection of small obstacles (e.g., pipes with diameter smaller than 20mm). The noisy measurements of RGB-D sensors also require additional probabilistic filtering on each voxel's occupancy (e.g., OctoMap [1]), which imposes an inherent trade-off between processing time and affordable resolution. Multi-line spinning lidars are another type of sensor that has been used on UAVs. Besides the high-cost, their bulky size, weight ($\sim$1kg), low resolution (e.g., 2° for 16 lines lidars), and low frame-rate (e.g., 10Hz) have significantly prevented the adoption to UAVs for obstacle avoidance.

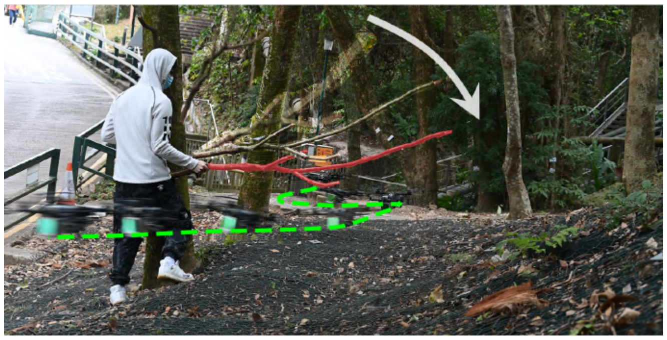


Fig. 1. Our system is able to detect and avoid dynamic small obstacles such as tree branches. The green dashed line is the path flied by the UAV, the white arrow indicates the movements of the tree branch. The current tree branch is highlighted in red. Video is available at https://youtu.be/mjtmpEYwQsI.

In this paper, we present an autonomous UAV system aiming to fulfill intelligent flight in cluttered and dynamic environments. The system-level contribution is the proof of feasibility of using only a low-cost, light-weighted, and small form factor 3D lidar and onboard processing to achieve high-rate and safe navigation in unknown cluttered environments (see Fig. 1). We summarize our contributions as follows:

1) Design of a complete autonomous lidar-based UAV for the avoidance of small and dynamic obstacles. It is lightweight (1.8kg in total) and compact (280mm wheelbase) while equipped with onboard computing (a DJI Manifold 2-C, with Intel i7 8550U CPU) and sensing (a solid-state lidar: LIVOX AVIA) modules. The system performs full state estimation and high-resolution mapping at 50Hz with onboard computing;
2) Time-accumulated KD-Tree: a novel mapping approach naturally balancing the LiDAR resolution and dynamic objects. This approach enables efficient and timely (i.e., 50Hz) detection of obstacles, static or dynamic, large or small, in the scene;
3) Field tests demonstrating the effectiveness of the UAV system and the proposed approach. Real-world experi-

Manuscript received February 24, 2021; revised May 6, 2021; accepted July 13, 2021. This work is supported by DJI under the grant number 200009538.

*These authors have contributed equally to this work.

[1]All authors are with Department of Mechanical Engineering, University of Hong Kong. {kongfz,yixicai}@connect.hku.hk, {xuweii, fuzhang}@hku.hk

[2]https://github.com/hku-mars/dyn_small_obs_avoidance.git

ments show that with only onboard sensing and computation, our UAV system can safely navigate in various indoor and outdoor environments while reliably avoiding dynamic small obstacles (down to 9mm) in the scene.

II. RELATED WORK

A. Navigation in cluttered environments

In recent years, the localization and mapping technology of unmanned vehicles have developed rapidly, especially vision-based and lidar-based methods. For the application of vision-based UAVs, Visual-Inertial Odometry (VIO) is one state-of-the-art technique, which gives stable odometry by only using camera images and IMU measurements [2]–[5]. Most existing works of autonomous aerial vehicle systems are based on visual sensors and have achieved outstanding performance in static complex environments.

The standard process of a vision-guided aerial vehicle is to obtain the location from the VIO algorithm and combine it with the depth image from a depth camera to build a map (e.g., Occupancy grid map [1], Euclidean Signed Distance Field (ESDF) map [6] or point clouds map [7]), and then generate and execute a trajectory on the established map. Liu *et al.* [8] use a uniform resolution volumetric occupancy grid map to establish a short-range planner for high-speed quadrotor navigation. Oleynikova *et al.* [9] presents a UAV system using a high-rate local replanner based on ESDF map, which can avoid newly-detected obstacles and generate feasible trajectories online in 50ms. Lopez *et al.* [10] proposed a state-based motion primitive planning on current point cloud map and achieved high-speed avoidance. Tordesillas *et al.* [11] proposed a framework that generates a fast trajectory while always having a safe trajectory in known area and achieves up to 7.8m/s speed in cluttered environments. The Nanomap [12] and Mapless planner [13] check collision not only in the current FoV point cloud, but also in the past FoVs. Nanomap [12] even achieved 10m/s velocity flight in forests considering pose uncertainties.

Besides vision-based UAV, there are also many works on lidar-based UAVs. Liu *et al.* [14] generate safe flight corridors (SFC) consisting of multiple ellipsoids from point clouds of a 360° lidar and optimize the trajectory in the flight corridor. Mihir *et al.* [15] proposed a method to explore a mine site autonomously, which utilizes control-based motion primitives to find future-safe paths in an ESDF map built from lidar data. Zhang *et al.* [16] achieved a 10m/s flight in forests using a set of static motion primitives assuming a constant flying velocity in the lidar frame.

B. Navigation in dynamic environments

The approaches mentioned above have a common assumption that the environment is static, so the rate of map updating and replanning is usually low. To avoid dynamic obstacles, it requires both the environment perception (i.e., mapping) and trajectory replanning to be performed at a very high rate. Sanchez *et al.* [17] generates a progressive optimal trajectory allowing the UAV to avoid moving people at a very low speed. Allen *et al.* [18] use kinodynamic RRT* algorithm to make the UAV avoid a dynamic sword rapidly, but the perception and replanning are both performed by offboard sensors (i.e., motion capture systems) and computers. Wang *et al.* [19] use an RGB-D camera on a quadrotor UAV to avoid moving people, but its ability to avoid very small objects is not known. Falanga *et al.* [20] use an event camera to fulfill a high speed (10m/s relative speed) objects avoidance, having only 3.5ms overall latency. However, the event cameras do not have a complete perception of the environment.

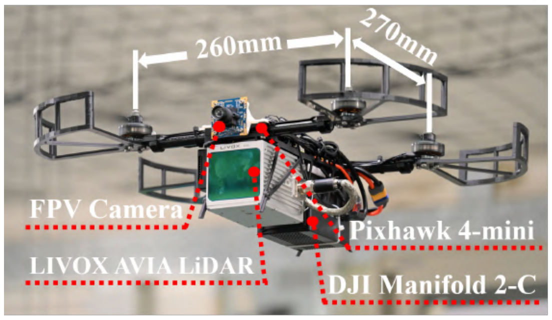


Fig. 2. Our quadrotor UAV. The camera is used for visualization only.

C. Navigation considering small obstacles

In practical applications, it is common for UAVs to encounter small obstacles (e.g., power lines outdoor, small pipes and/or stair rails in indoors). Existing works are usually based on visual sensors and detect such small objects by reasoning the image. Madaan *et al.* [21] trains a convolution network to segment wires from images and reconstructs them to fulfill the avoidance. Zhang *et al.* [22] proposes a method to detect the power line using convolutional and structured features. These methods are usually not general and costly for UAVs.

Our work considers the three challenges all at once. We emphasize a wholistic system design from sensing, perception, estimation, to planning, instead of focusing on one component alone. Compared with the existing work, our developed system is lightweight, fast, and able to safely navigate in complex indoor and outdoor environments while avoiding dynamic small obstacles.

III. SYSTEM HARDWARE

In this section, we present the hardware design of our quadrotor testbed aimed at achieving good autonomous flying performance, being an integrated, durable, and aggressive UAV, as shown in Fig. 2.

A. Fully onboard hardware architecture

The UAV hardware system consists of a Livox AVIA lidar, a DJI Manifold 2-C onboard computer (i7 8550U CPU, 1.8GHz Basic Frequency, quad-core), and a Pixhawk4-mini flight controller. We also add a monocular camera on the UAV to gain first-person view (FPV) images for better visualization. Note that the camera is not used in the perception system.

Compared with vision-based UAVs, our system only uses a lidar (with a built-in consumer-grade IMU) to navigate the UAV and to sense the environment. The measured point cloud and IMU data are transmitted to the onboard computer, which performs all the state estimation, mapping, and motion (re-) planning (see Section IV). At last, the flight controller Pixhawk4-mini performs the trajectory tracking control.

TABLE I
SYSTEM SPECIFICATIONS

Thrust weight ratio	3
Weight (lidar included)	1.8kg
Size (with propeller)	44cmx38cmx15cm
Size (without propeller)	26cmx27cmx15cm
Maximum flight time	10min
Battery	5S 3300mAh 50C

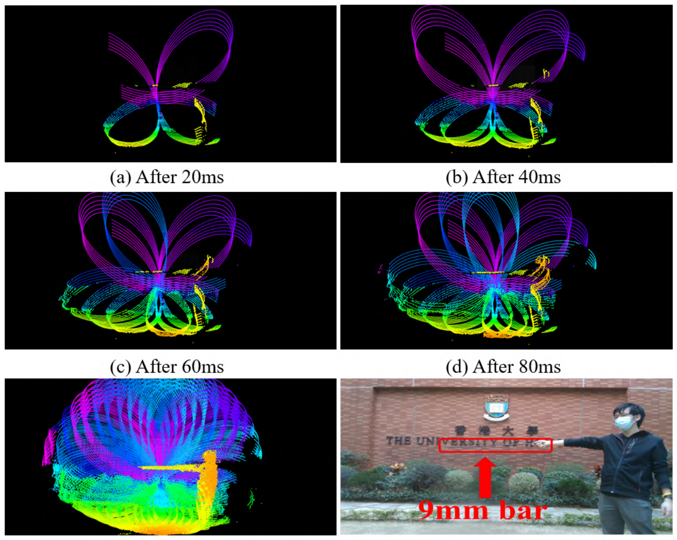
TABLE II
LIDAR PARAMETERS

Detection Range	1m – 450m
FoV	$70.4° \times 77.2°$
Point Rate	240,000points/sec
Maximum Frame Rate	2500Hz
Build-in IMU Rate	200Hz
Range Precision	2cm
False Alarm Ratio	<0.0003%
Weight	498g
Size	76mm×65mm×91mm

Besides the avionic system mentioned above, we also design and optimize our specific quadrotor frame. To ensure that the UAV has enough mobility and small size, we choose T-motor F90 motors and 7042 propellers as the power plant. As shown in Table I, the final UAV has a moderate weight (1.8kg in total), size (280mm wheelbase), and flight endurance (over 10 mins). With over 3-g thrusts, the UAV is able to perform aggressive maneuvers to quickly respond to dynamic obstacles.

B. Lidar characteristics

Lidar is a type of sensor which emits laser pulses and receives the reflected pulse to gain depth information of the environment. Unlike multi-line spinning lidars used on existing UAVs [14], we use a solid-state lidar, the Livox AVIA, which features many advantages that are suitable for aerial applications: (1) shown in Table II, the sensor has a weight and size that can be adapted to a small-scale UAV; (2) the sensor has a very long detection range, allowing obstacle detection and avoidance at high-speed flight; (3) the sensor has an extremely low false alarm rate, producing only three noise points per one million points. This could eliminate the time-consuming ray-casting occupancy filtering as in OctoMap [1] and ESDF [6]; (4) the sensor can output point clouds data at a very high rate (up to 2500Hz), as opposed to the 10∼20Hz output rates of multi-line spinning lidars. This allows timely detection of dynamic obstacles; (5) the sensor uses 6 laser heads to scan the front area simultaneously in a rosellete pattern, forming a circular FoV. Since it covers the entire FoV very quickly, the sensor is very suitable for detecting line-shaped small objects. Furthermore, the sensor uses a non-repetitive scanning [23] that significantly boosts the resolution even at stationary. An example of the sensor measurements for a 9mm bar are seen in Fig. 3, it can be seen that the bar is detected only after 20ms and the resolution increases over the accumulation time.



(a) After 20ms (b) After 40ms (c) After 60ms (d) After 80ms (e) After 1000ms (f) Camera image

Fig. 3. Point clouds scanning results of a 9mm diameter bar over different accumulation time. The bar can be detected in the first 20ms data.

IV. INTEGRATED ONBOARD PERCEPTION AND MOTION PLANNING

A. Overview

In this section, we detail the software and algorithm design. The overview of the system workflow is seen in Fig. 4 and Algorithm 6. A typical autonomous UAV consists of three functional parts: navigation, planning, and control. In our system, navigation and planning modules are both running on the onboard computer, and the tracking control is running on the onboard flight controller Pixhawk4-mini. For the navigation task, we use FAST-LIO [24] to compute the location of the UAV and build a point cloud map of the environment, both at 50Hz. After receiving a new scan of point cloud and IMU data from the lidar, FAST-LIO estimates the current UAV state in a tightly-coupled iterated Kalman filter. The estimated state is, in turn, used to project the new scan of point cloud to the world frame (Line 1) and then added to a local map (Line 2), which is a time-accumulated KD-tree detailed in Section IV-B. The updated local map triggers a collision check on the current trajectory under tracking. If any collision occurs (Line 3), a new trajectory is re-planned from the state on the current trajectory that is $30ms$ after (to account for the re-planning time, Line 4). The re-planned trajectory replaces the current

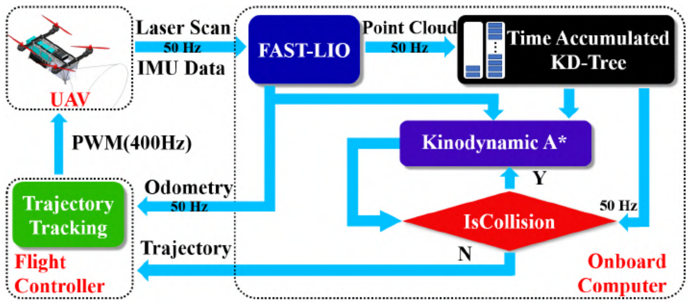


Fig. 4. The software architecture of our overall system. Perception and planning are all running on the onboard computer.

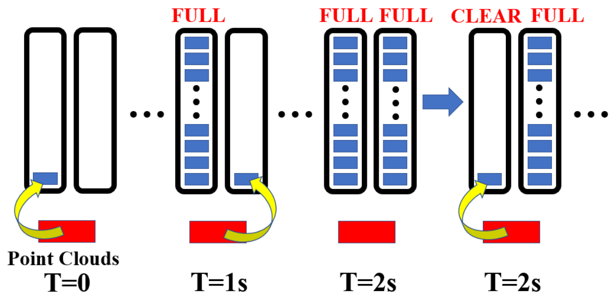


Fig. 5. Update of the two time-accumulated KD-Trees with new scans of points.

Algorithm 1: Software architecture workflow

```
Input: New LiDAR Scan $Scan$ and IMU data $IMU$
Manual Input: Goal point $G$
Output: Odometry $Odom$, Trajectory $Traj_c$
1 $[Odom, Ptclouds] \leftarrow$ FAST-LIO$(Scan, IMU)$;
2 $TimeAccKDTrees$.add$(Ptclouds)$;
3 if IsCollision$(TimeAccKDTrees, Traj_c)$ then
4     $S \leftarrow$ GetState$(Traj_c, T_c + 0.03)$;
5     $Traj_c \leftarrow$ kinoA*search$(TimeAccKDTrees, S, G)$;
6 SendToFlightController$(Odom, Traj_c)$;
```

one (Line 5) and is sent to the flight controller for tracking (Line 6).

We defer the local map representation (Line 2) to the next section. For the trajectory planning (Line 5), we choose the motion primitive-based method, which has been successfully used in high-speed planning [10] and dynamic objects avoidance [20]. More specifically, we directly adopt the Kinodynamic A* search algorithm implemented in [25, 26]. For collision check (Line 3), we perform the check on a set of discrete points along the trajectory by searching its nearest neighbors on the time-accumulated KD-trees. If the point on the trajectory is too close (i.e., safety clearance) to the nearest point in the map, a collision occurs. The number of checking points are assigned according to the length of each motion primitive segment of the trajectory (as apposed to a fixed number 5 in the original method [26]). Finally, the trajectory tracking controller is a cascaded PID controller implemented on Pixhawk4-Mini, and we tune the parameters appropriately beforehand.

B. Time-accumulated KD-Tree

Most existing works rely on occupancy grid map [8, 9, 27], where the space is discretized into small voxels. To filter out sensor noise, ray-casting is usually conducted to estimate the occupancy probability of each voxel along the ray of a point measurement (e.g., OctoMap [1]). Besides filtering out noise points, such ray-casting also enables to distinguish unknown spaces from the free or occupied ones, which could be useful in some planning tasks. Euclidean Signed Distance Field (ESDF) map [6, 28] builds on top of the occupancy grid map and further maintains the signed distance of each voxel to its nearest occupied voxels. Such distance provides the gradient information allowing the optimization of a smooth trajectory. However, it takes additional time to compute the distance information.

Due to the very long measuring range of our lidar (up to 450m as in Table II), ray-casting is very time-consuming, especially when the voxel size is small to accommodate small objects. On the other hand, the lidar has a very low noise ratio (lower than 0.0003%) that makes the ray-casting used in OctoMap unnecessary. In this paper, we directly plan on the point clouds received from the estimation and mapping module. To enable efficient collision check [10], we organize the point clouds into a KD-tree structure.

Considering all historic points in the collision check leads to over-conservative motion planning since the space traveled by dynamic objects will be mistakenly considered as occupied [29]. Moreover, maintaining all points in a single KD-tree is time-consuming due to the large number. On the other hand, using points in the current scan only [10] will lead to incomplete coverage of the FoV and miss potential obstacles on the trajectory.

We propose to use a temporally local map. That is, only the most recent point cloud is used for collision check. We accumulate the point cloud over a certain time, called the accumulation time. For the livox AVIA lidar, the accumulation time is set to one second, which produces sufficiently high-resolution point clouds (see Fig. 3) while retains only recent points on dynamic objects. We further downsample the raw point clouds at a prescribed resolution (e.g., 10cm) to lower the computation. Note that the down-sampling resolution, unlike the voxel size in OctoMap, does not affect the detection of objects smaller than the resolution, but may only inflate the obstacle space slightly.

The local map is organized into two KD-Trees, one static and one dynamic growing (see Fig. 5), each KD-Tree stores up to the accumulation time point clouds data. The update of each KD-tree is detailed in Algorithm 13, where $H = 50$ denotes the maximum number of lidar scans stored in one KD-Tree, and $N = 2$ denotes the number of KD-Trees. In the beginning, the two KD-Trees are both empty. Once a new scan of points arrives, the KD-Tree that is still not full is obtained (Line 1 - 5). Then, the new scan of points is added to the point cloud already on the tree (Line 10). The updated point clouds are downsampled at the prescribed resolution (Line 11) and used

Algorithm 2: Time-accumulated KD-Tree

```
Input: Point clouds $NewPoints$ in the new scan
Output: Two KD-Trees $KDTree[2]$
Parameters: $H = 50 \quad N = 2$
1  if $ScanInputNum \geq H * N$ then
2      $ScanInputNum = 0$;
3      $TreeInputNum = 0$;
4  else
5      $TreeInputNum = \textbf{floor}(ScanInputNum/H)$;
6  if $ScanInputNum \bmod H = 0$ then
7      $CloudAccumulate.\textbf{Clear}()$;
8      $CloudAccumulate = NewPoints$;
9  else
10     $CloudAccumulate.\textbf{add}(NewPoints)$;
11 $CloudFilt = \textbf{VoxelGridFilter}(CloudAccumulate)$;
12 $KDTree[TreeInputNum].\textbf{build}(CloudFilt)$;
13 $ScanInputNum = ScanInputNum + 1$;
```

to build up a new KD-Tree replacing the existing one (Line 12). The above process repeats until the current KD-Tree is full (Line 6), where the new scan is accumulated from scratch (Line 7-8) and saved to the other KD-Tree (this will override the existing point cloud already on the other KD-Tree).

The two KD-Trees save point cloud up to twice the accumulation time, which is sufficient to cover the lidar FoV with great details. When performing collision checks, both KD-Tree should be used: the static one provides information of the environments while the dynamic one timely detects dynamic obstacles at the lidar scan rate (i.e., 50Hz).

We would like to point out that building only one KD-Tree (as apposed to two in our approach) is not feasible. If a single KD-Tree is used, a purging of the tree (when it is full) will delete all map points and lead to over-optimistic collision check. One may also wonder why not use more than two KD-Trees. While this is certainly feasible, it requires to performing collision checks on all trees, which in turn increases the query time. To study the effect of the number of KD-Trees, we performed a quick comparison in Table. III, where a total number of 100 scans of points are organized into different numbers of KD-Trees. As expected, when N increases, the average time for adding a new scan decreases since the number of scans to build on each KD-Tree reduces. However, at the same time, as each point on the trajectory needs to carry out a nearest neighbor search on every KD-Tree, the average search time per each checking point increases linearly with N. As a result, the overall planning time increases with N, the number of KD-Trees.

Remark: while the proposed time-accumulated KD-Trees is very efficient in computation time (see Section V), it shares the same drawback of a point-cloud map, the inability to distinguish unknown spaces from free ones. Moreover, similar to all other local maps, the time-accumulated KD-Trees maintain points over a local time window, so points of obstacles moving out of the window will not be considered

TABLE III
TIME-ACCUMULATED KD-TREES WITH DIFFERENT PARAMETERS

N	H	Update time	Search time/point	Planning time (15m)
2	50	1.06ms	4us	11.5ms
4	25	0.663ms	8us	20ms
5	20	0.574ms	10us	25ms

by the planner. This will lead to two problems: (1) the UAV may collide with obstacles that lie in the unmapped area or move out of the local map; (2) a planner based on such a partial map may fail to plan a globally feasible trajectory in very complicated environments (e.g., maze). In practice, the first issue could be addressed by constraining the trajectory within the LiDAR FoV, while the second one by a lower-rate global planner on an additional global map.

V. RESULTS

In this section, we present experiment results to validate our system. We first test the minimal size of both static and dynamic obstacles that our UAV could avoid. Then, we present some more challenging real-world experiments, including an indoor office corridor, an outdoor forest environment, and an outdoor cluttered environment full of tree crowns (see Fig. 6). All the environments are unmodified except the possible dynamic small obstacles (e.g., narrow bar, tree branches) intentionally added to the scene. Each experiment is successfully repeated at least twice. In all experiments, the parameters of the system are set as follows according to the actual situation: the down-sampling resolution is set to 10cm when building the two time-accumulated KD-Trees. The safety clearance between UAV and obstacles is set to 45cm considering the actual size of the UAV and the possible obstacle shrink caused by point cloud down-sampling. For the kinodynamic A* based planning [26], we set $v_{\max}$ = 2m/s, $a_{\max}$ = 2m/s^2 (or $v_{\max}$ = 5m/s, $a_{\max}$ = 3m/s^2 for high speed experiment), and the time duration of one motion primitive is $T_s = 0.6$s. The snapshots of experiments are shown in Fig. 6 (a-c). Due to the use of lidar, a byproduct of our system is a high-resolution high-accuracy 3D map of the environments built in real-time as shown in Fig. 6 (d-f). We supply a video attachment (also available at https://youtu.be/mjtmpEYwQsI) that better visualizes the flight performance.

A. Indoor experiments with static and dynamic obstacles

First, we conduct a series of experiments to determine the smallest static obstacle that our system is able to detect, and found that a bar with the minimum diameter of 9mm can be reliably mapped, shown in Fig. 7(a). Then we keep waving the 9mm bar in front of the UAV and it could still be successfully avoided as shown in Fig. 7(b). After this simple validation experiment, we test our system in more realistic indoor environments.

Instead of flying in a man-made indoor environment [10, 26], we choose an unmodified office corridor, as shown in Fig. 6(a). The UAV can autonomously fly to a 9 meters away goal point while avoiding both the static and dynamic obstacles with a maximum velocity of 2m/s. During the flight, we suddenly raise a narrow-bar shape obstacle (diameter 20mm)

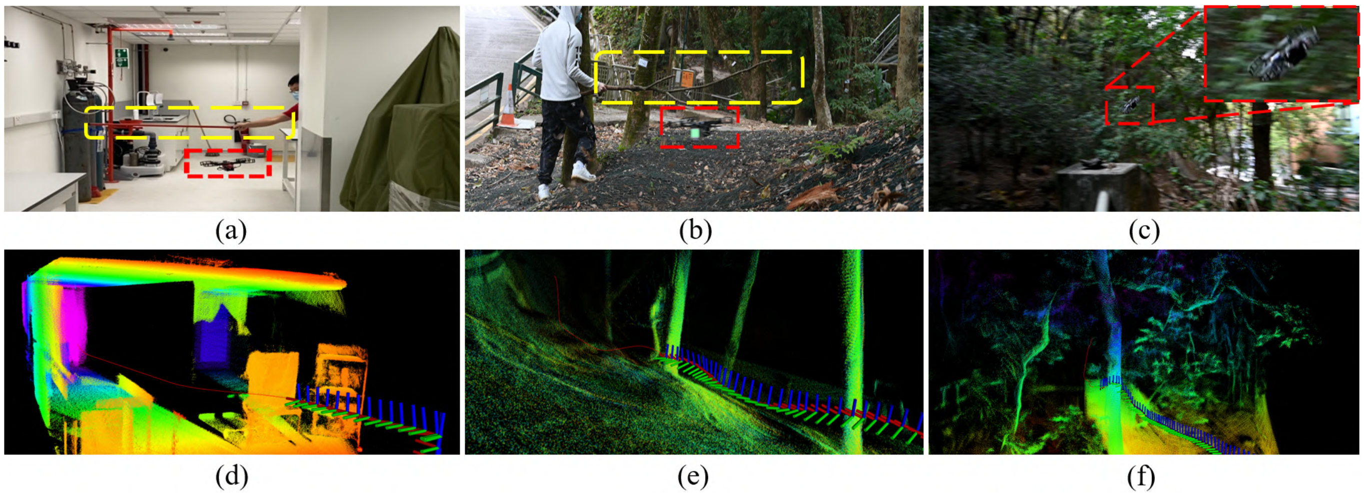


Fig. 6. We test our developed system in both indoor and outdoor environments: (a) an indoor office corridor with a narrow bar of diameter 20mm being intentionally raised to block the UAV flight path; (b) outdoor forest with a tree branch of diameter 10-20mm being intentionally lowered to block the UAV flight path; (c) a hillside with cluttered tree crowns. (d-f), the point cloud map built in real-time as the UAV flies in the three tested environments. Notice that (e) is the scene in (b) but rendered at an apposite view angle.

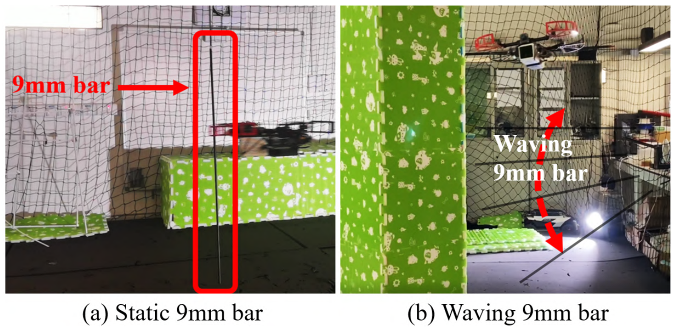


Fig. 7. Avoiding a static and moving bar of 9mm diameter. It is the thinnest bar that our UAV system can avoid.

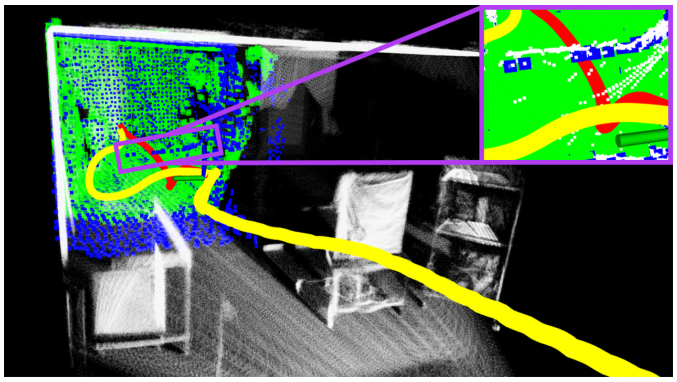


Fig. 8. Avoiding a dynamic 20mm diameter bar in an indoor environment. The white point clouds represent all points accumulated so far, the blue and green point clouds represent points in the first and second KD-tree of our local map, respectively, used for re-planning. The yellow curve is the current trajectory and the red one is the re-planned one after detecting the new bar position. The upper right corner is a zoomed view of the points collected on the moving bar.

when the UAV is about 2.5m away and hold it to the UAV flight path. Still, our system can detect this dynamic small obstacle and avoid it in a very short time. As shown in Fig. 8, a smooth trajectory (the yellow curve) is first generated to the target point. With the narrow-bar raising, the trajectory is detected to have a collision with the updated environment, and the planning module starts replanning. After several times of replanning, a final safe trajectory (the red curve) that passes under the bar is planned. In the upper right corner of Fig. 8, we show the zoomed-in view of the narrow bar where the blue points represent the points in time-accumulated KD-Trees used for replanning and white points are points in the past. It is seen that the bar movement profile is clearly captured by the LiDAR points. By using a temporally local map, the re-planner is able to utilize the space where dynamic obstacles swept over (indicated by white points in the upper right corner) to avoid the bar at its current position (indicated by the blue points in the upper right corner).

B. Outdoor experiments with natural and dynamic obstacles

In order to validate the system in more natural and complex environments, we also carry out experiments in a forest. In the first scene, the quadrotor passes several trees and a dynamic branch (see Fig. 6(b)) to reach the target point 15m away. During the flight, we suddenly swept down a tree branch

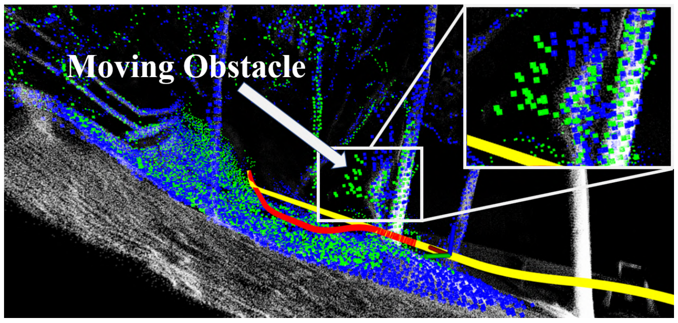


Fig. 9. Avoiding a dynamic tree branch in outdoor environment. The meanings of point clouds and trajectories are the same as those in Fig. 8.

(diameter 10-20mm) when the UAV is about 2.5m away and hold it on the UAV flight path (see Fig. 1). Fig. 9 shows the moment the re-planning is triggered. It can be seen that the tree branch movement profile is clearly seen (the zoomed view in the upper right corner). Once the tree branch falls within the safety clearance of the current trajectory under tracking (the yellow trajectory), the re-planning is triggered to produce a new trajectory (the red trajectory) that avoids the tree branch.

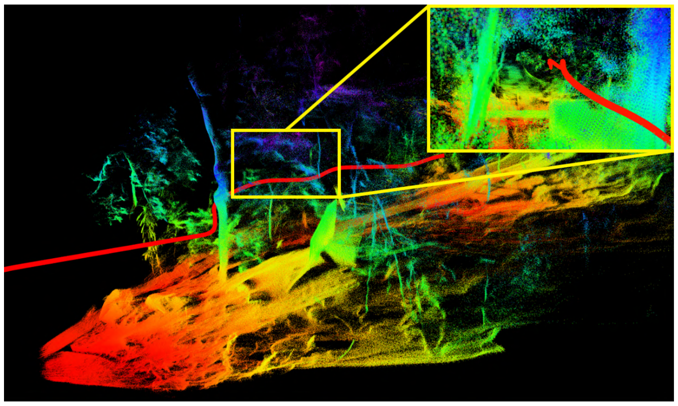


Fig. 10. Global point cloud map of the second outdoor environment, with leafy trees, stones and a water pipe. The upper right corner shows the narrow corridor and the leaves around the trajectory.

Unlike the indoor experiment, all points on the tree branch are on the KD-Tree. This is because the tree branch falls within the safety clearance, which triggers the re-planning, within twice the accumulation time.

High-speed experiments are also carried out in the first outdoor environment. Peak speed of 5.5m/s and average speed of 3m/s are achieved, which shows our approach and system are capable of handling such challenging situations.

The second outdoor environment is a very complex hillside full of cluttered tree crowns, stones, water pipes, and with significant height variation, as shown in Fig. 10. We set the target point of the UAV to be 18 meters away from its takeoff position. Our method is able to find a safe path through cluttered tree crowns and flies to the target point. A part of the path is in a narrow corridor as shown in the upper right corner of Fig. 10 but the UAV still succeeds.

More than ten experiments are additionally conducted but not included in this paper due to the space limit. We refer readers to the accompanying video for a more comprehensive demonstration. Generally, as long as the environment is not too cluttered (like mazes where the local planner may fail to find a feasible trajectory due to the partial local map) and the dynamic small obstacle does not appear out of the LiDAR's measuring range (i.e., below 1m as shown in Table II), our system would always succeed.

C. Breakdown of running time

In general, our system achieves real-time capability and is able to avoid dynamic small obstacles. Fig. 11 shows a timing breakdown of each module for the 15m target in the first outdoor environment, our system took an average of 16.5ms in total, including 3ms for state estimation and mapping, 1ms for updating time-accumulated KD-Trees, and 12.5ms for kinodynamic A* search. Note that our re-planning is triggered only when the current trajectory collides with an obstacle instead of re-planning at a fixed rate, hence the worst case in Fig. 11 rarely occurs.

D. Comparison of mapping method

We finally study the applicability of OctoMap [1] in detecting small dynamic objects and compare its performance

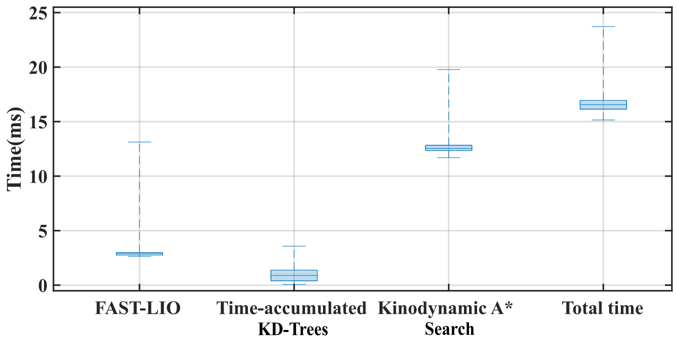


Fig. 11. Running time statistics of each module for the 15 meters away target in the first outdoor forest environment.

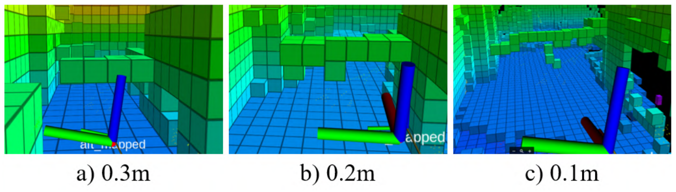


Fig. 12. Different resolution of OctoMap detecting dynamic 20mm bar.

with our time-accumulated KD-Trees. Taking the indoor office corridor as an example, we build OctoMaps at 0.3m, 0.2m, and 0.1m resolution, respectively. Due to the very low false alarm rate of our lidar (see Table. II), we tune the probabilities for hits and misses from the default value [0.7/0.4] to [0.93/0.48]. With these parameter tuning, the Octomap can reliably detect the dynamic bar at all these three resolutions as shown in Fig. 12.

TABLE IV
AVERAGE UPDATE TIME OF OCTOMAP AND TIME-ACCUMULATED KD-TREES

Environment	Octo (0.3m)	Octo (0.2m)	Octo (0.1m)	KD-Trees
Indoor Env	1.6ms	2.76ms	8.2ms	1.09ms
Outdoor Env1	9.6ms	19.7ms	67.6ms	1.06ms
Outdoor Env2	2.4ms	4.2ms	15.4ms	1.25ms

The strength of our time-accumulated KD-tree is mainly on the computation efficiency. We compare the update time of OctoMap in different resolutions with that of our time-accumulated KD-Trees (0.1m downsampling) in the above three environments (one indoor, two outdoor). As shown in Table IV, our approach achieves the lowest processing time in all cases. As expected, the difference between OctoMap and time-accumulated KD-tree is small in indoor environments due to the confined space. In open outdoor environments, the superiority of our time-accumulated KD-tree is more evident. It is also noticed that OctoMap takes less time in the second outdoor environment than the first one although the target position is further (18m versus 15m), this is because the second outdoor environment is much more cluttered, full of leafy trees, so most LiDAR measurements are very short in range. On the other hand, our time-accumulated KD-trees take consistently lower time as it depends on the number of points (which is nearly constant in every new scan) instead of the points' locations.

VI. Conclusion And Future Work

This paper proposes an autonomous UAV system where all the state estimation, mapping, and trajectory planning are performed onboard based solely on lidar and inertial measurements. The design of the hardware and software system is presented. Various indoor and outdoor experiments are conducted. Results show that the proposed system is able to fly safely in some cluttered environments while avoiding dynamic small obstacles.

The proposed system uses an existing Kinodynamic A* (re-) planning method, where the control (i.e., acceleration) space is discretized to produce motion primitives. The searched trajectory has a discontinuous acceleration trajectory, preventing the UAV from higher flying speeds requiring accurate trajectory tracking. The small number (i.e., 27) of control space motion primitives also limits the UAV from operating in an even tighter space or higher re-planning rate. While these two issues can be mitigated by a finer discretization of the control space, the planning time would dramatically increase. In the future, we would like to improve the re-planning module to enable more challenging environments and higher flying speeds.

References

[1] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, Apr. 2013.

[2] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "Svo: Semidirect visual odometry for monocular and multicamera systems," *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 2016.

[3] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[4] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen, "Autonomous aerial navigation using monocular visual-inertial fusion," *Journal of Field Robotics*, vol. 35, no. 1, pp. 23–51, Jan. 2018.

[5] N. Bucki, J. Lee, and M. W. Mueller, "Rectangular Pyramid Partitioning using Integrated Depth Sensors (RAPPIDS): A Fast Planner for Multicopter Navigation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4626–4633, July 2020, arXiv: 2003.01245.

[6] H. Oleynikova, Z. Taylor, M. Fehr, J. Nieto, and R. Siegwart, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-Board MAV Planning," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1366–1373, Sept. 2017, arXiv: 1611.03631.

[7] M. Liu, "Robotic Online Path Planning on Point Cloud," *IEEE Transactions on Cybernetics*, vol. 46, no. 5, pp. 1217–1228, May 2016, conference Name: IEEE Transactions on Cybernetics.

[8] Sikang Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden: IEEE, May 2016, pp. 1484–1491.

[9] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online UAV replanning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea: IEEE, Oct. 2016, pp. 5332–5339.

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

[11] J. Tordesillas, B. T. Lopez, and J. P. How, "FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments," *arXiv:1903.03558 [cs]*, May 2020, arXiv: 1903.03558.

[12] P. R. Florence, J. Carter, J. Ware, and R. Tedrake, "NanoMap: Fast, Uncertainty-Aware Proximity Queries with Lazy Search Over Local 3D Data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, May 2018, pp. 7631–7638.

[13] J. Ji, Z. Wang, Y. Wang, C. Xu, and F. Gao, "Mapless-Planner: A Robust and Fast Planning Framework for Aggressive Autonomous Flight without Map Fusion," *arXiv:2011.03975 [cs]*, Nov. 2020, arXiv: 2011.03975.

[14] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, July 2017.

[15] M. Dharmadhikari, T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis, "Motion Primitives-based Path Planning for Fast and Agile Exploration using Aerial Robots," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020, pp. 179–185.

[16] J. Zhang, C. Hu, R. G. Chadha, and S. Singh, "Falco: Fast likelihood-based collision avoidance with extension to human-guided navigation," *Journal of Field Robotics*, p. rob.21952, Apr. 2020.

[17] J. L. Sanchez-Lopez, M. Wang, M. A. Olivares-Mendez, M. Molina, and H. Voos, "A Real-Time 3D Path Planning Solution for Collision-Free Navigation of Multirotor Aerial Robots in Dynamic Environments," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1-2, pp. 33–53, Feb. 2019.

[18] R. E. Allen and M. Pavone, "A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance," *Robotics and Autonomous Systems*, vol. 115, pp. 174–193, May 2019.

[19] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous Flights in Dynamic Environments with Onboard Vision," *arXiv:2103.05870 [cs]*, Mar. 2021, arXiv: 2103.05870.

[20] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, p. eaaz9712, Mar. 2020.

[21] R. Madaan, "Wire detection, reconstruction, and avoidance for unmanned aerial vehicles," Master's thesis, Carnegie Mellon University, Pittsburgh, PA, August 2018.

[22] H. Zhang, W. Yang, H. Yu, H. Zhang, and G.-S. Xia, "Detecting power lines in uav images with convolutional features and structured constraints," *Remote Sensing*, vol. 11, p. 1342, 06 2019.

[23] Z. Liu, F. Zhang, and X. Hong, "Low-cost retina-like robotic lidars based on incommensurable scanning," *IEEE/ASME Transactions on Mechatronics*, p. 1–1, 2021.

[24] W. Xu and F. Zhang, "FAST-LIO: A Fast, Robust LiDAR-inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter," *arXiv:2010.08196 [cs]*, Oct. 2020, arXiv: 2010.08196.

[25] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 2872–2879.

[26] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight," *arXiv:1907.01531 [cs]*, July 2019, arXiv: 1907.01531.

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

[28] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4423–4430.

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