Received April 6, 2018, accepted May 9, 2018, date of publication May 14, 2018, date of current version July 25, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2836192

Dynamic Obstacles Rejection for 3D Map Simultaneous Updating

WENJUN SHI[1,2,3], (Student Member, IEEE), JIAMAO LI[1,3], (Member, IEEE), YANQING LIU[1,2,3], (Student Member, IEEE), DONGCHEN ZHU[1,2], (Student Member, IEEE), DONGDONG YANG[3], AND XIAOLIN ZHANG[1,3]

[1]Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China
[2]University of Chinese Academy of Sciences, Beijing 100049, China
[3]Shanghai Eyevolution Technology Company Ltd., Shanghai 200050, China

Corresponding author: Jiamao Li (jmli@mail.sim.ac.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61671014, in part by the Shanghai Committee of Science and Technology, China, under Grant 16JC1420503, and in part by the National Natural Science Foundation of China under Grant 61601448.

ABSTRACT We present a simple yet highly efficient method to eliminate spurious trails of dynamic objects for 3-D point cloud map updating. First, we extract the view overlaps based on view frustum filter. Then, we obtain spurious trails via bidirectional searching of view overlaps using a KD tree. Finally, in terms of the situation where moving objects occlude part of background due to the limits of the RGB-D camera, we design a ray tracing principle-based filter to supplement the missing background in the whole point cloud map. Our method can be integrated into any SLAM or 3-D reconstruction systems with RGB-D data input, and it is suitable for both static and dynamic environments. We validate our approach in real-world scenes of our laboratory office using a Kinect system for robot obstacles avoidance and navigation. Moreover, experiments on the KITTI odometry benchmark illustrate that the proposed approach is highly efficient for dynamic spurious trails rejection and 3-D map updating.

INDEX TERMS View frustum culling, 3D map updating, dynamic, spurious trails, ray tracing, obstacles avoidance, navigation.

I. INTRODUCTION

The success of Simultaneous Localization and Mapping (SLAM) has significantly improved the autonomous robot navigation techniques. In many applications, robots need to perceive the world through mapping 3D structure of the environment. However, most 3D mapping approaches are based on the assumption of static environments, while in many applications the environments where the robots carry out their tasks are usually dynamic. The ability that robots detect the dynamic objects automatically and update the 3D map simultaneously like the human is crucial for navigation.

For this reason, many approaches attempt to refine 3D reconstruction process in dynamic scenes, while most of them mainly focus on the accuracy of camera poses. The main consideration is to extract moving objects from dynamic scenes and eliminate them in the process of reconstruction. These approaches can be categorized into two classes. Some methods directly regard the moving objects as outliers and remove them. The most popular technique is to implement RANSAC regression to filter out dynamic objects [1], [2]. These methods are robust to noise and large scale complex scenes. Meanwhile, correct transformations can be obtained in real-time. However, the RANSAC may not work when the environment contains more dynamic points than the static background points. Furthermore, the dynamic objects would not be rejected clearly because these methods are usually based on feature points. In other words, the side-effect is that 3D maps still contain spurious trails when there exist dynamic objects in the scene.

Other methods mainly utilize inter-frame information to compute extra motion information or estimate background model, which is further implemented to discriminate moving object and static background. Some of them try to extract moving objects in the 2D images [3]–[6], while some others are based on 3D point cloud [7]–[9]. The work of [4] combined dense moving object segmentation with dense SLAM, where it proposed to utilize dense optical flow to improve the dense segmentation of dynamic objects. The dynamic points can be further removed from the energy function. Some latest works [5], [6] proposed to use the boundaries of dynamic

2169-3536 © 2018 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

objects to help segment the dense dynamic points and reduce the influence of dynamic objects by motion removal. Reference [3] considered to extract moving objects based on the estimation of the background model, which is represented by the nonparametric model from depth scenes. However, due to its dependence on background modeling algorithm, it is difficult and time-consuming to estimate a robust background model from complex scenes in real world. Other works focused on detection of dynamic objects in 3D space, inspired by the scene flow and 3D vector field, for example [7], [9]. Reference [7] extended a motion segmentation method based on sparse subspace clustering to distinguish between static scene parts and multiple moving objects. Reference [9] proposed to extract moving objects based on 3D normal vectors which declared good performance. But we highly suspect the real-time performance because it covers excessive computations. These methods are more robust in dynamic environments, but both of them rely on the computations of some extra information which hinder their real-time applicability. Thus they are not suitable for obstacle avoidance and navigation.

In this paper, we propose a simple yet highly efficient 3D map updating method which is built upon the well established algorithms for camera pose estimation. Unlike [10] and [11] which updated the 3D map through long-term static scene reconstruction, our method ensures a correct representation of any object whether static or dynamic from the robots' viewer at present moment. Besides, [10], [11] need statistical probability of reconstruction maps of the dynamic scene at different moments. For instance, they may need to scan the environment in the morning and evening respectively, which is hard to achieve remarkable performance in robots navigation.

More recently, [12] shares the same goal of proposed method, but experiments show that the method in [12] is only suitable for rotating LIDAR system. We further notice that the real-time performance and reconstruction accuracy of this method strongly depend on the resolution of OctoMap [13]. And no experiments in [12] support a common case where the camera and object move together. In contrast, our method is suitable for all RGB-D data including RGB-D images from depth camera and stereo camera. Moreover, our method is directly based on 3D point cloud map in any resolution and promise the real-time performance in large even complicated scenes. The key contributions of our work are as follows:

- We design a real-time method based on view frustum culling filter to discriminate between spurious trails and objects.
- In terms of the situation where moving objects occlude some background due to the limits of the RGB-D camera, we design a ray tracing principle based filter to supplement the missing background in the whole 3D point cloud map.
- We provide a generalized 3D map updating method which can be integrated into any SLAM or reconstruction systems easily and validate it is suitable for both static and dynamic environments in robot navigation and 3D reconstruction.

The paper is organized as follows. In Section II we formulate the problem, introducing some notations too. Section III describes our method based on view frustum culling filter and its optimization through the ray tracing. In Section IV we present the experiments we have done in different environments and discuss the results. Finally, in Section V we conclude the paper and give our future research direction.

II. PROBLEM STATEMENT

Frame-to-frame accumulation is the most basic and widely used method in 3D point cloud mapping. At the moment $t-1$, we obtain a whole point cloud map of all old frames $F(1) \sim F(t-1)$ as $M(t-1)$. At the moment t, we get another point cloud frame of the scene as $F(t)$. After the estimation of the transformation $T(t)$ from $F(t)$ to the whole map $M(t-1)$, we can use this formula

$$M(t) = M(t-1) + T(t) * F(t) \tag{1}$$

to get the current whole point cloud map $M(t)$. Step by step like this, we can reconstruct the whole scene where the environment keeps static. However, in real world, most environments are usually dynamic. From the view of mobile robots, the simplest case is that the sensor is static while there exist several moving objects in the scene. But in real applications a more common case is that the sensor and objects are generally all on the move, where the velocities and moving directions of them are usually different as shown in Fig. 1(a). According to equation (1), if the scene contains dynamic objects whether the sensor is static or not, there will exist some spurious trails of the dynamic objects which we don't need in the whole map $M(t)$ as Fig. 1(b) displayed. Our task is to detect and remove these trails also called "ghosts" from the map. We use a simplified model to simulate this problem as shown in Fig. 2, including four subfigures (a)∼(d). Although in practical applications the cameras have both translations and rotations, here for convenience of illustration, we assume that the camera only has a translational motion in Fig. 2. In fact, the principle of implementation is the same. Subfigure (a) in Fig. 2 shows the 2D-plane model of $M(t-1)$, where green parts represent the moving object at time $t-1$. In contrast, subfigure (b) shows the 2D-plane model of $F(t)$, where green parts represent the moving object at time t. And subfigure (c) illustrates mapping result just from equation (1), while (d) is the result we expect. Through comparisons between the points of (c) and (d), it can be easily concluded that our goal is to extract and remove the green trails under the blue backgrounds in $M(t-1)$. It is observed that the blue backgrounds in subfigure (a) and (b) correspond to the view overlap between $M(t-1)$ and $F(t)$, which are defined as M_{over} and F_{over} respectively in counterparts. Furthermore, points in blue backgrounds can be classified into three classes. Those which belong to both F_{over} and M_{over} are static background points, while those which belong to F_{over} but don't belong to M_{over} are

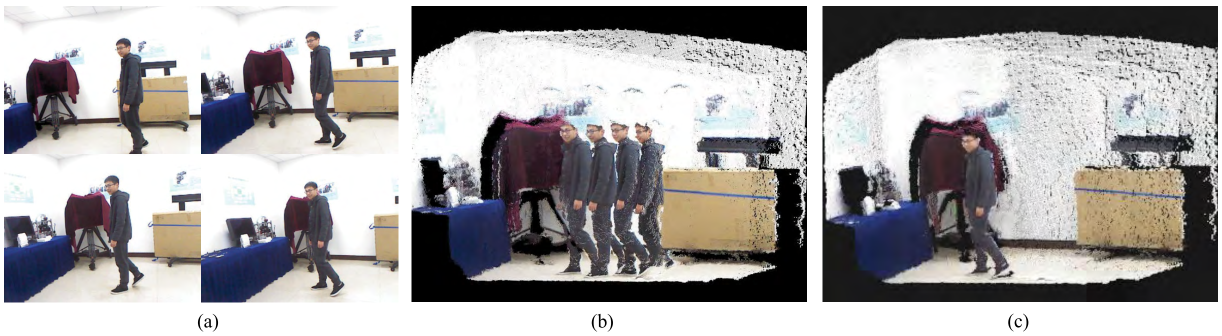


FIGURE 1. Schematic view of the existing problem for traditional frame-to-frame accumulation under dynamic scene: (a) shows four RGB frames captured from a moving Kinect at time $t-3, t-2, t-1, t$ under the scene containing a moving person; (b) is the reconstructed 3D map of the scene, using GICP [14] to get the camera pose of each frame and mapping the scene by equation(1). (c) is the scene's 3D map without spurious trails, adding our method before equation(1).

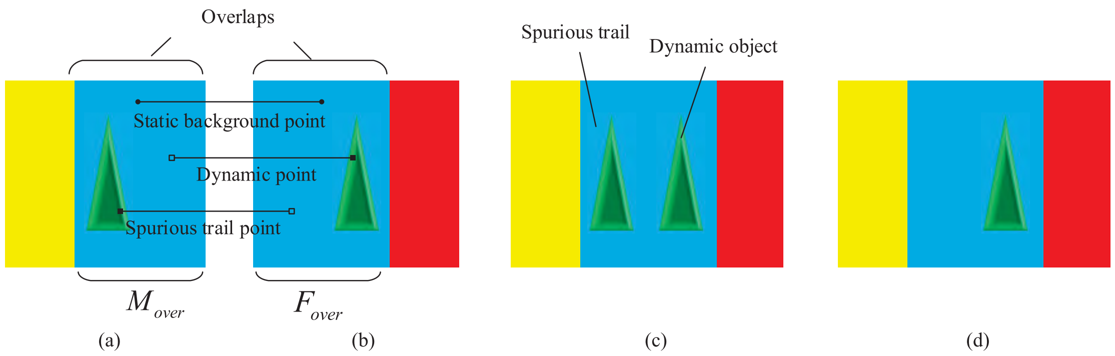


FIGURE 2. Diagrammatic sketch simulates the registration of the $M(t-1)$ and $F(t)$. (a) represents the $M(t-1)$ with a historical spurious trail; (b) represents $F(t)$ with a real dynamic object at current moment; (c) simulates the 3D map result reconstructed of (a) and (b) which is an example of result shown in Fig. 1(b); (d) is the result that we expect.

real moving object points. And points we want to remove are those which belong to M_{over} but don't belong to F_{over}, i.e., spurious trails.

Considering the problem is oriented towards the robot applications of autonomous obstacles avoidance and navigation, we hope to keep the real-time performance in 3D mapping. Thus we discard plans where extra information such as optical flow or edges is used to extract dynamic objects. Instead, we determine to achieve the extraction and removal of spurious trails directly in non-order point cloud data without computing any other extra information.

III. PROPOSED METHOD

Based on analyses in Section II, we propose a simple yet effective map updating algorithm for both static and dynamic environments. The basic processes are as follows. Firstly, we implement a novel filtering technique to extract view overlap points mentioned before. Then a bidirectional search algorithm is utilized to extract spurious trails of moving objects. Finally, with regard to the possible background deficiency problem in some situations, we leverage the ray tracing principle to supplement the background, which leads to more reliable results. Details are presented in this section.

A. EXTRACT VIEW OVERLAPS BY FRUSTUM CULLING

Considering the imagery features of the RGB-D camera and the characteristics of the point cloud structure, we apply a view frustum culling filter to get the view overlap between historical 3D point cloud map model $M(t-1)$ and the current point cloud frame $F(t)$. To our best knowledge, we firstly introduce the view frustum culling filter to point cloud view overlap extraction. We implement it in three steps:

STEP 1

We utilize existing robust SLAM or 3D reconstruction systems like RTABMAP [15] and StereoScan [16] to estimate the accurate transformation $T(t)$ for each frame between the camera coordinate system and the world coordinate system. Because we assume that the camera coordinate system of the first frame coincides with the world coordinate system, the coordinate system of 3D map $M(t-1)$ coincides with the world coordinate system. And then we can obtain the camera pose $P_t(x, y, z, roll, pitch, yaw)$ of $F(t)$ in the coordinate system of $M(t-1)$ as well as the world coordinate system through the transformation.

STEP 2

We compute the horizontal and vertical fields of view for the camera in degrees H_{fov}, V_{fov} according to the camera internal parameters f_x, f_y and the size of images $width, height$ as follows:

$$H_{fov} = 2\arctan(\frac{width}{2f_x}) \quad (2)$$

$$V_{fov} = 2\arctan(\frac{height}{2f_y}) \quad (3)$$

Moreover, we set the near plane $Z_{near} = 0.8$ and the far plane $Z_{far} = 4.0$ considering the imagery features of the depth camera in Kinect [17]. Then we set the $P_t(x, y, z, roll, pitch, yaw)$ obtained from $Step1$ as the camera pose at time t and use H_{fov}, V_{fov} to estimate a frustum-shaped [18] field of view in the 3D space for this camera pose, as shown in Fig. 3. We construct the view frustum using six intersecting infinite planes and figure out the plane equation for each one of those six planes $Up, Down, Left, Right, Near, Far$. Detailed implementations can be found in [19] and [20].

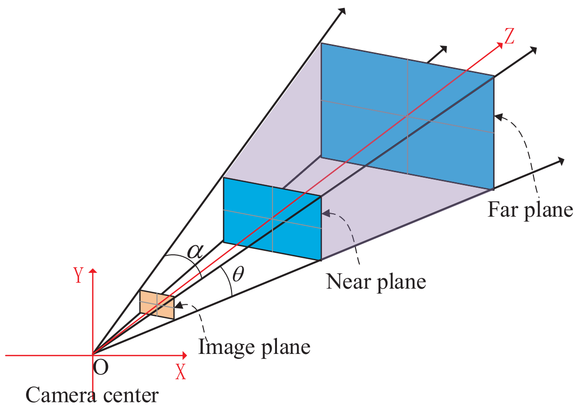


FIGURE 3. The geometry diagrammatic sketch of the field of view for the camera solid. α and θ are the horizontal and vertical field of view for the camera; the pink frustum field between the near plane and the far plane is the view frustum.

STEP 3

We put coordinate of each point of $M(t-1)$ and $T(t) * F(t)$ into the six plane equations from $Step2$ and judge whether the point should be in view frustum or not. Then the view overlaps M_{over} and F_{over} can be obtained as shown in Fig. 4 (c) and (d).

B. EXTRACT AND REMOVE HISTORICAL SPURIOUS TRAILS

In this subsection, we mainly aim to extract historical spurious trails of moving objects in the view overlap $M(t-1)$. After carrying out the procedures of Subsection A we obtain both M_{over} and F_{over}. Recall the definitions in Section II, if the point in M_{over} doesn't exist in F_{over}, it belongs to spurious trails. Thus here we need to check the points of M_{over} to identify whether they exist in F_{over} or not. In practice, we build a KD tree for the point cloud F_{over}. Then we set each point of M_{over} as a search point, and search its nearest neighbor point in F_{over}, where the nearest distance between them can also be obtained. More precisely, the point should belong to the static background if its nearest neighbor distance is smaller than the resolution of the point cloud F_{over}. Otherwise the point should belong to the spurious trails. Then we remove these trails in $M(t-1)$ and obtain $M_{left}(t-1)$. Finally, we accumulate $M_{left}(t-1)$ and $F(t)$ by equation (1), which theoretically leads to result shown in Fig. 4 (a).

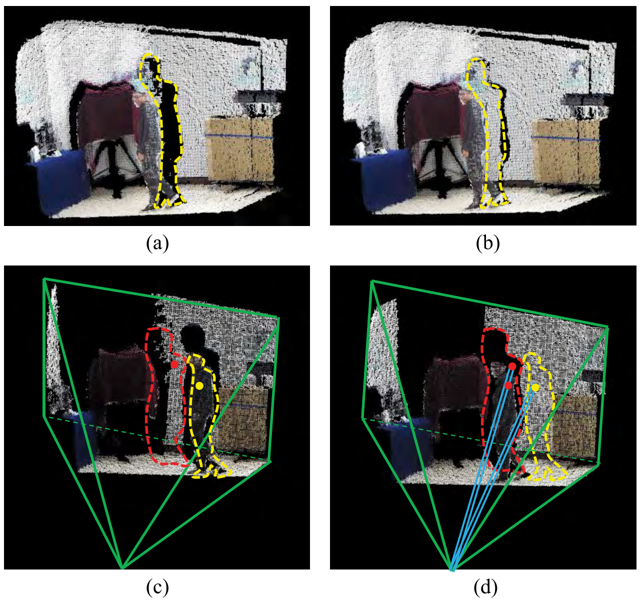


FIGURE 4. (a) shows the result reconstructed of two frames from Fig. 1(a) via proposed method without the PRTF presented in Subsection C of Section III, where some static backgrounds dotted in yellow lines are lost. (b) shows the result using PRTF to retrieve most missing backgrounds. (c) and (d) expound the geometrical principle of the PRTF: the green lines encircle view frustum overlaps of the two frames, where (c) represents the overlap from $M(t-1)$ and (d) is the overlap from $F(t)$; the red and yellow dotted lines in (c) and (d) represent the "spurious trails" detected by procedures of Subsection A and B – the yellow lines indicate the real spurious trails while the red lines indicate part of static backgrounds; the blue lines encircle a cylinder field based on ray tracing.

We can obtain good results just via procedures of Subsection A and B in real time, and the historical spurious trails can be well removed as shown in Fig. 4 (a). In practical SLAM system, the small amount of deficient backgrounds considered in Subsection C will be partially supplemented in 3D map in the future because of the movement of the camera. Therefore, it has no adverse effects on robot obstacle avoidance and autonomous navigation.

C. SUPPLEMENT BACKGROUNDS BASED ON PROJECTION RAY TRACING FILTER (PRTF)

However, in some 3D reconstruction applications, moving objects might occlude some backgrounds in frame $F(t)$ because of the limits of the RGB-D camera. These occluded background points exist in M_{over} but disappear in F_{over}, which also satisfy the judging condition of spurious trails points. Thus they will be removed as trails shown in Fig. 4 (a), which further leads to the deficiency of small amount of backgrounds. Here for the completeness of proposed method and for some reconstruction results which required strict

Algorithm 1 Proposed Spurious Trails Rejection Algorithm

```
Input: $N$ RGB-D frames; Flag: use PRTF (1) or not (0)
Output: updated 3D point cloud map $M(t)$
 1: for $i$-th RGB-D frame do
 2:     Transform the frame to point cloud $F(t)$
 3:     if $i = 1$ then
 4:         $T = I_4$ (identity matrix with dim = 4); Camera pose $P_t = T$; $M(t) = F(t)$
 5:     else
 6:         Obtain accurate transformation $T(t)$ between $F(t-1)$ and $F(t)$ via GICP
 7:         $T = T * T(t)$; New camera pose $P_t = T$
 8:         View overlap $M_{over} = function_{FrustumCulling}(P_t, M(t-1))$
 9:         View overlap $F_{over} = function_{FrustumCulling}(P_t, F(t) * T)$
10:         Construct KD tree for $F_{over}$
11:         for each point in $M_{over}$ do
12:             Search the nearest point in KD tree and obtain nearest distance $d$
13:             if $d > R$ ($R$ is the resolution of point cloud ) then
14:                 Set the point to "spurious trails"
15:                 if Flag == 1 then
16:                     Connect the point to camera center $O(x, y, z)$ from $P_t$ to get a line segment
17:                     Count the point where the distance from it to the line segment is smaller than $R/2$ and obtain $m$
18:                     if the number $m < 2$ then Set the point to "spurious trails"
19:                     else The point doesn't belong to "spurious trails"
20:                     end if
21:                 end if
22:             else The point doesn't belong to "spurious trails"
23:             end if
24:         end for
25:         Eliminate "spurious trails" from $M(t-1)$ and obtain $M_{left}(t-1)$
26:         $M(t) = M_{left}(t-1) + F(t) * T$
27:     end if
28:     $F(t-1) = F(t) * T$ ; $M(t-1) = M(t)$
29:     Output updated 3D point cloud map $M(t)$
30: end for
```

precision, we design a filter named "Projection Ray Tracing Filter" (PRTF) to solve this problem. PRTF is partially inspired by [12] which is based on ray tracing principle. As the Fig. 4 (c) and (d) shown, the red and yellow dotted lines represent the "spurious trails" points detected by procedures of Subsection A and B, which exist in M_{over} but disappear in F_{over}. Apparently, the yellow dotted lines indicate the real spurious trails and the red dotted lines indicate part of static backgrounds. For each point P_i in the dotted field, we use a straight line to connect it to the camera position point in F_{over}. Then, we make a cylinder using this line as axle wire and half the resolution of the point cloud as its radius, as blue field shown in Fig. 4 (d). If there are no other points in this cylinder region in F_{over}, P_i should belong to the static background and we can put it back to the whole point cloud map. Otherwise, P_i should be a part of the spurious trails. The final result using PRTF is shown in Fig. 4 (b), where the missing background is supplemented.

The specific implementation and details of our whole approach refer to the Algorithm 1. From Algorithm 1 we can see that the parameter R is crucial for the proposed algorithm. We process all point clouds by VoxelGrid sampling at a fixed resolution R throughout the whole course of the experiments. Actually, the parameter R is set based on the scene size and the resolution of original point cloud in mapping process. And it is independent of the moving velocities of objects. Based on this setting we make following discussions. If the object moves fast, the spurious trails of it can be easily eliminated. If an object moves slowly enough, we can barely distinguish between its current state and its historical state. Concretely, the distance which the object moves between two adjacent key frames is less than the resolution of point cloud. It will not be regarded as historical spurious trails nor be eliminated at current moment. But its movement will be definitely accumulated over time. Therefore, the distance which the object moves between some key frame in the future and current key frame must be larger than the point cloud resolution. As a result, the historical spurious trails will be eliminated and updated to a new state at that time.

TABLE 1. The parameters and different setups for all experiments.

Experiments	A. 1) & 2)	B. 1)	B. 2)
Sensor Parameters	Kinect: fx = 525.0, fy = 525.0 cx = 319.5, cy = 239.5	Kinect: fx = 525.0, fy = 525.0 cx = 319.5, cy = 239.5	stereo camera (KITTI Odometry dataset, i.e., sequence 05): fx = 707.1, fy = 707.1 cx = 601.9, cy = 183.1
Image Resolution	640*480	640*480	1226*370
View Frustum Parameters	$H_{fov} = 62.7°, V_{fov} = 49.1°$ $Z_{near} = 0.8m, Z_{far} = 4.0m$	$H_{fov} = 62.7°, V_{fov} = 49.1°$ $Z_{near} = 0.8m, Z_{far} = 4.0m$	$H_{fov} = 81.8°, V_{fov} = 29.3°$ $Z_{near} = 0.8m, Z_{far} = 15.0m$
Location Algorithm (from)	GICP(no loop closure)	RTABMAP (loop closure)	Stereoscan(loop closure)
Platform	Handle Kinect & Intel Core i7-5820K CPU	Kinect on TurtleBot robot & Intel Core i7-5820K CPU	KITTI Odometry dataset & Intel Core i7-5820K CPU
Point Cloud Resolution: R	$0.02m$	$0.05m$	$0.1m$
Environment Size	$5 \sim 10m^2$	$20 \sim 50m^2$	$100 \sim 1000m^2$

IV. RESULTS

Due to the fact that most methods for the dynamic scene reconstruction focused on the accurate solution of the camera trajectory, there is no benchmark with static background truth for dynamic scene mapping. Although we can evaluate our method in a simulated environment like [12], we decide to experiment directly in real environments leading to a richer set of scenes. In subsection A, we implement experiments in the scene of local corner reconstruction to show the details of dynamic spurious trails elimination. We also analyze the computational efficiency of proposed method to illustrate its real-time performance.

Our method can be easily ported to any of these SLAM systems because it is implemented before the mapping process of each frame. In details, we utilize our algorithm which is packaged into a function to process point clouds $F(t)$ and $M(t-1)$. In this process our algorithm can eliminate the historical spurious trails in $M(t-1)$ by comparing with $F(t)$. After that, point cloud map $M(t)$ without trails can be obtained by equation (1), which is suitable for robot navigation. In subsection B, we choose two state-of-the-art open source SLAM systems for experiments to prove this point. One is for indoor dynamic scenes and the other is for outdoor scenes of KITTI odometry dataset [21].

In order to make it clearer for readers to understand which configurations/parameters/methods are used for each experiment, we list them all in the Table 1.

A. EXPERIMENTS OF LOCAL SCENE RECONSTRUCTION: SHOWING MORE DETAILS

To further show the details of dynamic spurious trails elimination, we test our method in the local scene, such as a corner of room, reconstruction containing moving objects. The point cloud resolution is controlled to 0.02m through VoxelGrid [22] sampling, so the parameter of PRTF is chosen as half resolution $R/2$ according to the principle of our algorithm. The parameters of the view frustum are calculated according to equation (2) and (3). More paramenters can be found in Table 1 column "A. 1) & 2)". We first handle a Kinect to scan one corner of our lab office, and simultaneously some moving objects are arranged to move optionally in the corner with motions. More precisely, in current situation the camera and objects are all on the move where the moving velocities and directions are different. Then we compute the camera pose of each frame using GICP method [14], [22] and reconstruct the scene 3D map by equation (1).

1) EXPERIMENTS WITH SINGLE MOVING OBJECT

In this subsection we display reconstruction results in the local corner scene with only one moving object. The scan images of local corner are shown in Fig. 1 (a), and the reconstruction results directly based on equation (1) are shown in Fig. 1 (b). To prove the efficiency of our method, we implement it to eliminate historical spurious trails first and then reconstruct the 3D map using equation (1) as shown in Fig. 1 (c). It is observed that when there exists one moving object in the scene, our method can accurately eliminate spurious trails generated from traditional reconstruction process, which might mislead the passable decisions of mobile robots.

2) EXPERIMENTS WITH MULTIPLE MOVING OBJECTS

In this subsection, we aim to test proposed method in the scene with multiple moving objects. As the four images in Fig. 4 (a) displayed, we arrange for three persons to move optionally in the corner including motions like translation, rotation, and occlusion. We reconstruct the scene 3D map by equation (1) without using any moving objects rejection methods, as shown in Fig. 5(b).

In addition, we take advantage of the efficient estimation of scene flow proposed in [8] to eliminate moving objects from an RGB-D sequence. This method can be chosen because it is the only one that realizes real-time computation of extra motion information, i.e., scene flow, for dynamic objects. We leverage the efficient solution from [8] to compute the scene flow of each old point cloud frame and obtain the mask of moving objects based on geometric clustering. Then, the mask is used for elimination of moving objects in each old point cloud frame. Finally, these old frames are implemented

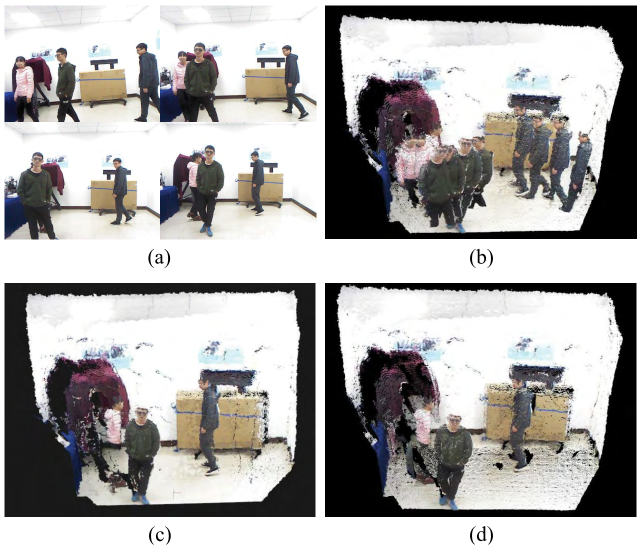


FIGURE 5. 3D reconstruction results of the scene containing multiple moving objects. (a) shows the scan images of the dynamic scene. (b) shows the oblique front view of 3D map via [14] without using any moving objects rejection methods. (c) represents the mapping result of [14] using [8] to eliminate moving objects first. In contrast, (d) corresponds to the result after removal of spurious trails via proposed method.

for scene 3D reconstruction. Note that registration method is still GICP [14], [22] while the difference is that here the point cloud frames no longer contain moving objects. The result is shown in Fig. 5 (c). Due to the restrictions of clustering, this method is not stable which indicates in some frames moving objects cannot be eliminated completely especially around boundaries of them. As shown in Fig. 5 (c), there are still several spurious trails around edges of moving objects. These trails are fragmentary and sparse which seem negligible in the figure but might disturb the normal robot navigation process. In contrast, the image in Fig. 5 (d) corresponds to result after removal of spurious trails for each former frame using proposed method. There are no fragmentary and sparse boundary trails that existed in (c). It can be concluded that our method works well under dynamic scenes with multiple moving objects.

Meanwhile, we compare the run time between [8] and proposed method in the elimination process of spurious trails, which is shown in Table 2. It is observed that

TABLE 2. The comparison of computational efficiency between proposed method and [8] called Fast-SF here. (Evaluated at half resolution of the RGB-D sequence shown in Fig. 5 (a) where the image size is 640 × 480 and one core of the Intel Core i7-5820K CPU, 3.30GHz×12).

Time t	Method	
	Fast-SF ($F(t)$ size / Runtime)	Ours ($M(t-1)$ size / Runtime)
t=1	58118 points / 45.3 ms	0 points / 0 ms
t=2	59890 points / 47.1 ms	58118 points / 2.1 ms
t=3	60235 points / 51.9 ms	93680 points / 3.7 ms
t=4	59924 points / —	128789 points / 5.9 ms

proposed method without using background supplement satisfies real-time requirement. In details, [8] takes about 50ms on average to compute scene flow for each down-sampled point cloud frame containing about 58,000 points. Then it takes at least 10ms to obtain the point cloud without moving objects by the binary mask. Thus [8] needs at least 60ms to process one frame, which is hard to fit some SLAM systems with high frame rate. On the contrary, our experimental statistics discover that proposed method promises less than 10ms for processing each frame $F(t)$ when the map $M(t-1)$ contains less than 350,000 points, which keeps the real-time performance in robot navigation tasks. Besides, proposed method attempts to eliminate points in the existed map $M(t-1)$ which are different from the scene under the robot's current view $F(t)$, also called spurious trails. Therefore it is also suitable for updating 3D map under semi-static situations such as a moved chair. However, methods like [7] and [8] only compute scene flow with adjacent frames thus they cannot update non-real-time changes of semi-static objects such as a moved chair, which burdens the normal robot navigation process.

B. GLOBAL SCENE MAPPING: SHOWING THE FUNCTION OF THE ALGORITHM IN SLAM SYSTEM

To demonstrate the practicability of our method for 3D map simultaneous updating, we integrate it into several typical SLAM systems. We will prove the effectiveness, real-time performance and robustness of our method through 3D mapping process in following indoor and outdoor environments.

1) INDOOR ENVIRONMENT: EXPERIMENTS IN LAB ENVIRONMENT

We further embed proposed method in RTABMAP [15] system and reconstruct 3D map for our lab office and hallway existing highly moving objects on TurtleBot robot platform [23], [24]. In test process, we implement the wheel odometry information as supplementation to ensure the accuracy of each key frame camera pose under dynamic scenes. This platform was chosen not only because of its representativeness, but also because of the robustness of its localization algorithm in dynamic scenes with a large number of moving objects. And considering that the area of the scene in this experiment is about dozens of square meters, we make the point cloud resolution is 0.05m. More paramenters and setup can be found in Table 1 column "B. 1)".

Fig. 6(a) and Fig. 7(a) show the reconstructed 3D maps of our lab office and hallway with dynamic objects via the system in [15] respectively. Obviously, there exist plenty of spurious trails in the maps when the TurtleBot robot moves. This problem directly leads the robot to do wrong decisions, i.e., it will decide on stopping even the moving object has gone away because it considers the spurious trails as obstacles in the front. In contrast, proposed method can realize the simultaneous 3D map updating and the robot could perceive the current scene synchronously, which leads to better navigation. For comparison, we present

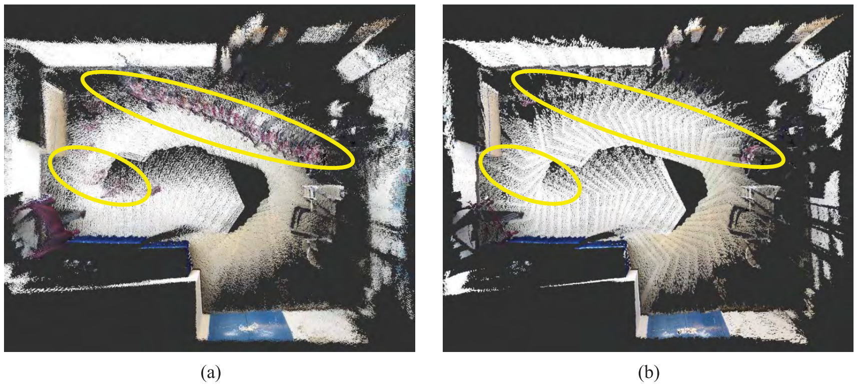


FIGURE 6. **The 3D reconstruction results of our lab office. (a) is the result via the system in [15] and there exist plenty of spurious trails encircled by yellow lines. (b) is the final result using the system in [15] added our approach, where spurious trails in yellow circles are eliminated completely.**

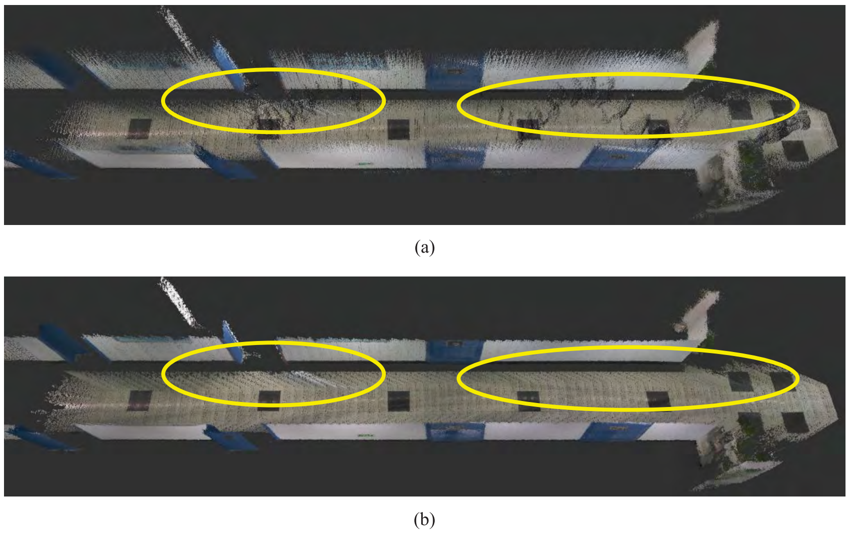


FIGURE 7. **The 3D reconstruction results of hallway. (a) is the result via the system in [15] and there exist plenty of spurious trails encircled by yellow lines. (b) is the final result using the system in [15] with our approach, where spurious trails in yellow circles are eliminated completely.**

the final result via RTABMAP added our approach as shown in Fig. 6(b) and Fig. 7(b).

2) OUTDOOR ENVIRONMENT: EXPERIMENTS ON KITTI ODOMETRY DATASET

In this subsection, we test our method on outdoor dynamic scenes. The KITTI Odometry benchmark [25] contains stereo sequences recorded from a car in urban and highway environments. The stereo sensor has a ∼54cm baseline and works at 10Hz. We select three representative sequences 02, 05 and 07 with moving objects from this challenging real-world benchmark to further prove the validity of our algorithm for RGB-D data even from a stereo sensor.

In the existing open source stereo SLAM systems, we selected Stereoscan [16] which has good localization and reconstruction results in practical tests. Because outdoor scenes cover a large area with size about a few hundred square meters, we control the point cloud resolution to 0.1 m for the real-time performance of the mapping algorithm. And the view frustum's far plane parameter Z_{far} is different with that in previous experiments. More details are shown in Table 1 column "B. 2)".

Firstly, we use the LIBELAS stereo matching [26] to get the disparity map for each stereo frame (including two images captured from left and right cameras) and turn it into depth map as the "D" of RGB-D data. Then, we utilize the system Stereoscan proposed in [16] to obtain global poses of the stereo sequences. Finally, we reconstruct the whole scene in dense 3D point cloud map with both global poses and RGB-D frames. For comparison, we display part of 3D map of sequence 02 which contains some moving objects in Fig. 8(a). After processing via proposed method, we obtain

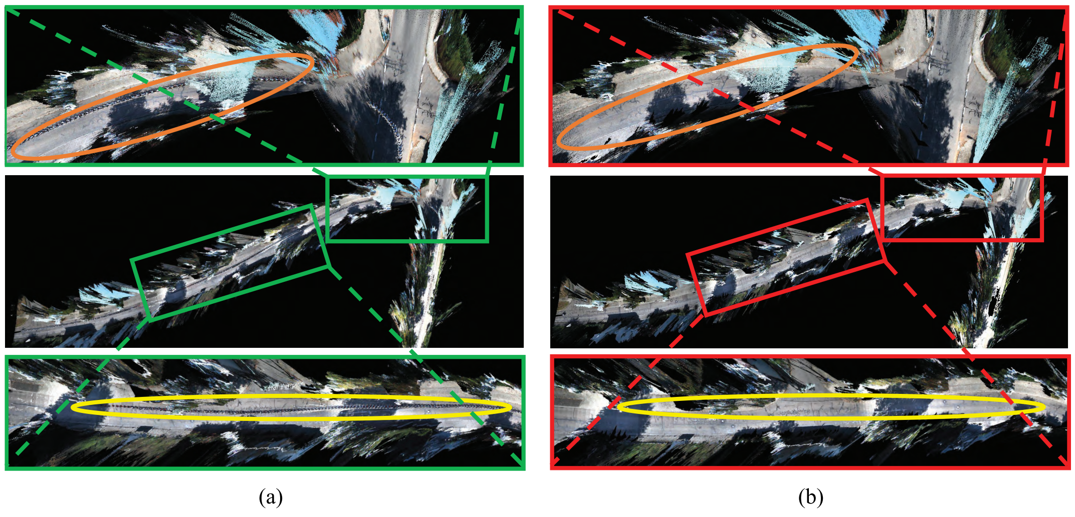


FIGURE 8. Sample results of the scene 3D reconstruction of the KITTI odometry datasets where numerous moving objects exist. (a) shows the result obtained by [16], and the zoom-in regions show the immense "ghosts" effects from the moving objects. (b) is our result which accurately removes all "ghosts" and performs better.

results as shown in Fig. 8 (b) where the spurious trails are removed completely. It can be easily concluded that proposed method has outstanding performance on removing spurious trails and updating 3D map for dynamic environments.

If the estimation of camera position and pose in SLAM system is inaccurate, the view frustum obtained by proposed method cannot overlap with the real frustum completely and the final overlap area can be inaccurate. Therefore, proposed method may falsely eliminate part of background and remain some dynamic spurious trails. However, proposed method is in terms of point cloud mapping which can process each key frame in real-time. Although the localization failure of one key frame leads to the deviations when eliminating spurious trails in mapping, our method can update the whole map in time and modify original errors with the subsequent accurate relocation and global optimization in SLAM system.

V. CONCLUSION AND PERSPECTIVES

We have presented a 3D point cloud map updating approach for RGB-D data which is generated from a depth camera or a stereo camera. Unlike other methods, our approach based on view frustum culling filter and ray tracing principle focuses only on 3D map updating without caring about the estimation of camera poses. As a result, it can remove dynamic spurious trails in real-time in standard CPUs and update 3D maps simultaneously. Furthermore, it can be integrated into any SLAM or reconstruction systems easily and it is suitable for both static and dynamic environments in robot navigation or 3D reconstruction. The comparison to other dynamic objects rejection methods in real environments containing multiple moving objects shows very competitive performance of proposed approach. Experiments both in real lab office and on KITTI dataset illustrate that how the proposed approach for dynamic spurious trails rejection enhances both robots navigation and scene 3D reconstruction.

Our work in progress will mainly focus on the optimization and GPU acceleration of proposed PRTF. Further work will include extending the current research to trajectory tracking of moving objects, action and intention analysis based on moving information.

REFERENCES

[1] D.-H. Kim, S.-B. Han, and J.-H. Kim, *Visual Odometry Algorithm Using an RGB-D Sensor and IMU in a Highly Dynamic Environment*. Springer, 2015.

[2] T. S. Leung, "Visual navigation aid for the blind in dynamic environments," in *Proc. Comput. Vision Pattern Recognit. Workshops*, 2014, pp. 579–586.

[3] D. H. Kim and J. H. Kim, "Effective background model-based RGB-D dense visual odometry in a dynamic environment," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1565–1573, Dec. 2016.

[4] Y. Wang and S. Huang, "Towards dense moving object segmentation based robust dense RGB-D slam in dynamic scenarios," in *Proc. (ICARCV)*, Dec. 2014, pp. 1841–1846.

[5] 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, MAr. 2017.

[6] S. Li and D. Lee, "RGB-D SLAM in dynamic environments using static point weighting," *IEEE Robot. Autom. Lett.*, vol. 2, no. 4, pp. 2263–2270, Oct. 2017.

[7] D. Kochanov, A. Ošep, J. Stückler, and B. Leibe, "Scene flow propagation for semantic mapping and object discovery in dynamic street scenes," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 1785–1792.

[8] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast odometry and scene flow from RGB-D cameras based on geometric clustering," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2017, pp. 3992–3999.

[9] C. Jiang, P. D. Paudel, Y. Fougerolle, D. Fofi, and C. Demonceaux, "Static and dynamic objects analysis as a 3D vector field," in *Proc. Int. Conf. 3D Vis.*, 2017.

[10] M. Fehr *et al.*, "TSDF-based change detection for consistent long-term dense reconstruction and dynamic object discovery," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 5237–5244.

[11] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3D map maintenance in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2014, pp. 3712–3719.

[12] F. Ferri, M. Gianni, M. Menna, and F. Pirri, "Dynamic obstacles detection and 3D map updating," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2015, pp. 5694–5699.

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

[14] A. V. Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," *Robot. Sci., Syst.*, vol. 2, no. 4, p. 435, Jun. 2009.

[15] M. Labbe and F. Michaud, "Online global loop closure detection for large-scale multi-session graph-based slam," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2014, pp. 2661–2666.

[16] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3D reconstruction in real-time," in *Proc. Intell. Vehicles Symp.*, 2011, pp. 963–968.

[17] (2017). *Kinect for Xbox One*. Accessed: Dec. 7, 2017. [Online]. Available: https://www.xbox.com/en-US/xbox-one/accessories/kinect

[18] U. Assarsson and T. Moller, "Optimized view frustum culling algorithms for bounding boxes," *J. Graph. Tools*, vol. 5, no. 1, pp. 9–22, 2000.

[19] J. Antonio and F. Ramires. (2017). *OpenGL Lighthouse 3D—View Frustum Culling Tutorial*. Accessed: Dec. 4, 2017. [Online]. Available: http://www.turtlebot.com/

[20] D. S. Kora. (2000). *Efficient View Frustum Culling*. [Online]. Available: http://old.cescg.org/CESCG-2002/DSykoraJJelinek/

[21] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2012, pp. 3354–3361.

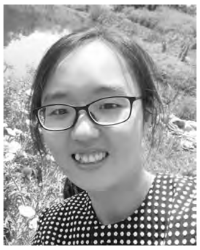
[22] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011, pp. 1–4.

[23] (2017). *TurtleBot*. Accessed: Dec. 4, 2017. [Online]. Available: http://www.turtlebot.com/

[24] (2017). *The Robot Operating System (ROS)*. Accessed: Dec. 4, 2017. [Online]. Available: http://www.ros.org/

[25] F. Delorme, S. Slempkes, G. Alibert, B. Rose, and J. Brandon, "Butt-jointed DBR laser with 15 nm tunability grown in three MOVPE steps," *Electron. Lett.*, vol. 31, no. 15, pp. 1244–1245, Jul. 1995.

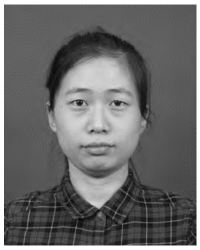
[26] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proc. Asian Conf. Comput. Vis.*, 2010, pp. 25–38.



WENJUN SHI received the B.S. degree from the Nanjing University of Aeronautics and Astronautics, China, in 2015. She is currently pursuing the Ph.D. degree in information and communication engineering with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China. She is also an Intern Student with Shanghai Eyevolution Technology Company Ltd., Shanghai. Her current research includes 3-D reconstruction, 3-D scene understanding, and visual odometry.

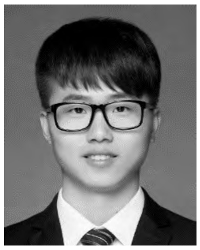
JIAMAO LI received the Ph.D. degree from the Tokyo Institute of Technology, Japan, in 2012. He is currently an Associate Professor with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, China. His current research interests include computer vision, machine vision, 3-D micro-imaging, and artificial intelligence.

YANQING LIU received the B.E. degree in Internet of Things engineering from Shandong University, Shandong, China, in 2014. He is currently pursuing the Ph.D. degree in communication and information systems with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, and the University of Chinese Academy of Sciences. His research interests include robot vision, visual odometry, visual simultaneous localization and mapping, and robotics.



DONGCHEN ZHU received the B.S. degree from Wuhan University, China, in 2013. She is currently pursuing the Ph.D. degree in information and communication engineering with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China.

Her current research includes computer vision, stereo vision, 3-D reconstruction, and artificial intelligence.



DONGDONG YANG received the M.Sc. degree from the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, China, in 2017.

He is currently with Shanghai Eyevolution Technology Company Ltd., Shanghai, China. His research interests include computer vision, and mobile robots planning and navigation.

XIAOLIN ZHANG received the Ph.D. degree from Yokohama National University in 1995. He was a Professor with the Tokyo Institute of Technology, Japan, from 2012 to 2013.

He is currently a Professor with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, China. His research interests include bionics, brain science, computer vision, and artificial intelligence.