Perception and Avoidance of Multiple Small Fast Moving Objects for Quadrotors with Only Low-cost RGBD Camera

Minghao Lu†1, Han Chen†2 and Peng Lu∗1

Abstract— **The autonomous navigation of unmanned aerial vehicles in a rapidly changing environment, such as avoiding small fast moving objects with onboard sensing, still remains a challenge. In this paper, we propose a complete system that only relies on a lightweight RGBD camera to achieve fast and accurate perception and avoidance of small dynamic obstacles, whereas navigating in a complex environment. Firstly, we detect the moving objects by Yolo-Fastest in RGB frame, obtain the 3D information with the depth image, and track the multiple detected objects with our proposed 3D-SORT (Simple Online and Real-time Tracking in Three-dimensional Space) algorithm. To achieve fast dynamic avoidance, we design an effective method to generate the optimized smooth trajectory to dodge all the static and dynamic obstacles with the predicted moving objects' trajectories. Finally, we integrate the above methods on our UAV platform, and demonstrate the performance of our system by testing thoroughly in simulation and real-world experiments.**

I. Introduction

With robotics developing from its birth to the present, an intention always stands unchanged: people hope that robots can help humans undertake some heavy and dangerous work, such as exploring dangerous and unknown environments [1]. Unmanned aerial vehicles (UAVs), especially micro UAVs, have become the best choice to explore the unknown environment because of their motion flexibility in space.

After the development of aerial autonomy technology in recent decades, autonomous flight of UAVs in static unknown environments has been achieved [2]. However, those fast moving objects in the environment still pose severe threats to UAVs, such as bats in the cave and birds in the sky. In real-world applications, UAVs attacked by birds are often reported [3], and UAVs may also face with intentional attacks. Therefore, it is necessary to enable UAVs to avoid attacks from small fast-moving objects.

From a technical point of view, the difficulty for the micro UAVs to avoid fast objects mainly comes from three aspects. First, the latency of perceiving small objects is supposed to be short enough, and shorter than the theoretical upper bound. The latency bound is related to the maximal sensing distance and the object detecting time cost. Second, the motion planning method should be effective in computing, also it should be feasible for navigation tasks. Finally, all the algorithms are deployed on an onboard computer with very limited computing resources, to run in real-time.

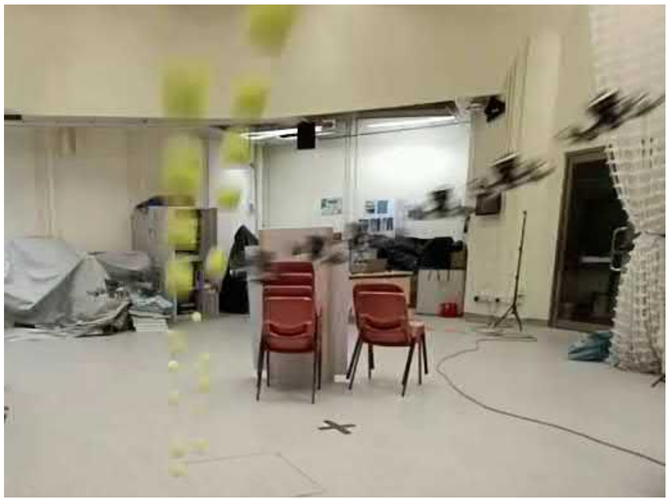

Fig. 1: Sequence of our UAV dodging the multi-balls attack. The two tennis balls are thrown out simultaneously. No motion capture is used.

The proposed system in this paper, to our best knowledge, is the first one that tackles all the above mentioned challenges. In this paper, we consider the perception and avoidance of small fast-moving objects (balls) for quadrotors using only a low-cost depth camera. Comparing to existing studies [3]–[5], we only use a low-cost depth camera instead of event cameras, and we consider multiple objects which also introduces more challenges to the multi-object tracking and obstacle avoidance. Furthermore, our avoidance targets in tests are tennis balls which are smaller, and more difficult to perceive, than the basketball considered in existing works.

To achieve the goal, we propose a 3D-SORT (Simple Online and Real-time Tracking in 3D Space) algorithm to track the multiple objects and estimate their velocity and acceleration in 3D space. A network is trained carefully to detect small objects at a relatively far distance. Then, we simplify the trajectory optimization in [6] and extend it to the dynamic environment to generate the spatial-temporal optimal trajectory. Compared to the method in the related works [3]–[5], our approach can generate a much more safe and energy-saving trajectory to avoid dynamic objects. We perform comprehensive tests in simulation and real-world to validate the performance of our method. Our contributions are summarized as follows:

1) We propose a novel 3D-SORT algorithm that can

† Equal contribution

∗ Corresponding Author

1 Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, China. email: minghao0@connect.hku.hk, lupeng@hku.hk

2 Department of Aeronautical and Aviation Engineering, The Hong Kong Polytechnic University, Hong Kong, China. email: stark.chen@connect.polyu.hk

This research is supported by the Seed Funding for Strategic Interdisciplinary Research Scheme and Platform Technology Fund.

achieve a fast and effective tracking of multiple fast-moving objects. The algorithm is composed of unique state estimation and data association method for information fusion.

2) We propose a hierarchical trajectory generation method composed of a kinodynamic path searching method and a gradient-based optimization. Even both static and dynamic obstacles are considered, the whole procedure is finished in milliseconds. The energy and time cost and safety are much improved compared to the state-of-the-art works.
3) We are the first to demonstrate that quadrotors can avoid multiple small (radius < 0.035 m) fast-moving (speed $>$ 5 m/s) objects attacking from a close (< 5 m) distance with only a low-cost depth camera. We will release our source code[1] to the public.

II. Related Work

Among most existing related works that demonstrate successful avoidance towards fast moving objects, the system can be regarded to be composed of two major parts: objects perception and vehicle motion planning. In practice, the perception part brings greater challenges and plays a more critical role in the system. Also, the related motion planning methods in the motion planning part still remain to be improved in safety and optimality.

A. Perception of small fast moving objects

To avoid fast moving objects, the two critical factors are the perception latency and the detection range of distance [7]. To our best knowledge, all the existed works that dodge the fast moving objects are based on event cameras [3]–[5], because event cameras benefit from the low latency in observing dynamic events. A learning-based method is also proposed to merge the two-stage algorithm into a series of neural networks to estimate both the ego-motion and the motion of independently moving objects [4]. Besides, Some works explore the techniques specifically about the motion compensation and the motion segmentation based on event cameras [8], [9]. However, the resolution for those light-weight (typically below 50 g) event cameras is only about 320 by 240 pixels, which is much lower than that of the state-of-the-art standard cameras for robotic applications [3]. This will lead to greater position estimation errors of objects. Also, the event camera is only sensitive to dynamic objects when they are close, otherwise it is difficult to distinguish moving objects from the background events [7]. For depth cameras, a similar case is that a drone can catch a falling ball with only an RGB camera to perceive the ball's state [10], suggesting that avoiding a flying ball with an RGBD camera is also possible. In recent years, some researchers achieved successful avoidance towards relatively slow moving obstacles (such as walking pedestrians) with onboard vision [11]–[13]. Also, object tracking methods with image input is extensively studied over years [14], [15]. However, the research about dodging fast, small objects with only a single onboard RGBD camera during autonomous navigation tasks still remains blank. For lidars, learning-based object detecting and tracking methods show robust results [16], but all the related methods are pretty far from real-time computing on the micro onboard computers, also the message update rate and the sparse spatial resolution of lidar is not satisfactory for detecting fast moving small objects. One most recent work demonstrates the avoidance towards small objects with a solid-state lidar on the UAV [17], however, as no object detection and velocity estimation is involved, all objects are considered in the same way as for static objects.

B. Dynamic Obstacle Avoidance

The obstacle avoidance problem for UAVs, especially for quadrotors, has been wildly studied and explored in recent years. It is achieved by planning the continuous safe motion primitives based on the kinematic model and executing the primitives with the flight controller. Several complete solutions in the community have demonstrated fast, robust flights in a clustered static environment, and the motion primitives can be solved in real-time [2], [18], [19]. For avoiding dynamic obstacles, model predictive control [12] is one feasible solution, and it can exploit the full dynamics of the aircraft [20]. A gradient-based polynomial trajectory optimization method [11] shows a big improvement in computation efficiency and trajectory optimality. For fast small objects, researchers proposed methods for catching flying balls earlier than avoiding them. An effective sampling-based motion primitive generation method [21] is proposed to show a high success rate to catch a randomly thrown ball. To avoid fast objects, the existed works [3]–[5] all choose to adopt a primary and simple method to plan the motion, such as artificial potential field (APF), because it is very easy to compute and the computation time cost is short enough. However, such a method is not suitable for complex scenarios when dense static obstacles and multiple fast objects are existing, because of the local minimum problem and the absence of complete collision check.

III. Perception of Small Fast-Moving Objects

In this section, we will first present the multi-object detection. Then, we will present our proposed 3D-SORT algorithm for the tracking of multiple small fast-moving objects using RGB and depth images.

A. Detection of Small Fast-Moving Objects

Assume that there is a specific object in the environment attacking the drone. At time t, a total of N objects enter the camera field of view, we use YOLO algorithm to detect the objects in the RGB frame. Although there are some fast object detecting methods directly based on clustering the spatial information, such methods are not robust for small objects because they are hard to be distinguished from noise, also the clustering algorithm can not distinguish objects when they are close [3], [11], [12]. Given the detection result,

[1]https://github.com/arclab-hku/fast-dodging

we can obtain the central pixel position (u_c, v_c) and pixel scale (s_b, r_b) of the objects, where s_b represents the area of the detection box, and r_b represents the aspect ratio of the detection box. By projecting the region of interest (ROI) of the detection boxes to depth image frame, we can get the depth of the objects, expressed as d. Then the position of an object in world frame can be described by

$$M_1 M_2 [x, y, z, 1]^T = d[u, v, 1]^T, \tag{1}$$

where M_1 represents the camera intrinsic matrix, and M_2 represents the rotation matrix from world frame to camera frame. The detection result at time t can be presented as a set ξ_t defined as:

$$\xi_t = \{(p_i, c_i) | i \in (1, N)\}, \tag{2}$$

where N is the amount of detected moving objects, $p_i \in \mathbb{R}^3$ denotes the object position in 3D space, and $c_i = [s_b^i, r_b^i]^T$ is the pixel scale vector, i is the index of the dynamic object among multiple ones.

B. Tracking of Small Fast-Moving Objects

To avoid multiple fast-moving objects, it is necessary to predict their trajectories to plan the avoidance beforehand. However, it is difficult to predict the trajectories for fast-moving objects using low-cost depth cameras. In this section, we propose a 3D-SORT algorithm, which is presented in **Algorithm 1**, to achieve this.

Our proposed 3D-SORT algorithm is distinct from the 2D version [14], [15]. The 3D-SORT algorithm can track and predict the 3D position of the objects due to its unique state estimation and data association techniques as follows:

Algorithm 1 Proposed 3D-SORT

```
Input: Detection indices $\xi_t = \{(P_i, C_i) | i \in [1, N]\}$, Tracking indices $\chi_t = \{(\widehat{P}_h, \widehat{C}_h) | h \in [1, B]\}$
1:  Compute cost matrix $\mathcal{T}_{B\times N}$ using Equation 7
2:  Assign prediction and detection via Hungarian algorithm
3:  Update matched pairs set $\mathcal{M}_p$, unmatched detection set $U_d$, unmatched tracker set $U_t$
4:  for pair in $\mathcal{M}_p$ do
5:      if $\mathcal{T}(pair) > h_{min}$ then
6:          update the tracker by two KFs to obtain $(\widehat{P}, \widehat{C})$
7:      else
8:          $U_d \leftarrow U_d \cup (P, C)$ in pair
9:  for $(P, C)$ in $U_d$ do
10:     $\chi_t \leftarrow \chi_t \cup (\widehat{P}, \widehat{C})$
11: for $(\widehat{P}, \widehat{C})$ in $U_t$ do
12:     if $Age_{(\widehat{P}, \widehat{C})} > Age_{max}$ then
13:         $\chi_t \leftarrow (\widehat{P}, \widehat{C}) \setminus \chi_t$
```

1) State Estimation: Here we describe our state estimation method. We conduct separate state estimation for the object's space motion and pixel scale. The inter-frame displacements of each object in world frame is approximated as a linear constant acceleration model, and the change in pixel scale is approximated as a first-order differential model. According to the detection results in (2), the motion model of each target can be formulated as

$$P_{i_\kappa} = \left[p_{i_\kappa}^T, \dot{p}_{i_\kappa}^T, \ddot{p}_{i_\kappa}^T\right]^T, P_{i_{\kappa+1}} = \mathbf{A}^{P_i} P_{i_\kappa} + v_\kappa^{P_i}, z_\kappa^{P_i} = \mathbf{H}^{P_i} P_{i_\kappa} + w_\kappa^{P_i}, \tag{3}$$

$$\mathbf{A}^{P_i} = \begin{bmatrix} I_{3\times3} & \Delta t \cdot I_{3\times3} & 0.5\Delta t^2 \cdot I_{3\times3} \\ 0_{3\times3} & I_{3\times3} & \Delta t \cdot I_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & I_{3\times3} \end{bmatrix}, \tag{4}$$

$$\mathbf{H}^{P_i} = \begin{bmatrix} I_{3\times3} & I_{3\times3} & 0_{3\times3} \end{bmatrix}, \tag{5}$$

where κ denotes the κth time stamp, Δt is the time interval between two adjacent frames of images, v is the normally distributed process noise and w is the normally distributed measurement noise. Basically, the pixel scale model ($C_{i_\kappa} = \left[c_{i_\kappa}^T, \dot{c}_{i_\kappa}^T\right]^T$) is similar to Equation (3)-(5), the difference is only the state transition matrix A and the measurement matrix H:

$$\mathbf{A}^{C_i} = \begin{bmatrix} I_{3\times3} & \Delta t \cdot I_{3\times3} \\ 0_{3\times3} & I_{3\times3} \end{bmatrix}, \ \mathbf{H}^{C_i} = \begin{bmatrix} I_{3\times3} & 0_{3\times3} \end{bmatrix}. \tag{6}$$

Using this, we can create two Kalman filters to conduct the prediction and update of the two models (**Algorithm 1** Line 10). The posterior probability outputs of the filters are $\widehat{P}_i$ and $\widehat{C}_i$ (Line 6). Compared to previous methods, the motion model with constant acceleration can predict the motion state of the objects more accurately and generate better trajectory. The pixel scale is only related to the data association of the objects, which will be presented in the next part.

2) Data Association: In assigning detected objects to existing targets, we firstly project $\widehat{P}_i$ to pixel frame according to Equation 1, then according to $\widehat{C}_i$, we can get the prediction bounding boxes in the current image. The assignment cost matrix combines location and color features. The location cost matrix is computed as the intersection-over-union (IOU) distance between each detection and all predicted bounding boxes from the existing targets, and the color cost matrix is computed as Bhattacharyya Coefficient between the HSV channels histogram of each detection ROI and all predicted ROI. Although we may encounter such a situation that the appearance of the objects is very similar, the lightning situation on each object varies. Thus, the color information in HSV space can distinguish similar objects at different positions. For the ith $(i = 1, 2, ..., B)$ prediction box and hth $(h = 1, 2, ..., N)$ detection box, the cost matrix $\mathcal{T}_{ih}$ can be expressed as

$$\mathcal{T}_{ih} = a * f_{iou}(i, h) + b * f_{hsv}(i, h), \quad a + b = 1, \tag{7}$$

$$f_{iou}(i, h) = \frac{\Omega_{ih}}{A_i + A_h - \Omega_{ih}}, \tag{8}$$

$$f_{hsv}(i, h) = 1 - \sqrt{1 - \frac{1}{\sqrt{\bar{H}_i \bar{H}_h n_{max}^2}} \sum_{n=1}^{n_{max}} \sqrt{H_i(n) H_h(n)}}, \tag{9}$$

where A_i and A_h are the area of the prediction and detection bounding boxes, respectively. Ω_{ih} is the overlap area of the two bounding boxes, $H()$ is the HSV histogram function, $\bar{H}$ is the mean of $H()$, and n is the value of H or S or V channel (see Fig. 2). $\mathcal{T}_{ih} \in (0, 1)$ indicates the matching degree where a larger value means a better match. The assignment is solved optimally using the Hungarian algorithm (**Algorithm 1** Line

1-3). Additionally, a minimum threshold h_{min} is imposed to reject assignments where the cost between detection and target is less than h_{min} (Line 5), and trackers lost of matching for three continuous times will be deleted (Line 11-13). The process of the assignment method is shown in Fig. 2.

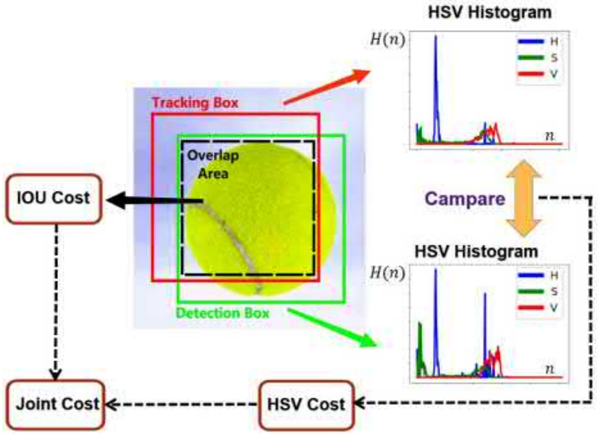

Fig. 2: Principle of our data association method. The ratio of the overlapping area of the detection box and the tracking box to the total area represents the spatial correlation, and the comparison of the HSV histogram in the detection box ROI and the prediction box ROI represents the color correlation.

3) Trajectory Prediction: Based on the filtered 3D position p_{t_0}, velocity v_{t_0} and acceleration a_{t_0} at current time t_0, we can represent the position trajectory $p_b(t)$ of the ith dynamic object at any future time t as

$$p_b^i(t) = p_{t_0}^i + v_{t_0}^i(t-t_0) + \frac{1}{2}a_{t_0}^i(t-t_0)^2. \tag{10}$$

IV. Trajectory Planning

In this section, we present our trajectory planning method based on gradient optimization with kinodynamic and spatial constraints. The planner takes the safety corridor, the dynamic objects' trajectory, and the initial and goal states of the vehicle as the inputs. The output is a dynamically feasible and safe trajectory.

A. Hierarchical planning

At the beginning of the trajectory planning, a collision-free path connects the start point and the goal is searched. We modify the original 3D hybrid A* algorithm [18] to search a kinodynamic-feasible path efficiently and directly on point cloud. A KD-tree [22] is built up with the most recent point cloud instead of a voxel map to reduce latency. In the safety check function, we do the nearest neighbor query with the sample point on the proposed path segment. If the distance to the neighbor is greater than the pre-assigned safety margin, then the queried sample point is safe.

Then, the safety corridor is built, composed of a series of polyhedrons or spheres connected end to end. The trajectory is allowed to vary inside the corridor, which is more flexible than directly optimizing the trajectory with the guidance of the path [**?**], and the computing efficiency can be improved.

For brevity, we directly define the SFC as a list of $m-1$ convex polyhedrons, which is generated from a path of m waypoints. Each polyhedron $\mathscr{H}_j$ $(0<j<m)$ is defined as

$$\mathscr{H}_j = \left\{ \mathbf{x} \in \mathbb{R}^3 \mid \left(\mathbf{x} - \hat{p}_j^k\right)^{\mathrm{T}} \vec{n}_j^k \le 0, k = 1,2,\cdots,N_j \right\}, \tag{11}$$

where N_j is the number of faces of the j^{th} polyhedron, and $\hat{p}_j^k, \vec{n}_j^k$ are the point and the corresponding normal vector on the face. At last, the trajectory is optimized under the constraints of SFC.

B. Trajectory optimization

With MINCO (minimum control) class [6], we can directly control the spatial and temporal profile of a trajectory. Thus, we are able to solve a safe trajectory in a very short time, which is optimal in time and energy cost (as the user-defined objective function), while satisfying dynamic feasibility.

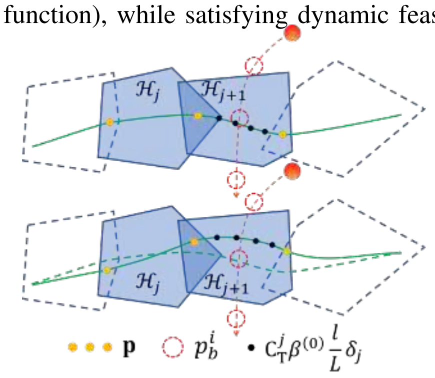

Fig. 3: The optimization of the trajectory to avoid a dynamic object. Some sample points on the trajectory (black dots) are checked not safe (inside the red dashed circle) in the upper figure. After the optimization, the sample points are all safe, as shown in the lower figure. The optimization changes the position of the original waypoints (yellow dots), as well as the time allocation of the trajectory pieces. Each waypoint must stay inside the overlapping part of the two adjacent polyhedrons, and the trajectory only varies inside the safety corridor.

We use a series of polynomials to define the trajectory in each dimension (x, y, and z) in 3D space, denoted as:

$$p_j(t) = \sum_{j=1}^{m-1} \mathbf{C}_j^{\mathrm{T}} \beta\left(t - T_{j-1}\right), \ T_{j-1} \le t < T_j \tag{12}$$

where $\mathbf{C}_j \in \mathbb{R}^{6\times3}$ is the coefficient matrix of the j^{th} piece and $\beta(t) = \left[1, t, \cdots, t^5\right]^{\mathrm{T}}$ is the time vector. The states including position, velocity, and acceleration at the joint point between polynomial pieces are continuous, and the order of polynomials is 5, so the angles of the quadrotor's body will be changing smoothly along the planned trajectory according to the differential flatness property.

We define the optimization problem formulation as follows:

$$\begin{aligned}
\min_{\mathbf{C},\mathbf{T}} \ & G = S_e + \rho\left(T_{m-1} - T_0\right) + \lambda_v S_v + \lambda_a S_a + \lambda_c S_c + \lambda_d S_d \\
\text{s.t.} \ & p_j^{[s]}\left(T_j\right) = p_{j+1}^{[s]}(0) = \bar{p}_j, \\
& p_j\left(T_j\right) \in \mathscr{H}_j \cap \mathscr{H}_{j+1}, T_j - T_{j-1} > 0, \\
& \forall j \in \{1, \cdots, m-1\}, \\
& p_1^{[s]}(0) = \bar{p}_s, p_{m-1}^{[s]}\left(T_{m-1}\right) = \bar{p}_f,
\end{aligned} \tag{13}$$

where $\mathscr{C}(x) = max(x,0)^3$ is a cubic function, ρ is the weight of the flight time of the whole trajectory. $[s]$ represents a set of derivatives of the highest order s. We choose $s = 2$, so the position, velocity and acceleration of the joint point between polynomial segments are all equal. $\bar{p}_j \in \mathbb{R}^{(s+1)\times 3}$ is the interval condition, $\bar{p}_s, \bar{p}_f \in \mathbb{R}^{(s+1)\times 3}$ are the start and final states, respectively. S_e, S_v, S_a, S_c, S_d are the cost for the energy, the maximum velocity constraint, the maximum acceleration constraint, the collision with safety corridor, and the collision with dynamic objects respectively. λ_v, λ_a, λ_c, and λ_d are the corresponding weights. $\mathbf{C}$ is the overall coefficient matrix representing $\left(\mathbf{C}_1^{\mathrm{T}}, \mathbf{C}_2^{\mathrm{T}}, \cdots, \mathbf{C}_{m-1}^{\mathrm{T}}\right)^{\mathrm{T}} \in \mathbb{R}^{2(m-1)s\times 3}$. We define the time allocation vector $\mathbf{T}$ as:

$$\begin{aligned}\mathbf{T} &= (T_1 - T_0, T_2 - T_1, \cdots, T_{m-1} - T_{m-2})^{\mathrm{T}} \\ &= (\delta_1, \delta_2, \cdots, \delta_{m-1})^{\mathrm{T}} \in \mathbb{R}_+^{m-1}\end{aligned}. \tag{14}$$

The detailed definition of the costs are as follows:

$$\begin{aligned}
S_e &= \textstyle\sum_{j=1}^{m-1}\sum_{l=0}^{L-1}\left\|\mathbf{C}_j^{\mathrm{T}}\beta^{(2)}\left(\frac{l}{L}\delta_j\right)\right\|_2^2\frac{\delta_j}{L},\\
S_v &= \textstyle\sum_{j=1}^{m-1}\sum_{l=0}^{L-1}\mathscr{C}\left(\left\|\mathbf{C}_j^{\mathrm{T}}\beta^{(1)}\left(\frac{l}{L}\delta_j\right)\right\|_2^2 - v_{\max}^2\right)\frac{\delta_j}{L},\\
S_a &= \textstyle\sum_{j=1}^{m-1}\sum_{l=0}^{L-1}\mathscr{C}\left(\left\|\mathbf{C}_j^{\mathrm{T}}\beta^{(2)}\left(\frac{l}{L}\delta_j\right)\right\|_2^2 - a_{\max}^2\right)\frac{\delta_j}{L},\\
S_c &= \textstyle\sum_{j=1}^{m-1}\sum_{l=0}^{L-1}\sum_{k=1}^{N_j}\mathscr{C}\left(\left(\mathbf{C}_j^{\mathrm{T}}\beta^{(0)}\left(\frac{l}{L}\delta_j\right) - \hat{p}_j^k\right)^{\mathrm{T}}\vec{n}_j^k\right)\frac{\delta_j}{L},\\
S_d &= \textstyle\sum_{j=1}^{m-1}\sum_{l=0}^{L-1}\sum_{i=1}^{N}\mathscr{C}\left(-\left\|\mathbf{C}_j^{\mathrm{T}}\beta^{(0)}\left(\frac{l}{L}\delta_j\right) - \hat{p}_b^i\right\|_2^2 + d_s^2\right)\frac{\delta_j}{L},
\end{aligned} \tag{15}$$

where L is the sample number on each piece of the trajectory, and N is the number of dynamic objects. $d_s = e_{ctl} + e_{pos} + r_{sum}$ is the safety radius to guarantee the safety between the vehicle trajectory and the obstacle at any moment. It is composed of the sum of characteristic radius of the dynamic object and vehicle r_{sum}, the estimated responding error of the entire control system e_{ctl}, and the object position estimation error e_{pos}. e_{pos} can be estimated from posterior estimated covariance matrix $\mathbf{P}^{P_i}$ of the KF, $e_{pos} = \sqrt{\mathbf{P}^{P_i}[1,1] + \mathbf{P}^{P_i}[2,2]t_f + \mathbf{P}^{P_i}[3,3]t_f^2/2}$ $(t_f = \frac{l}{L}\delta_j + \sum_{q=1}^{j-1}\delta_q)$. v_{max} and a_{max} are the upper bound of the velocity and acceleration magnitude, respectively. $\hat{p}_b^i$ can be obtained by forwarding the object position via Equation 10, in a discrete form $\hat{p}_b^i = p_b^i(t_f)$. In addition, S_e is the integration of a polynomial essentially, and it is easy to calculate the closed-form result in each iteration of the optimization. The other costs are accumulated in discrete form in our code by iterating the sample points along the trajectory.

To solve the optimization problem efficiently, we need the gradient w.r.t the joint points $\mathbf{p}$ between the trajectory pieces, written as $\partial G(\mathbf{p},\mathbf{T})/\partial\mathbf{p}$. Also, we use $\partial G(\mathbf{p},\mathbf{T})/\partial\mathbf{T}$ to represent the gradient w.r.t the time vector $\mathbf{T}$. The required gradients can be derived by the Gradient Propagation Law:

$$\frac{\partial G(\mathbf{p},\mathbf{T})}{\partial\mathbf{p}} = \frac{\partial G}{\partial\mathbf{C}}\frac{\partial\mathbf{C}}{\partial\mathbf{p}},\ \frac{\partial G(\mathbf{p},\mathbf{T})}{\partial\mathbf{T}} = \frac{\partial G}{\partial\mathbf{T}} + \frac{\partial G}{\partial\mathbf{C}}\frac{\partial\mathbf{C}}{\partial\mathbf{T}}, \tag{16}$$

Since we can calculate $\partial\mathbf{C}/\partial\mathbf{p}$ and $\partial\mathbf{C}/\partial\mathbf{T}$ efficiently referring [6], and the gradient $\partial G/\partial\mathbf{C}$, $\partial G/\partial\mathbf{T}$ can be derived following the cost definition in (19), the gradients of the objective function are finally obtained. In addition, the constraints in (17) can all be eliminated by the method in [6], the optimization problem is transformed to an unconstrained one and we can solve it efficiently with optimization algorithms such as L-BFGS.

V. Experiments and Evaluations

A. Implementation Details

The perception and avoidance of obstacles are tested and verified in the Robot Operation System (ROS)/Gazebo simulation environment first and then in the real-world experiment. We present the hardware design of our quadrotor platform for the experiment. We use a Q250 airframe, carrying an Intel RealSense D435i depth camera, and an NVIDIA Jetson Xavier NX running Ubuntu 18.04 as onboard computer. The controller of the UAV is a PixRacer running the PX4 flight stack. The overall UAV platform only weighs 0.98 kg, with a size of $178 \times 178 \times 75$ mm. The whole hardware platform is pretty light-weight with a thrust-weight ratio of 3.125, and the theoretical maximal acceleration of the aircraft is about 29 m/s^2 for horizontal maneuvers. For the simulation tests of the trajectory planning method, we use a laptop computer with an Intel i7 8565U CPU running at about 2.8 GHz. An overview of our system is shown in Fig. 4.

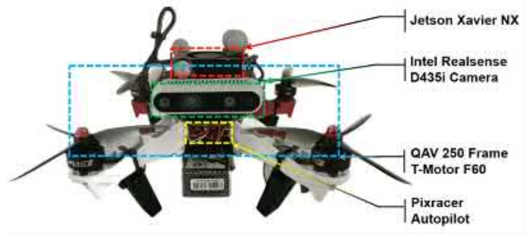

Fig. 4: Overview of our UAV system.

B. Evaluation of Object Detection

In order to achieve the fastest real-time detection speed, we choose Yolo-FastestV2[2] algorithm, which is a fast and light target detection algorithm. Tennis balls (size from 2.57∼2.70 inches in diameter) are selected as the small objects to attack the UAV, and we collected about 2000 tennis ball images as the dataset to train the model. The final obtained model is only 477.4KB, running on the GPU of Jetson Xavier NX.

To choose the resolution, we need to trade off the detection speed against the maximum distance that the tennis balls can be detected. A basic rule is that higher network input resolution can help to detect farther objects, but the computation time cost will be longer. To choose a reasonable resolution, we design action time $t_{act} = t_{hit} - t_{est}$ as the indicator, where t_{hit} is the time from the first successful object detection to the collision on the drone (the collision is recognized in IMU data), t_{est} is the time from the first successful detection to the first object motion estimation, which needs three continuous detection. t_{act} indicates the time remained for the motion planning module and the propulsion system to avoid

[2]https://github.com/dog-qiuqiu/Yolo-FastestV2

moving objects, and it should be as long as possible. With different input resolution, the detection performance is shown in TABLE I. We fix the camera and throw balls at the camera from about 5 m for 20 times, respectively, and calculate the average value for the time results. In conclusion, $672*672$ input size is selected because it results the longest t_{act}.

TABLE I: Detection Performance of Different Resolution

Input Size	Performance		
	Time Cost(ms)	***Max Distance***(m)	***Action Time***(ms)
736*736	42	4.3	220
672*672	23	3.9	270
512*512	14	2.2	120

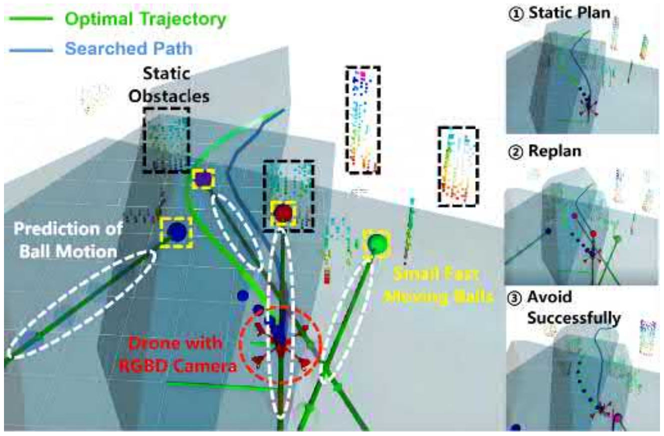

Fig. 5: Planning of optimized trajectory of UAV to dodge ball attack while flying in a complex Gazebo simulation environment.

C. Evaluation of Object Tracking in 3D Space

To validate the accuracy of estimated trajectories, we compare the result of 3D-SORT algorithm with the ground truth data provided by Vicon. The object is fixed on a pole and fitted with reflective balls, and waved by hand with a max velocity of about $5m/s$(shown in Fig. 6). We evaluate the performance of our method based on the open-source trajectory evaluation tool EVO[3]. The mean APE (Absolute Pose Error) achieves $0.038m$, and the mean AVE (Absolute Velocity Error) achieves $1.712m/s$. The evaluation of position and velocity estimation is shown in Fig. 7. It can be seen that the estimated position and velocity closely match the ground truth provided by the Vicon.

To further demonstrate the benefit of using the constant acceleration model, we compare with the constant velocity model which is usually in current drone obstacle avoidance studies [7]. We predict the trajectory one second into the future based on the current state estimates of the two models with a sampling time of 0.05s, and compare to the ground truth data. The predicted trajectory of constant velocity model is only a straight line while constant acceleration model is a second-order polynomial. According to Fig.8, obviously the prediction trajectory of constant acceleration model can better represent the motion trend of the obstacle, which gives more advantage to the collision avoidance.

[3]https://github.com/MichaelGrupp/evo

Compared with the method proposed in paper [11], running point clouds in real-time on a mobile platform is very resource-intensive, and it is difficult to detect small objects using low-resolution point clouds, which are generally unrecognized or close to noise. Our pipeline is more effective for small objects and costs less computing resources. The running time of the 3D-SORT is within 5 ms, and the overall tracking by detecting pipeline is within 30 ms. The comparison result is presented in TABLE II.

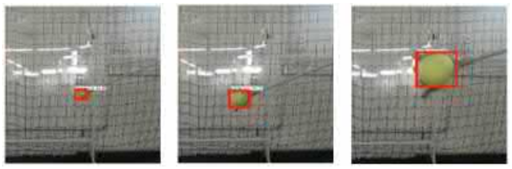

Fig. 6: Tracking result displayed in RGB image.

TABLE II: Perception Method Comparison

Methods	Effective Distance(m)	Time Cost(ms)
Method [11](1280*720)	/	Can't be real-time
Ours(1280*720)	**3.9**	**29**
Method [11](640*480)	0.9	30
Ours(640*480)	**3.0**	**25**

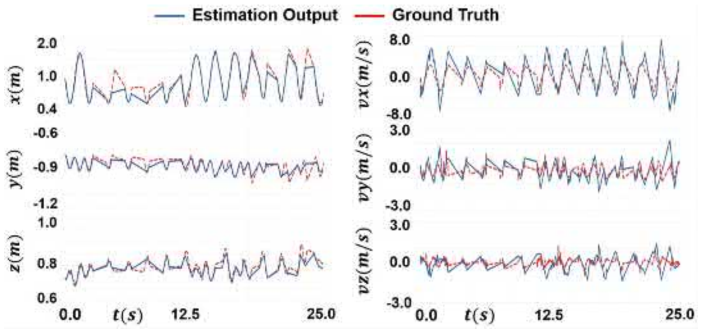

Fig. 7: The comparison of the estimated states of the ball by our method and ground truth obtained by Vicon.

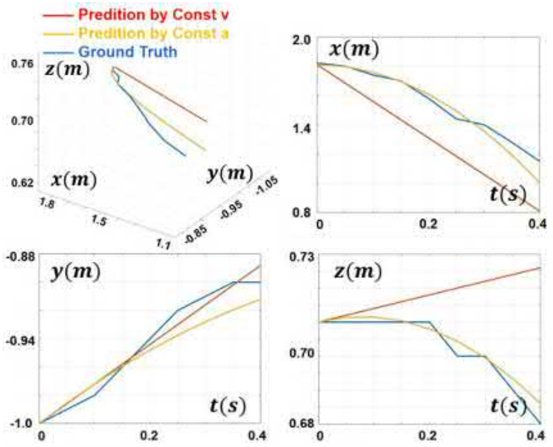

Fig. 8: Comparison of the obstacle future trajectory by constant velocity and constant acceleration model.

D. Evaluation of Attack Avoidance

Firstly, we present several flight tests in Gazebo simulation. The obstacles hit the UAV from different directions

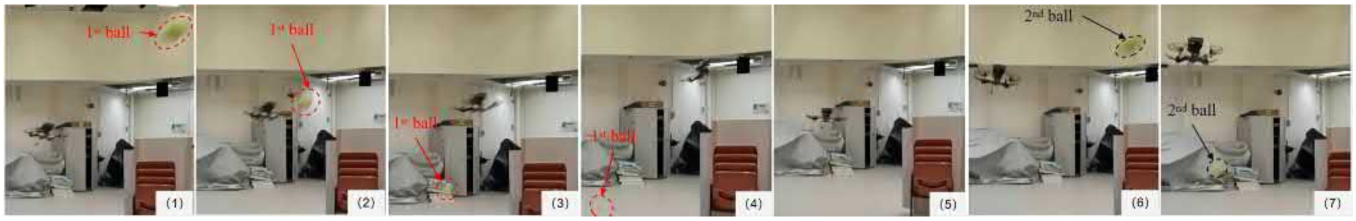

Fig. 9: Screenshots of the UAV dodging two balls thrown continuously while navigating the unknown environment. No mocap is used. The 2nd ball was thrown out in sequence (5) and the UAV successfully avoided it in sequence (6) and (7).

with an average velocity of 6 m/s, the acceleration is equal to the gravitational acceleration 9.8 m/s^2, and the states of the obstacles are simulated as ROS topic and published with timestamps. The simulated drone model is modified with the same weight and trust weight ratio as our real vehicle, the parameters of the PX4 flight controller also keep the same. Fig. 5 presents the attack and planning result in simulation.

To study the influence on flight safety from the different components of our trajectory planning system, we give distinct parameter settings of the algorithm, including the maximum polyhedron size s_H (for building the SFC), the control sampling number N_{acc} of the hybrid A-star path searching, and the maximum velocity and acceleration constraint v_{max}, a_{max}. We set the minimum distance between the drone and dynamic objects d_{min}, the trajectory generation time cost t_{comp} and the successful rate η ($d_{min} < 0.4m$ for failure) as indicators. The drone flies in the simulation world with only dynamic objects, for each parameter configuration the data is collected after 10 attacks. Each attack contains 4 objects at speed 6 m/s starting from 1 to 6 meters away from the vehicle. The results are compared in TABLE III. v_{max} shown in TABLE III is only for avoiding dynamic objects, in our tests it is set to 2 m/s if no dynamic objects are detected. The parameters in row 1 are the default value we use in simulation. We can conclude from rows 2, 4, and 5 that limiting the corridor size or the maximum velocity or acceleration will drop the successful avoidance rate towards dynamic objects because the solution space of trajectory optimization is narrowed, and sometimes the feasible solution does not exist. Increasing the control sampling number of hybrid A-star will cause a much heavier computational load (only necessary for dense static obstacles), and the larger computing latency will lead to a collision with the close and fast objects.

In TABLE IV, we summarize the success rate of avoidance under different obstacle conditions (initial distance of objects, latency), each result comes after 50 tests in simulation. We add the pre-assigned time delay at different levels after the motion planner received the objects' state to simulate the time cost of the object detection and tracking module. The avoidance success rate is about 98% when the reaction time t_{act} is greater than 0.25 s, thus our system is verified practical for real applications according to the maximal average $t_{act} = 0.27$ s recorded in TABLE IV.

To verify the advantage of our trajectory planning method in flight safety and optimality (energy cost and trajectory duration), we compare it with the artificial potential field (APF) method [3], and a relative velocity planning method [23]. We set t_{all} and J to evaluate the trajectory optimality in navigation tasks, and flight success rate η and the minimum distance d_{min} are utilized to justify safety. t_{all} is the flight time to reach the target point, and E_A is the integral of the UAV acceleration modulus during the flight. Note that the APF method used in [3] does not consider avoiding static obstacles, which indicates the huge superiority in environment adaptability of our method. When comparing with APF, only dynamic objects exist in simulation for fair, and static obstacles are set up when compared with [23]. We conducted 20 repeated flight tests for each approach, starting from position $(-6, 0, 1.5)$ to $(10, 0, 1.5)$ with 4 randomly initialized dynamic object (the same configuration as in TABLE III, attacking once towards the current position of the drone during the navigation. a_{max} for the three approaches is 29 m/s^2. v_{max} is set to 2 m/s, but when any dynamic object is detected $v_{max} = 6m/s$. The results are shown in TABLE V. Our method shows superiority in all four indicators. The APF method does not involve any collision check between the vehicle trajectory and objects' trajectories, also the vehicle kinodynamic constraints are not respected, thus no promise in flight safety, let along optimality. The method in [23] assumes the moving objects are of constant velocity, which differs from the objects in our tests. More importantly, [23] prioritizes avoiding closer obstacles but can not guarantee a feasible solution for further obstacles' avoidance, where the planner sometimes fails.

TABLE III: Parameter analysis of trajectory planning

s_H(m)	N_{acc}	v_{max}(m/s)	a_{max}(m/s^2)	d_{min}(m)	t_{comp}(ms)	η(%)
5	5	6	20	0.56	3.32	100
1.5	5	6	20	0.31	3.95	70
5	21	6	20	0.33	36.40	90
5	5	2	20	0.26	3.51	50
5	5	6	5	0.39	4.25	60

TABLE IV: Success Rate of Attack Avoidance Under Different Conditions in Simulation, 50 trials

Ball number	**Ball Velocity**			**Action Time t_{act}**		
	$5m/s$	$9m/s$	$12m/s$	$> 0.25s$	$0.1 - 0.25s$	$< 0.1s$
1	100%	100%	94%			
2	100%	90%	88%	98%	90%	84%
4	92%	80%	72%			

TABLE V: Benchmark tests for trajectory planning

Method	$E_A(m/s^2)$	t_{all}(s)	d_{min}(m)	η(%)
Ours	**3852**	**8.23**	**0.58**	**95**
APF	5670	12.94	0.28	65
[23]	4835	11.61	0.26	75

E. Experiments of Dodging Multiple Small Fast-moving Balls

We perform two types of experiments on dodging small fast-moving tennis balls: continuous attack and simultaneous attack. In both experiments, the UAV is required to navigate in an unknown environment to avoid static obstacles. The UAV uses its onboard VIO FLVIS[4] for localization instead of using a motion capture system. Please refer to our video for the results [5].

In the first type of experiment, we throw out multiple balls continuously. As seen in Fig. 9, the drone successfully avoided the two balls without colliding with any static obstacles. Because the balls are moving quickly, the UAV also has to perform aggressive maneuvers to avoid them.

In the second type of experiment, we throw out multiple tennis balls simultaneously. In this case, the UAV has to track multiple balls simultaneously to plan a trajectory to avoid them. As seen in Fig. 1, the UAV successfully avoided them without colliding with static obstacles.

We also performed avoiding four balls thrown from different directions and the UAV also successfully avoided them. In the experiments, the distance between UAV and ball is about 5 m, and the velocity of the ball after throwing is about 5 m/s. The reader can refer to the video for more tests.

VI. Conclusion

In this paper, we propose a complete autonomous UAV system to dodge fast and small objects in the navigation tasks. To our best knowledge, we are the first to demonstrate the avoidance of fast objects with the onboard vision from an RGBD camera, and promote the avoidance into complex navigation tasks. Our system achieves satisfactory perception precision with sensors of a lower price than the event camera based works. In addition, our motion planning method shows better safety and optimality in time and energy cost, while being capable of performing navigation tasks.

Our UAV can avoid fast-moving balls most of the time. However, failure cases also exist, most of which are related to the field of view of the camera. If the ball flies outside of the field of view, the tracking may fail. However, in this case, as the ball is outside of the view, it also does not collide with the UAV. One future work would be to increase the field of view by adding more cameras. Although we took motion blur into consideration, we did not consider lighting conditions. Another interesting future work would be to enhance the robustness of the perception in terms of different lighting conditions.

References

[1] G.-Z. Yang, J. Bellingham, P. E. Dupont, P. Fischer, L. Floridi, R. Full, N. Jacobstein, V. Kumar, M. McNutt, R. Merrifield, *et al.*, "The grand challenges of science robotics," *Science robotics*, vol. 3, no. 14, p. eaar7650, 2018.

[2] 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, 2019.

[4]https://github.com/HKPolyU-UAV/FLVIS

[5]https://www.youtube.com/watch?v=n1rlnFTOBGE

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

[4] N. J. Sanket, C. M. Parameshwara, C. D. Singh, A. V. Kuruttukulam, C. Fermüller, D. Scaramuzza, and Y. Aloimonos, "Evdodgenet: Deep dynamic obstacle dodging with event cameras," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10651–10657, IEEE, 2020.

[5] B. He, H. Li, S. Wu, D. Wang, Z. Zhang, Q. Dong, C. Xu, and F. Gao, "Fast-dynamic-vision: Detection and tracking dynamic objects with event and depth sensing," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3071–3078, IEEE, 2021.

[6] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *arXiv preprint arXiv:2103.00190*, 2021.

[7] D. Falanga, S. Kim, and D. Scaramuzza, "How fast is too fast? the role of perception latency in high-speed sense and avoid," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1884–1891, 2019.

[8] T. Stoffregen, G. Gallego, T. Drummond, L. Kleeman, and D. Scaramuzza, "Event-based motion segmentation by motion compensation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7244–7253, 2019.

[9] A. Mitrokhin, C. Ye, C. Fermüller, Y. Aloimonos, and T. Delbruck, "Ev-imo: Motion segmentation dataset and learning pipeline for event cameras," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6105–6112, IEEE, 2019.

[10] K. Su and S. Shen, "Catching a flying ball with a vision-based quadrotor," in *International Symposium on Experimental Robotics*, pp. 550–562, Springer, 2016.

[11] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous flights in dynamic environments with onboard vision," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1966–1973, IEEE, 2021.

[12] J. Lin, H. Zhu, and J. Alonso-Mora, "Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2682–2688, IEEE, 2020.

[13] T. Eppenberger, G. Cesari, M. Dymczyk, R. Siegwart, and R. Dubé, "Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10528–10535, IEEE, 2020.

[14] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*, pp. 3645–3649, IEEE, 2017.

[15] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE international conference on image processing (ICIP)*, pp. 3464–3468, IEEE, 2016.

[16] M. Simon, K. Amende, A. Kraus, J. Honer, T. Samann, H. Kaulbersch, S. Milz, and H. Michael Gross, "Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[17] F. Kong, W. Xu, Y. Cai, and F. Zhang, "Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7869–7876, 2021.

[18] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[19] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, *et al.*, "Fast, autonomous flight in gps-denied and cluttered environments," *Journal of Field Robotics*, vol. 35, no. 1, pp. 101–120, 2018.

[20] M. Jacquet and A. Franchi, "Motor and perception constrained nmpc for torque-controlled generic aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 518–525, 2020.

[21] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE transactions on robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[22] J. L. Blanco and P. K. Rai, "nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees." `https://github.com/jlblancoc/nanoflann`, 2014.

[23] H. Chen and P. Lu, "Real-time identification and avoidance of simultaneous static and dynamic obstacles on point cloud for uavs navigation," *Robotics and Autonomous Systems*, vol. 154, p. 104124, 2022.