2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

Autonomous Flights in Dynamic Environments with Onboard Vision

Yingjian Wang, Jialin Ji, Qianhao Wang, Chao Xu and Fei Gao

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636117

***Abstract*— In this paper, we introduce a complete system for autonomous flight of quadrotors in dynamic environments with onboard sensing. Extended from existing work, we develop an occlusion-aware dynamic perception method based on depth images, which classifies obstacles as dynamic and static. For representing generic dynamic environment, we model dynamic objects with moving ellipsoids and fuse static ones into an occupancy grid map. To achieve dynamic avoidance, we design a planning method composed of modified kinodynamic path searching and gradient-based optimization. The method leverages manually constructed gradients without maintaining a signed distance field (SDF), making the planning procedure finished in milliseconds. We integrate the above methods into a customized quadrotor system and thoroughly test it in real-world experiments, verifying its effective collision avoidance in dynamic environments.**

I. Introduction

Thanks to the maturity of autonomous navigation technology, aerial robots, especially quadrotors, have made impressive progress in the last decade. However, it is still challenging to let quadrotors operate autonomously in dynamic environments due to trifold reasons. Firstly, the perception of dynamic obstacles is hard to satisfy the efficiency and accuracy requirements at the same time, making the quadrotor fragile in a complex dynamic environment. Moreover, to make a quadrotor fly smoothly among moving obstacles, the onboard planner must fully utilize the obstacle position and velocity estimation while retaining high efficiency. Finally, the system integration requires robustness from all modules, including localization, perception, and planning, under limited onboard sensing and computing resources. These challenges render autonomous flights in an unknown dynamic environment hard, making robotic community always lacks a convincing systematic solution.

In this paper, we bridge this research gap by introducing a complete system composed of an accurate dynamic environment perception module and an accompanied trajectory generation method. For dynamic environment perception, we classify the obstacles into static ones and dynamic ones by clustering, tracking, and voting the point cloud data from a depth sensor, inspired by [1]. To address the issue of occlusion, we propose an occlusion-aware Kalman filter to estimate the motion states of dynamic objects, including position, velocity, size, and associated uncertainty, making the classification more accurate and robust. We then fuse the motionless points into an occupancy grid map and model the dynamic ones as several moving ellipsoids for the subsequent trajectory generation.

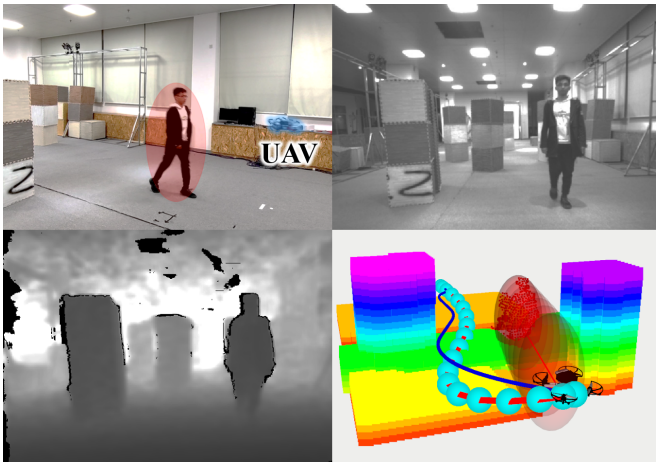

Fig. 1. Experimental results of autonomous flight in a dynamic environment with onboard sensing. Top left: a flying UAV in front of a walking person. Top right: onboard grayscale image. Bottom left: depth image. Bottom right: the optimized B-spline trajectory (blue points and red curve) between the detected moving person(red ellipsoids with a red line indicating the velocity) and the static grids.

Considering the static and the dynamic obstacles simultaneously, we propose a hierarchical trajectory generation method. Firstly, we design a kinodynamic path searching method to search for a safe, feasible, and minimum-time initial path by dynamic safety check. After parameterizing the initial path into a B-spline curve, we refine the curve with a gradient-based optimization. Extending our previous work [2], we obtain gradients by manually constructed penalties without maintaining an SDF, which dramatically lowers the computational burden.

Compared to existing state-of-the-art works [2, 3], our proposed method can generate safe trajectories in more generic environments composed of cluttered static and dynamic obstacles. We perform comprehensive tests in simulation and real world to validate the robustness of our method. Summarizing our contributions as follows:

1) We develop an occlusion-aware dynamic environment perception method based on [1], which efficiently yet accurately classifies the obstacles into static ones and dynamic ones. Moreover, we represent both classes of obstacles as two different data structures to handle generic complex environments.
2) We propose a hierarchical trajectory generation method composed of a kinodynamic searching method and

All authors are with the State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, 310027, China, and also with the Huzhou Institute of Zhejiang University, Huzhou, 313000, China. {yj_wang, jlji, qhwangaa, cxu, fgaoaa}@zju.edu.cn

This work was supported by National Natural Science Foundation of China under Grant 62003299

a gradient-based optimization. We leverage the environment representation mentioned above to construct gradients manually, which avoids the overhead of maintaining an SDF, making the whole procedure finished in milliseconds.

3) We integrate our proposed methods into a fully autonomous quadrotor system and release our software for the community's reference[1].

II. Related Work

A. Dynamic Perception

There are mainly two categories of methods for dynamic perception. One is to use imagery data. In [4], authors adopt the Frame-Difference (FD) method [5] to detect moving objects, which requires that UAVs hover steady. However, frequent hovering will greatly affect the smoothness of flight in dynamic avoidance missions. Besides, some work utilize outliers of the feature tracking module in feature-based vision system to detect dynamic objects [6]–[8]. However, it requires dense feature points, which is hard to satisfy in generic environments. Extended from optical flow, scene flow method [9, 10] computes 3D velocity of each pixel. This technique, however, is unable to run in real-time on a computationally-constrained platform. Another approach is using detector [11], or segmentation networks [12], which behaves well in detection of predefined classes such as pedestrians or cars [13, 14], but cannot handle generic cases.

Apart from using imagery data, point-cloud-based methods aim to estimate the motion of objects leveraging 3D information. Kinect Fusion [15] implements Iterative Closest Point (ICP) [16] to deal with moving objects. However, this approach would result in high computational overhead. Some work [17, 18] use U-map [19] to detect obstacles which considers that points with similar depth belong to the same object. This method divides the point cloud into individual objects without dynamic classification, which does not represent the dynamic environment accurately. Our approach builds upon and extends a dynamic obstacle detection and tracking algorithm [1], which tracks clusters and classifies them as dynamic or static. Varying from them, we conduct tracking with occlusion-awareness and replace their human detector with our proposed re-free strategy to address the issue of erroneous occupancy caused by temporary standstills.

B. Dynamic Planning

Most planning methods in dynamic environments inherit the ones in static environments and just design some techniques to handle the moving obstacles. Reactive methods, including velocity obstacles (VO) [20]–[22], inevitable collision states (ICS) [23, 24] and artificial potential field (APF) [25], leverage the information of current moving obstacles and only compute one-step actions, which makes them quite short-sighted due to the lack of long horizon. Seder et.al [26] first adopt a focused D* to find a reference path without considering obstacle motions and then generates admissible local trajectories around the reference path using a dynamic window algorithm. In [27], the RRT and forward SR sets are utilized to find the reference path and avoid moving obstacles, respectively.

In recent studies, Gao et.al [28] use semi-definite relaxation on nonconvex Quadratical Constraint Quadratic Programming (QCQP) to optimize a piecewise polynomial trajectory in dynamic environments. However, this method and other global planners [29, 30], assume that locations and velocities of all obstacles are known, which cannot work in autonomous drone platform. In [3], the authors obtain objects' necessary state information by onboard detection method and utilize chance-constrained nonlinear model predictive control for obstacle avoidance in dynamic environments. However, it regards all obstacles as ellipsoids, which makes it cannot handle complex static obstacles. Beyond that, nonlinear optimization is prone to fall into a local minimum without a proper initial guess.

III. Dynamic Environment Perception

In this section, we describe a dynamic perception module built upon [1]. Given a point cloud generated from depth image at time $\tau \in \mathbb{R}$, we filter it to reduce sensor noise and lower computational overhead, and then cluster the filtered point cloud into individual objects per frame using DBSCAN [31], resulting in a set of m clusters $C_\tau = \{c_\tau^1, c_\tau^2, ..., c_\tau^m\}$. To estimate moving agents' necessary states, we then conduct occlusion-aware tracking and dynamic classification.

A. Occlusion-aware Tracking

We denote a list of cluster sets over k frames separated by Δt as $\widetilde{C}_{t,k}$ = $\{C_{t-k\cdot\Delta t}$,..., $C_{t-\Delta t}$, $C_t\}$. Given $\widetilde{C}_{t,k}$, we use an occlusion-aware tracking algorithm extended from [1] to obtain their **tracking history** $H_t^i = \{c^*_{t-n\cdot\Delta t}, ..., c^*_{t-\Delta t}, c_t^i\}$ which include indices of the same cluster.

As shown in Fig.2, for each cluster at time t_0, we will apply a Kalman filter to it initially with a conservative motion model, similar to the work of Azim et al. [32], resulting in a list of Kalman filters $K_{t_0} = \{\kappa_{t_0}^1, \kappa_{t_0}^2, ..., \kappa_{t_0}^l\}$. Then, in every later frame at time t_i, we firstly propagate all Kalman filters in $K_{t_{i-1}}$ forward and then associate centroids of clusters in the current frame to the forward propagated positions by k-Nearest Neighbor searching [33]. If the association of cluster $c_{t_0}^j$ successes, the cluster will inherit its corresponding tracking history. Otherwise, we suppose the unassociated cluster is a newly appeared object and create a new Kalman filter for it. To address the short-time occlusion issue, we mark the unassociated clusters lost and keep their Kalman filters running until they are tracked or lost for a threshold time t_d, as shown in the right figure of Fig.3.

Moreover, it is essential that we attempt to associate current clusters to forward propagated Kalman filters rather than clusters in the previous frame like [1]. As shown in the right figure of Fig.3, when there are several moving agents, the association without considering motion information may be incorrect due to the movements of dynamic obstacles.

[1]https://github.com/ZJU-FAST-Lab/dynamic-flight

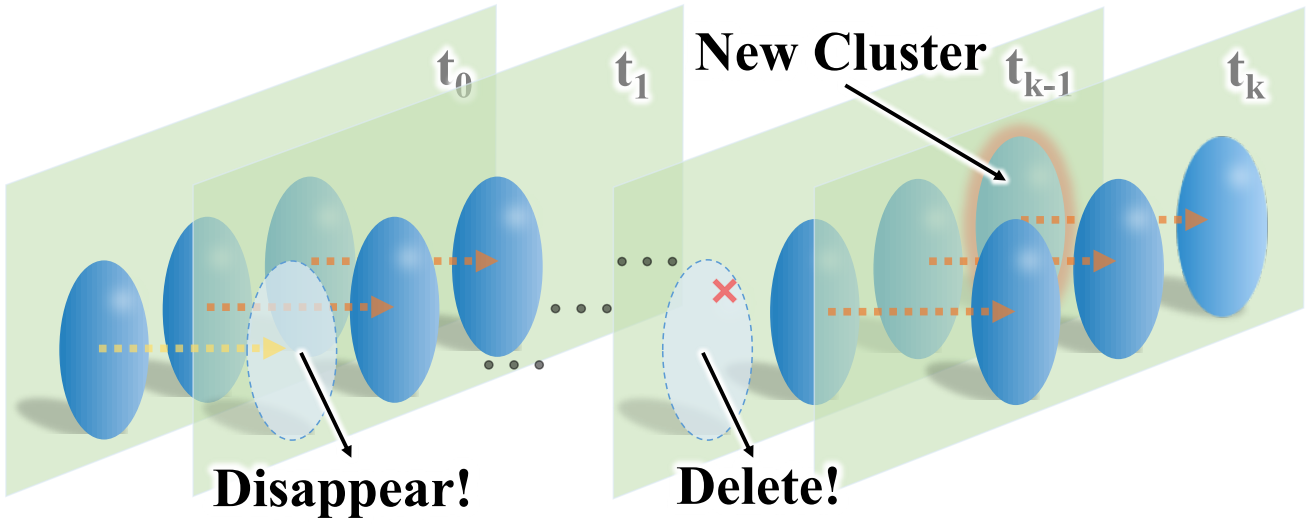

Fig. 2. Non-connected KFs in the previous frame will keep propagating until their corresponding cluster disappears for specific frames (i.e., the white ellipsoids in t_1 and t_{k-1}), while non-connected clusters in the current frame will be marked as new objects and begin their tracking (frame t_{k-1}).

Compared with it, we track clusters with forward propagated Kalman filters, efficiently avoiding plenty of incorrect associations.

B. Classification as Dynamic or Static

We then classify the generated cluster as either static or dynamic. Inspired by [1], we obtain the motion of clusters by comparing point clouds from two frames being δ seconds apart. Firstly, we build kdTree with the dense, non-filtered point cloud from the reference frame at time $t-\delta$. Then we measure the global nearest neighbor distance $d^{i,j}$ of each point $p^{i,j}$ belonging to the cluster c_t^i in the current frame at time t by the kdTree built before. Then velocity of each point is calculated by $v = \frac{d^{i,j}}{\delta}$ and the points with $v > v_d$ vote for dynamic. After voting, a cluster will be considered dynamic obstacle if the absolute or relative amount of votes for being dynamic surpass respective thresholds l_{dyn}^{abs} or l_{dyn}^{rel}.

To guarantee that each point must be observed in both current frame and reference frame, there are two cases that should be noted to improve performance of classification. One is that when the FOV changes between two frames, we only allow points which appear in the overlapped area to vote. Another is the occlusion caused by dynamic objects. To address the issue, firstly we associate current cluster c_t^i to its corresponding cluster $c_{t-\delta}^j$ by the tracking history H_t and project their 3D centroids onto the image plane. Self-occlusion is supposed to happen if $depth[q_{t-\delta}^j] < depth[q_t^i]$ and all points of c_t^i can vote, where q denotes the projected 2D point of 3D point p. Otherwise, for points $p_t \in \mathbb{R}^3$ belonging to other clusters at time t, we firstly reproject them onto the image plane at time t-δ,

$$\begin{bmatrix} \hat{q}_{t-\delta} \\ 1 \end{bmatrix} = T_{t-\delta}^{-1} \begin{bmatrix} p_t \\ 1 \end{bmatrix}, \tag{1}$$

$$\begin{bmatrix} q_{t-\delta} \\ 1 \end{bmatrix} = K\hat{q}_{t-\delta}, \tag{2}$$

where $T_{t-\delta}$ is the transformation matrix of camera pose at time $t-\delta$, K is the intrinsic matrix of the camera and $\hat{q}_{t-\delta}$ is 3D point in the camera frame.

We then suppose point p_t is occluded if $|\hat{q}_{t-\delta}|_Z - depth[q_{t-\delta}] > \epsilon$, where $|\bullet|_Z$ denotes the z-coordinate of a 3D point and the $depth[q_{t-\delta}]$ and ϵ is set related to the velocity threshold v_d to reduce sensor noise. Occluded points will be excluded from vote.

Finally, to improve the robustness of perception, it is essential to check the consistency of classification results over a time horizon t_h. Namely, a cluster is judged as dynamic or static finally only if its classification results in all frames over t_h keep the same.

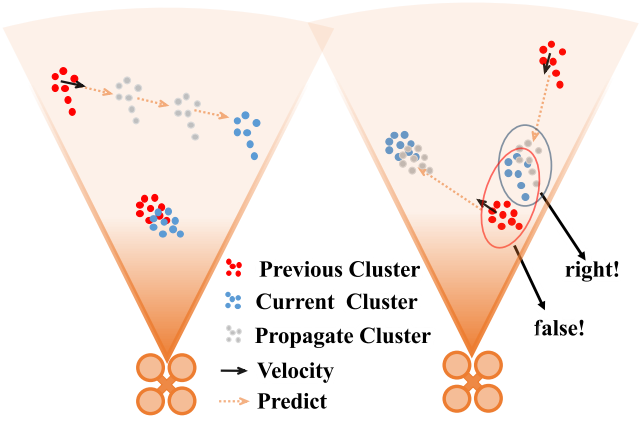

Fig. 3. The left figure shows our tracking algorithm could address the issue of occlusion. The right figure shows our method could avoid incorrect association, which would happen in the association without propagation.

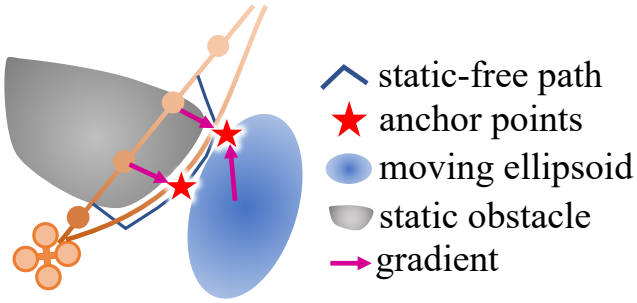

Fig. 4. Our previous work [2] search a collision-free path around static obstacles to generate gradients between control points and anchor points. However, these gradients may conflict with gradients from dynamic obstacle and make trajectory into local minimum. Thus, a path searching method considering dynamic objects is required.

C. Dynamic Environment Representation

In order to allow for autonomous flight of UAV in a dynamic environment, we represent the obstacles respectively according to their dynamic or not.

For dynamic agents, we model them as moving 3D ellipsoids with velocity $\mathbf{v}^o$:

$$(\mathbf{p}-\mathbf{p}^o)^T\Theta^o(\mathbf{p}-\mathbf{p}^o)=1, \tag{3}$$

where $\Theta^o = R^T diag(\frac{1}{(l_x+r)^2}, \frac{1}{(l_y+r)^2}, \frac{1}{(l_z+r)^2})R$. $\mathbf{p}^o, l_x, l_y, l_z, r$ are estimated position and axis-length of ellipsoid and inflate radius.

For classified static point cloud, we fuse it into occupancy 3D grid map, which is efficient and easy to use for trajectory planning. Significantly, we introduce **Re-Free** strategy to eliminate the erroneously occupied space caused by the temporary standstill of moving objects and address the issue of time-lag in [1] caused by δ in classification. In detail, we fuse all points in current frame into the occupancy grid map initially. Furthermore, only if an agent is considered dynamic, we free all related occupancy grids according to its tracking history. In this way, the quadrotor may treat moving objects as static obstacles in the beginning and re-frame them after classification. Sufficient experiments prove that this implementation could significantly improve the robustness and safety of UAVs with limited FOV in obstacle avoidance missions. We finally represent the dynamic environment with moving ellipsoids and occupancy grids.

IV. Dynamic Planning

In this section, we describe an ESDF-free gradient-based dynamic planning method based on our previous work [2]. In [2], anchor points are constructed manually to generate gradients by searching a collision-free path around each colliding segment of the initial trajectory. However, this procedure cannot handle dynamic obstacles, which makes it get stuck into a local minimum easily, shown in Fig.4. In fact, most gradient-based planning methods like [3] would face the same problem if the initial trajectory clamps between two near obstacles.

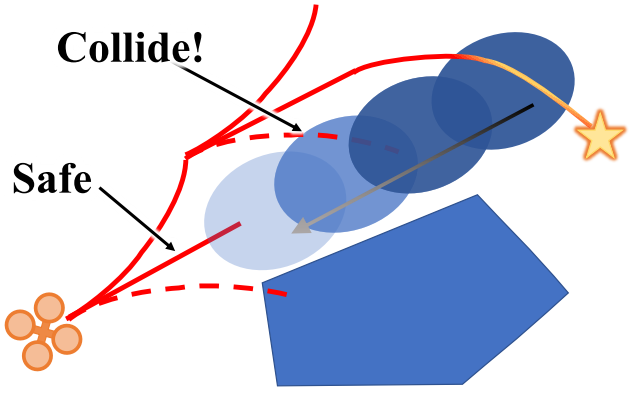

Fig. 5. An illustration of the dynamic obstacle-avoiding kinodynamic path searcher. The red curve indicates the motion primitives (full line means safe and dotted line means unsafe), while the yellow curve represents the analytic expansion scheme which speeds up searching [34]. Importantly, we predict the motion of dynamic obstacle and check that if the state at moment τ collides with predicted ellipsoid at the same time.

To solve this problem, we obtain a dynamic obstacles-avoiding initial trajectory for subsequent optimization by a dynamic obstacle avoiding kinodynamic searching method.

A. Dynamic Obstacles-avoiding Kinodynamic Searching

Similar to [34], our method generates motion primitives by sampling the control inputs u_i and durations T_i for each node. We define the cost of a trajectory as

$$J(T)=\int_0^T ||u(\tau)||^2 d\tau + \rho T, \tag{4}$$

where T is the total duration and ρ is a tunable weight. Heuristic cost is defined as $J(T_h)$, where T_h is obtained by solving a closed form Optimal Boundary Value Problem.

Differently, the valid check of each motion primitive is modified by considering the collision check of the dynamic obstacles represented as ellipsoids, as shown in Fig.5. If the collision condition between the motion primitive at time τ and the predicted ellipsoid at the same time is satisfied, this motion primitive will be considered unsafe. The minimum distance between a point and an ellipsoid can not perform in closed form [35]. Thus, the collision condition is

$$C_{ij} := \{||\mathbf{p_i}-\mathbf{p_j^o}||_{\Theta^o} \le 1\}, \tag{5}$$

where $\mathbf{p_j^o}$ and Θ^o are the estimated position and ellipsoid matrix in III-C. After searching for an initial trajectory that lies near the optimum, our subsequent B-spline based optimization will find that optimum.

B. B-Spline Trajectory Optimization

Based on our previous work [2], the trajectory is parameterized by a uniform B-spline curve Φ, which is a piecewise polynomial uniquely determined by its degree p_b, a knot span Δt, and N_c control points $Q_i \in \mathbb{R}^3$.

According to the property that the k^{th} derivative of a B-spline is still a B-spline with degree $p_{p,k} = p_b - k$, the

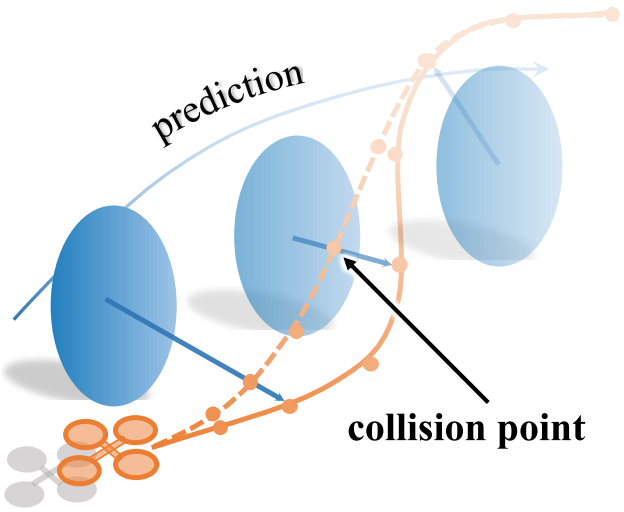

Fig. 6. An illustration of a initial curve (dotted line) and the corresponding trajectory optimized by dynamic collision penalty (full line). If a control point collides with predicted ellipsoids, the gradient will push it out of moving obstacles.

velocity, acceleration and jerk control points $\mathbf{V}_i$, $\mathbf{A}_i$ and $\mathbf{J}_i$ are calculated by

$$\mathbf{V}_i = \frac{\mathbf{Q}_{i+1} - \mathbf{Q}_i}{\Delta t}, \mathbf{A}_i = \frac{\mathbf{V}_{i+1} - \mathbf{V}_i}{\Delta t}, \mathbf{J}_i = \frac{\mathbf{A}_{i+1} - \mathbf{A}_i}{\Delta t}. \quad (6)$$

Then the optimization problem is then formulated as:

$$\min_{\mathbf{Q}} J = J_s + \lambda_f J_f + \lambda_c J_c + \lambda_d J_d, \quad (7)$$

where J_s is the smoothness cost defined by

$$J_s = \sum_{i=1}^{N_c-2} ||\mathbf{A}_i||^2 + \sum_{i=1}^{N_c-3} ||\mathbf{J}_i||^2, \quad (8)$$

J_f, J_c, J_d are penalty terms for the constraints of feasibility, avoiding the static and dynamic obstacles. λ_f, λ_c and λ_d are weights for each penalty terms.

Thanks to the convex-hull property of B-spline, dynamic feasibility of the whole trajectory can be guaranteed sufficiently if we constrain all the velocity and acceleration control points:

$$J_f = \sum_{i=1}^{N_c-1} g(||\mathbf{V}_i||^2 - v_m^2) + \sum_{i=1}^{N_c-1} g(||\mathbf{A}_i||^2 - a_m^2), \quad (9)$$

where $g(x) = \max\{x, 0\}^3$ is a penalty function and v_m, a_m are the limits on velocity and acceleration.

For static obstacles, we follow the method of our previous work [2]. We firstly search for a collision-free path around static obstacles. Then fixed anchor points and corresponding vectors are generated to determine the static collision penalty and gradients, shown in Fig.4. The penalty is

$$J_c = \sum_{i=1}^{N_c} \sum_{j=1}^{N_i} g\left((\mathbf{Q}_i - \mathbf{p}_{ij}) \cdot \mathbf{v}_{ij}\right), \quad (10)$$

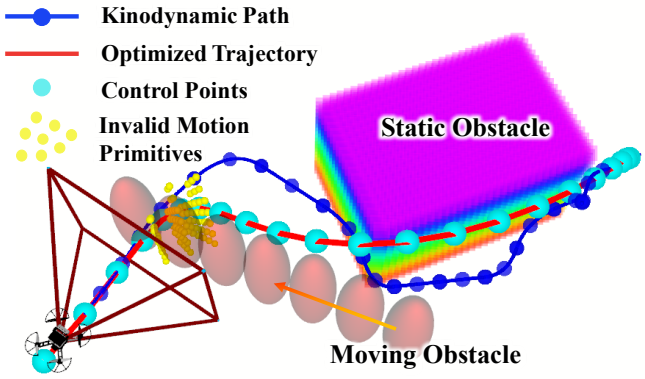

Fig. 7. A result of optimization considering both static and dynamic obstacles in simulation. The blue line with blue dots is the path searched by dynamic obstacle-avoiding kinodynamic searching and the red line is the optimized trajectory with control points (cyan points). The yellow points are invalid motion primitives which collide with predicted moving obstacles.

where $\{\mathbf{p}_{ij}, \mathbf{v}_{ij}\}$ is a pair of anchor point and vector, and N_i is the number of $\{\mathbf{p}_{ij}, \mathbf{v}_{ij}\}$ pairs recorded by the single control point $\mathbf{Q}_i$.

For dynamic obstacles,

$$J_d = \sum_{i=1}^{N_c} \sum_{j=1}^{N_d} g\left(1 - \mathbf{x}_{ij}^T \Theta_j^o \mathbf{x}_{ij}\right), \quad (11)$$

where $\mathbf{x}_{ij} = \mathbf{Q}_i - \mathbf{p}_j^o - i \cdot \Delta t \cdot \mathbf{v}_j^o$, and $\{\mathbf{p}_j^o, \mathbf{v}_j^o, \Theta_j^o\}$ which represents each ellipsoid is obtained from III-C. A initial curve and its corresponding trajectory optimized by dynamic collision penalty are as shown in Fig.6.

We reparameterize the searching result obtained from IV-A to B-spline as the initial guess and utilize the L-BFGS [2] to minimize the cost function (7). A simulated result trajectory of the optimization influenced by both static and dynamic obstacles is shown as Fig.7.

V. Experiments and Benchmarks

1) Implementation Details: In dynamic perception module, we set the time span $\xi = 0.3s$ and the dynamic velocity threshold $v_{nn} = 0.2m/s$. For a cluster, the absolute and relative threshold of vote for being dynamic are set as $l_{dyn}^{abs} = 100$ and $l_{dyn}^{rel} = 0.8$, and the time horizon is set as $t_h = 0.3s$ to improve the robustness of classification.

In dynamic planning module, we set the order of B-spline as $p_b = 3$. The time span Δt of B-spline alters around 100ms, which is determined by the path length searched by our modified kinodynamic A* and the number of control points $N_c = 20$. We use the same L-BFGS solver as [2] whose complexity is linear on the same relative tolerance. To further enforce safety, a collision checker for both static and dynamic obstacles with enough clearance is performed. The optimization stops only if no collision is detected. In real-world experiments, we adopt a similar quadrotor platform of [36] but use Intel RealSense D455 differently, which provides broader FOV.

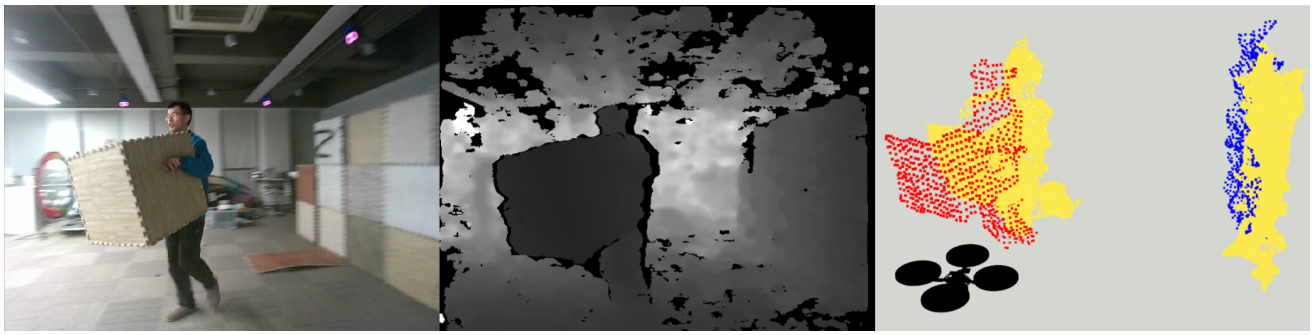

Fig. 8. The RGB-D images and the classification result. As the left figure shown, the moving person and left boxes are classified as dynamic and static and marked with red or blue points respectively, while the yellow point cloud is from the reference frame.

2) Real-World Experiments: We present several experiments in unknown cluttered environments with limited camera FOV. In the first experiment, the drone flies through static objects, avoiding a man walking toward it simultaneously. As shown in Fig.9, the moving person is modeled as an ellipsoid with velocity, and static obstacles are presented as 3D occupancy grids. The person's speed is around 1.6 m/s and the speed limit of UAV is set as 1.5 m/s. A collision is unavoidable if there is no dynamic perception and prediction. The optimized trajectory also meets the performance in the simulation like Fig. 7.

Another experiment requires the UAV to navigate from a start point to an endpoint while avoiding a man walking across the flight course with higher speed and plenty of static obstacles. Fig.10 shows a series of optimized trajectories in a dynamic environment and a snapshot of the flying UAV and the walking person. With a limited maximum perception range of depth sensor around 4m, the UAV classifies obstacles as dynamic or static, estimates the velocity, and builds a 3D grid map below 30ms, indicating that our system can run efficiently in real-time. The person walks around 2m/s, while the UAV achieves a speed of 2.5m/s in the cluster environment during the experiment.

In addition, we also conduct plenty of experiments to especially test our proposed system's robustness. More details are available in the attached video.

3) Benchmark Comparisons: In this section, we both compare the dynamic perception module and dynamic planning module.

Firstly, we compare our dynamic perception module with [3]. We carry on both simulated and real-world experiments. In simulated tests, a $40 \times 10 \times 3m$ environment is generated with static obstacles and moving agents. In real-world experiments, we record a dataset of which one human walks back and forth in a cluster environment. The experiment's snapshot is shown in Fig.8. We then use both perception methods and compare results with ground truth measurements. The method adopted in [3] treats each obstacle as individual ellipsoids without dynamic classification, which results in that the positions and velocities of static obstacles will be frequently misestimated due to the sudden FOV change and occlusion caused by dynamic agents.

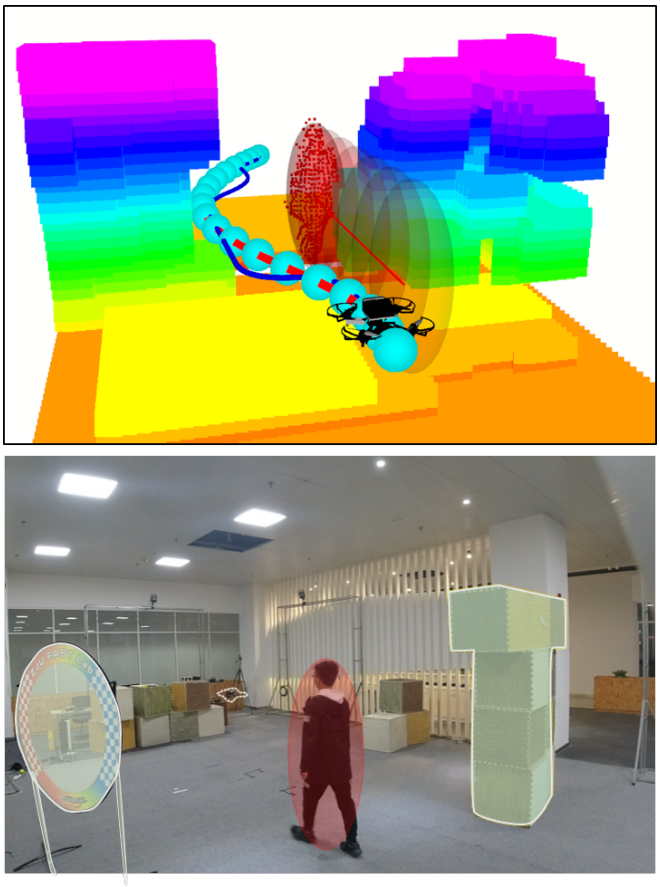

Fig. 9. Top: the optimized trajectory between a moving person and static obstacles in narrow space. Bottom: the experiment scenario in which the static obstacles are masked with yellow and the dynamic one is masked with red.

In each comparison above, we calculate the average estimation errors of position and velocity comparing with the ground truth measurements, and the results are shown in Tab.I. It can be observed that our method could handle more generic environments and perform more accurately.

Afterwards, for dynamic planning, we compare our work

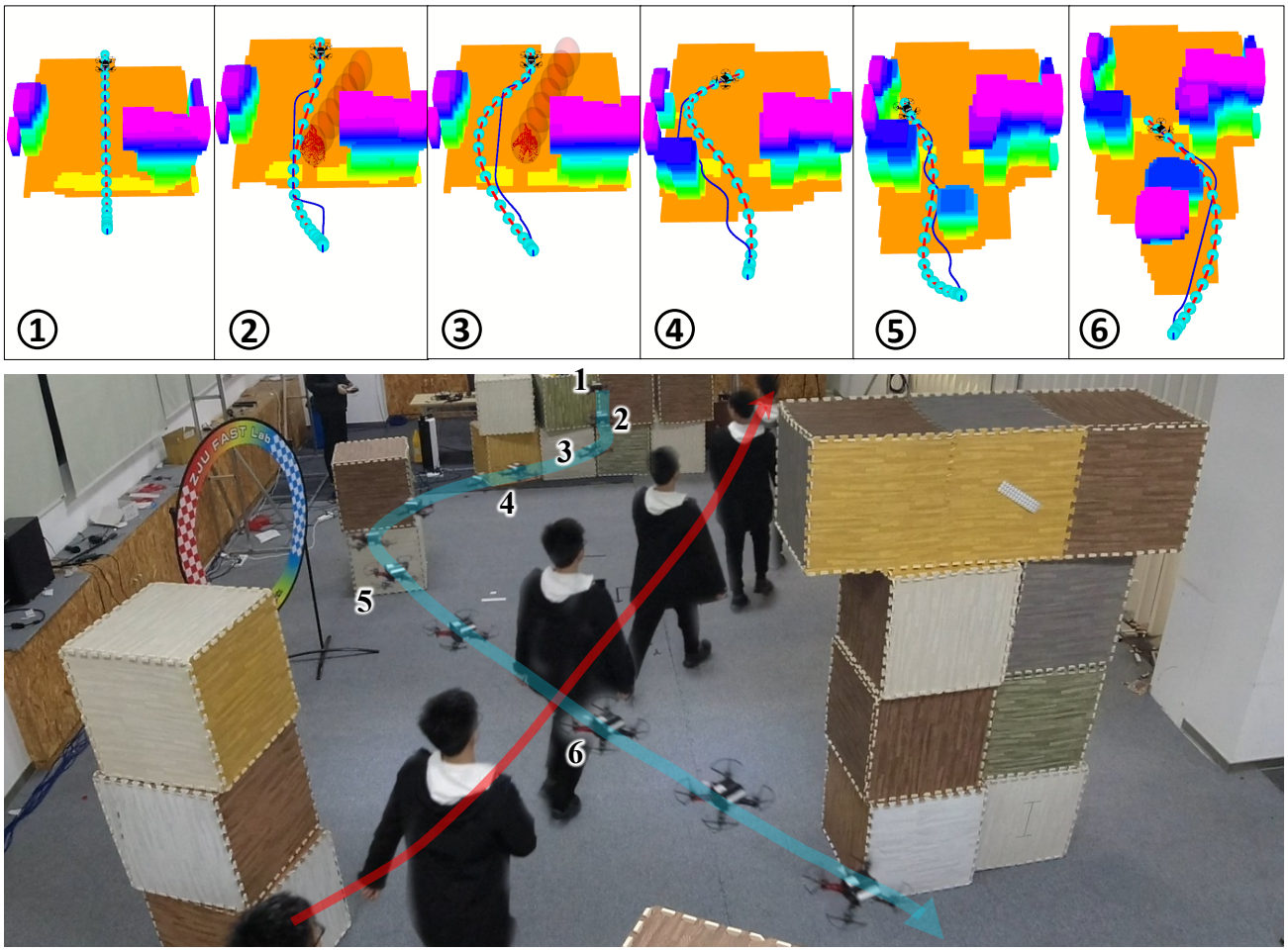

Fig. 10. A sequence of optimized trajectories and corresponding snapshot during the experiment. An UAV is required to fly from a start point to a goal, avoiding an obliquely walking person and a plenty of obstacles. Top: estimated position and velocity of the moving person and optimized trajectory. Bottom: Snapshot of the experiment.

TABLE I
DYNAMIC PERCEPTION COMPARISON

Scenario	Method	$error_{pos}$(m)	$error_{vel}$(m/s)
$simulation$	Ours	**0.11**	**0.19**
$experiment$	Method [3]	0.14	0.36
$real-world$	Ours	**0.18**	**0.29**
$experiment$	Method [3]	0.31	0.57

TABLE II
DYNAMIC PLANNING COMPARISON

Scenario	Method	t_{arrive}(s)	l_{traj}(m)	v_{mean}(m/s)	t_{opt}(ms)
$N=20$,	Ours	15.75	31.1	2.07	**5.76**
$v=1m/s$	Method [37]	14.7	30.0	2.04	15
$N=20$,	Ours	16.2	32.1	2.02	**6.34**
$v=2m/s$	Method [37]	15.75	31.3	2.01	14
$N=50$,	Ours	16.7	36.1	2.31	**6.27**
$v=1m/s$	Method [37]	16.2	35.3	2.21	17

with [37]. In [37], authors assume that locations and velocities of all obstacles are known and do not classify them as dynamic or static. Therefore, we only compare the dynamic planning module in a fully dynamic environment without static obstacles. Each planner runs for twenty times of different obstacle velocities and densities from the same start points to endpoints. In each simulated experiment, obstacles move with constant velocities and reverse when they arrive at the boundary of $20\times20\times3m$ map. The average performance statistics and the computation time are shown in Table.II

From Table.II, we conclude that the performance of our generated trajectory is comparable to [37]. However, our proposed planner needs a much lower computation budget. This is because the time complexity of our method is $O(N_c)$, since one control point only affects nearby segments thanks to the local support property of B-spline, while [37] plans the pose of a quadrotor with nonlinear model predictive control, which considers more constrains. Moreover, our method would not fall into a local minimum, while [37] would.

VI. CONCLUSION

In this paper, we introduce a robust vision-based system with onboard sensing for UAV in the dynamic environment and implement it on a computational power-limited quadrotor platform. Firstly we adopt an accurate yet efficient dynamic

perception module, which classify obstacles as dynamic or static, obtain necessary state information of moving obstacles, and build a high-quality static occupancy grid map for navigation. Then, our proposed gradient-based planning framework generates local collision-free trajectories without maintaining ESDF by a carefully designed optimization in less than 7ms. Sufficient real-world experiments and benchmark comparisons validate that our system is effective, robust, and highly lightweight in unknown cluster environment.

REFERENCES

[1] T. Eppenberger, G. Cesari, M. Dymczyk, R. Siegwart, and R. Dubé, "Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking," *arXiv preprint arXiv:2007.10743*, 2020.

[2] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, 2020.

[3] 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)*. IEEE, 2020, pp. 2682–2688.

[4] H. Cheng, L. Lin, Z. Zheng, Y. Guan, and Z. Liu, "An autonomous vision-based target tracking system for rotorcraft unmanned aerial vehicles," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1732–1738.

[5] C. Zhan, X. Duan, S. Xu, Z. Song, and M. Luo, "An improved moving object detection algorithm based on frame difference and edge detection," in *Fourth international conference on image and graphics (ICIG 2007)*. IEEE, 2007, pp. 519–523.

[6] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "Mid-fusion: Octree-based object-level multi-instance dynamic slam," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5231–5237.

[7] K. Berker Logoglu, H. Lezki, M. Kerim Yucel, A. Ozturk, A. Kucukkomurler, B. Karagoz, E. Erdem, and A. Erdem, "Feature-based efficient moving object detection for low-altitude aerial platforms," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2119–2128.

[8] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust dense mapping for large-scale dynamic environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7510–7517.

[9] W. Wu, Z. Wang, Z. Li, W. Liu, and L. Fuxin, "Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds," *arXiv preprint arXiv:1911.12408*, 2019.

[10] S. L. Francis, S. G. Anavatti, and M. Garratt, "Detection of obstacles in the path planning module using differential scene flow technique," in *2015 International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*. IEEE, 2015, pp. 53–57.

[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[13] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Towards reaching human performance in pedestrian detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 973–986, 2017.

[14] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 5636–5643.

[15] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE international symposium on mixed and augmented reality*. IEEE, 2011, pp. 127–136.

[16] S. Bouaziz, A. Tagliasacchi, and M. Pauly, "Sparse iterative closest point," in *Computer graphics forum*, vol. 32, no. 5. Wiley Online Library, 2013, pp. 113–123.

[17] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for mav flight," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 50–56.

[18] P. Skulimowski, M. Owczarek, A. Radecki, M. Bujacz, D. Rzeszotarski, and P. Strumillo, "Interactive sonification of u-depth images in a navigation aid for the visually impaired," *Journal on Multimodal User Interfaces*, vol. 13, no. 3, pp. 219–230, 2019.

[19] F. Arnell and L. Petersson, "Fast object segmentation from a moving camera," in *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* IEEE, 2005, pp. 136–141.

[20] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[21] D. Wilkie, J. Van Den Berg, and D. Manocha, "Generalized velocity obstacles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 5573–5578.

[22] J. Van Den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3475–3482.

[23] T. Fraichard and H. Asama, "Inevitable collision statesa step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[24] D. Althoff, J. J. Kuffner, D. Wollherr, and M. Buss, "Safety assessment of robot trajectories for navigation in uncertain and dynamic environments," *Autonomous Robots*, vol. 32, no. 3, pp. 285–302, 2012.

[25] N. Malone, H.-T. Chiang, K. Lesser, M. Oishi, and L. Tapia, "Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1124–1138, 2017.

[26] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1986–1991.

[27] H.-T. L. Chiang, B. HomChaudhuri, A. P. Vinod, M. Oishi, and L. Tapia, "Dynamic risk tolerance: Motion planning by balancing short-term and long-term stochastic dynamic predictions," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3762–3769.

[28] F. Gao and S. Shen, "Quadrotor trajectory generation in dynamic environments using semi-definite relaxation on nonconvex qcqp," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 6354–6361.

[29] C. Cao, P. Trautman, and S. Iba, "Dynamic channel: A planning framework for crowd navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5551–5557.

[30] D. Zhu, T. Zhou, J. Lin, Y. Fang, and M. Q.-H. Meng, "Online state-time trajectory planning using timed-esdf in highly dynamic environments," *arXiv preprint arXiv:2010.15364*, 2020.

[31] K. Khan, S. U. Rehman, K. Aziz, S. Fong, and S. Sarasvady, "Dbscan: Past, present and future," in *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*, 2014, pp. 232–238.

[32] A. Azim and O. Aycard, "Detection, classification and tracking of moving objects in a 3d environment," in *2012 IEEE Intelligent Vehicles Symposium*. IEEE, 2012, pp. 802–807.

[33] Y. Tian, L. Deng, and Q. Li, "A knn match based tracking-learning-detection method with adjustment of surveyed areas," in *2017 13th International Conference on Computational Intelligence and Security (CIS)*. IEEE, 2017, pp. 447–451.

[34] 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.

[35] A. Y. Uteshev and M. V. Goncharova, "Point-to-ellipse and point-to-ellipsoid distance equation analysis," *Journal of computational and applied mathematics*, vol. 328, pp. 232–251, 2018.

[36] 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.

[37] H. Zhu and J. Alonso-Mora, "Chance-constrained collision avoidance for mavs in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 776–783, 2019.