Robotics and Autonomous Systems 108 (2018) 115–128

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Motion removal for reliable RGB-D SLAM in dynamic environments

Yuxiang Sun [a], Ming Liu [a,*], Max Q.-H. Meng [b,*]

[a] Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong, China
[b] Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China

HIGHLIGHTS

- An on-line RGB-D data-based motion removal approach is proposed.
- The approach does not require prior knowledge from moving objects.
- The approach improves RGB-D SLAM in various dynamic scenarios.

ARTICLE INFO

Article history:
Received 28 January 2018
Received in revised form 27 June 2018
Accepted 8 July 2018
Available online 17 July 2018

Keywords:
Motion removal
Codebook model
Dynamic environments
RGB-D SLAM

ABSTRACT

RGB-D data-based Simultaneous Localization and Mapping (RGB-D SLAM) aims to concurrently estimate robot poses and reconstruct traversed environments using RGB-D sensors. Many effective and impressive RGB-D SLAM algorithms have been proposed over the past years. However, virtually all the RGB-D SLAM systems developed so far rely on the static-world assumption. This is because the SLAM performance is prone to be degraded by the moving objects in dynamic environments. In this paper, we propose a novel RGB-D data-based motion removal approach to address this problem. The approach is on-line and does not require prior-known moving-object information, such as semantics or visual appearances. We integrate the approach into the front end of an RGB-D SLAM system. It acts as a pre-processing stage to filter out data that are associated with moving objects. Experimental results demonstrate that our approach is able to improve RGB-D SLAM in various challenging scenarios.

1. Introduction

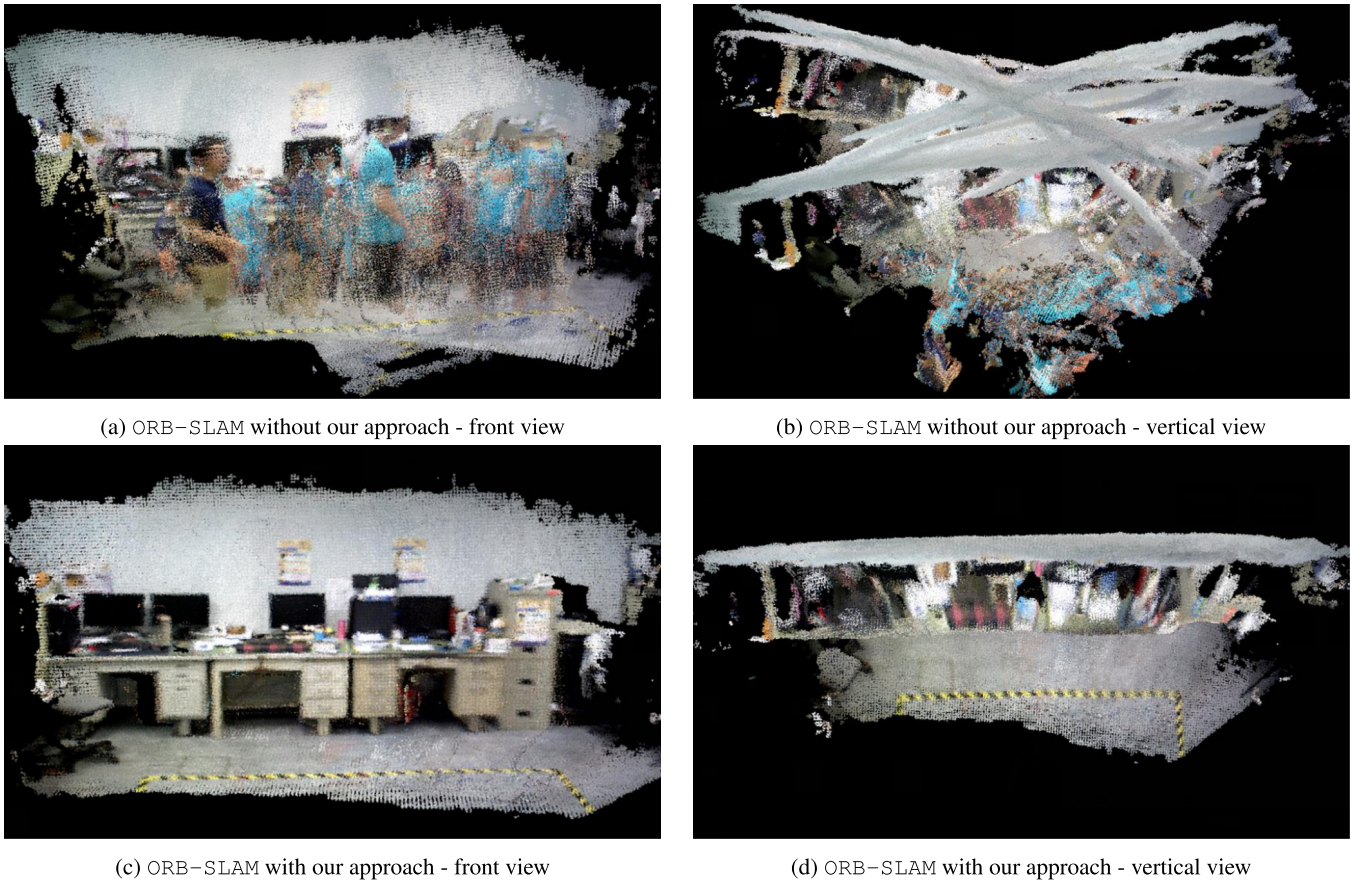
Simultaneous Localization and Mapping (SLAM) is a fundamental step for many robotic applications. It concurrently estimates robot poses and reconstructs traversed environment models. Many effective SLAM algorithms using visual sensors, such as monocular cameras [1], stereo cameras [2] and RGB-D cameras [3], have been proposed over the past years. Related technologies, such as the augmented reality [4] and autonomous driving [5], have benefited from the development of the SLAM technology. It is worth noting that the advent of the RGB-D cameras has changed the computer vision world [6]. They provide colored point clouds with real-scale distance information, which greatly benefits the dense 3-D environment reconstruction. Many impressive RGB-D SLAM systems have been developed [7–12] in recent years, and most of them adopt the graph optimization framework. We refer readers to this survey [13] to get an overview of the progress on the graph SLAM algorithm.

However, virtually all the current RGB-D SLAM algorithms are proposed under the *static-world assumption*. It requires that there is no moving object existing in the environment during the traversal of robots. The data associations in the SLAM front end can be hindered by moving objects. With incorrect data associations fed into the SLAM back end, the graph optimization process could be severely jeopardized, which finally leads to a catastrophic failure for the localization and mapping processes. Thus, the technology of RGB-D SLAM is still vulnerable in dynamic environments.

The data associations in the SLAM front end consists of two components, namely, the short-term data association and the long-term data association [13]. The short-term data association determines adjacent pose estimations, while the long-term one has an impact on the loop detection. Take as an example the sparse feature-based RGB-D SLAM. Standard robust estimators, such as the RANdom SAmple Consensus (RANSAC) algorithm [14], are usually employed in the SLAM front end to reject outlier feature associations. However, it is hard to reliably reject outliers when moving objects are not trivial in the camera field of view. In such a case, the outliers are unavoidably used for computing the robot poses, which makes the pose estimation erroneous. When a robot returns to a previously visited place where moving objects have

* Corresponding authors.
E-mail addresses: yxsun@link.cuhk.edu.hk (Y. Sun), eelium@ust.hk (M. Liu), max.meng@cuhk.edu.hk (M.Q.-H. Meng).

https://doi.org/10.1016/j.robot.2018.07.002
0921-8890/



(a) ORB−SLAM without our approach - front view

(b) ORB−SLAM without our approach - vertical view

(c) ORB−SLAM with our approach - front view

(d) ORB−SLAM with our approach - vertical view

Fig. 1. The qualitative comparison for the point-cloud maps built by ORB−SLAM [11] without and with our motion removal approach in a dynamic environment. (a) and (b) show the map produced by the original ORB−SLAM system. The map quality is significantly degraded. (c) and (d) show the map produced by the integrated SLAM system. With our approach, the ORB−SLAM system is able to build a correct map. The comparison clearly demonstrates the negative effects caused by moving objects in such a dynamic environment, and the enhanced performance provided by our approach.

gone away, the loop detection would be confused by matching the same scene but with different visual appearances. If we eliminate moving objects at the first exploration of a place, we can compare the image frames using just the static feature points, which would lead to a much more reliable loop detection result. Moreover, we can merely use the static feature points to get accurate robot pose estimations. Therefore, eliminating moving objects is able to reduce the incorrect data associations, which is critical to improve the SLAM performance.

In this paper, we develop a novel RGB-D data-based motion removal approach to address the problem of RGB-D SLAM in dynamic environments. We refer to the dense pixel-wisely moving-object segmentation as *motion removal*. Our approach serves as a pre-processing stage to filter out data that are associated with moving objects. With our approach, incorrect data associations can be greatly reduced in the SLAM front end.

Fig. 1 qualitatively compares the resulting point-cloud maps produced by the ORB−SLAM system [11] and the system integrated with our motion removal approach in a dynamic environment. In this scenario, two persons are playing with a basketball in an office room, where the moving objects are the two persons and the basketball. We perform the SLAM algorithm in this environment using a hand-held Asus Xtion RGB-D camera, which is moved in a circle-like trajectory in the scene. Comparing Fig. 1a with b, we could see that the quality of the resulting map is substantially degraded. Almost no object or scene structure is correctly aligned. This is mainly caused by the jeopardized camera pose estimations due to the incorrect data associations. Moreover, moving objects are recorded as spurious objects in the resulting map, which makes the map virtually useless for future applications, such as the map-based navigation. The point-cloud map built with our motion removal approach is displayed in Fig. 1c and d. The models for the desks, monitors, wall, floor and scene structure are correctly built. Virtually no point from the moving objects is recorded in the map. The holes caused by motion removal in the point-cloud map are complemented by the frame fusion with redundant scans. Fig. 1 clearly illustrates the negative effect caused by moving objects in dynamic environments. It should be noted that the tested environment shown in Fig. 1 is a small-scale environment. We believe that using such an example would be sufficient to illustrate the negative impact caused by moving objects. This is because large-scale environments are generally more challenging for SLAM algorithms. By performing the experiments in such a small-scale environment, the ORB-SLAM system has already given the unsatisfied performance. The performance in large-scale dynamic environments would not be better than this. The main contributions of this paper are summarized as follows:

1. We propose a novel RGB-D data-based on-line motion removal approach. A foreground model is built and updated incrementally. No prior information of moving objects, such as semantics or visual appearances, is needed.
2. We explain why we use motion removal to address the problem of RGB-D SLAM in dynamic environments. The experimental results confirm that the robustness of RGB-D SLAM can be increased with motion removal.
3. We integrate our motion removal approach with an RGB-D SLAM system. Evaluations and method comparisons are

performed with the widely used TUM RGB-D benchmark dataset [15].

The remainder of this paper is organized as follows. Section 2 reviews the related work. Section 3 formulates the problem of RGB-D SLAM in dynamic environments and explains why we use motion removal to address this problem. Sections 4 and 5 give the overview and the details of our approach, respectively. The experimental results are presented in Section 6. We conclude this paper and discuss the future work in the last section.

2. Related work

For Visual SLAM in dynamic environments, the mainstream solution is to identify the features or pixels that are associated with moving objects. This process is referred to as motion segmentation and many approaches have been proposed. We generally divide the motion segmentation approaches into two categories: sparse methods and dense methods. The sparse methods identify feature points from moving objects, while the dense ones segment moving objects pixel-wisely.

2.1. Sparse motion segmentation

Lin et al. [16] extended the SLAM and Moving Object Tracking (SLAMMOT) algorithm [17] with visual information. They assumed that adding a new feature from moving objects into the SLAM system can degrade the SLAM performance. Moving-object features were detected based on this assumption by examining the SLAM performance with and without adding a new feature. Ozden et al. [18] detected and tracked moving-object features by selecting motion models from an over-complete set of motion hypothesis. The approach was not suitable for long sequences due to the huge amount of combinatorial trials of motion models. Zou et al. [19] was the first to use multiple cameras for SLAM in dynamic environments. Feature points from dynamic objects were identified based on a two-step verification scheme. In the first step, intra-camera outliers were found by reprojection errors. The outliers were considered as the candidates of dynamic feature points. In the second step, the outliers were verified by checking whether they were inter-camera inliers. If they were not inter-camera outliers, the feature points were considered as dynamic points. Tan et al. [20] observed that feature points from dynamic objects normally aggregated together while the feature points from static objects usually distributed evenly. Based on this observation, a prior-based adaptive RANSAC algorithm was proposed to discriminate moving-object features against static-object features. Wang et al. [21] proposed a method to group neighboring feature points that share close scene flows as clusters. The method employed the clustered feature pairs to compute the transformation matrix instead of directly selecting random feature pairs. The feature points from static objects were indicated from the largest inlier group.

2.2. Dense motion segmentation

The major limitation of the sparse methods is that moving objects are unavoidably recorded as spurious objects in the resulting maps. In order to build clear maps, moving objects are required to be densely segmented. As aforementioned, we refer to the dense motion segmentation as *Motion Removal*. Our approach falls into this category.

Wang et al. [22] proposed an off-line motion removal method based on the long-term video analysis algorithm [23]. The method merged the regions that shared the same motion model and considered the largest group as static objects. Jiang et al. [24] proposed an off-line trajectory clustering method based on the sparse subspace clustering algorithm [25]. Dense motion segmentation was realized using a multi-seeded region growing technique with the classified moving-object trajectories. As the method relies on the consistency of the tracked features, it suffers from tracking lost and object occlusions. Sevilla-Lara et al. [26] improved the optical flow estimation with semantic segmentation. The satisfied performance was promising for motion removal. However, as the method relies on the prior-known semantic information of objects, it is not suitable for applications in unknown environments. Vineet et al. [27] proposed a semantic fusion algorithm by modifying the Truncated Signed Distance Function (TSDF) fusion scheme [28] with the information from semantic segmentation. The algorithm accelerated the data fusing in dynamic spaces by assigning higher weights to the voxels that are labeled as moving objects, which was able to avoid fusing the depth data from moving objects into the static scene. Similar as [26], the method relies on the prior-known semantic knowledge, which limits the applications in unknown environments. Jiang et al. [29] proposed an off-line flow field analysis method to discriminate the static flows against the dynamic flows with a 3-D range finder. Individual object flows were grouped using the proposed sparse flow clustering technique. Moving objects were pixel-wisely segmented using a region growing algorithm with the clustered flows. The method relies on the feature-rich point-cloud data produced by long-range 3-D Lidars so that the ego-motion compensation can be performed using the `loam_velodyne` algorithm [30]. Therefore, the method is not suitable for RGB-D cameras which can only provide point clouds with limited geometric features due to the short range measurements.

Our RGB-D data-based motion removal approach differs the related work in mainly two aspects. Firstly, our approach is on-line. No future data or batch data processing is required. Thus, it is able to support the on-line SLAM process. We believe that this feature is of great significance for robotic applications, because there exist scenarios that robots must localize themselves real-timely and moving objects could not be totally evacuated to leave only static objects in the environments for mapping. For instance, many international airports open 24 h a day. Moving objects always exist in such environments that robots are required to do on-line localization and mapping in dynamic environments. Secondly, our approach does not rely on prior-known information of moving objects, such as semantics or visual appearances. It is applicable in unknown environments, where motions cannot be previously known. We consider this as a key benefit of our approach. In real-world environments, the motion status of objects is indeterminate, so it is difficult to predict moving objects in advance. In addition, enumerating all the possible moving objects is almost impracticable. Having no assumptions on moving objects enables our approach to work in the most general case.

The most similar work to ours is the RGB-D camera-based motion removal method presented in [31]. The method is on-line and developed without prior-known object information. However, as the method relies on the Maximum-a-Posterior scheme to determine the foreground, it could segment only one cluster of moving objects with close depth measurements. In most cases, the method only segments one moving object at a time. Different from [31], our approach models the foreground in different classes, so there is no limit on the number of moving objects for segmentation. In addition, our method is equipped with the on-line learning capability, which allows it to update the foreground model incrementally. The information from disappeared or newly appeared moving objects can be cleared or added into the model timely, while the method proposed in [31] has no such on-line incremental learning capability.

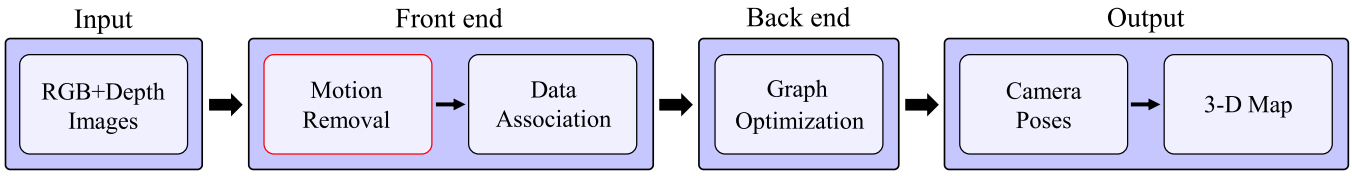


Fig. 2. We use motion removal as the solution to address the problem of RGB-D SLAM in dynamic environments. The diagram depicts a typical graph SLAM structure. The front end constructs the graph for the SLAM problem. The back end optimizes the graph to generate the optimal estimations for the camera poses. In our approach, *Motion Removal* is embedded into the SLAM front end, which is highlighted with a red box in the figure. It acts as a pre-processing stage to filter out data that are associated with moving objects. The figure is best viewed in color.

3. Problem statement

This section describes the problem of RGB-D SLAM in dynamic environments and explains why we use motion removal to address this problem. The SLAM problem can be naturally described in a graph structure [32], where the vertexes encode robot poses or landmark positions, and the edges represent constraints between vertexes. The constraints are pose transformations estimated from odometry, landmark reprojections, etc. The SLAM problem is to find the vertexes with the measured constraints. The graph SLAM consists of two parts: the front end which constructs the graph, and the back end that minimizes a non-linear error function to find the optimal configurations for the vertexes.

In the probabilistic view [33], finding the optimal values in graph SLAM is to solve a Maximum-A-Posterior (MAP) problem. Let $\mathcal{X}$ and $\mathcal{L}$ denote camera poses and landmark positions, where $\mathcal{X} = \{\mathbf{x}_t : t = 0, 1, \ldots, n\}$, $\mathcal{L} = \{\ell_t : t = 0, 1, \ldots, n\}$, $\mathbf{x}_t$ and ℓ_t represent the camera pose and landmark position at time t. Note that each ℓ_t is a set of landmark positions. The MAP estimation is to find the optimal value $\mathcal{X}^*$ and $\mathcal{L}^*$ by maximizing the posterior:

$$\mathcal{X}^*, \mathcal{L}^* = \arg\max_{\mathcal{X},\mathcal{L}} p(\mathcal{X}, \mathcal{L}|\mathbb{Z}, \mathbb{U}) \tag{1}$$

where $\mathbb{Z} = \{\mathbf{z}_t : t = 1, 2, \ldots, n\}$ and $\mathbb{U} = \{\mathbf{u}_t : t = 1, 2, \ldots, n\}$ are the measurements and the control inputs. According to the Bayesian rule, the posterior in (1) is expressed as:

$$p(\mathcal{X}, \mathcal{L}|\mathbb{Z}, \mathbb{U}) = \frac{p(\mathbb{Z}|\mathcal{X}, \mathcal{L}, \mathbb{U})p(\mathcal{X}, \mathcal{L}|\mathbb{U})}{p(\mathbb{Z}|\mathbb{U})} \tag{2}$$

Let $\eta = p(\mathbb{Z}|\mathbb{U})^{-1}$, where η is a constant because it is independent of $\mathcal{X}$ and $\mathcal{L}$. The term $p(\mathbb{Z}|\mathcal{X}, \mathcal{L}, \mathbb{U})$ reduces to $p(\mathbb{Z}|\mathcal{X}, \mathcal{L})$ by dropping the irrelevant conditioning variable $\mathbb{U}$. Thus, the posterior in (1) is expressed as:

$$p(\mathcal{X}, \mathcal{L}|\mathbb{Z}, \mathbb{U}) = \eta p(\mathbb{Z}|\mathcal{X}, \mathcal{L})p(\mathcal{X}, \mathcal{L}|\mathbb{U}) \tag{3}$$

In RGB-D SLAM, the odometry is normally computed from visual measurements, so it can be generalized into the measurement model. Thus, we ignore the variable $\mathbb{U}$ and assume there is no prior knowledge about $\mathcal{X}$ and $\mathcal{L}$. The MAP estimation reduces to a maximum likelihood problem:

$$\mathcal{X}^*, \mathcal{L}^* = \arg\max_{\mathcal{X},\mathcal{L}} \kappa p(\mathbb{Z}|\mathcal{X}, \mathcal{L}) \tag{4}$$

where $\kappa = \eta p(\mathcal{X}, \mathcal{L})$ is a new constant. We assume that the measurement $\mathbb{Z}$ is generated according to the Gaussian distribution $\mathcal{N}\big(\zeta_t(\mathbf{x}_t, \ell_t), \Xi_t\big)$, where $\zeta_t(\cdot)$ is the measurement function at time t, Ξ_t is the error covariance at time t [34]. This leads to the likelihood function at time t:

$$\begin{aligned} &p(\mathbf{z}_t|\mathbf{x}_t, \ell_t) \\ &= \exp\Big\{-\frac{1}{2}\big[\mathbf{z}_t - \zeta_t(\mathbf{x}_t, \ell_t)\big]^T \Xi_t^{-1}\big[\mathbf{z}_t - \zeta_t(\mathbf{x}_t, \ell_t)\big]\Big\} \end{aligned} \tag{5}$$

The formulation in (5) is determined according to the individual problem. In bundle adjustment, for instance, $\zeta_t(\cdot)$ is a projective function from 3-D points to 2-D image pixels, $\mathbf{z}_t$ represents the corresponding 2-D pixel coordinates. For odometry, $\zeta_t(\cdot)$ models the relative transformations between consecutive poses, $\mathbf{z}_t$ denotes the measured consecutive pose transformations. For loop closure, $\zeta_t(\cdot)$ computes the relative transformation between the loop candidates, $\mathbf{z}_t$ is the measured transformation between the poses. Note that for the latter two cases, the difference operation are performed on manifold $\mathbb{SE}(3)$.

The data associations in the SLAM front end construct the error terms in (5). In other words, the correspondences between measurements and measurement functions are determined by data associations. For instance, the short-term data associations determine the pose estimation. The long-term data associations determine the loop closure. Let Δ_t and Γ_t denote the sets of correctly and wrongly associated measurements at time t. There exists:

$$\mathbf{z}_t = \{\mathbf{z}_t^r \in \Delta_t\} \cup \{\mathbf{z}_t^s \in \Gamma_t\} \tag{6}$$

where $r = 1, 2, \ldots, |\Delta_t|$ and $s = 1, 2, \ldots, |\Gamma_t|$, $|\Delta_t|$ and $|\Gamma_t|$ are the cardinalities of the two sets. The likelihood in (4) factorizes into:

$$p(\mathbb{Z}|\mathcal{X}, \mathcal{L}) = \prod_{t=1}^{n}\Big[\prod_{r=1}^{|\Delta_t|} p(\mathbf{z}_t^r|\mathbf{x}_t, \mathbf{l}_t)\prod_{s=1}^{|\Gamma_t|} p(\mathbf{z}_t^s|\mathbf{x}_t, \mathbf{l}_t)\Big] \tag{7}$$

Substituting (5) into (7) and taking the negative logarithm of both sides of (7), the likelihood maximization problem turns into a non-linear least square problem:

$$\begin{aligned} &-\log p(\mathbb{Z}|\mathcal{X}, \mathcal{L}) \\ &= \sum_{t=1}^{n}\frac{1}{2}\Big\{\sum_{r=1}^{|\Delta_t|}\big[\mathbf{z}_t^r - \zeta_t(\mathbf{x}_t, \ell_t)\big]^T \Xi_{\Delta,t}^{-1}\big[\mathbf{z}_t^r - \zeta_t(\mathbf{x}_t, \ell_t)\big] + \\ &\quad \sum_{s=1}^{|\Gamma_t|}\big[\mathbf{z}_t^s - \zeta_t(\mathbf{x}_t, \ell_t)\big]^T \Xi_{\Gamma,t}^{-1}\big[\mathbf{z}_t^s - \zeta_t(\mathbf{x}_t, \ell_t)\big]\Big\} \end{aligned} \tag{8}$$

where $\Xi_{\Delta,t}$ and $\Xi_{\Gamma,t}$ are the error covariance for the two cases.

The formula (8) indicates that the optimizer tries to adjust the variables to relax the errors caused by the wrong data associations. If the wrong errors are significant, the whole optimization would be severely corrupted. In dynamic environments, wrong data associations are mainly caused by the moving objects. Removing moving objects helps to eliminate the wrong square error terms in (8), which reduces the influence on the optimizer to the minimal level. This explains why we use motion removal as a solution to address the problem of RGB-D SLAM in dynamic environments. Fig. 2 shows our idea of integrating motion removal into the front end of RGB-D SLAM. The motion removal module acts as a pre-processing stage to filter out data that are associated with moving objects. With incorrect data associations eliminated, the SLAM performance can be improved.

4. The approach overview

The idea of our motion removal approach is straightforward. Fig. 3 shows the flowchart of our approach. It consists of two

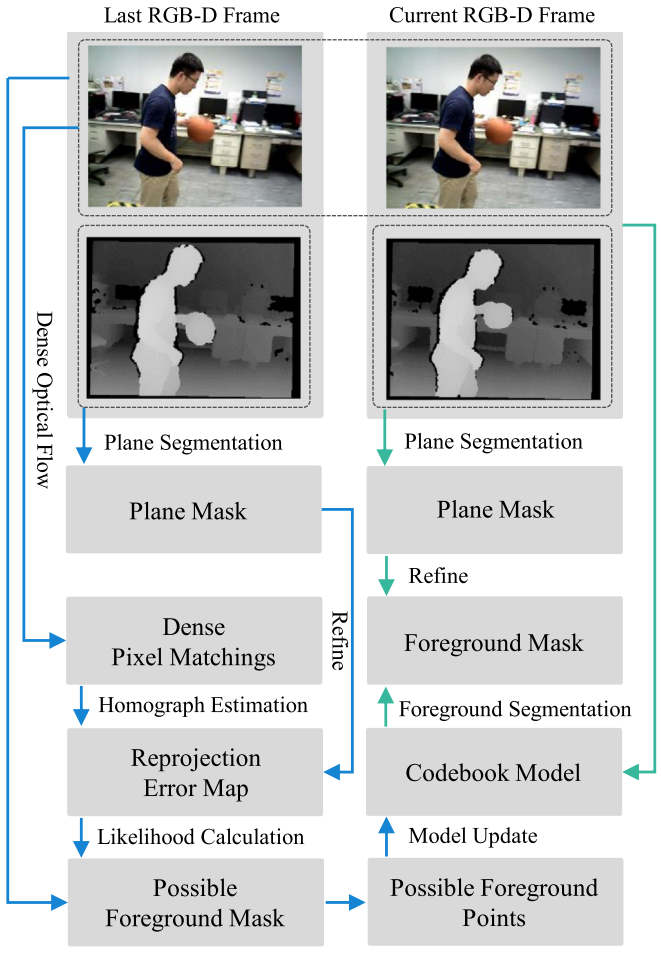


Fig. 3. The overview of our approach. The first two rows show the last and the current RGB and depth images. The blue and green arrows represent the procedures carried out in the *Learning* and *Inference* processes, respectively. Note that the two processes run simultaneously and our approach is on-line. The *Foreground Segmentation* procedure runs before the *Model Update* procedure in each iteration. The figure is best viewed in color.

on-line parallel processes: the *Learning* process that builds and updates the foreground model; the *Inference* process that pixel-wisely segments the foreground with the built model.

In the *Learning* process, we firstly employ a dense optical flow algorithm to find the 2-D pixel matchings between the two consecutive RGB images, which we refer to the last image and the current image. Secondly, we estimate the 2-D homography transformation between the two images using the dense pixel matchings. We warp the corresponding pixel coordinates in the current image with the homography transformation. The reprojection errors are found by subtracting the pixel coordinates in the last image with the warped ones. We generate the *Reprojection Error Map* from the subtraction results, which roughly indicates the moving objects. Thirdly, we derive the foreground likelihood pixel-wisely with the *Reprojection Error Map*. A scheme is devised to generate a *Possible Foreground Mask* with the likelihood. The *Possible Foreground Points* is a set of RGB-D points that are believed to be the foreground. It is found from the last RGB-D frame with the mask. Finally, we use the RGB-D information from the *Possible Foreground Points* to build and update the foreground model.

In the *Inference* process, we pixel-wisely compares the current RGB-D frame with the model to segment the foreground. It is worth noting that the final foreground segmentation results are not employed for the model update. This is because false segmentation results could contaminate the model and propagate negatively in future iterations.

In our approach, we use the codebook model [35,36] for the foreground modeling. The codebook model is a type of non-parametric model. We adapt the codebook learning and inference mechanisms from [37]. To the best of our knowledge, this is the first time that the codebook-based method has been used in the context of dealing with the SLAM problem in dynamic environments. Note that our approach is different from the codebook-based background subtraction algorithm [38]. Firstly, the background subtraction algorithm aims to construct a pixel-wise background model, so codebooks are built for each pixel in the image. In our approach, the foreground is learned as a whole, so we only maintain one foreground codebook model. Secondly, the background subtraction algorithm requires that cameras remain static during the whole process, while our approach allows free camera motions. Lastly, we use both the RGB and the depth data in our approach, while the background subtraction algorithm merely uses the RGB data. Our model contains the 4-channel RGB-D information that includes both the 3-channel RGB data and the 1-channel depth data.

In our approach, we assume that planes are static objects, so pixels from planes are not considered as moving objects. Planes are predominant features in man-made environments. Many SLAM approaches incorporate the plane information [39–43] into their algorithms. It is natural and without loss of generality to use this assumption. We identify the plane points with the RANSAC-based plane segmentation algorithm [44] implemented in the Point Cloud Library [45] with the GPU acceleration enabled. The algorithm segments the points that lie in the largest plane in the scene. The plane segmentation results are employed to refine the *Reprojection Error Map* in the *Learning* process, as well as the final foreground segmentation result in the *Inference* process.

5. The proposed approach

This section presents the details of the *Learning* and *Inference* processes of our approach.

5.1. The Learning process

We have four steps in the *Learning* process: finding the dense pixel matchings, generating the reprojection error map, finding the possible foreground points and modeling the foreground.

5.1.1. Finding the dense pixel matchings

We use the *EpicFlow* [46] to find the dense pixel matchings between two consecutive RGB images. A sample *EpicFlow* result is displayed in Fig. 4. As we can see, there are different optical flow values on the basketball and the person, including both the orientations and the magnitudes. The dense pixel matchings between the two consecutive images are found with the following formula:

$$\xi_i^{t-1} = \xi_i^t - \mathbf{v}(\xi_i^t) \tag{9}$$

where ξ_i^t is the location of pixel i in the current image, $\mathbf{v}(\cdot)$ is the optical flow vector, ξ_i^{t-1} is the corresponding pixel location in the last image. All the variables are two-element vectors.

5.1.2. Generating the reprojection error map

The reprojection error map roughly indicates the moving objects. The values with large reprojection errors are likely to be the foreground. Let π_t^{t-1} denote the 2-D homography transformation from the current RGB frame to the last RGB frame. The reprojection error for the pixel matchings is expressed as:

$$\varepsilon(\xi_i^{t-1}, \xi_i^t; \pi_t^{t-1}) = \left\| \xi_i^{t-1} - \pi_t^{t-1} \xi_i^t \right\| \tag{10}$$

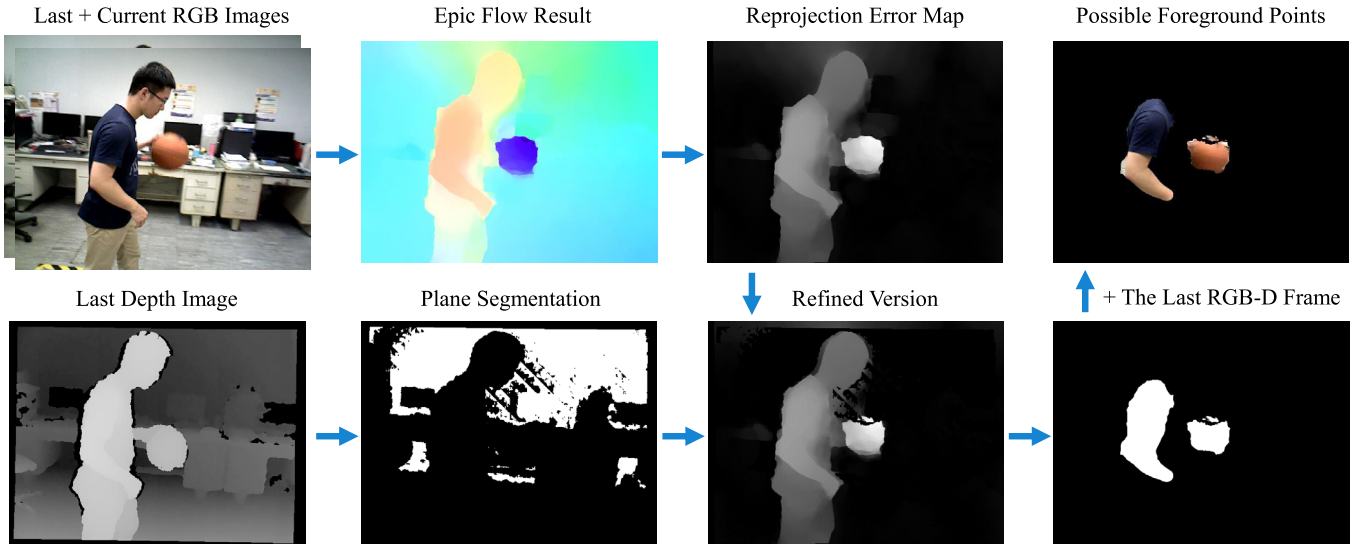
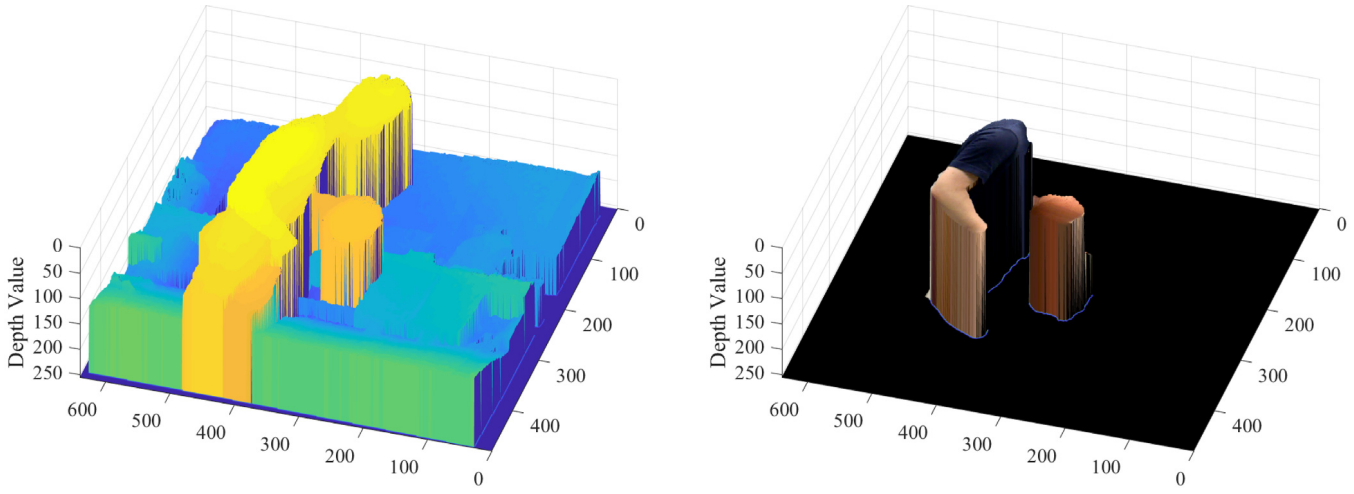


Fig. 4. This figure shows the process to find the possible foreground points. The shown scenario is a person playing with a basketball in an office room. The person and the basketball are the moving objects. Firstly, two consecutive RGB images are taken as input for the dense optical flow. The results are described in a color image, in which the color encodes the orientation and the intensity encodes the magnitude. Secondly, the *Reprojection Error Map* is generated with the optical flow results. The intensity values encode the reprojection errors. The *Plane Segmentation* is a binary mask in which the white pixels denote the found plane. It is used to refine the *Reprojection Error Map*. Finally, the *Possible Foreground Points* mask that is shown in the bottom right is derived from the refined *Reprojection Error Map*. The *Possible Foreground Points* are obtained from the mask with the last RGB-D image. The figure is best viewed in color.



(a) Depth values for the last frame.

(b) Possible foreground points.

Fig. 5. The 3-D view for the whole depth image and the possible foreground RGB-D points derived from Fig. 4. In order to display the depth values, they are normalized to 0–255. (a) shows the depth values for the whole image (depth increases from yellow to blue). (b) shows the 3-D surface plot of the possible foreground points with color information. The figure is best viewed in color.

where $\|\cdot\|$ represents the 2-norm and $\varepsilon(\cdot)$ is a scalar. π_t^{t-1} is found by minimizing the following least squares all over the pixel matchings:

$$\pi_t^{t-1} = \arg\min_{\pi} \sum_i \varepsilon(\xi_i^{t-1}, \xi_i^t; \pi_t^{t-1}) \tag{11}$$

The least square optimization is prone to be corrupted by outlier pixel matchings due to the relaxation for large errors. In our approach, we use the Least-Median-of-Squares (LMedS) algorithm [47] as the robust estimator. Unlike the RANSAC algorithm, the LMedS algorithm does not need threshold tunings, but requires that the ratio of outliers is below 50% [48]. Fortunately, this is true in most cases. The number of the pixel matchings from moving objects is usually less than the half of the total number pixel matchings.

The algorithm randomly selects 4 pairs of pixel matchings to compute π_t^{t-1}, and then repeats this process for a few times. For each trial, we compute the median reprojection error with all the pixel matchings:

$$\varrho_k = \underset{i}{\text{median}}\, \varepsilon(\xi_i^{t-1}, \xi_i^t; {}^k\pi_t^{t-1}), \tag{12}$$

where ϱ_k is the median reprojection error for the kth trial. Finally, the algorithm finds the minimum median reprojection error among all these trials:

$$\varrho_{min} = \arg\min_k \varrho_k \tag{13}$$

The π_t^{t-1} corresponding to ϱ_{min} is employed for our reprojection error map generation. As aforementioned, we assume that planes are static and use the plane segmentation results to refine the reprojection error map. The pixels that belong to planes are excluded for the reprojection error map generation.

Fig. 4 shows a sample for the depth data-based plane segmentation, and the reprojection error map with its refined version. Note that in order to display the reprojection error values, they are normalized to $0-255$ in the map figure.

5.1.3. Finding the possible foreground points

Let φ_i^{t-1} denote the likelihood of pixel i being the foreground. We model the likelihood as:

$$\varphi_i^{t-1} = L_i^{t-1} \exp\left[\varepsilon(\xi_i^{t-1}, \xi_i^t; \pi_t^{t-1})\right] \tag{14}$$

where $\exp(\cdot)$ is an exponential function, L_i^{t-1} denotes the label for pixel i from the plane segmentation results, 0 represents that the point is from a plane, and 1 represents non-plane. The pixels with larger reprojection errors give higher likelihoods, and hence more likely to be the foreground.

We normalize the likelihood φ_i^{t-1} to $0-1$ and let $\bar{\varphi}_i^{t-1}$ denote the normalized likelihood, which indicates the probability of a pixel being the foreground. The mask for the possible foreground is determined by the following formula:

$$I_m^{t-1}(i) = 255 \times \mathcal{U}\left[\log \frac{\bar{\varphi}_i^{t-1}}{1-\bar{\varphi}_i^{t-1}}\right] \tag{15}$$

where $I_m^{t-1}(i)$ represents the intensity value of the mask image, $1-\bar{\varphi}_i^{t-1}$ represents the probability of a pixel being the background, $\mathcal{U}(\cdot)$ is the step function where the function value equals 1 if the argument is equal or greater than 0, otherwise, the function value is 0. In (15), the $\log(\cdot)$ function compares the foreground and background probabilities. For example, if the foreground probability is greater than the background probability, the ratio of the two probabilities is greater than 1. Then the $\log(\cdot)$ function gives a positive value. Through the step function $\mathcal{U}(\cdot)$, the pixel value is assigned as 255. The bottom right sub-figure of Fig. 4 displays a sample possible foreground mask. Note that in our experiments, possible foreground masks are refined using erosion operations to reduce noises at object boundaries. Fig. 5 displays the possible foreground viewed in 3-D.

5.1.4. Modeling the foreground

We use the non-parametric codebook model in our approach. A codebook model comprises a set of codewords. A codeword is a type of container that clusters points with close RGB-D values. Moreover, a codeword records the ranges of RGB-D values from the similar points. We only maintain one foreground codebook model in our approach. The model contains a number of codewords, which record the RGB-D value ranges of the moving-object points.

The possible foreground points are used as the seeds for the foreground modeling. Let $\mathcal{P}_{t-1}$ denote the set of possible foreground points at time $t-1$, and $\mathcal{C}$ denote the codebook model. We use the bold font to represent vectors in this section. Each codeword has the following variables:

- $\boldsymbol{\psi}$: the high learning boundary
- $\boldsymbol{\phi}$: the low learning boundary
- $\boldsymbol{\mu}$: the high inference boundary
- $\boldsymbol{\upsilon}$: the low inference boundary
- $\boldsymbol{\delta}$: the offset to create a learning boundary
- τ : the time record for the latest update

Algorithm 1: The foreground modeling algorithm

```
Data: P_{t-1}, C, ψ, φ, μ, υ, δ, τ, ϑ, ϵ, ρ
begin
    for each point p^i_{t-1} in P_{t-1} do
        ϑ = 0;
        // check if p^i_{t-1} fits in a codeword
        for each codeword c^j in C do
            if p^i_{t-1} ⊡ c^j then
                ϑ = 1;
                // update the codeword c^j
                if G(p^i_{t-1}) ⪰ μ(c^j) then
                    μ(c^j) ≐ G(p^i_{t-1}) ;
                end
                if G(p^i_{t-1}) ⪯ υ(c^j) then
                    υ(c^j) ≐ G(p^i_{t-1}) ;
                end
                if G(p^i_{t-1}) ⊕ δ ⪰ ψ(c^j) then
                    ψ(c^j) ≐ ψ(c^j) ⊕ ϵ ;
                end
                if G(p^i_{t-1}) ⊖ δ ⪯ φ(c^j) then
                    φ(c^j) ≐ φ(c^j) ⊖ ϵ ;
                end
                τ(c^j) = t − 1 ;
                break;
            end
        end
        // create a new codeword c
        if ϑ = 0 then
            ψ(c) ≐ G(p^i_{t-1}) ⊕ δ ;
            φ(c) ≐ G(p^i_{t-1}) ⊖ δ ;
            μ(c) ≐ G(p^i_{t-1}) ;
            υ(c) ≐ G(p^i_{t-1}) ;
            τ(c) = t − 1 ;
        end
    end
    // clear inactive codewords
    for each codeword c^i in C do
        if t − 1 − τ(c^i) > ρ then
            delete the codeword ;
        end
    end
end
```

where τ is a scalar that records the time for the latest update of the model, $\boldsymbol{\psi}, \boldsymbol{\phi}, \boldsymbol{\mu}, \boldsymbol{\upsilon}, \boldsymbol{\delta}$ are four-element vectors in which each element corresponds to each channel in the RGB-D data. For each element, we have:

$$\boldsymbol{\phi}_n \le \boldsymbol{\upsilon}_n \le \boldsymbol{\mu}_n \le \boldsymbol{\psi}_n, \quad n \in \{1, 2, 3, 4\} \tag{16}$$

where the subscript $(\cdot)_n$ represents the nth element for the nth channel of the RGB-D data.

The pseudo-code for the foreground modeling is shown in Algorithm 1. The flag ϑ indicates whether there is a codeword from $\mathcal{C}$ to accommodate a tested point. The scalar ρ is a threshold to determine whether the codeword is inactive. For each point p_{t-1}^i in $\mathcal{P}_{t-1}$, we check if the point can be accommodated by a codeword in the codebook model. We use the symbol $\boxdot$ to represent the accommodation. $\mathrm{p}_{t-1}^i \boxdot \mathrm{c}^j$ is established when the following condition is met for all the channels:

$$\boldsymbol{\phi}_n \le \mathcal{G}_n(\mathrm{p}_{t-1}^i) \le \boldsymbol{\psi}_n \tag{17}$$

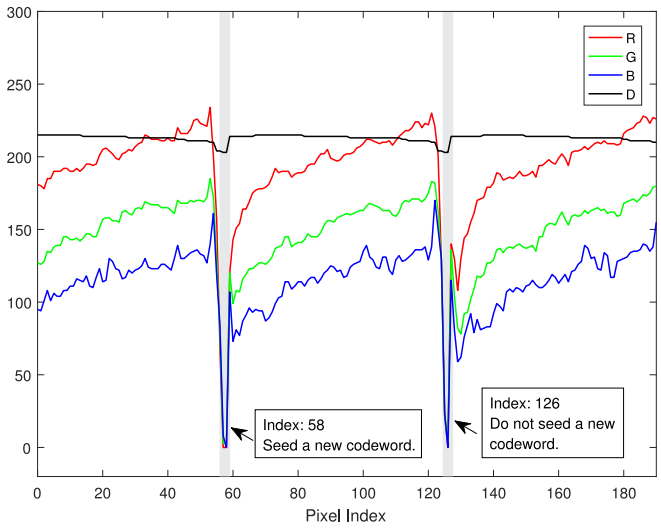


Fig. 6. Plot of the 4-channel RGB-D data from a number of possible foreground points. The depth values are normalized to 0–255. The vertical axis shows the pixel values. The figure is best viewed in color.

where c^j is a codeword in $\mathcal{C}$, $\mathcal{G}_n(\cdot)$ extracts the nth channel value from a point. If $\mathrm{p}^i_{t-1} \boxdot \mathrm{c}^j$ is established, we set the flag ϑ to 1 and update the codeword.

Let the function $\mathcal{G}(\cdot)$ denote the set of $\mathcal{G}_n(\cdot)$, which generates a four-element vector by extracting the 4 channel values from a point. We compare the point values with the boundary variables channel-by-channel. The symbols $\succeq$, $\preceq$, $\oplus$, $\ominus$, $\doteq$ represent the element-wise operations for *greater-than*, *less-than*, *addition*, *minus* and *assignment*.

In Algorithm 1, the statements 7-9 check each channel with the condition:

$$\mathcal{G}_n(\mathrm{p}^i_{t-1}) \succeq \boldsymbol{\mu}_n(\mathrm{c}^j) \tag{18}$$

If a channel value of the point is greater than the corresponding channel value of the boundary variable $\boldsymbol{\mu}(\mathrm{c}^j)$, the channel value $\boldsymbol{\mu}_n(\mathrm{c}^j)$ is updated using:

$$\boldsymbol{\mu}_n(\mathrm{c}^j) \coloneqq \mathcal{G}_n(\mathrm{p}^i_{t-1}) \tag{19}$$

where $\coloneqq$ is the symbol for scalar assignment. Similarly, we check and update the low boundary $\boldsymbol{\upsilon}(\mathrm{c}^j)$ for each channel. The statements 7-12 stretch the inference boundaries $\boldsymbol{\mu}$ and $\boldsymbol{\upsilon}$ to the maximum and the minimum values in the accommodated points. The statements 13-18 slowly adjust the current learning boundaries if the conditions are met. $\boldsymbol{\epsilon}$ is a constant vector with 4 positive-value elements.

If there is no codeword that can accommodate the point p^i_{t-1}, we create a new codeword c. Each channel value of the point is employed as a seed to generate the boundary values.

$$\begin{aligned}
\boldsymbol{\psi}_n(\mathrm{c}) &\coloneqq \mathcal{G}_n(\mathrm{p}^i_{t-1}) + \boldsymbol{\delta}_n \\
\boldsymbol{\phi}_n(\mathrm{c}) &\coloneqq \mathcal{G}_n(\mathrm{p}^i_{t-1}) - \boldsymbol{\delta}_n \\
\boldsymbol{\mu}_n(\mathrm{c}) &\coloneqq \mathcal{G}_n(\mathrm{p}^i_{t-1}) \\
\boldsymbol{\upsilon}_n(\mathrm{c}) &\coloneqq \mathcal{G}_n(\mathrm{p}^i_{t-1})
\end{aligned} \tag{20}$$

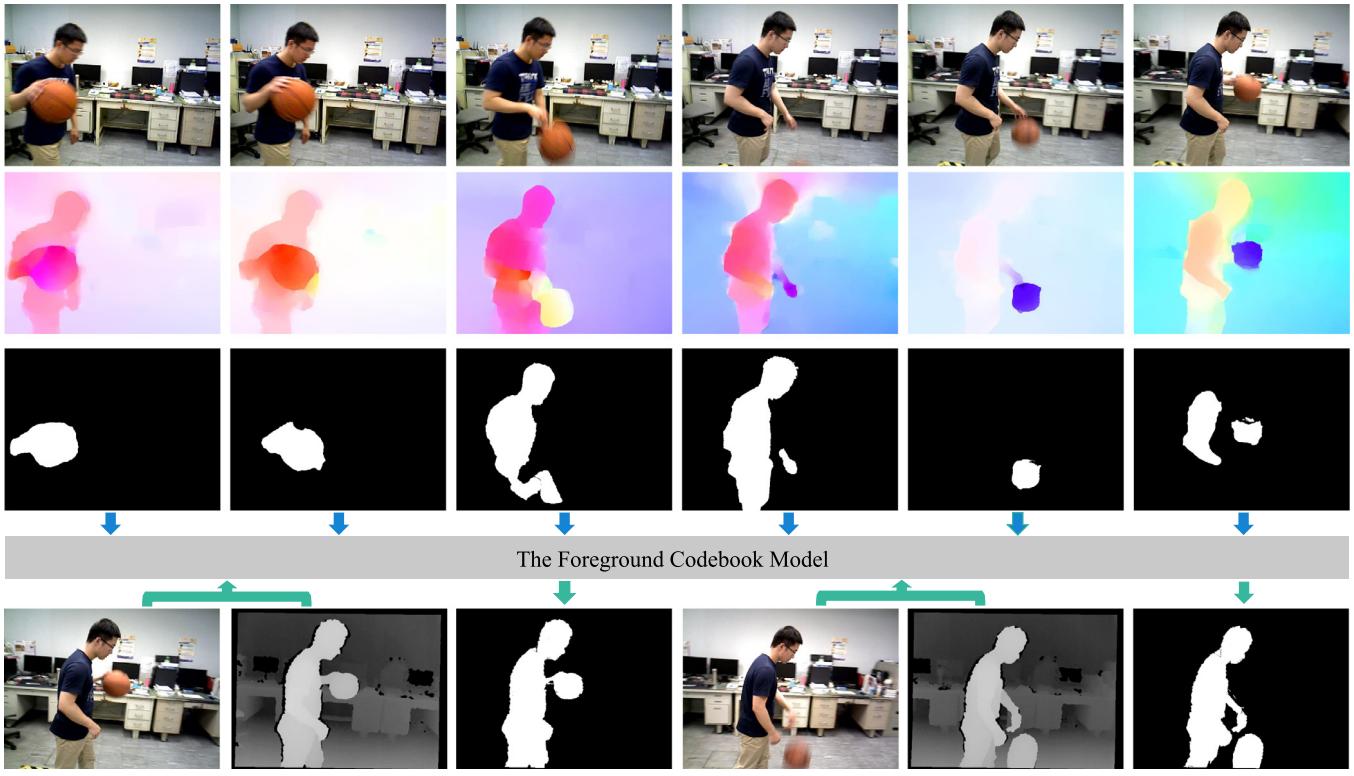


Fig. 7. This figure demonstrates examples for the model update and the motion segmentation. The shown scenario is a person playing with a basketball in an office room. The person and the basketball are the moving objects. The blue and green arrows schematically indicate the directions of the data flows in the *Learning* and *Inference* processes, respectively. Only the key data in the processes are shown in this figure. The 1st row shows the RGB images, which are displayed in chronological order from left to right. The 2nd row shows the corresponding optical flow results computed from two consecutive images. The 3rd row shows the generated mask images for the possible foreground points. The possible foreground points are masked from the mask images and taken as input to update the foreground codebook model. The bottom row demonstrates two examples for the motion segmentation. The input is a pair of registered RGB and depth images, and the output is the foreground mask. The timestamps of the two RGB-D frames are later than those in the 1st row. Because our codebook model is able to accumulate information, we can segment the moving objects correctly even though some motion cues for the model update are not sufficient. The figure is best viewed in color.

Algorithm 2: The foreground Segmentation algorithm

```
Data: P̄_t, C, λ
for each point p_t^i in P̄_t do
    L(p_t^i) = 0 ;
    // check if p_t^i fits in a codeword
    for each codeword c^j in C do
        if p_t^i ⊙ c^j then
            L(p_t^i) = 1 ;
            break ;
        end
    end
end
```

The statements 24-27 implement (20) channel-by-channel. The inference boundaries $\boldsymbol{\mu}(\mathrm{c})$ and $\boldsymbol{\upsilon}(\mathrm{c})$ are gradually expanded as new points are accommodated during the model update. The variable $\boldsymbol{\delta}$ is a constant vector with 4 positive-value elements.

We delete inactive codewords from the codebook model when we finished the model update in each iteration. A codeword is identified as inactive when the following condition is satisfied:

$$t - 1 - \tau(\mathrm{c}^i) > \rho \tag{21}$$

where c^i is the examined codeword, $t-1$ is the time index because we are checking the last frame. The clearance for inactive codewords is able to eliminate moving objects that have disappeared for a certain period of time.

Fig. 6 plots the 4-channel RGB-D data from a number of possible foreground points. Assume that we build and update a codebook model using this sequence of points. We start from the initial point at the index 0. Undoubtedly, the initial point seeds a codeword because there is no codeword existing before. Let we set the values of each channel of $\boldsymbol{\delta}$ to 10. Obviously, the point No. 58 cannot fit in this codeword, so it seeds a new codeword. This new codeword will accommodate the point No. 126 due to the close RGB-D values. Thus, the point No. 126 will not seed a new codeword.

5.2. The Inference process

In the *Inference* process, the foreground is segmented using the codebook model $\mathcal{C}$. Let $\overline{\mathcal{P}}_t$ denote the point set of the whole frame at time t, and $L(\mathrm{p}_t^i)$ denote the binary label for a point p_t^i, where $L(\mathrm{p}_t^i) = 0$ denotes the background and $L(\mathrm{p}_t^i) = 1$ denotes the foreground. The segmentation problem is to determine the label for each point.

The foreground segmentation algorithm is shown in Algorithm 2. We use the symbol $\odot$ to represent that the point p_t^i can be covered by a codeword c^j. If $\mathrm{p}_t^i \odot \mathrm{c}^j$ is true, the point is recognized as a foreground point. We introduce a tolerance variable $\boldsymbol{\lambda}$ to adjust the determination, which is a four-element vector. $\mathrm{p}_t^i \odot \mathrm{c}^j$ is established when the following condition is satisfied for all the channels:

$$\boldsymbol{\upsilon}_n - \boldsymbol{\lambda}_n \le \mathcal{G}_n(\mathrm{p}_t^i) \le \boldsymbol{\mu}_n + \boldsymbol{\lambda}_n \tag{22}$$

If $\boldsymbol{\lambda}_n$ is positive, the segmentation is loosened and more points are classified as foreground. Otherwise, the segmentation is tightened up and less points are classified as foreground.

As aforementioned, we use the plane segmentation results to refine the foreground segmentation. Let $L_p(\overline{\mathcal{P}}_t)$ denote the plane segmentation results, where 0 represents a plane point and 1 otherwise. The refined foreground segmentation $L_r(\overline{\mathcal{P}}_t)$ is obtained point-wisely by:

$$L_r(\overline{\mathcal{P}}_t) = L(\overline{\mathcal{P}}_t) \cap L_p(\overline{\mathcal{P}}_t) \tag{23}$$

Table 1
The characteristics of the tested sequences. Low dynamic sequences are denoted with a superscript star. Others are high-dynamic sequences. Due to the absence of the ground truth for the `fr2/d/person/v` sequence, the speed is not calculated.

Sequences	Duration s	Trans. speed m/s	Rot. speed deg/s
fr3/w/half	35.90	0.2229	18.3213
fr3/w/rpy	30.64	0.0912	21.0986
fr3/w/static	24.93	0.0119	1.3983
fr3/w/xyz	28.82	0.2083	5.5099
fr3/w/half/v	41.22	0.2617	8.5957
fr3/w/rpy/v	27.29	0.0550	18.7280
fr3/w/static/v	27.41	0.0116	1.4223
fr3/w/xyz/v	31.16	0.2208	4.9529
fr3/s/half*	37.29	0.1807	19.1091
fr3/s/xyz*	42.56	0.1322	3.5860
fr2/d/person*	141.64	0.1262	5.4669
fr2/d/person/v*	135.13	–	–

Table 2
The codebook parameters used in our experiments. The parameters $\boldsymbol{\delta}$ and $\boldsymbol{\lambda}$ have 4-channel values for R, G, B and D. The parameter ρ is a scalar.

Parameters	$\boldsymbol{\delta}$				$\boldsymbol{\lambda}$				ρ
	R	G	B	D	R	G	B	D	
Values	10	10	10	5	5	5	5	10	10

where $\cap$ represents the AND logic operation. (23) indicates that only the non-plane points are considered as the foreground.

Fig. 7 shows some examples for the model update and the foreground segmentation. As we can see, the mask images for the possible foreground may not cover the whole moving objects at some frames, for instance, the 5th mask image only cover the basketball. The motion cues here is not sufficient for the model update. However, we updates the foreground model incrementally, so the model is able to accumulate information from previous frames. Thus, we can see that our approach is able to correctly segment the whole foreground although the motion cues are not sufficient at some frames.

6. Experimental results and discussions

6.1. Experiment setup

We performed the experiments using the public TUM RGB-D dataset. In the TUM dataset, the `Dynamic Objects` sequences are designed to evaluate SLAM algorithms in dynamic environments. Among the sequences, the `desk_with_person` and the `sitting` sequences depict low-dynamic scenarios, while the `walking` sequences depict high-dynamic scenarios [31]. Note that there is no moving object in the preceding section of the `desk_with_person` sequences. This part could be considered as recorded in a static environment. The characteristics of the selected sequences in this paper are listed in Table 1. The words `fr`, `w`, `s`, `d`, `half`, `v` stand for `freiburg`, `walking`, `sitting`, `desk`, `halfsphere`, `validation`, respectively. All the images were processed at the 640×480 resolution in our algorithms. The algorithms run on a desktop PC with 3.6 GHz Intel Core i7 CPU and 16 GB RAM.

We left the parameters in the open-source implementation of EpicFlow [49] unchanged. The parameters of our motion removal approach used in the experiments are listed in Table 2. Note that these parameters were chosen empirically and we found that the tuning of these parameters does not influence the motion removal performance dramatically.

In our approach, we use DVO SLAM [7] as the SLAM system. We integrated our motion removal approach into the front end of the DVO SLAM. Our approach acts as a pre-processing stage to filter out the data that were associated with moving objects. We evaluated the original DVO SLAM system and the integrated SLAM system, respectively. The performance improvements brought by our motion removal approach were quantitatively demonstrated.

Table 3
ATE in meters for the results without and with our motion removal approach. Low-dynamic sequences are denoted with a superscript star. Others are high-dynamic sequences. The results demonstrate that our approach greatly reduces the absolute trajectory error over all the high-dynamic sequences.

Sequences	Without our approach				With our approach				Improvements			
	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.
fr3/w/half	0.5287	0.4780	0.4018	0.2260	**0.0668**	0.0613	0.0585	**0.0266**	**87.37%**	87.18%	85.44%	**88.23%**
fr3/w/rpy	0.7304	0.6730	0.6649	0.2837	**0.0729**	0.0647	0.0581	**0.0335**	**90.02%**	90.39%	91.26%	**88.19%**
fr3/w/static	0.2120	0.1628	0.1134	0.1358	**0.0334**	0.0263	0.0191	**0.0207**	**84.25%**	83.85%	83.16%	**84.76%**
fr3/w/xyz	0.5966	0.5334	0.4508	0.2672	**0.0657**	0.0554	0.0453	**0.0354**	**88.99%**	89.61%	89.95%	**86.75%**
fr3/w/half/v	0.3735	0.3142	0.2305	0.2019	**0.1341**	0.1047	0.0857	**0.0838**	**64.10%**	66.68%	62.82%	**58.49%**
fr3/w/rpy/v	0.9115	0.8740	0.8677	0.2588	**0.1195**	0.0896	0.0764	**0.0791**	**86.89%**	89.75%	91.20%	**69.44%**
fr3/w/static/v	0.2016	0.1365	0.0785	0.1485	**0.0197**	0.0182	0.0167	**0.0075**	**90.23%**	86.67%	78.73%	**94.95%**
fr3/w/xyz/v	0.8778	0.7102	0.5067	0.5158	**0.0597**	0.0528	0.0496	**0.0279**	**93.20%**	92.57%	90.21%	**94.59%**
fr3/s/half*	**0.0616**	0.0524	0.0431	**0.0324**	0.0664	0.0540	0.0392	0.0386	**−7.83%**	−3.21%	9.11%	**−19.01%**
fr3/s/xyz*	**0.0505**	0.0393	0.0336	0.0317	0.0514	0.0431	0.0373	**0.0280**	**−1.80%**	−9.80%	−11.12%	**11.81%**
fr2/d/person*	0.0853	0.0834	0.0868	**0.0180**	**0.0759**	0.0692	0.0717	0.0313	**11.00%**	17.05%	17.44%	**−73.96%**
fr2/d/person/v*	0.1468	0.1305	0.1170	0.0672	**0.1202**	0.1075	0.1025	**0.0538**	**18.11%**	17.64%	12.44%	**19.90%**

Table 4
Translational drift (RPE) in m/s for the results without and with our motion removal approach. Low-dynamic sequences are denoted with a superscript star. Others are high-dynamic sequences. The results demonstrate that our approach is able to reduce the translational drift over all the high-dynamic sequences.

Sequences	Without our approach				With our approach				Improvements			
	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.
fr3/w/half	0.3284	0.2074	0.0832	0.2546	**0.0611**	0.0550	0.0514	**0.0268**	**81.38%**	73.50%	38.15%	**89.49%**
fr3/w/rpy	0.4644	0.3357	0.2175	0.3210	**0.0968**	0.0823	0.0681	**0.0510**	**79.15%**	75.48%	68.68%	**84.11%**
fr3/w/static	0.2451	0.1277	0.0231	0.2093	**0.0307**	0.0229	0.0162	**0.0205**	**87.47%**	82.10%	29.68%	**90.19%**
fr3/w/xyz	0.4019	0.2801	0.1640	0.2882	**0.0668**	0.0557	0.0455	**0.0369**	**83.37%**	80.11%	72.28%	**87.18%**
fr3/w/half/v	0.2682	0.1529	0.0532	0.2203	**0.1043**	0.0778	0.0505	**0.0694**	**61.12%**	49.12%	5.06%	**68.49%**
fr3/w/rpy/v	0.3907	0.2303	0.0909	0.3156	**0.1692**	0.1050	0.0598	**0.1327**	**56.68%**	54.40%	34.20%	**57.95%**
fr3/w/static/v	0.1853	0.0955	0.0311	0.1589	**0.0228**	0.0194	0.0172	**0.0119**	**87.71%**	79.68%	44.62%	**92.49%**
fr3/w/xyz/v	0.4614	0.2624	0.0527	0.3795	**0.0567**	0.0485	0.0399	**0.0293**	**87.72%**	81.53%	24.24%	**92.27%**
fr3/s/half*	**0.0466**	0.0346	0.0235	**0.0312**	0.0547	0.0444	0.0347	0.0318	**−17.29%**	−28.34%	−47.35%	**−2.05%**
fr3/s/xyz*	0.0360	0.0245	0.0165	0.0264	**0.0357**	0.0276	0.0218	**0.0225**	**1.06%**	−12.87%	−31.94%	**14.78%**
fr2/d/person*	**0.0147**	0.0116	0.0097	**0.0091**	0.0213	0.0150	0.0117	0.0151	**−44.29%**	−29.94%	−20.60%	**−64.65%**
fr2/d/person/v*	0.0171	0.0145	0.0127	0.0090	**0.0165**	0.0141	0.0122	**0.0085**	**3.53%**	2.90%	4.25%	**5.17%**

Table 5
Rotational drift (RPE) in deg/s for the results without and with our motion removal approach. Low-dynamic sequences are denoted with a superscript star. Others are high-dynamic sequences. The results demonstrate that our approach is able to reduce the rotational drift over all the high-dynamic sequences.

Sequences	Without our approach				With our approach				Improvements			
	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.	RMSE	Mean	Median	S.D.
fr3/w/half	6.6125	4.2811	2.3583	5.0395	**1.9004**	1.7405	1.7156	**0.7629**	**71.26%**	59.34%	27.25%	**84.86%**
fr3/w/rpy	9.0292	6.7447	4.3548	6.0029	**2.5936**	2.2320	1.9845	**1.3210**	**71.28%**	66.91%	54.43%	**77.99%**
fr3/w/static	4.2761	2.2631	0.5148	3.6282	**0.8998**	0.6253	0.3908	**0.6470**	**78.96%**	72.37%	24.08%	**82.17%**
fr3/w/xyz	8.6593	5.5484	3.4288	6.6483	**1.5950**	1.3660	1.1754	**0.8236**	**81.58%**	75.38%	65.72%	**87.61%**
fr3/w/half/v	5.4066	3.4424	1.5721	4.1692	**2.6987**	2.0379	1.4747	**1.7692**	**50.09%**	40.80%	6.20%	**57.56%**
fr3/w/rpy/v	6.7382	4.2855	2.1695	5.1998	**3.0329**	2.3219	1.6276	**1.9512**	**54.99%**	45.82%	24.98%	**62.47%**
fr3/w/static/v	3.2292	1.7556	0.5812	2.7103	**0.6447**	0.5017	0.3955	**0.4050**	**80.03%**	71.42%	31.96%	**85.06%**
fr3/w/xyz/v	8.7977	5.2338	1.4675	7.0716	**1.6194**	1.4211	1.2838	**0.7766**	**81.59%**	72.85%	12.52%	**89.02%**
fr3/s/half*	2.4747	1.8168	1.1745	1.6802	**2.2677**	1.7948	1.2656	**1.3861**	**8.36%**	1.21%	−7.76%	**17.51%**
fr3/s/xyz*	**0.9956**	0.8114	0.6836	0.5769	1.0362	0.8903	0.7904	**0.5302**	**−4.08%**	−9.72%	−15.61%	**8.09%**
fr2/d/person*	**0.5986**	0.4994	0.4384	**0.3300**	0.7744	0.6102	0.5005	0.4767	**−29.37%**	−22.20%	−14.17%	**−44.45%**
fr2/d/person/v*	**0.6271**	0.5406	0.4718	**0.3179**	0.6501	0.5555	0.4736	0.3376	**−3.65%**	−2.76%	−0.38%	**−6.19%**

6.2. Evaluation of our approach

We employed the widely used metrics Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) for the quantitative evaluations [15]. The RPE contains both the translational drift error and the rotational drift error. In Tables 3–5, the columns `Without Our Approach` show the results produced by the original DVO SLAM system. The columns `With Our Approach` show the results produced by the integrated SLAM system. We present the values of Root Mean Square Error (RMSE), Mean Error, Median Error and Standard Deviation (S.D.) in this paper. The improvements brought by our approach are calculated using the following formula:

$$\digamma = (1 - \frac{\beta}{\alpha}) \times 100\% \tag{24}$$

where $\digamma$ represents the improvement value, α represents the value obtained without our approach, β represents the value obtained with our approach. We highlight the RMSE values in the tables, because they are prone to be influenced by large or occasional errors [50]. They can better indicate the robustness of the SLAM system compared to the mean and median values. We also highlight the S.D. values, because they indicate the stability of the system.

According to Tables 3–5, the average RMSE improvement values for the ATE, translational drift and rotational drift in the high-dynamic sequences are 85.63%, 78.08% and 71.22%, respectively. This demonstrates that our approach greatly improves the DVO SLAM system in these high-dynamic scenarios. Moreover, the tables indicate that our approach gives better improvements in the `static` and `xyz` sequences, and inferior improvements in the `half` and `rpy` sequences. As indicated from Table 1, the camera speed of the `half` and `rpy` sequences is generally larger than that of the `static` and `xyz` sequences. We find that our approach could be degraded by the fast camera motions. In the *Learning* process,

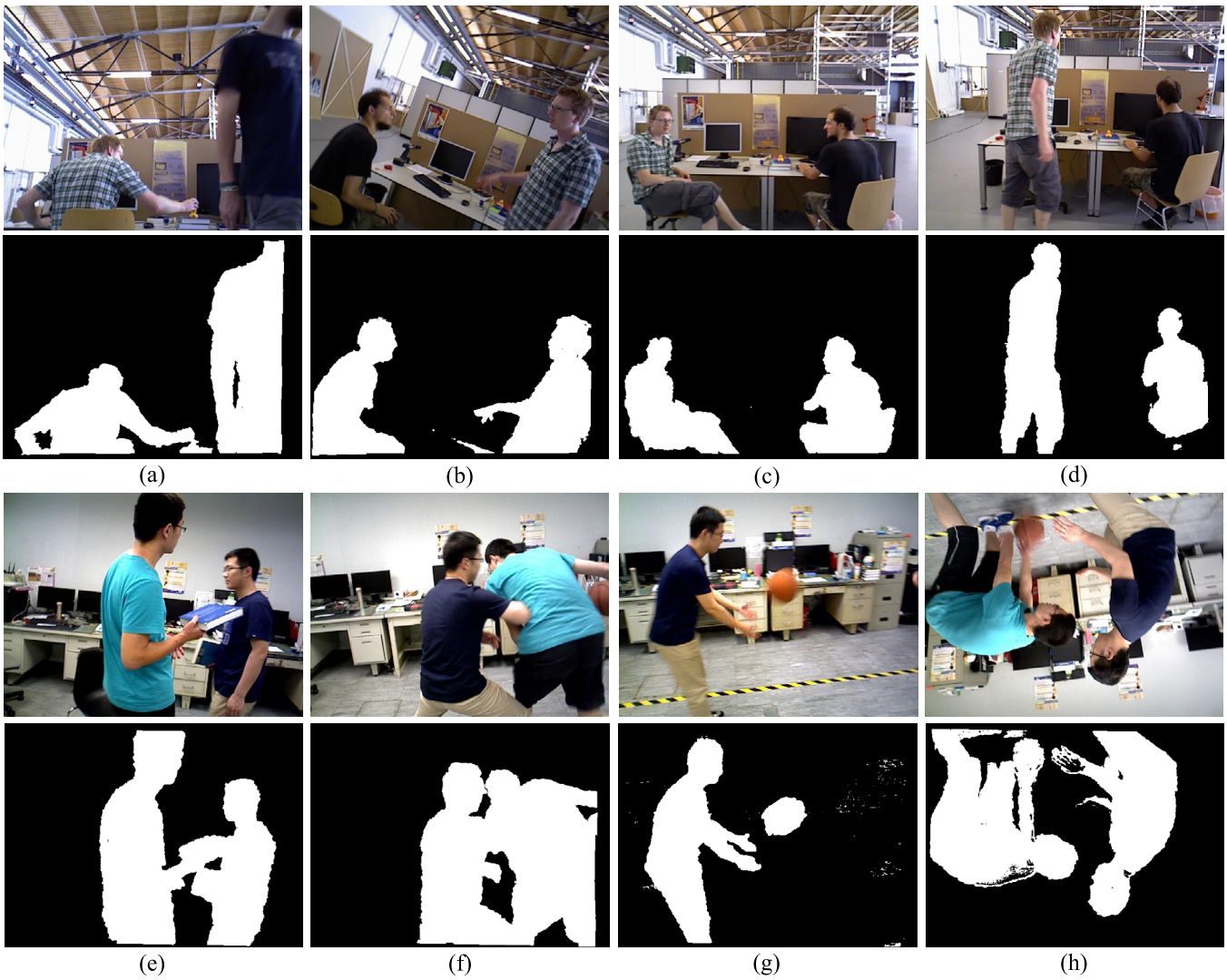


Fig. 8. The selected motion removal results provided by our approach. The sub-figures of the first and the third rows are the figures from the TUM `walking` sequences and our sequences. The mask images show the motion removal results. The figure clearly indicates that our motion removal approach is able to effectively remove the moving objects in various challenging scenarios.

fast camera motions appear to cause large parallax between consecutive images, which may degrade the optical flow estimation. As we use the optical flow results to determine the dense pixel matchings, the large parallax could result in less accurate pixel matchings, with which the homography estimation could be degraded and the reprojection errors would become inaccurate. The inaccurate reprojection errors would lead to incorrect foreground likelihood, and hence false possible foreground points could be generated. With these false points, the foreground model may be contaminated through the model update. Finally, in the *Inference* process, the false foreground segmentation could be produced by using the incorrect foreground model. Thus, in order to get better results, we require that the parallax between consecutive frames is small. We consider it as a limitation of our approach.

From Tables 3–5, we find that our approach gives less improvements in the low-dynamic sequences. We think the reason is that the low-dynamic motions could be easily identified as outliers, so that they can be eliminated from computing the camera poses in the DVO SLAM algorithm. Moreover, as the low-dynamic moving objects do not move from place to place, they can hardly hinder the loop closure detection. Thus, the low-dynamic motions could not significantly degrade the SLAM process. The DVO SLAM system can work well and produce relative good results, which leaves very limited space for our improvement. From the tables, we also find that our approach gives negative results in some low-dynamic sequences. We believe the reason is that the reprojection errors from the static and moving objects do not differ notably in these cases. The static objects may be wrongly identified as the possible foreground points, which contaminates the foreground model and lead to false foreground segmentation results.

We record two persons walking and playing with a basketball in an office room using a hand-held Asus Xtion Pro Live camera. Fig. 8 qualitatively displays the selected motion removal results using both the TUM `walking` sequences and our sequences. The results indicate that our motion removal approach is able to effectively remove the moving objects in various challenging scenarios, such as the results shown in Fig. 8b, e and f. However, there are still some incorrectly labeled pixels. For instance, in Fig. 8h where the camera is almost turning upside down, some false negatives appear at the hand of the right person. We find that the camera overexposures at these pixels, so the color of these pixels cannot be accommodated by the built foreground model. False negatives also appear at the foot of the left person. We think the reason is that the foot is too far away from the main body. Thus, no codeword can cover the depth information of the points at this area. Similar false negatives can be found in Fig. 8c. The RGB and depth values of the points at the hand

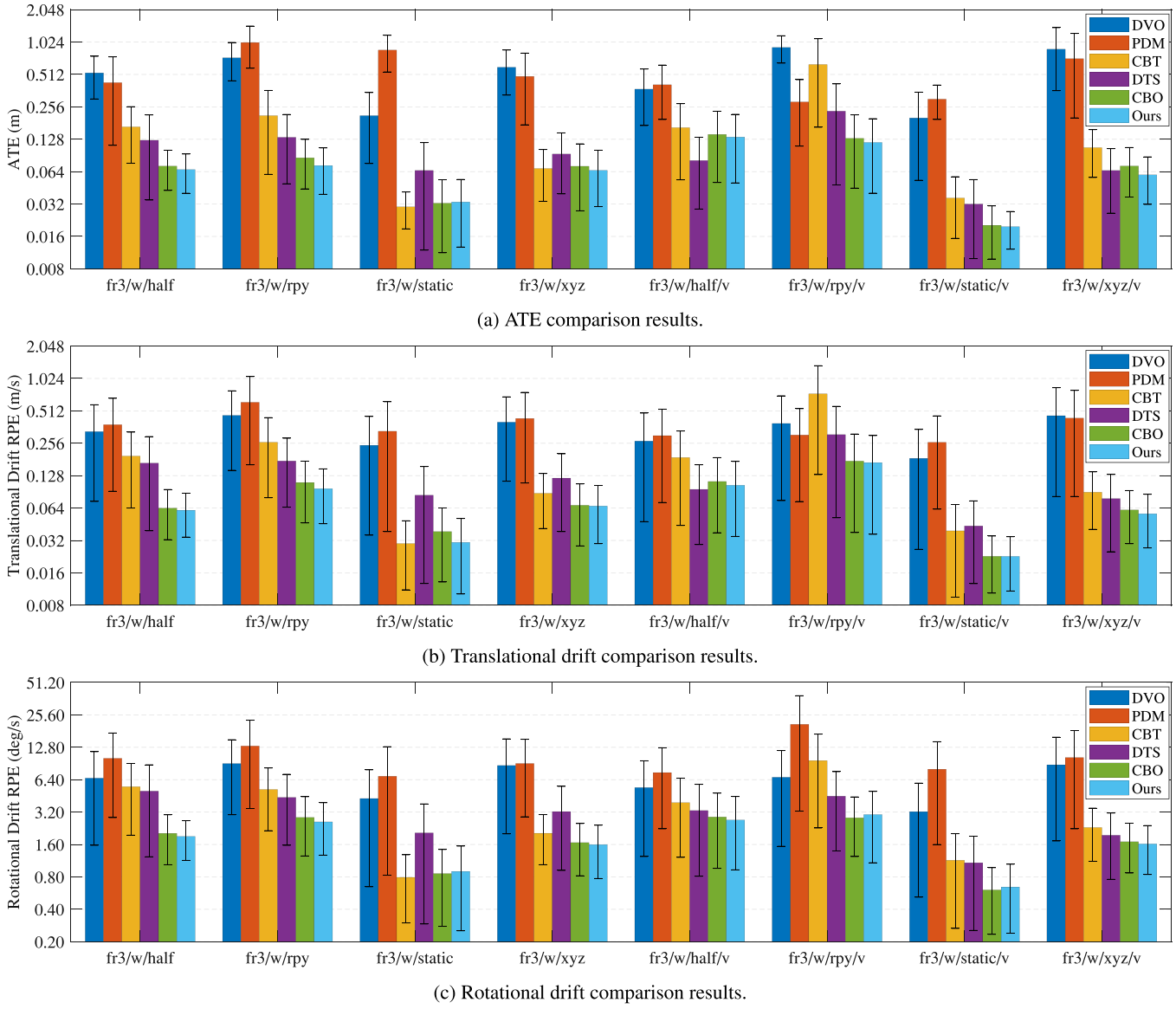


Fig. 9. The comparison between different methods. Experiments were performed using the TUM `walking` sequences. The sub-figures (a), (b) and (c) show the ATE results, the translational drift results, and the rotational drift results, respectively. The vertical axis of each sub-figure is ticked in the logarithm scale. The results presented in the error bars are RMSE values with standard deviations. Small values correspond to better performance. The optimal performance of our proposed approach is clearly revealed by the comparison. This figure is best viewed in color.

of the right person is far away from the main body so that these pixels are misclassified as background. There are also some false positives, for instance, the pixels at the book and desk in Fig. 8a and b. We believe that these false positives are caused by the close RGB and depth values between the foreground and the background. In Fig. 8g, we find that there are some salt-and-pepper noises. We think these are caused by the depth measurement noises from the RGB-D camera.

6.3. Method comparison

We implemented two variants of our approach: one merely using the 3-channel RGB data, and the other one merely using the 1-channel depth data. The two variants are named as CBT (CodeBook Three channel) and CBO (CodeBook One channel), respectively. We compare both of them with our approach in this section. To ensure a fair comparison, we use the same set of parameters listed in Table 2 for both the two variants. The CBT and CBO approaches are identical to our approach except how many channels are used.

Scene flow is able to infer motions in the 3-D Euclidean space [51]. Herbst et al. [52] developed a dense motion segmentation method with the proposed dense RGB-D scene flow algorithm. We implemented this motion removal method by replacing the scene flow algorithm with PD-flow [53], which is a real-time RGB-D flow algorithm. The method is named as PDM (PD-flow-based Motion removal) and compared with our approach. According to [52], motions are indicated by the RANSAC outliers of the point correspondences associated by the scene flow field between consecutive RGB-D frames. In this paper, we used the open-source RANSAC-based point correspondence rejector in the Point Cloud Library to determine the outliers.

Note that all these motion removal methods were integrated with the DVO SLAM system individually. What we compare are the integrated DVO SLAM systems. They are evaluated using the TUM `walking` sequences with the ATE and RPE metrics. Smaller errors correspond to better performance. We also compare all the integrated SLAM systems with our most similar work [31], which

employs a motion Detection, Tracking and Segmentation (DTS) framework.

Fig. 9 displays the comparison results. Note that the word DVO denotes the results obtained with the original DVO SLAM system. The comparison results show that our approach ensures the smallest ATE and RPE values across almost all the sequences, which clearly demonstrates that our approach gives the best performance. Among all the other methods, CBO gives the closest performance to ours in most sequences. However, the performance of CBT is not satisfied, especially in the sequences with fast camera motions. This indicates that the depth information contributes significantly in our approach and combining both the RGB and depth information is a benefit here. The DTS method can also give satisfied performance. It is better than CBT in most sequences. We think this is because the DTS method uses both the RGB and the depth information while CBT just uses the RGB information.

We also find that the PDM method degrades the SLAM and visual odometry performance in most cases. The reason is that the PDM method could not correctly segment moving objects. Many false detected motions come from the object boundaries and distant objects. The point correspondences become less accurate due to the unstable depth measurements at these areas. These imprecise point correspondences dominate the outliers of the RANSAC output and become the false motion removal results that finally degrade the SLAM performance. This also confirms that just using robust estimators, such as the RANSAC algorithm, to reject outliers in 3-D registration is not sufficient. Development of effective motion removal methods is necessary to solve the RGB-D SLAM problem in dynamic environments.

7. Conclusions

We proposed here a novel RGB-D data-based motion removal approach to address the problem of RGB-D SLAM in dynamic environments. Our approach requires no prior-known moving-object information, such as semantics or visual appearance. It is an on-line method and relies solely on the information obtained until the current frame. No future information or batch data processing is required. In addition, the on-line learning capability allows our approach accumulating the foreground information incrementally and updating the foreground model timely. Extensive experiments were performed using the public TUM RGB-D dataset. The results indicated that our approach was able to improve the RGB-D SLAM performance in various challenging environments, especially in the high-dynamic scenarios. The comparison results demonstrated that our approach achieved the best performance among all the methods.

However, our approach still presents several limitations. Firstly, we require small parallaxes between consecutive frames. The reason is that the optical flow estimation could be degraded by the large parallax. With the degraded optical flow, reprojection would become erroneous and the foreground model could be contaminated. Then, false foreground segmentation would be generated. In the future, we plan to use an RGB-D camera with a higher frame rate to alleviate this problem. Secondly, we assume that static objects dominate the scene. In particular, the ratio of the outlier pixel matchings should not exceed 50%. This is required by the LMedS robust estimator that is employed to optimize the homography estimation. The algorithm would be confused to discriminate the inliers against the outliers if the ratio is more than 50%. For instance, the algorithm would fail in the scenarios that more than half of the pixels come from moving objects. Thirdly, our approach may degrade the performance of RGB-D SLAM in low-dynamic environments. This is because there may be not sufficient differences between the reprojection errors from the static objects and the dynamic objects. Thus, static objects may be identified as the possible foreground and used to update the foreground model. Lastly, our approach cannot run real-timely at the current status. The most time-consuming process is the dense optical flow estimation, which takes around 7 s per frame, while the core operations in our algorithm just take around 1 s per frame. In the future, we will try fast dense optical flow algorithms and accelerate the approach with FPGA devices.

Acknowledgments

Research presented in this paper was partially supported by the Hong Kong RGC GRF grant #14205914 and #14200618, ITC ITF grant #ITS/236/15, and Shenzhen Science and Technology Innovation project JCYJ20170413161616163 awarded to Max Q.-H. Meng, and partially supported by the Research Grant Council of Hong Kong SAR Government, China, under Project No. 11210017 and No. 16212815 and No. 21202816, the National Natural Science Foundation of China (Grant No. U1713211) awarded to Prof. Ming Liu. The authors would like to thank Jiyu Cheng and Jiankun Wang for acting as moving objects in the experiment.

References

[1] G. Younes, D. Asmar, E. Shammas, J. Zelek, Keyframe-based monocular SLAM: design, survey, and future directions, Robot. Auton. Syst. 98 (2017) 67–88.
[2] R. Gomez-Ojeda, F.A. Moreno, D. Scaramuzza, J. Gonzalez-Jimenez, PL-SLAM: a stereo SLAM system through the combination of points and line seegments, arXiv preprint arXiv:1705.09479, 2017.
[3] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J.J. Leonard, J. McDonald, Real-time large-scale dense RGB-D SLAM with volumetric fusion, Int. J. Rob. Res. 34 (4–5) (2015) 598–626.
[4] E. Marchand, H. Uchiyama, F. Spindler, Pose estimation for augmented reality: a hands-on survey, IEEE Trans. Vis. Comput. Graphics PP (99) (2016) 1–1.
[5] S. Song, M. Chandraker, C.C. Guest, High accuracy monocular sfm and scale correction for autonomous driving, IEEE Trans. Pattern Anal. Mach. Intell. 38 (4) (2016) 730–743.
[6] L. Shao, J. Han, D. Xu, J. Shotton, Computer vision for RGB-D sensors: Kinect and its applications [special issue intro.], IEEE Trans. Cybern. 43 (5) (2013) 1314–1317.
[7] C. Kerl, J. Sturm, D. Cremers, Dense visual SLAM for RGB-D cameras, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 2100–2106.
[8] F. Endres, J. Hess, J. Sturm, D. Cremers, W. Burgard, 3-D mapping with an RGB-D camera, IEEE Trans. Robot. 30 (1) (2014) 177–187.
[9] C. Kerl, J. Stuckler, D. Cremers, Dense continuous-time tracking and mapping with rolling shutter RGB-D cameras, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2264–2272.
[10] M. Labbé, F. Michaud, Online global loop closure detection for large-scale multi-session graph-based slam, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2014, pp. 2661–2666.
[11] R. Mur-Artal, J.D. Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, IEEE Trans. Robot. 33 (5) (2017) 1255–1262.
[12] https://ieeexplore.ieee.org/document/8206375/.
[13] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J.J. Leonard, Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age, IEEE Trans. Robot. 32 (6) (2016) 1309–1332.
[14] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395.
[15] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, A benchmark for the evaluation of RGB-D SLAM systems, in: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, IEEE, 2012, pp. 573–580.
[16] K.-H. Lin, C.-C. Wang, Stereo-based simultaneous localization, mapping and moving object tracking, in: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, IEEE, 2010, pp. 3975–3980.
[17] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, H. Durrant-Whyte, Simultaneous localization, mapping and moving object tracking, Int. J. Robot. Res. 26 (9) (2007) 889–916.
[18] K.E. Ozden, K. Schindler, L. Van Gool, Multibody structure-from-motion in practice, IEEE Trans. Pattern Anal. Mach. Intell. 32 (6) (2010) 1134–1141.
[19] D. Zou, P. Tan, Coslam: Collaborative visual slam in dynamic environments, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2) (2013) 354–366.
[20] W. Tan, H. Liu, Z. Dong, G. Zhang, H. Bao, Robust monocular SLAM in dynamic environments, in: Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on, IEEE, 2013, pp. 209–218.

[21] Y. Wang, S. Huang, Motion segmentation based robust RGB-D SLAM, in: Intelligent Control and Automation (WCICA), 2014 11th World Congress on, IEEE, 2014, pp. 3122–3127.
[22] Y. Wang, S. Huang, Towards dense moving object segmentation based robust dense RGB-D SLAM in dynamic scenarios, in: 2014 13th International Conference on Control Automation Robotics Vision (ICARCV), 2014, pp. 1841–1846.
[23] P. Ochs, J. Malik, T. Brox, Segmentation of moving objects by long term video analysis, IEEE Trans. Pattern Anal. Mach. Intell. 36 (6) (2014) 1187–1200.
[24] C. Jiang, D.P. Paudel, Y. Fougerolle, D. Fofi, C. Demonceaux, Static-map and dynamic object reconstruction in outdoor scenes using 3-d motion segmentation, IEEE Rob. Autom. Lett. 1 (1) (2016) 324–331.
[25] E. Elhamifar, R. Vidal, Sparse subspace clustering: Algorithm, theory, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2765–2781.
[26] L. Sevilla-Lara, D. Sun, V. Jampani, M.J. Black, Optical flow with semantic segmentation and localized layers, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 3889–3898.
[27] V. Vineet, O. Miksik, M. Lidegaard, M. Nießner, S. Golodetz, V.A. Prisacariu, O. Kähler, D.W. Murray, S. Izadi, P. Pérez, P.H.S. Torr, Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 75–82.
[28] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohi, J. Shotton, S. Hodges, A. Fitzgibbon, KinectFusion: Real-time dense surface mapping and tracking, in: 2011 10th IEEE International Symposium on Mixed and Augmented Reality, 2011, pp. 127–136.
[29] C. Jiang, P.D. Paudel, Y. Fougerolle, D. Fofi, C. Demonceaux, Static and dynamic objects analysis as a 3D vector field, in: International Conference on 3D Vision (3DV), Qingdao, China, 2017.
[30] J. Zhang, S. Singh, Low-drift and real-time lidar odometry and mapping, Auton. Robots 41 (2) (2017) 401–416.
[31] Y. Sun, M. Liu, M.Q.H. Meng, Improving RGB-D SLAM in dynamic environments: A motion removal approach, Robotics Auton. Syst. 89 (2017) 110–122.
[32] K. Khosoussi, S. Huang, G. Dissanayake, Good, bad and ugly graphs for SLAM, in: RSS Workshop on the Problem of Mobile Sensors, 2015.
[33] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, F. Dellaert, iSAM2: Incremental smoothing and mapping using the Bayes tree, Int. J. Robot. Res. (2011) 0278364911430419.
[34] V. Ila, J.M. Porta, J. Andrade-Cetto, Information-based compact Pose SLAM, IEEE Trans. Robot. 26 (1) (2010) 78–93.
[35] T. Kohonen, Learning vector quantization, in: Self-Organizing Maps, Springer, 1995, pp. 175–189.
[36] B.D. Ripley, Pattern Recognition and Neural Networks, Cambridge university press, 2007.
[37] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, " O'Reilly Media, Inc.", 2008.
[38] K. Kim, T.H. Chalidabhongse, D. Harwood, L. Davis, Real-time foreground–background segmentation using codebook model, Real-Time Imaging 11 (3) (2005) 172–185.
[39] F. Servant, E. Marchand, P. Houlier, I. Marchal, Visual planes-based simultaneous localization and model refinement for augmented reality, in: Pattern Recognition, 2008. ICPR 2008. 19th International Conference on, IEEE, 2008, pp. 1–4.
[40] J. Martínez-Carranza, A. Calway, Unifying planar and point mapping in monocular SLAM, in: BMVC, Citeseer, 2010, pp. 1–11.
[41] Y. Taguchi, Y.D. Jian, S. Ramalingam, C. Feng, Point-plane SLAM for hand-held 3D sensors, in: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, 2013, pp. 5182–5189.
[42] E. Fernández-Moral, W. Mayol-Cuevas, V. Arévalo, J. González-Jiménez, Fast place recognition with plane-based maps, in: 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 2719–2724.
[43] R.F. Salas-Moreno, B. Glocken, P.H. Kelly, A.J. Davison, Dense planar slam, in: Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on, IEEE, 2014, pp. 157–164.
[44] D. Holz, S. Holzer, R.B. Rusu, S. Behnke, Real-time plane segmentation using RGB-D cameras, in: Robot Soccer World Cup, Springer, 2011, pp. 306–317.
[45] R.B. Rusu, S. Cousins, 3D is here: Point cloud library (PCL), in: 2011 IEEE International Conference on Robotics and Automation, 2011, pp. 1–4.
[46] J. Revaud, P. Weinzaepfel, Z. Harchaoui, C. Schmid, EpicFlow: Edge-preserving interpolation of correspondences for optical flow, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1164–1172.
[47] P.J. Rousseeuw, Least median of squares regression, J. Amer. Statist. Assoc. 79 (388) (1984) 871–880.
[48] Z. Zhang, Parameter estimation techniques: A tutorial with application to conic fitting, Image Vis. Comput. 15 (1) (1997) 59–76.
[49] EpicFlow Website [Online]. Available: http://lear.inrialpes.fr/src/epicflow/.
[50] C. Kerl, J. Sturm, D. Cremers, Robust odometry estimation for RGB-D cameras, in: Robotics and Automation (ICRA), 2013 IEEE International Conference on, IEEE, 2013, pp. 3748–3754.
[51] A. Letouzey, B. Petit, E. Boyer, Scene flow from depth and color images, in: J. Hoey, S. McKenna, E. Trucco (Eds.), BMVC 2011 - British Machine Vision Conference, in: Proceedings of the British Machine Vision Conference, BMVA Press, Dundee, United Kingdom, 2011, pp. 46:1–11.
[52] E. Herbst, X. Ren, D. Fox, RGB-D flow: Dense 3-D motion estimation using color and depth, in: 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 2276–2282.
[53] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, D. Cremers, A primal-dual framework for real-time dense RGB-D scene flow, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 98–104.

Yuxiang Sun received his Ph.D. degree from The Chinese University of Hong Kong (CUHK), Hong Kong, China, in 2017, the master's degree from University of Science and Technology of China (USTC), Hefei, China, in 2012, and the bachelor's degree from Hefei University of Technology (HFUT), Hefei, China, in 2009. He is now a research associate at the Robotics Institute, Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology (HKUST), Hong Kong, China. His current research interests include visual SLAM, autonomous vehicles, motion detection, deep learning, etc. He is a recipient of the Best Student Paper Finalist Award at the IEEE ROBIO 2015.

Ming Liu received the BA degree in Automation at Tongji University in 2005. During his master study at Tongji University, he stayed one year in Erlangen-Nünberg University and Fraunhofer Institute IISB, Germany, as a master visiting scholar. He graduated as a Ph.D. student from the Department of Mechanical and Process Engineering of ETH Zürich in 2013, supervised by Prof Roland Siegwart. He is now affiliated with ECE department, CSE department and Robotics Institute of Hong Kong University of Science and Technology. He is a founding member of Shanghai SWing Automation Ltd. Co. He is also coordinating and involved in NSF projects, and National 863-Hi-Tech-Plan projects in China. As a team member, he won the second place of EMAV'09 (European Micro Aerial Vehicle Competition) and two awards from IARC'14 (International Aerial Robot Competition). He won the Best Student Paper Award as the first author for MFI 2012 (IEEE International Conference on Multisensor Fusion and Information Integration), the Best Paper Award in Information for ICIA 2013 (IEEE International Conference on Information and Automation) as the first author and the Best Paper Award Finalists as a co-author, the Best RoboCup Paper Award for IROS 2013 (IEEE/RSJ International Conference on Intelligent Robots and Systems), the Best Conference Paper Award for IEEE-CYBER 2015, the Best Student Paper Finalist for RCAR 2015 (IEEE International conference on Real-time Computing and Robotics), the Best Student Paper Finalist for ROBIO 2015, the Best Student Paper Award for IEEE-ICAR 2017 and the Best Paper in Automation Award for IEEE-ICIA 2017. He won twice the innovation contest Chunhui Cup Winning Award in 2012 and 2013. He won the Wu Weijun AI award in 2016. He was the Program Chair of IEEE-RCAR 2016; the Program Chair of International Robotics Conference in Foshan 2017; He is the Conference Chair of ICVS 2017. Ming Liu's research interests include dynamic environment modeling, deep-learning for robotics, 3D mapping, machine learning and visual control.

Max Q.-H. Meng received his Ph.D. degree in Electrical and Computer Engineering from the University of Victoria, Canada, in 1992. He joined the Chinese University of Hong Kong in 2001 and is currently Professor and Chairman of Department of Electronic Engineering. He was with the Department of Electrical and Computer Engineering at the University of Alberta in Canada, serving as the Director of the Advanced Robotics and Teleoperation Lab and holding the positions of Assistant Professor (1994), Associate Professor (1998), and Professor (2000), respectively. He is affiliated with the State Key Laboratory of Robotics and Systems at Harbin Institute of Technology and the Honorary Dean of the School of Control Science and Engineering at Shandong University, in China. His research interests include robotics, medical robotics and devices, perception, and scenario intelligence. He has published some 600 journal and conference papers and led more than 50 funded research projects to completion as PI. He has served as an editor of several journals and General and Program Chair of many conferences including General Chair of IROS 2005 and General Chair of ICRA 2021 to be held in Xi'an, China. He is an elected member of the Administrative Committee (AdCom) of the IEEE Robotics and Automation Society. He is a recipient of the IEEE Millennium Medal, a Fellow of the Canadian Academy of Engineering, a Fellow of HKIE, and a Fellow of IEEE.