arXiv:2303.00132v1 [cs.RO] 28 Feb 2023

Onboard dynamic-object detection and tracking for autonomous robot navigation with RGB-D camera

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

***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 LiDAR and their high computation cost for learning-based data processing make those methods not applicable to lightweight 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. 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 method to prevent mismatches and use the Kalman filter with the constant acceleration model to track detected obstacles. In addition, our system includes an optional and auxiliary learning-based module to enhance the obstacle detection range and dynamic obstacle identification. The users can determine whether or not to run this module based on the available computation resources. The proposed method is implemented in a lightweight quadcopter, and the experiments prove that the algorithm can make the robot detect dynamic obstacles and navigate dynamic environments safely.**

I. Introduction

Lightweight autonomous robots are widely used in various indoor applications. The environments of those applications usually involve humans, vehicles, and other robots, which can be highly dynamic and unpredictable. Under such circumstances, the robots must perceive dynamic obstacles accurately in real time for safe navigation. However, many indoor robots, such as lightweight UAVs, are only equipped with a computation-limited onboard computer and an RGB-D camera, making the GPU-demanding deep-learning-based methods from the autonomous driving field unsuitable. As a result, developing an onboard dynamic obstacle detection and tracking method based on an RGB-D camera becomes crucial for autonomous robot applications in dynamic environments.

There are mainly three challenges in onboard 3D dynamic obstacle detection using an RGB-D camera. First, the onboard computation resources are limited in lightweight indoor robots, making GPU-demanding learning-based methods [1][2] not applicable. Second, most lightweight depth cameras' range and field of view (FOV) are very narrow,

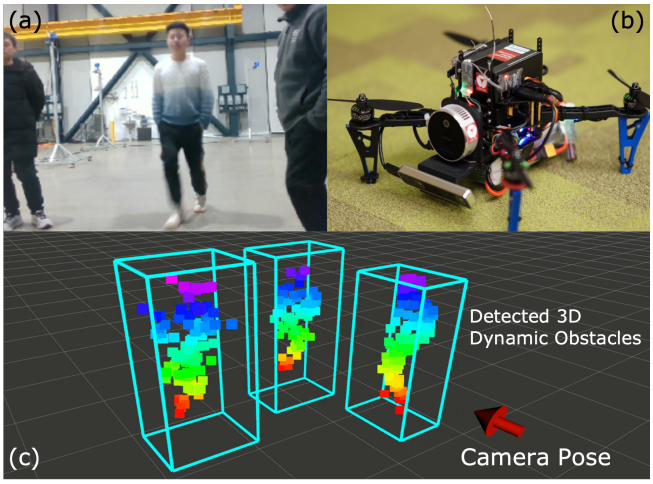

Fig. 1. The onboard dynamic obstacle detection results from the proposed DODT algorithm. (a) The camera RGB view. (b) An example of an autonomous robot with an RGB-D camera. (c) The onboard 3D dynamic obstacle detection results shown as blue bounding boxes with point clouds.

which makes obstacles either too close or too far not detectable. For example, the ideal depth range of the popular Intel RealSense D435i depth camera is from 0.3m to 3.0m. 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 paper presents an onboard 3D dynamic obstacle detection and tracking (DODT) method based on an RGB-D camera. We propose a novel ensemble detection strategy combining multiple computationally efficient but low-accuracy detectors to obtain fast, accurate obstacle detection results. Besides, the proposed method adopts our feature-based data association and uses the constant-acceleration Kalman filter to track each obstacle. Then, we use both point cloud and velocity criteria to identify dynamic obstacles. Finally, the system includes an optional and auxiliary learning-based module to enhance the detection range and dynamic obstacle identification when the robot's computation resources are enough. Fig. 1 shows an example of detection results. The main contributions of this work are:

- **Efficient Ensemble Detection:** Our algorithm runs

*The authors contributed equally.

Zhefan Xu, Xiaoyang Zhan, Yumeng Xiu, Christopher Suzuki, and Kenji Shimada are with the Department of Mechanical Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213, USA. zhefanx@andrew.cmu.edu

multiple computationally efficient and low-accuracy detectors in parallel and adopts an ensemble detection strategy to obtain accurate results with high efficiency.
- **Feature-based Association and Tracking:** Our feature-based data association method prevents tracking mismatches and use the constant-acceleration Kalman filter for better obstacle state estimation.
- **Auxiliary Learning-based Detection Module:** Our optional learning-based module is applied to improve the detection range and dynamic obstacle identification.

II. Related Work

Obstacle detection and tracking algorithms are designed based on robot sensors and data representations. The choice of sensors can vary from LiDARs [7][8][9][10], event cameras [11][12], and RGB-D cameras [3][4][6] based on the robot platforms. Among them, the RGB-D camera is one of the most popular sensors for indoor robots, and there are mainly two conventional ways of using the RGB-D camera:

Image-based methods: Most of this category of methods utilize the depth image for 3D obstacle detection. In [13], they use the depth image to generate the U-depth map and V-depth map to estimate the states of obstacles and demonstrate the safe navigation ability with static obstacles. Later, Lin et al. [14] adopt a similar U-depth map to detect and track obstacles and represent them as 3D ellipsoids. To further improve the obstacle dimension estimation accuracy, the restricted V-depth map is applied in [15]. In [6], the dynamic obstacles detected from the depth and U-depth map are identified by the estimated velocities. The dynamic obstacle detection results are combined with the occupancy map for navigating dynamic environments. Unlike previous depth image-based methods, Lu et al. [16] apply the YOLO detector to avoid fast and small dynamic obstacles. Sun et al. [17] apply image difference to detect all dynamic points from RGB images. Logoglu et al. [18] combine the 3-image-difference and epipolar constraints to determine dynamic obstacles. Scene flow, an extension of the optical flow, is applied in [19] [20] to detect the velocity of each pixel and identify dynamic points. Some other methods detect and segment dynamic obstacles in the 2D image planes for improving SLAM robustness. In [21][22][23][24], their approaches focus on removing dynamic obstacles in the image to reduce the state estimation errors. Qiu et al. [25] detect the pedestrian skeletons to improve the SLAM optimization.

Point cloud-based methods: Unlike the image-based methods, the point cloud-based methods directly detect 3D obstacles using the point cloud geometry information. In [3], the point cloud clustering method is combined with the YOLO detector for human detection. Following a similar clustering detection idea, Wang et al. [4] show indoor dynamic obstacle avoidance using a quadcopter. To improve the obstacle tracking robustness, Chen et al. [5] propose to use the point cloud feature vectors and object track points to find correct object matches and estimate their states. In [26], a KD-Tree map is built directly from the LiDAR point cloud for dynamic obstacle avoidance. Min et al. [27] represent dynamic obstacles in the dynamic occupancy map and leverage kernel inference to reduce computation. Similarly, in [28], a dual-structure particle-based dynamic occupancy map is used to represent dynamic environments and classify obstacle particles into static and dynamic.

However, both image-based methods and point cloud-based methods can inevitably have misdetection due to noises from the images, point clouds, and complicated environment structures. Realizing that different detectors have different error sources for misdetection, we propose a novel ensemble method to overcome the shortcomings of each simple detector. To balance computation and performance, we also suggest using the learning-based method as an optional and auxiliary module instead of a required end-to-end obstacle detection and tracking tool, making our method applicable to robots with a wide range of computational resources.

III. Methodology

A. System Overview

There are mainly three modules in the proposed system framework: the detection module, the tracking module, and the identification module, as shown in Fig. 2. The detection module is divided into a main non-learning module and an auxiliary learning-based module. The non-learning part takes the depth image and ensembles two non-learning detectors to detect generic obstacles, while the learning-based module uses the aligned RGB-D image to directly detect dynamic obstacles and ensemble the results with the non-learning detection module. The details of each detector will be discussed in Sec. III-B with the ensemble detection described in Sec. III-C. Then, the refined 3D bounding boxes of obstacles will be used in the tracking module (Sec. III-D) to find the correct history matches and use the histories to estimate the obstacle states. With the obstacle states and tracking histories, the identification module (Sec. III-E) is applied to classify obstacles as static and dynamic. Finally, the system outputs dynamic obstacles' bounding boxes with their point clouds, and the dynamic obstacle regions are cleaned 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.

U-depth Detector: The U-depth detector for obstacle detection is mentioned in the previous works [13][14][6]. Overall, the detector takes the depth image to generate 3D bounding boxes of static and dynamic obstacles' bounding boxes. 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

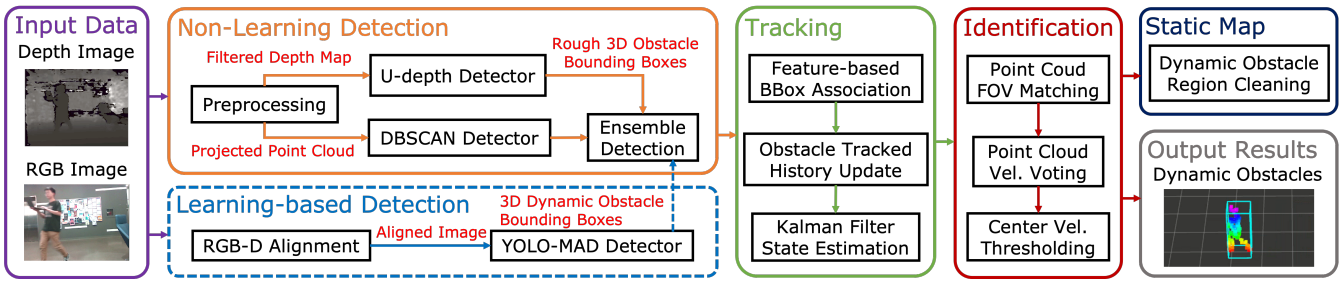

Fig. 2. The proposed dynamic obstacle detection and tracking system (DODT) framework. The input data are the RGB-D images. The non-learning detection module first uses the depth image to detect generic obstacles. Then, the tracking module is applied to track and estimate the obstacles states. With the 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.

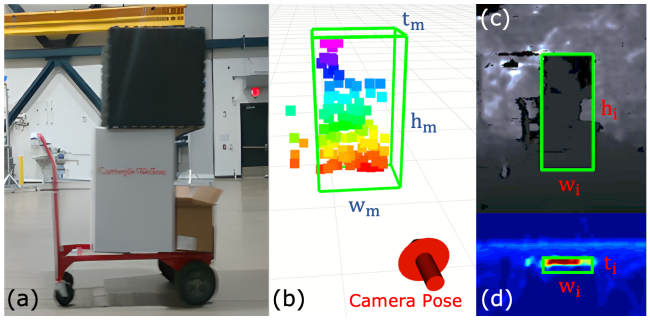

Fig. 3. Illustration of the U-depth detector. (a) The camera RGB view. (b) The detected 3D bounding box with the obstacle point cloud. (c) The 2D detection on the depth map. (d) The 2D detection on the U-depth map.

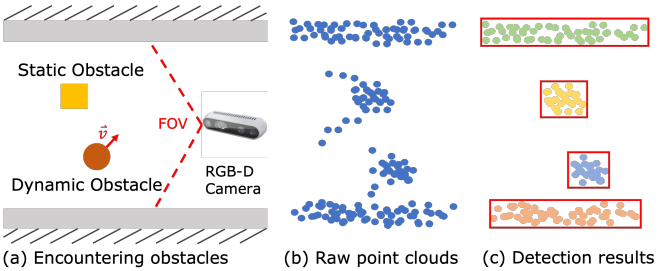

Fig. 4. Illustration of the DBSCAN detector. (a) The robot encounters obstacles in a corridor. (b) The raw point cloud data from the RGB-D camera are unstructured and noisy. (c) The DBSCAN detector takes the filtered point cloud and performs clustering to get obstacles' bounding boxes.

a depth image, we can compute the U-depth map using the column depth value histogram. Fig. 3c and Fig. 3d 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. 3d (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. 3c. 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. 3b).

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. 4b. 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. 4c). Similar to the U-depth detector, the DBSCAN detector does not need a training dataset and only requires a few computation resources.

YOLO-MAD Detector: The previously mentioned detectors rely on geometric structures of either depth images or point clouds. As a result, 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[1], 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 $\mathcal{R}_{\text{box}}$:

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

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 $\mathcal{S}_{\text{MAD}}$:

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

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

[1]https://github.com/dog-qiuqiu/FastestDet

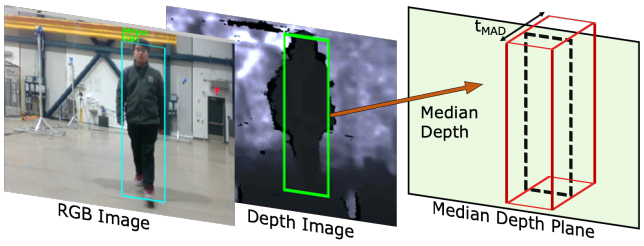

Fig. 5. Illustration of the YOLO-MAD detector. The RGB image is used to get the 2D detection result, and then the bounding box on the depth image is obtained. With the 2D result on the depth image, the 3D bounding box is calculated by the proposed median absolute deviation (MAD) method.

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

The proposed ensemble detection algorithm follows a pairwise manner presented in Alg. 1. When we obtain two sources of detection results, we go through each bounding box $\mathrm{b_{d1}}$ from one detector's results (Line 4). For the bounding box $\mathrm{b_{d1}}$, the algorithm finds the bounding box $\mathrm{b_{match1}}$ with the highest intersection-over-union (IOU) score from the other detection bounding boxes (Line 5). Following the same way, the bounding box $\mathrm{b_{match2}}$ is obtained by finding the highest IOU match of $\mathrm{b_{match1}}$ 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 9-10). 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

This section describes the data association and tracking module for matching obstacles temporally and estimating

Algorithm 1: Ensemble Detection Algorithm

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

their states. 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.

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:

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

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:

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

where we take the exponential of the negative L2 norm of the feature difference. With the similarity 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 $sim_{\max}$. Note that 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. In addition, we also need to ensure that the highest similarity score is 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. 6a and b, a scenario is presented where a person approaches the wall with the point clouds of all obstacles shown in Fig. 6c. 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

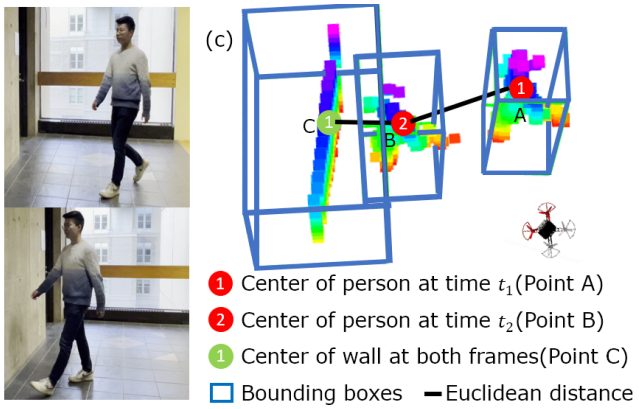

Fig. 6. Illustration of the issue with the center-distance-based data association method. (a) The RGB image at time t_1. (b) The RGB image at time t_2. (c) The center-distance-based data association method might fail by incorrectly associating the current detected person with the wall.

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:

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

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:

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

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:

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

and the system measurement is defined as:

$$Z_t = HX_t + R, \tag{8}$$

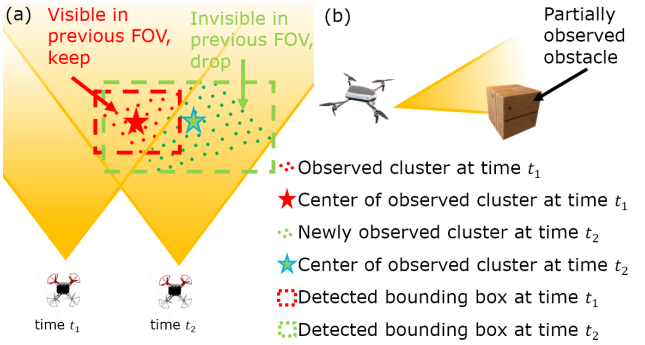

Fig. 7. Illustration of removing the invalid points using the field of view (FOV) criteria. (a) The analysis of the observed obstacle's point cloud at different time. (b) The robot detecting a partially visible obstacle.

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. In the first step, all the bounding boxes of obstacles with the center velocity $\mathbf{V}_{center}$ less than a threshold T_{vel} will be classified as static. After that, 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 $\mathbf{V}^i_{vote}$ 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:

$$\frac{N_{vote}}{N_{valid}} > T_{ratio}. \tag{9}$$

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 $\mathbf{V}^i_{vote}$, it will be removed from the dynamic voting process. The valid velocity should satisfy the condition:

$$angle(\mathbf{V}^i_{vote}, \mathbf{V}^j_{center}) < \frac{\pi}{2}, \tag{10}$$

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. 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,

Fig. 8. Illustration of using the OptiTrack motion capture system to estimate the error of the proposed dynamic obstacle detection and tracking method.

the classification results will be used for dynamic obstacle identification, skipping all the processes mentioned above.

IV. Result and Discussion

A. Implementation Details

To evaluate the performance of the proposed method, we conduct experiments in dynamic environments. The algorithm is implemented based on C++ and ROS, running on two customized autonomous quadcopters with the Intel NUC and NVIDIA Jetson Xaiver NX onboard computer, respectively. We use the RGB-D images from Intel RealSense D435i camera as the system inputs, which provides 640 × 480 pixels images with 87° by 58° field of view. The visual-inertial odometry (VIO) algorithm [29] is applied for the robot state estimation and to transform the detection results from the camera to the world frame. The autonomous quadcopters use the PX4-based flight controller for our flight tests. All the computations, including dynamic obstacle detection and tracking, mapping, planning and state estimation, are performed real-time on the robots' onboard computers.

B. Performance Benchmarking

To quantitatively analyze the proposed algorithm's performance, we conduct comparison experiments with the state-of-the-art dynamic obstacle detection and tracking methods in the UAV platform [14][4][6]. The comparison results of the average position and velocity errors are shown in Table I. The experiment's position and velocity errors are measured by comparing the ground truth measurement from the OptiTrack motion capture system shown in Fig. 8. Table I shows that our DODT method has the lowest position errors among all the methods, and our velocity error is the second least, comparable to Method III [6]. From our observations, we notice that the image-based method [14] is very sensitive to background noises, leading to many false positive defections in the backgrounds. Similarly, the point cloud-based method [4] tends to generate false-positive detection around the obstacles' edges, the places with the most point cloud noises. Although methods [4][6] apply the dynamic obstacle identification technique to filter the false-positive detection, their identification requires a long time horizon which causes latency in the dynamic obstacle detection. Since our algorithm applies the ensemble detection with the learning-based module, it can reduce the false-positive detection and improve the dynamic obstacle identification

TABLE I

Benchmarking of Detection and Tracking Errors.

Method	Position Error (m)	Velocity Error (m/s)
Method I [14]	0.28	0.47
Method II [4]	0.18	0.29
Method III [6]	0.19	**0.21**
DODT (Ours)	**0.11**	0.23

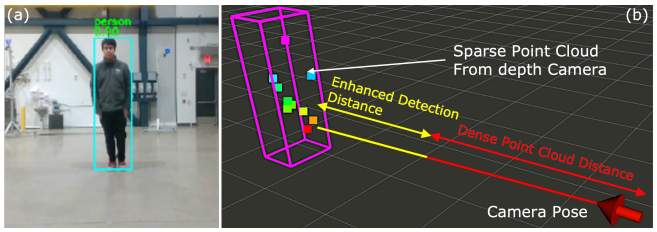

Fig. 9. Illustration of enhancing detection range by the auxiliary learning-based module. The red line measures the maximum ideal range to produce dense point cloud data for the DBSCAN and U-depth detectors to detect obstacles. The yellow line indicates the increased detection distance.

speed by using the "mutual agreements" (IOU criteria) and the classification from the learning-based detector.

The result illustration of enhancing detection range by the auxiliary learning-based module is visualized in Fig. 9. In Fig. 9b, we label our depth camera's dense point cloud distance (around 3m). 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. 9a) even though the obstacle is in a sparse point cloud region. Fig. 9b 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.

C. Runtime Analysis

The runtime of the entire system is shown in Table II. Note that we use the Intel NUC onboard to measure the runtime, and the 3D obstacle detection runtime includes the auxiliary learning-based module. Overall, one can see that the total runtime is 19.12ms which indicates our algorithm can run over 50Hz. The 3D obstacle detection spends the most time among all the modules in our framework. From the detector runtime of the Intel NUC (shown as the blue bar) in Fig. 10, we can see that the YOLO-MAD detector takes 14.3ms in an iteration which is 75.7% of the total detector runtime. Similarly, for the NVIDIA Xavier NX onboard computer, the YOLO-MAD takes 59.5% of the entire detector runtime. As we suggest in Sec. III-B, the YOLO-MAD detector should be used as an optional and auxiliary module when the computational resources are enough for real-time applications. The experiments show that if the user decides not to use the learning-based module, the detection frame rate on Intel NUC and Xaiver NX can increase from around 50Hz and 25Hz to around 210Hz and 60Hz, respectively.

TABLE II

THE RUNTIME OF EACH MODULE OF THE PROPOSED SYSTEM.

System Modules	Time (ms)	Portion (%)
3D Obstacle Detection	18.90	98.85%
Data Association and Tracking	0.10	0.52%
Dynamic Obstacle Identification	0.12	0.63%
System Total Runtime	**19.12**	**100.00%**

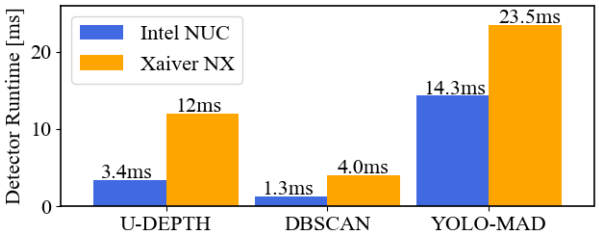

Fig. 10. The runtime comparison of the U-depth, the DBSCAN, and the YOLO-MAD detector on the Intel NUC and the NVIDIA Xavier NX.

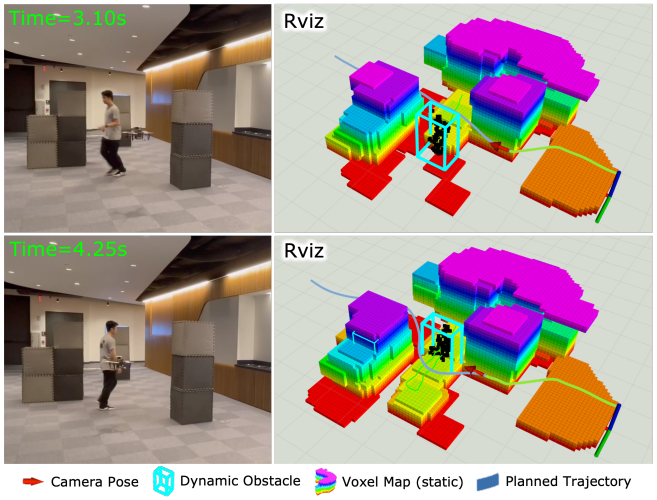

Fig. 11. Autonomous robot navigation in dynamic environments using the proposed algorithm. The onboard obstacle detection results (blue bounding boxes) can help the robot modify its planned path to avoid obstacles safely.

D. 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 [30][31] 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. 12 shows the example experiments with results. The first example experiment (Fig. 12a-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. 12d-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.

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. 11. 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 paper presents our onboard 3D dynamic obstacle detection and tracking (DODT) algorithm for autonomous robots navigating dynamic environments. Our method adopts an ensemble detection strategy to obtain refined detection results by combing multiple computationally efficient but low-accuracy detectors. In addition, the proposed feature-based data association method prevents incorrect matches of obstacles with detected histories. The constant-acceleration Kalman filter is used to estimate the states of obstacles. 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 handheld and autonomous flight experiments in dynamic environments prove that our system can help robots detect dynamic obstacles to navigate dynamic environments. From our experiment observations, the performance of our current algorithm is mainly bottlenecked by the sensor's field of view (FOV). So, future improvements can be made by using the multiple-camera system sensor fusion.

VI. ACKNOWLEDGEMENT

The authors would like to thank TOPRISE CO., LTD and Obayashi Corporation for their financial support in this work.

REFERENCES

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

[2] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 529–10 538.

[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 *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 528–10 535.

[4] 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)*. IEEE, 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," *Robotics and Autonomous Systems*, vol. 154, p. 104124, 2022.

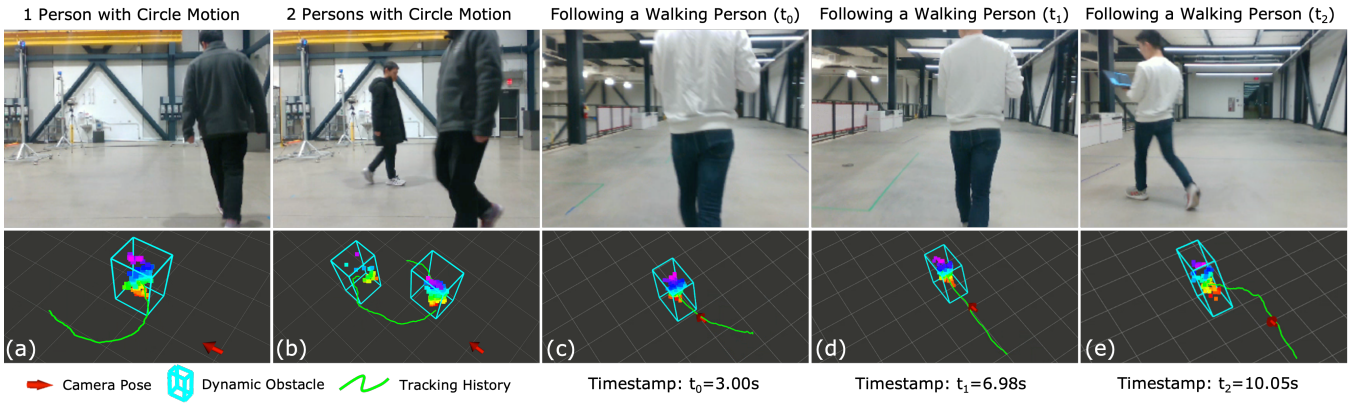

Fig. 12. The dynamic obstacle detection and tracking experiments with a handheld robot camera. The blue bounding boxes containing point clouds visualize the dynamic obstacles' detection results, and the tracking histories are shown as green curves. Figures (a) and (b) show the detection results of persons walking in circles. Figures (c), (d), and (e) show the long-distance dynamic obstacle detection and tracking ability following a walking person.

[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," *arXiv preprint arXiv:2209.08258*, 2022.

[7] X. Liu, G. V. Nardari, F. C. Ojeda, Y. Tao, A. Zhou, T. Donnelly, C. Qu, S. W. Chen, R. A. Romero, C. J. Taylor *et al.*, "Large-scale autonomous flight with real-time semantic slam under dense forest canopy," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5512–5519, 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 *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 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. 37 715–37 724, 2018.

[10] D. Yoon, T. Tang, and T. Barfoot, "Mapless online detection of dynamic objects in 3d lidar," 05 2019, pp. 113–120.

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

[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 Robotics and Automation Letters*, vol. 3, no. 3, pp. 2032–2039, 2018.

[13] 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.

[14] 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.

[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, p. 6537, 2022.

[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 Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 657–11 664, 2022.

[17] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving rgb-d slam in dynamic environments: A motion removal approach," *Robotics and Autonomous Systems*, vol. 89, pp. 110–122, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015302232

[18] K. B. Logoglu, H. Lezki, M. K. Yucel, A. Ozturk, A. Kucukkomurler, B. Karagoz, A. Erdem, and E. Erdem, "Feature-based efficient moving object detection for low-altitude aerial platforms," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 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," 11 2019.

[20] 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)*, 2015, pp. 53–57.

[21] Y. Sun, M. Liu, and M. Q.-H. Meng, "Improving rgb-d slam in dynamic environments: A motion removal approach," *Robotics and Autonomous Systems*, vol. 89, pp. 110–122, 2017.

[22] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1168–1174.

[23] W. Dai, Y. Zhang, P. Li, Z. Fang, and S. Scherer, "Rgb-d slam in dynamic environments using point correlations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 373–389, 2020.

[24] I. Ballester, A. Fontán, J. Civera, K. H. Strobl, and R. Triebel, "Dot: Dynamic object tracking for visual slam," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 705–11 711.

[25] Y. Qiu, C. Wang, W. Wang, M. Henein, and S. Scherer, "Airdos: dynamic slam benefits from articulated objects," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8047–8053.

[26] 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.

[27] Y. Min, D.-U. Kim, and H.-L. Choi, "Kernel-based 3-d dynamic occupancy mapping with particle tracking," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5268–5274.

[28] G. Chen, W. Dong, P. Peng, J. Alonso-Mora, and X. Zhu, "Continuous occupancy mapping in dynamic environments using particles," *arXiv preprint arXiv:2202.06273*, 2022.

[29] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[30] 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," *arXiv preprint arXiv:2209.07003*, 2022.

[31] 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 *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 250–256.