Onboard Dynamic-Object Detection and Tracking for Autonomous Robot Navigation With RGB-D Camera

Zhefan Xu ⑩, Xiaoyang Zhan ⑩, Yumeng Xiu ⑩, *Member, IEEE*, Christopher Suzuki ⑩, and Kenji Shimada ⑩

Abstract—Deploying autonomous robots in crowded indoor environments usually requires them to have accurate dynamic obstacle perception. Although plenty of previous works in the autonomous driving field have investigated the 3D object detection problem, the usage of dense point clouds from a heavy Light Detection and Ranging (LiDAR) sensor and their high computation cost for learning-based data processing make those methods not applicable to small robots, such as vision-based UAVs with small onboard computers. To address this issue, we propose a lightweight 3D dynamic obstacle detection and tracking (DODT) method based on an RGB-D camera, which is designed for low-power robots with limited computing power. Our method adopts a novel ensemble detection strategy, combining multiple computationally efficient but low-accuracy detectors to achieve real-time high-accuracy obstacle detection. Besides, we introduce a new feature-based data association and tracking method to prevent mismatches utilizing point clouds' statistical features. In addition, our system includes an optional and auxiliary learning-based module to enhance the obstacle detection range and dynamic obstacle identification. The proposed method is implemented in a small quadcopter, and the results show that our method can achieve the lowest position error (0.11 m) and a comparable velocity error (0.23 m/s) across the benchmarking algorithms running on the robot's onboard computer. The flight experiments prove that the tracking results from the proposed method can make the robot efficiently alter its trajectory for navigating dynamic environments. Our software is available on GitHub[1] as an open-source ROS package.

Index Terms—RGB-D perception, vision-based navigation, visual tracking, 3D object detection, collision avoidance.

I. INTRODUCTION

SMALL autonomous mobile robots, frequently employed in indoor scenarios, often operate in dynamic and unpredictable environments populated by humans, vehicles, and other robots. Ensuring safe navigation in such settings necessitates real-time, accurate perception of dynamic obstacles. However, many small robots are only equipped with onboard computers
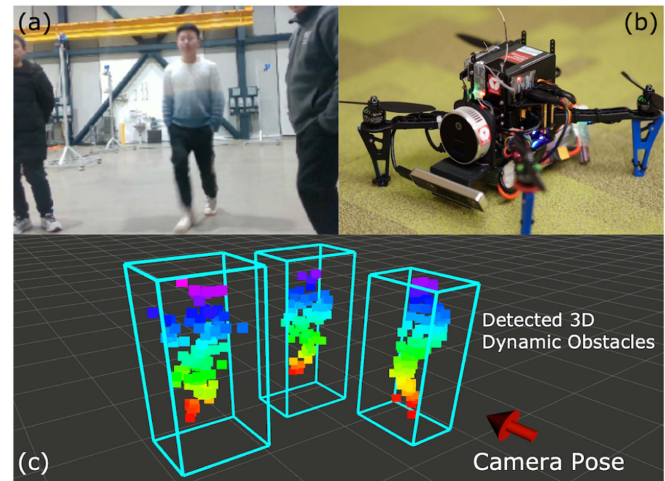
Fig. 1.   Onboard dynamic obstacle detection results from the proposed DODT algorithm. (a) The camera RGB view. (b) An example of an autonomous robot with an RGB-D camera. (c) The onboard 3D dynamic obstacle detection results shown as blue bounding boxes with point clouds.

with limited computational capabilities and rely on RGB-D cameras. This makes GPU-intensive learning-based methods, common in autonomous driving, unsuitable. Hence, the development of a lightweight RGB-D camera-based dynamic obstacle detection and tracking method is necessary for autonomous robots operating in dynamic environments.

There are three challenges in small mobile robots' detection and tracking. First, small mobile robots' onboard computation resources are limited, making GPU-demanding learning-based methods [1], [2] not applicable. Note that we define small mobile robots as those with weights below 1.5 kg, equipped with low-power (10–20 Watts) onboard computers measuring around 10 cm in length, shown in Fig. 1(b). Second, the range and field of view (FOV) of depth cameras suited for small mobile robots are limited, which makes obstacles either too close or too far and thus not detectable. For example, the ideal depth range of the popular Intel RealSense D435i depth camera is from 0.3 m to 3.0 m. This camera limitation makes some previous works [3], [4] only capable of tracking obstacles in the short range. Third, the noises from the depth value estimation of the camera are not negligible, especially for those noise-sensitive non-learning methods [5], [6]. The camera noises can make the detection algorithm not only estimate obstacle states inaccurately but also produce high-frequency false-positive and false-negative results, leading to confusion for obstacle avoidance planners.

To solve these issues, this letter presents an onboard 3D dynamic obstacle detection and tracking (DODT) method based on an RGB-D camera. In contrast to other low-computational algorithms [3], [4], [6], which employ a single detector, we propose a novel ensemble detection strategy combining multiple computationally efficient but low-accuracy detectors to obtain fast and more accurate obstacle detection results. Moreover, the proposed method incorporates feature-based data association, utilizing statistical features from point clouds, and employs the Kalman filter for obstacle tracking. This approach reduces tracking mismatches that can occur with the center-distance-based association methods employed by benchmarking algorithms. Then, we use both point cloud and velocity criteria to identify dynamic obstacles. Finally, the system introduces a novel usage for the learning-based detector as an auxiliary and optional module to enhance the detection range and dynamic obstacle identification when the robot's computation resources are enough. The contributions of this work are:

- *Efficient Ensemble Detection:* Different from other detection and tracking algorithms designed for low-computational robots with a single detector, the proposed algorithm runs multiple computationally efficient and low-accuracy detectors with a novel ensemble strategy to obtain more accurate results with high efficiency.
- *Feature-based Association and Tracking:* Unlike the traditional center-distance-based association methods in other algorithms, the feature-based association reduces tracking mismatches by utilizing statistical features from point clouds, improving the tracking accuracy.
- *Auxiliary Learning-based Detection Module:* The system incorporates a novel integration of the learning-based detector as an auxiliary module, enhancing the detection range and dynamic obstacle identification when the robot has sufficient computational resources.

II. RELATED WORK

Among small robots with limited computational power, such as UAVs and small UGVs, the detection and tracking methods can be categorized based on the sensors, including LiDARs [7], [8], [9], [10], event cameras [11], [12], and RGB-D cameras [3], [4], [6]. Among them, the RGB-D camera is one of the most popular sensors for small mobile robots, and there are mainly two ways of using the RGB-D camera.

Image-based methods: Most methods in this category leverage depth images for 3D obstacle detection. For instance, in [13], depth images are employed to generate U-depth maps and V-depth maps, enabling the estimation of obstacle states and proving safe navigation with static obstacles. Building on this, Lin et al. [14] adopt a similar U-depth map to detect and track obstacles, representing them as 3D ellipsoids. To enhance the accuracy of obstacle dimension estimation, the restricted V-depth map is introduced in [15]. In [6], dynamic obstacles identified from the depth and U-depth maps are characterized by their estimated velocities. These dynamic obstacle detection outcomes are integrated with the occupancy map to navigate dynamic environments. In contrast to prior depth image-based

approaches, Lu et al. [16] apply the YOLO detector to effectively avoid fast and small dynamic obstacles. Additionally, Sun et al. [17] employ image differences to identify all dynamic points from RGB images. Moreover, Logoglu et al. [18] combine the 3-image-difference technique with epipolar constraints to determine dynamic obstacles. They extend their approach by utilizing scene flow, an extension of optical flow, in [19], [20], for detecting the velocity of each pixel and identifying dynamic points. Some alternative methods focus on detecting and segmenting dynamic obstacles in 2D image planes to enhance SLAM robustness. In [17], [21], [22], [23], these approaches concentrate on removing dynamic obstacles from images to mitigate estimation errors, while Qiu et al. [24] detect pedestrian skeletons to improve SLAM optimization.

Point cloud-based methods: Unlike image-based methods, point cloud-based approaches directly detect 3D obstacles using geometric information from point clouds. In [3], a point cloud clustering method is combined with the YOLO detector for human detection. Wang et al. [4] employ a similar clustering-based detection approach for indoor dynamic obstacle avoidance using a quadcopter. To enhance obstacle tracking robustness, Chen et al. [5] propose using point cloud feature vectors and object track points to identify correct object matches and estimate their states. In [25], a KD-Tree map is directly constructed from the LiDAR point cloud for dynamic obstacle avoidance. Min et al. [26] represent dynamic obstacles in a dynamic occupancy map and employ kernel inference to reduce computation. Likewise, in [27], a dual-structure particle-based dynamic occupancy map is utilized to represent dynamic environments and classify obstacle particles as static or dynamic.

Both image and point cloud methods can suffer from misdetection due to noise and complex environments. To address this, we propose an ensemble method that leverages different detectors to mitigate their individual shortcomings. Additionally, we suggest using the learning-based method as an optional auxiliary module, enhancing adaptability for robots with varying computational resources.

III. METHODOLOGY

A. System Overview

Considering the payload and computational constraints of small mobile robots, both computational-intensive learning-based 3D object detectors and heavy LiDAR systems become impractical. To address this constraint, we have devised a lightweight detection and tracking framework comprising three core modules: the detection module, the tracking module, and the identification module, as shown in Fig. 2. The detection module comprises a non-learning and a learning-based component. The non-learning part employs depth images and two non-learning detectors for generic obstacle detection. Meanwhile, the learning-based module uses aligned RGB-D images for direct dynamic obstacle detection, and its results are combined with the non-learning module. Details of each detector are in Section III-B, with ensemble detection explained in Section III-C. Refined 3D bounding boxes are used in the tracking module (Section III-D) to estimate obstacle states using historical data.
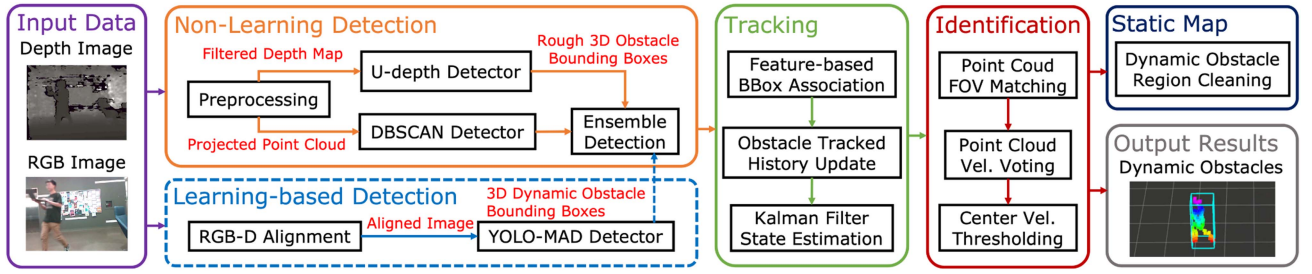
Fig. 2. Proposed dynamic obstacle detection and tracking system (DODT) framework. The input data are the RGB-D images. The non-learning detection module first uses the depth image to detect generic obstacles. Then, the tracking module is applied to track and estimate the obstacles states. With the identification module, the dynamic obstacles are identified from all detected obstacles. Finally, the output results show the dynamic obstacles' bounding boxes. The dynamic obstacle regions are cleaned in the static occupancy map. The optional learning-based detection module, presented in the blue dotted line, uses color and depth images to detect dynamic obstacles, enhancing the detection range, and dynamic obstacle identification.
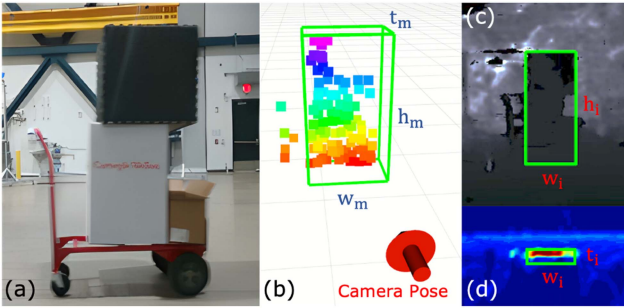

Fig. 3. Illustration of the U-depth detector. (a) The camera RGB view. (b) The detected 3D bounding box with the obstacle point cloud. (c) The 2D detection on the depth map. (d) The 2D detection on the U-depth map.
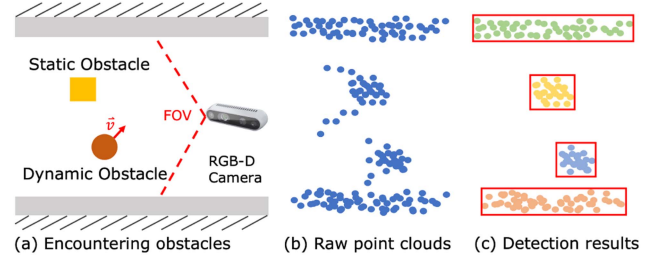

Fig. 4. Illustration of the DBSCAN detector. (a) The robot encounters obstacles in a corridor. (b) The raw point cloud data from the RGB-D camera are unstructured and noisy. (c) The DBSCAN detector takes the filtered point cloud and performs clustering to get obstacles' bounding boxes.

The identification module (Section III-E) classifies obstacles as static or dynamic based on state and tracking history. The system outputs dynamic obstacle bounding boxes, and dynamic obstacle regions are cleared in the static map for navigation.

B. 3D-Obstacle Detectors

This section introduces three computationally efficient but low-accuracy 3D obstacle detectors: the U-depth, the DBSCAN, and the YOLO-MAD detector. Note that all detection results are represented as axis-aligned bounding boxes. We select the U-depth and the DBSCAN detectors as the non-learning detectors due to their high computational efficiency demonstrated in small UAV 3D dynamic obstacle detection applications [4], [6]. Besides, their detection errors come from different sources (the depth image and the point cloud) obtained from the RGB-D camera, ensuring the ensemble strategy takes effect. For the learning-based detector, we choose an extremely lightweight implementation of a popular model and extend it into a 3D detector, which can run in real time on onboard computers without GPU acceleration.

U-depth Detector: The U-depth detector for obstacle detection is mentioned in the previous works [6], [13], [14]. Overall, the detector takes the depth image to generate 3D bounding boxes of static and dynamic obstacles. Fig. 3 visualizes sample detection results. There are three steps in the U-depth detector: (1) the U-depth map generation, (2) the line grouping on U-depth, and (3) the depth continuity search on the original depth image.

The U-depth map can be intuitively viewed as the top-down view from the camera. It has the same width as the original depth image, and its vertical axis from top to bottom indicates the increasing distance to the camera. When we get a depth image, we can compute the U-depth map using the column depth value histogram. Fig. 3(c) and (d) show a depth image and U-depth map pair. Then, we can perform the line grouping method on the generated U-depth map to get the 2D bounding box of the obstacle of width w_i and thickness t_i shown in Fig. 3(d) (note that i indicates the image plane). With the obstacle width w_i, we do the depth value continuity check on the original depth image to get the height h_i of the obstacle shown in Fig. 3(c). After having both 2D bounding boxes in the U-depth map and the original depth image, we can triangulate 3D points into the camera frame and perform coordinate transform to get the obstacle position and dimension of the world/map coordinate frame (Fig. 3(b)).

DBSCAN Detector: Unlike the image-based detector, the DBSCAN detector uses point cloud data to detect obstacles. DBSCAN is an unsupervised machine-learning algorithm for clustering which can automatically determine the cluster number. The illustration of the DBSCAN detector is shown in Fig. 4. When the robot encounters obstacles, the raw point cloud data can be triangulated from the depth image as shown in Fig. 4(b). Note that because of the sensor, the point cloud data can be noisy on the obstacle boundaries. So, we apply the voxel filter proposed in [3] to remove the noise of the point cloud and then perform DBSCAN clustering to get obstacles' bounding boxes (Fig. 4(c)). Similar to the U-depth detector, the DBSCAN detector does not need a training dataset and only requires a few computation resources.
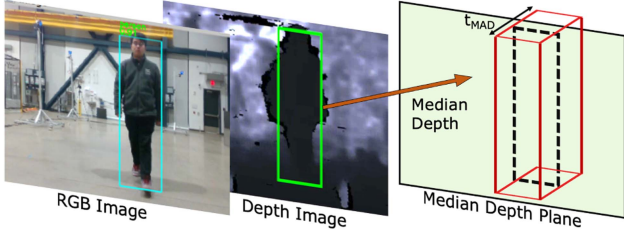
Fig. 5.   Illustration of the YOLO-MAD detector. The RGB image is used to get the 2D detection result, and then the bounding box on the depth image is obtained. With the 2D result on the depth image, the 3D bounding box is calculated by the proposed median absolute deviation (MAD) method.

YOLO-MAD Detector: The previously mentioned detectors rely on geometric structures of either depth images or point clouds. So, they cannot identify the type of obstacles (i.e., static or dynamic) and might even fail when the obstacles are far from the camera. To overcome these limitations, we introduce our 3D YOLO-MAD detector based on the 2D YOLOFastestDet, which can run real-time at an onboard CPU such as Intel NUC. The illustration of the YOLO-MAD detector is shown in Fig. 5. The detector first detects the 2D bounding box of each obstacle on the RGB image and finds the corresponding region on the aligned depth image. To find the depth and thickness of the 2D bounding box, we first calculate the median absolute deviation (MAD) based on the median depth value $\tilde{d}$ in the bounding box region $\mathcal{R}_{\text{box}}$:

$$\text{MAD} = \textbf{median}(|d_i - \tilde{d}|), \ \ d_i \in \textbf{depth}(\mathcal{R}_{\text{box}}), \quad (1)$$

where d_i is the depth value of ith pixel in the bounding box region $\mathcal{R}_{\text{box}}$. Then, we can search the minimum depth $d_{\min}$ and maximum depth $d_{\max}$ in the MAD range $\mathcal{S}_{\text{MAD}}$:

$$\mathcal{S}_{\text{MAD}} = \{d_i | \tilde{d} - n \cdot \text{MAD} \leq d_i \leq \tilde{d} + n \cdot \text{MAD}\}, \quad (2)$$

where n is a user-defined parameter. The obstacle's thickness t_{MAD} can be calculated based on the minimum and maximum depth values. The MAD range $\mathcal{S}_{\text{MAD}}$ can help filter the outlier depth values in the bounding box region from the background and the sensor noises. Finally, we can triangulate the points from the depth image at the median depth plane with the thickness to get the 3D obstacle's bounding box. Since this learning-based detector can still be computationally heavy for some extremely low-power onboard computers, we treat it as an optional and auxiliary module in our framework.

C. Ensemble Detection

This section introduces our proposed ensemble detection method to obtain refined obstacles' bounding boxes. In our framework, three detectors run in parallel and individually detect obstacles' bounding boxes. Since the previously mentioned detectors are designed to compensate for the detection accuracy for high-speed performance, they are all sensitive to different environments and sensor noises, leading to false positives and inaccurate obstacle dimension estimation. So, the intuition of the ensemble detection is to combine the detection results of different detectors and find their "mutual agreements" of detection results for reducing the noise effects. This technique can

---

Algorithm 1: Ensemble Detection Algorithm.

1:	$\mathcal{B}_{en} \leftarrow \emptyset$; ▷ ensembled bounding boxes
2:	$\mathcal{B}_{d1} \leftarrow \textbf{getDetBBox1}()$; ▷ detector1 results
3:	$\mathcal{B}_{d2} \leftarrow \textbf{getDetBBox2}()$; ▷ detector2 results
4:	**for** b_{d1} **in** $\mathcal{B}_{d1}$ **do**
5:	$\quad \mathcal{S}_{\text{iou1}}, b_{\text{match1}} \leftarrow \textbf{findBestIOUMatch}(b_{d1}, \mathcal{B}_{d2})$;
6:	$\quad \mathcal{S}_{\text{iou2}}, b_{\text{match2}} \leftarrow \textbf{findBestIOUMatch}$ $(b_{\text{match1}}, \mathcal{B}_{d1})$;
7:	$\quad \mathcal{C}_{\text{match}} \leftarrow b_{\text{match2}}$ **is** b_{d1};
8:	$\quad$ **if** $\mathcal{S}_{\text{iou1}} > \mathcal{S}_{\text{thr}}$ **and** $\mathcal{S}_{\text{iou2}} > \mathcal{S}_{\text{thr}}$ **and** $\mathcal{C}_{\text{match}}$ **then**
9:	$\quad\quad b_{en} \leftarrow \textbf{fuseBBoxes}(b_{d1}, b_{\text{match1}})$;
10:	$\quad\quad \mathcal{B}_{en}.\textbf{push\_back}(b_{en})$;
11:	**return** $\mathcal{B}_{en}$;

---

significantly improve detection robustness and accuracy with environment and sensor noises.

The proposed ensemble detection algorithm follows a pairwise manner presented in Algorithm 1. When we obtain two sources of detection results, we go through each bounding box b_{d1} from one detector's results (Line 4). For the bounding box b_{d1}, the algorithm finds the bounding box b_{match1} with the highest intersection-over-union (IOU) score from the other detection bounding boxes (Line 5). Following the same way, the bounding box b_{match2} is obtained by finding the highest IOU match of b_{match1} in the first detection bounding boxes (Line 6). Through this process, we want to find the bounding boxes that are detected by both detectors. Then, we need to ensure that the IOU score of their matched bounding boxes exceeds the predefined threshold and that their matched bounding boxes have the highest IOU score to each other (Line 8). Finally, we fuse two bounding boxes into a new ensembled bounding box (Lines 9 and 10). We adopt a conservative method for fusing bounding boxes: the new ensembled bounding box takes the maximum values in dimensions and the average value in positions. In our system framework (Fig. 2), we first ensemble detection results from the U-depth and DBSCAN detectors and then combine the YOLO-MAD results if the learning-based module is running.

D. Data Association and Tracking

Overall, the proposed module first applies the feature-based data association method to match the detected obstacles at the current time t_n with the obstacles at the previous time t_{n-1}. Then, it applies the Kalman filter with the constant-acceleration motion model to estimate the obstacles' states and add them to the estimation histories. In contrast to the constant-velocity model used in previous works [4], [6], [14], the constant acceleration model offers more accurate state estimation and dynamic obstacle identification.

Feature-based Data Association: The detected obstacles at the current time t_n are associated with the obstacles at the previous time t_{n-1} using the feature comparison. The feature vector of the obstacle O_i is defined as:

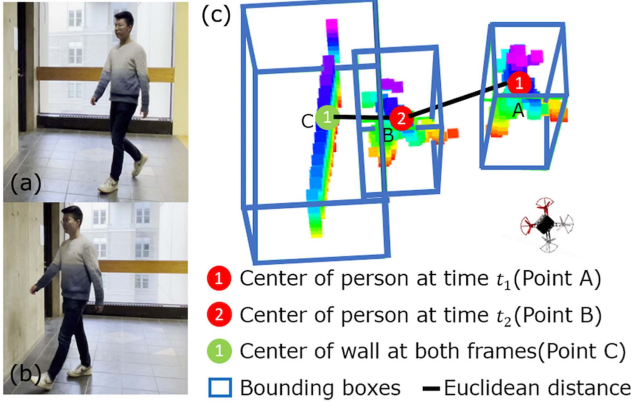$$feat(O_i) = [pos(i), dim(i), len(i), std(i)], \quad (3)$$

Fig. 6. Illustration of the issue with the center-distance-based data association method. (a) The RGB image at time t_1. (b) The RGB image at time t_2. (c) The center-distance-based data association method might fail by incorrectly associating the current detected person with the wall.

where $pos(i)$ is the obstacle's center position, $dim(i)$ is the obstacle's dimension in x, y and z direction, $len(i)$ is the obstacle's point cloud size, and $std(i)$ is the obstacle's point cloud standard deviation. Then, we perform normalization for the feature vector to reduce the effects from the different dimensions. After that, the similarity score between obstacles O_i and O_j is calculated using the following equation:

$$sim(O_i, O_j) = exp(-||feat(O_i) - feat(O_j))||_2^2), \quad (4)$$

where we take the exponential of the negative L2 norm of the feature difference. With the scores, the obstacle $O_i^{t_n}$ at the current time t_n can be matched with the obstacle $O_j^{t_{n-1}}$ at the previous time t_{n-1} with the highest similarity score sim_{max}. Instead of directly using the previous obstacle's feature, we apply the linear propagation to get the predicted obstacle's position and replace the previous obstacle's position with the predicted position in the feature vector. Also, the highest similarity score must be higher than a predefined threshold ($sim_{max} > T_{sim}$) to prevent incorrect associations.

The proposed feature-based data association method can overcome the drawback of traditional center-distance-based association, as shown in Fig. 6. In Fig. 6(a) and (b), a scenario is presented where a person approaches the wall with the point clouds of all obstacles shown in Fig. 6 c. Since the center of the wall (Point C) is closer to the person's position at the current time t_2 (Point B) than the person's position at the previous time t_1 (Point A), a center-distance-based tracking will associate the person with the wall. On the contrary, if the proposed feature-based association method is applied, the person and wall will not be matched together because of the obvious differences in the obstacles' dimensions, velocities, point cloud sizes, and standard deviations. So, the detected person at the current time t_2 will be correctly associated with the person at the previous time t_1.

Constant-Acceleration Kalman Filter: The states of each obstacle are estimated by the Kalman filter with a constant-acceleration motion model. Unlike the previous work [4], [6], where the velocities of obstacles are assumed to be constant, our method allows the obstacles' velocities to change without

increasing the complexity of the motion model too much. We will discuss all quantities in global map frame for simplicity. The obstacle states are defined as $X = [x, y, \dot{x}, \dot{y}, \ddot{x}, \ddot{y}]^T$, including the position, the velocity, and the acceleration in x and y directions. The measurement vector is the same as the obstacle state vector. To calculate the measurement of the velocity vector $\mathbf{V}_i$ and acceleration vector $\mathbf{A}_i$ at time t, we adopt the following equations:

$$\mathbf{V}_t = \frac{\mathbf{P}_t - \mathbf{P}_{t-1}}{\delta t}, \ \mathbf{A}_t = \frac{\mathbf{V}_t - \mathbf{V}_{t-1}}{\delta t}, \quad (5)$$

where δt is the time difference. Note that we take the data from several time differences δt to calculate smoother observations. In this way, the system model is described by:

$$X_{t|t-1} = AX_{t-1} + Bu_{t-1} + Q, \quad (6)$$

where A is the state transition matrix, Q is the covariance of the motion model noise, u is the control input, which is zero in this case. Since the acceleration model is assumed, the state transition matrix can be calculated by:

$$A = \begin{bmatrix} 1 & 0 & \delta t & 0 & \frac{\delta t^2}{2} & 0 \\ 0 & 1 & 0 & \delta t & 0 & \frac{\delta t^2}{2} \\ 0 & 0 & 1 & 0 & \delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (7)$$

and the system measurement is defined as:

$$Z_t = HX_t + R, \quad (8)$$

where the measurement matrix H is an identity matrix, and R is the covariance of measurement noise.

E. Dynamic Obstacle Identification

This section describes how to identify the status of an obstacle (dynamic or static). By default, any quantities defined in the following are at the current time t_n. As the first dynamic obstacle identification criteria, all the bounding boxes of obstacles with the center velocity $\mathbf{V}_{center}$ less than a threshold T_{vel} will be classified as static. Although the velocity criteria should theoretically filter out all static obstacles, the noises from detection and state estimation can cause false-positive dynamic obstacle identification. To reduce the false-positive identification results, in the second identification step, the module takes all valid points of an obstacle's point cloud to vote for its status. In this step, every point at the current time t_n is matched with its corresponding point at the time t_{n-k} by the nearest neighbor search. After determining the correspondence, the velocity of each point $\mathbf{V}_{vote}^i$ is calculated. Then, a point will vote for the obstacle as dynamic if its velocity exceeds a predefined threshold T_{vote}. If the ratio of dynamic votes N_{vote} over the number of valid points N_{valid} is higher than another threshold T_{ratio}, the obstacle will be identified as a dynamic obstacle:

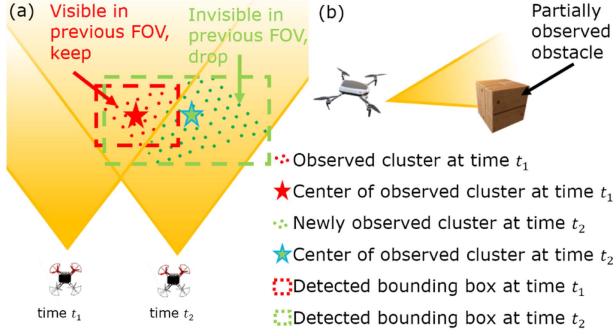$$\frac{N_{vote}}{N_{valid}} > T_{ratio}. \quad (9)$$

Fig. 7. Illustration of removing the invalid points using the field of view (FOV) criteria. (a) The analysis of the observed obstacle's point cloud at different time. (b) The robot detecting a partially visible obstacle.
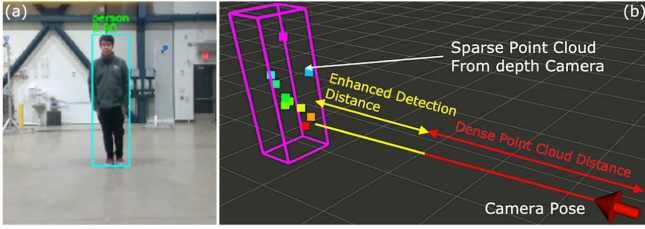

Fig. 8. Illustration of enhancing detection range by the auxiliary learning-based module. The red line measures the maximum ideal range to produce dense point cloud data for the DBSCAN and U-depth detectors to detect obstacles. The yellow line indicates the increased detection distance.
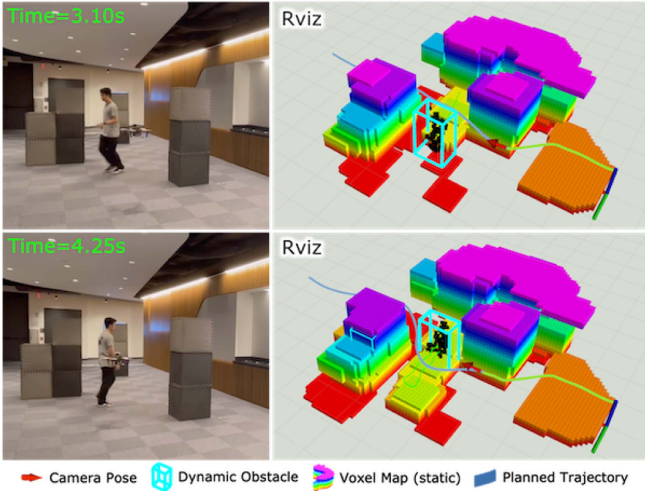

Fig. 9. Autonomous robot navigation in dynamic environments using the proposed algorithm. The onboard obstacle detection results (blue bounding boxes) can help the robot modify its planned path to avoid obstacles safely.

Before the dynamic voting process, it is necessary to drop the invalid points from the point cloud. First, if any point $p_{i,j}$ with the point cloud index i in obstacle j has an invalid velocity $\mathbf{V}_{vote}^{i}$, it will be removed from the dynamic voting process. The valid velocity should satisfy the condition:

$$angle(\mathbf{V}_{vote}^{i}, \mathbf{V}_{center}^{j}) < \frac{\pi}{2}, \qquad (10)$$

where we ensure that points with incorrect velocity estimations are removed. Second, if any point $p_{i,j}$ at time t_n is invisible at time t_{n-k}, it will also be removed from voting shown in Fig. 7.

TABLE I
BENCHMARKING OF THE DETECTION AND TRACKING RESULTS IN THE POSITION ERRORS, VELOCITY ERRORS, AND THE FALSE POSITIVE RATES

Method	Pos. Err. (m)	Vel. Err. (m/s)	FP Rate (%)
Method I [14]	0.28	0.47	N/A
Method II [4]	0.18	0.29	16.4%
Method III [6]	0.19	**0.21**	19.6%
DODT w/o Ens	0.17	0.30	18.6%
DODT w/o FAT	0.14	0.29	6.5%
DODT (Ours)	**0.11**	0.23	**3.7%**

The bold values indicate the best results among all.

Fig. 7(b) shows a scenario where a robot approaches a partially visible static obstacle. At the previous time t_1, only red points are visible; the detected center of the obstacle is the red star. At the current time t_2, the whole box is visible, and the center of the obstacle shifts a lot. In this case, the obstacle will have a large center velocity $\mathbf{V}_{center}$ and voting velocity $\mathbf{V}_{vote}$ due to incorrect points correspondence. Our method drops the newly observed points from the voting and identifies the obstacle as static. Finally, when the YOLO-MAD Detector is applied, its classification results will be used for dynamic obstacle identification, skipping all the processes mentioned above.

IV. RESULT AND DISCUSSION

To evaluate the performance of the proposed method, we conduct experiments in dynamic environments. The algorithm is implemented in C++, running on two customized quadcopters with the Intel NUC and NVIDIA Jetson Xavier NX onboard computers, respectively. All the computations are performed real-time on the robots' onboard computers.

A. Performance Benchmarking

To assess our algorithm's performance, we conduct comparative experiments with state-of-the-art dynamic obstacle detection and tracking methods in the UAV platform [4], [6], [14]. We also evaluate the impact of ensemble detection and feature-based association and tracking by comparing our method's performance with and without these features. For experiments without ensemble detection (DDOT w/o Ens), we use the U-depth detector due to its higher accuracy compared to the DBSCAN detector. In the absence of feature-based association and tracking (DODT w/o FAT), we apply center-distance-based association with the constant-velocity model. We employ various evaluation metrics, including position and velocity estimation errors and the false-positive detection rate. The false-positive rate is determined by dividing the number of misdetections (identifying static obstacles as dynamic) by the total number of detections. Ground truth measurements are acquired from the OptiTrack motion capture system. Table I summarizes the comparison results. Our DODT method exhibits the lowest position errors among all methods, with our velocity error ranking second, comparable to Method III [6]. Ensemble detection significantly reduces the false-positive detection rate by leveraging consensus among detectors and enhances obstacle position and velocity estimation accuracy. Additionally, the feature-based association
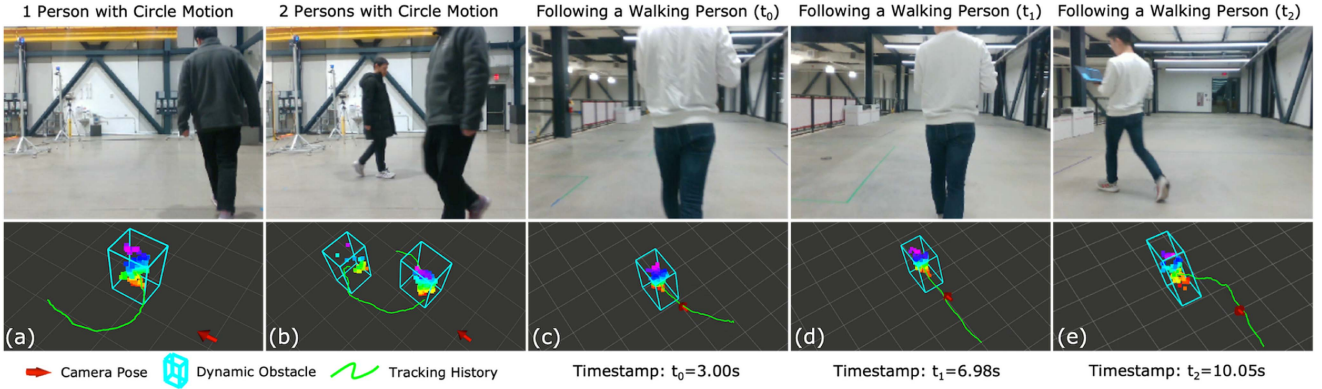
Fig. 10. Dynamic obstacle detection and tracking experiments with a handheld robot camera. The blue bounding boxes containing point clouds visualize the dynamic obstacles' detection results, and the tracking histories are shown as green curves. Figures (a) and (b) show the detection results of persons walking in circles. Figures (c), (d), and (e) show the long-distance dynamic obstacle detection and tracking ability following a walking person.

TABLE II
RUNTIME OF EACH MODULE OF THE PROPOSED SYSTEM

System Modules	Intel NUC (ms)	Xavier NX (ms)
U-depth detection	3.4	12.0
DBSCAN detection	1.3	4.0
YOLO-MAD detection	14.3	23.5
Feature-based Data Assoc.	0.03	0.08
Kalman filter tracking	0.07	0.17
Dynamic Obstacle Id.	0.12	0.33
System Total Runtime	**19.12**	**40.08**

The bold values mean the total while the rest of them are components.

and tracking method results in lower state estimation errors and a reduction in false-positive rates. From the experiment observation, this reduction in the state estimation errors and false-positive rates comes from fewer obstacle mismatches and more accurate velocity estimation.

The result illustration of enhancing detection range by the auxiliary learning-based module is visualized in Fig. 8. In Fig. 8(b), we label our depth camera's dense point cloud distance (around 3 m). Since both non-learning detectors, the U-depth and the DBSCAN detectors, require geometric information from either depth image or point cloud, detecting obstacles using the non-learning detectors outside the dense point cloud region can fail. On the contrary, the learning-based module can use the color image to detect obstacles (Fig. 8(a)) even though the obstacle is in a sparse point cloud region. Fig. 8(b) shows that our YOLO-MAD detector can successfully detect the dynamic obstacle (shown as the purple bounding box) in the sparse point cloud region with the increasing detection distance labeled as the yellow line.

B. Runtime Analysis

The runtime of the entire system is detailed in Table II, with measurements conducted on both the Intel NUC and Xavier NX onboard computers. Notably, the total runtime for the Intel NUC and Xavier NX is 19.12 ms and 40.08 ms, respectively, indicating the real-time performance on both platforms. The runtime breakdown results reveal that the YOLO-MAD detector consumes a significant portion of the processing time,

accounting for 75.7% and 59.5% of the total detector runtime on the Intel NUC and Xavier NX, respectively. As discussed in Section III-B, we recommend using the YOLO-MAD detector as an optional and auxiliary module based on the computational resources. Experiments demonstrate that if the user disables the learning-based module, the detection frame rate on the Intel NUC and Xavier NX can increase substantially, reaching around 210 Hz and 60 Hz, respectively, up from 50 Hz and 25 Hz.

C. Physical Experiments

To verify the proposed algorithm's performance in robot navigation, we conduct handheld experiments using the robot camera and do the autonomous navigation tests with the trajectory planner [28], [29] in dynamic environments.

Handheld Experiments: The handheld experiments are conducted by moving the robot's camera in dynamic environments to simulate the navigation trajectories. Fig. 10 shows the example experiments with results. The first example experiment (Fig. 10(a)–(b)) shows persons walking in circles in front of the camera. One can see that our proposed algorithm can detect multiple persons in the camera's FOV and track the history trajectories (shown as green curves) of dynamic obstacles. Note that we only visualized the past 3 seconds' history trajectories. The second example experiment (Fig. 10(d), (e)) lets the camera follow a walking person. The timestamp t_i denotes the time starting from when the first time the dynamic obstacle is detected. The detection results show that our method can allow the robot to perform long-distance detection and tracking of the dynamic obstacle. However, from the experiment observation, we also notice that the occlusion can cause losing track of the obstacles, which is the limitation of the current system. Besides, due to the camera range limitation, the robot can only detect and track the obstacles in the camera's field of view.

Navigation Experiments: We prepare the dynamic environment consisting of both static and dynamic obstacles to test the autonomous robot's navigation ability. The experiment is shown in Fig. 9. Note that the static occupancy voxel map is also used for static obstacle avoidance. In the experiment, the robot is required to navigate to the given goal position, which is

15 meters from the start location. During the navigation period, two persons (only one shown in the figure) are walking randomly as dynamic obstacles, and the robot must avoid them safely. The figure shows that the walking person is successfully detected as a dynamic obstacle, and the robot can efficiently modify its planned trajectory based on the dynamic obstacle's states.

V. CONCLUSION AND FUTURE WORK

This letter presents our lightweight 3D dynamic obstacle detection and tracking (DODT) algorithm for autonomous robots navigating dynamic environments with limited computation. Our method adopts an ensemble detection strategy to obtain refined detection results by combining multiple computationally efficient but low-accuracy detectors. In addition, the proposed feature-based data association and tracking method prevents incorrect matches of obstacles with detected histories. Besides, with the obstacles' state estimations, our dynamic obstacle identification module can classify the detected obstacles into static and dynamic. Finally, we propose using the learning-based method as an optional and auxiliary module to enhance the detection range and dynamic obstacle identification. Our experimental results show that our method has the lowest position error (0.11 m) and a velocity error (0.23 m/s) compared with benchmarking algorithms. In the flight experiments, our method enables the robot to adapt its trajectory efficiently for dynamic collision avoidance. Future improvements can be realized through sensor fusion by harnessing the capabilities of multiple-camera systems. Furthermore, more advanced tracking techniques can be investigated to address tracking losses caused by occlusion.

REFERENCES

[1] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3337.

[2] S. Shi et al., "PV-RCNN: Point-Voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10529–10538.

[3] T. Eppenberger, G. Cesari, M. Dymczyk, R. Siegwart, and R. Dubé, "Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 10528–10535.

[4] Y. Wang, J. Ji, Q. Wang, C. Xu, and F. Gao, "Autonomous flights in dynamic environments with onboard vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 1966–1973.

[5] H. Chen and P. Lu, "Real-time identification and avoidance of simultaneous static and dynamic obstacles on point cloud for UAVs navigation," *Robot. Auton. Syst.*, vol. 154, 2022, Art. no. 104124.

[6] Z. Xu, X. Zhan, B. Chen, Y. Xiu, C. Yang, and K. Shimada, "A real-time dynamic obstacle tracking and mapping system for UAV navigation and collision avoidance with an RGB-D camera," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 10645–10651.

[7] X. Liu et al., "Large-scale autonomous flight with real-time semantic SLAM under dense forest canopy," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 5512–5519, Apr. 2022.

[8] A. Moffatt, E. Platt, B. Mondragon, A. Kwok, D. Uryeu, and S. Bhandari, "Obstacle detection and avoidance system for small UAVs using a LiDAR," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2020, pp. 633–640.

[9] W. Shi, J. Li, Y. Liu, D. Zhu, D. Yang, and X. Zhang, "Dynamic obstacles rejection for 3D map simultaneous updating," *IEEE Access*, vol. 6, pp. 37715–37724, 2018.

[10] D. Yoon, T. Tang, and T. Barfoot, "Mapless online detection of dynamic objects in 3D LiDAR," in *Proc. 6th Conf. Comput. Robot Vis.*, 2019, pp. 113–120.

[11] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Sci. Robot.*, vol. 5, no. 40, 2020, Art. no. eaaz9712.

[12] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3D perception," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 2032–2039, Mar. 2018.

[13] H. Oleynikova, D. Honegger, and M. Pollefeys, "Reactive avoidance using embedded stereo vision for MAV flight," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 50–56.

[14] J. Lin, H. Zhu, and J. Alonso-Mora, "Robust vision-based obstacle avoidance for micro aerial vehicles in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 2682–2688.

[15] A. Saha, B. C. Dhara, S. Umer, K. Yurii, J. M. Alanazi, and A. A. AlZubi, "Efficient obstacle detection and tracking using RGB-D sensor data in dynamic environments for robotic applications," *Sensors*, vol. 22, no. 17, 2022, Art. no. 6537.

[16] M. Lu, H. Chen, and P. Lu, "Perception and avoidance of multiple small fast moving objects for quadrotors with only low-cost RGBD camera," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 11657–11664, Apr. 2022.

[17] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving RGB-D SLAM in dynamic environments: A motion removal approach," *Robot. Auton. Syst.*, vol. 89, pp. 110–122, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015302232

[18] K. B. Logoglu et al., "Feature-based efficient moving object detection for low-altitude aerial platforms," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2017, pp. 2119–2128.

[19] W. Wu, Z. Wang, Z. Li, W. Liu, and F. Li, "PointPWC-Net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3D point clouds," 2019, *arXiv:1911.12408*.

[20] S. L. Francis, S. G. Anavatti, and M. Garratt, "Detection of obstacles in the path planning module using differential scene flow technique," in *Proc. Int. Conf. Adv. Mechatron., Intell. Manufacture, Ind. Automat.*, 2015, pp. 53–57.

[21] C. Yu et al., "DS-SLAM: A semantic visual SLAM towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1168–1174.

[22] W. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, "RGB-D SLAM in dynamic environments using point correlations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 1, pp. 373–389, Jan. 2020.

[23] I. Ballester, A. Fontán, J. Civera, K. H. Strobl, and R. Triebel, "Dot: Dynamic object tracking for visual SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 11705–11711.

[24] Y. Qiu, C. Wang, W. Wang, M. Henein, and S. Scherer, "AirDOS: Dynamic SLAM benefits from articulated objects," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 8047–8053.

[25] F. Kong, W. Xu, Y. Cai, and F. Zhang, "Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 7869–7876, Apr. 2021.

[26] Y. Min, D.-U. Kim, and H.-L. Choi, "Kernel-based 3-D dynamic occupancy mapping with particle tracking," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 5268–5274.

[27] G. Chen, W. Dong, P. Peng, J. Alonso-Mora, and X. Zhu, "Continuous occupancy mapping in dynamic environments using particles," *IEEE Trans. Robot.*, 2023.

[28] Z. Xu, Y. Xiu, X. Zhan, B. Chen, and K. Shimada, "Vision-aided UAV navigation and dynamic obstacle avoidance using gradient-based B-spline trajectory optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 1214–1220.

[29] Z. Xu, D. Deng, Y. Dong, and K. Shimada, "DPMPC-Planner: A real-time UAV trajectory planning framework for complex static environments with dynamic obstacles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 250–256.