Robotics and Autonomous Systems 89 (2017) 110–122

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

Improving RGB-D SLAM in dynamic environments: A motion removal approach

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

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

HIGHLIGHTS

- We propose a motion removal approach with a freely moving RGB-D camera.
- Comparative results with and without motion removal using TUM dataset are given.
- The proposed motion removal approach benefits RGB-D SLAM in dynamic environments.

ARTICLE INFO

Article history:
Available online 21 November 2016

Keywords:
RGB-D SLAM
Dynamic environments
RANSAC
Motion removal

ABSTRACT

Visual Simultaneous Localization and Mapping (SLAM) based on RGB-D data has developed as a fundamental approach for robot perception over the past decades. There is an extensive literature regarding RGB-D SLAM and its applications. However, most of existing RGB-D SLAM methods assume that the traversed environments are static during the SLAM process. This is because moving objects in dynamic environments can severely degrade the SLAM performance. The static world assumption limits the applications of RGB-D SLAM in dynamic environments. In order to address this problem, we proposed a novel RGB-D data-based motion removal approach and integrated it into the front end of RGB-D SLAM. The motion removal approach acted as a pre-processing stage to filter out data that were associated with moving objects. We conducted experiments using a public RGB-D dataset. The results demonstrated that the proposed motion removal approach was able to effectively improve RGB-D SLAM in various challenging dynamic environments.

1. Introduction

Simultaneous Localization and Mapping (SLAM) has developed as a fundamental capability for robots over the past decades [1]. Lots of impressive SLAM systems have been developed and open-sourced [2]. However, almost all the theories and implementations of current SLAM approaches are built on the static world assumption. It requires that objects in the traversed environments must remain still during the SLAM process [3]. However, this assumption is usually not true in real world environments, because moving objects are ubiquitous and unavoidable in most cases. They can corrupt the odometry estimation and cause spurious objects recorded on the resulting map, which makes the map useless for further applications. A solution to solve this problem is to equip robots with sophisticated proprioceptive sensors [4], such as precise inertial systems. Robot poses can be reliably estimated by fusing information from the sensors [5]. Moving objects can be inferred by ego-motion compensation with the reliable odometry estimation. However, precise proprioceptive sensors are usually not cost effective. In most visual applications, cameras are usually the only available sensors. Therefore, the key challenge to address the RGB-D data-based Visual SLAM problem in dynamic environments is how to eliminate the negative effects caused by moving objects merely using visual sensors.

The advent of RGB-D cameras provides a low-cost solution to sense the 3-D world in point clouds [6]. RGB-D cameras, such as Kinect [7], have become standard equipments for robots today. Many effective 3-D SLAM algorithms have been proposed using RGB-D cameras [8–12]. Impressive SLAM results in static environments were reported. In most RGB-D SLAM systems, moving objects are usually implicitly tackled to increase the robustness of the systems. For instance, SLAM methods that use Iterative Closest Point (ICP) [13] algorithm for point-cloud registration normally adopt a two-step framework like the mechanism in [14]. They integrate the widely used robust estimator Random Sample Consensus (RANSAC) [15] algorithm into estimation pipeline. In the first step,

* Corresponding author.
E-mail addresses: yxsun@ee.cuhk.edu.hk (Y. Sun), eelium@ust.hk (M. Liu), qhmeng@ee.cuhk.edu.hk (M. Meng).

http://dx.doi.org/10.1016/j.robot.2016.11.012
0921-8890/© 2016 Elsevier B.V. All rights reserved.

RANSAC is used to roughly estimate the 3-D transformation by finding an optimum consensus model. In the second step, ICP is employed to refine the transformation with the initial guess from the first step. This framework increases the robustness of RGB-D SLAM. However, RGB-D cameras normally have a narrow Field-of-View (FOV) compared with laser range finders. When moving objects are not trivial in FOV, just rejecting outliers using the above mentioned framework is not sufficient to eliminate the negative effects caused by moving objects. Note that the outliers here mean the feature associations that are not consistent with the estimated transformation. In dynamic environments, the outliers normally come from moving objects. The wrongly detected inliers are often sufficient to corrupt the pose estimation. In addition, moving objects are prone to be recorded in resulting maps. When robots return to the place where the moving objects have disappeared, the loop detection can be severely confused.

To address this problem, we proposed to densely segment moving objects from image frames. We define the dense moving-object segmentation as *motion removal* in this paper. A novel motion removal approach based on RGB-D data was proposed. To validate our idea, we incorporated the proposed motion removal approach into the front end of an RGB-D SLAM system. The proposed approach acted as a pre-processing stage to filter out data that were associated with moving objects. We tested our approach using a public RGB-D dataset. The experimental results demonstrated that our approach was able to improve the RGB-D SLAM performance effectively. Note that part of this paper had been published in [16]. This paper is an extended version to solve the RGB-D SLAM problem in dynamic environments. The main contributions of this paper are summarized as follows:

- A novel motion removal approach was proposed. The approach was on-line and required only an RGB-D camera as the sensor. No human intervention at any stage was needed.
- Vector quantized depth images were adopted for motion segmentation, by which the benefits were demonstrated empirically.
- Experiments were performed using a public RGB-D dataset. The results demonstrated the effectiveness of our approach to improve RGB-D SLAM in various dynamic environments.

The remainder of this paper is organized as follows. Section 2 presents a review for related work. Section 3 explains why we use motion removal to improve the RGB-D SLAM performance. Section 4 presents the details of our proposed approach. Section 5 discusses the experimental results. Conclusions and future works are drawn in the last section.

2. Related work

We use motion removal to solve the RGB-D SLAM problem in dynamic environments. However, motion removal for images captured by moving cameras is an ill-posed problem in computer vision. Classical motion segmentation methods, such as background subtraction [17], become virtually useless when camera is not static. This is because both the foreground and the background are moving in images at the same time. Over the past years, lots of motion removal methods have been proposed. The fundamental challenge of motion removal is to disambiguate the motions induced by camera ego-motion and the motions caused by independently moving objects. It is hard to address this challenge without prior cues or assumptions. This section reviews selected motion removal methods which have been reported in recent literature.

We generally divide the motion removal approaches into two categories: the off-line approaches and the on-line approaches. Off-line refers to start processing after receiving the whole data. They cannot produce motion removal results immediately in response to each input image. Decisions must be made with knowledge of the future information [18]. Lots of early motion removal approaches adopt the off-line strategy. In [19], the method estimated a set of trajectories by tracking sparse salient points across video frames. The approach discriminated background trajectories and foreground trajectories using RANSAC algorithm with geometric constraints. Background and foreground appearance models were built with classified sparse points. Pixel-wise labeling was obtained using the built models. In [20], dense pixel correspondences were found using dense optical flow. Pixel-wise dense trajectories across a number of frames were computed using particle advection. The authors proposed a multi-frame epipolar constraint. With RANSAC algorithm, the trajectories which violated the constraint were determined as outliers. Moving objects were segmented according to the classified dense trajectories. Liu et al. proposed a learning-based motion removal approach [21]. Pixel-wise motion likelihoods were obtained by subtracting optical flow values with global ego-motion estimated by image homography. They selected a set of key frames through the whole video. The selection criterion is based on whether a frame can cover at least a part of moving objects. Foreground appearance model was learned from motion cues in the key frames. They employed the Gaussian Mixture Model (GMM) technique to build the foreground model. The foreground was segmented with the built GMM model. Zamalieva et al. estimated a temporal fundamental matrix from a number of consecutive frames [22]. Different from traditional fundamental matrix estimation method, they estimated the temporal fundamental matrix using tracklets calculated from dense optical flow. Moving objects can be determined by how well the corresponding tracklet fits the estimated scene geometry.

In contrast to off-line methods, on-line methods do not require future information. In RGB-D SLAM, moving objects are supposed to be removed in the SLAM front end before data associations. On-line methods can segment moving objects simultaneously with the SLAM process, which prevents moving objects hindering the data associations in SLAM. Thus, on-line methods are more suitable for SLAM applications. In [23], an on-line motion removal approach for Visual SLAM in dynamic environments was proposed. It segmented moving objects using multiple geometric constraints and dense optical flow algorithm. Motion removal was integrated into a Visual SLAM system. The reported results demonstrated that the SLAM performance was effectively improved. In [24], motion cues were obtained in the first frame by examining the dense optical flow with the epiploar constraint. The initial segmentation was propagated as a seed for following frames. The authors divided an image into a number of equal-size blocks. Background and foreground appearances were learned block-wisely in kernel density models with the propagated motion segmentation result. The motion segmentation and the model propagation were repeated iteratively. In [25], motion removal was realized real-timely on resource limited devices. The authors adopted the block technique. Each block was described using dual-mode Single Gaussian Model (SGM) with age. One SGM severed as an apparent model and the other one severed as a candidate model. The dual-mode SGM was different from the GMM with two models. The dual-model SGM provided two containers to receive data, which avoids foreground points contaminating the real background model. The two SGM models were swapped when the age of one model was greater than that of the other model. Blocks were mixed with propagation using ego-motion computed from homography. In [26], an RGB-D data-based motion removal approach was proposed. A large number of segmentation cues were combined to construct a Conditional Random Field (CRF) model. The segmentation cues included optical flow, visual appearance, color and depth discontinuities, etc. The training process of the method was to determine the weights for each cue in the energy function. The method assumed that an initial hand-labeled segmentation was given at the first iteration. The CRF segmentation result of the current frame served as the motion likelihood for the CRF model in the next frame. Moving objects were incrementally segmented from each frame using the propagation scheme. In [27], motion cues were obtained by examining super-pixel changes in consecutive frames. The authors observed that super-pixel changes usually happen on moving objects. They propagated super-pixels from current frame back to the last frame. The super-pixel with the largest overlap in the last frame was employed to calculate the Jaccard distance with the propagated super-pixel. An adaptive threshold was employed to determine whether the propagated super-pixel belongs to moving objects by the calculated Jaccard distance. Foreground and background appearance models were built using Mixture of Gaussians (MOG) technique with the classified super-pixels. Motion segmentation was further optimized using a graph-cut framework. The mostly relevant work to ours presented here is [28]. The authors improved the motion removal method developed in [29], and integrated it into an RGB-D SLAM system. They performed experiments on the public TUM RGB-D dataset [30]. The results demonstrated the effectiveness to improve the RGB-D SLAM performance.

3. Problem statement

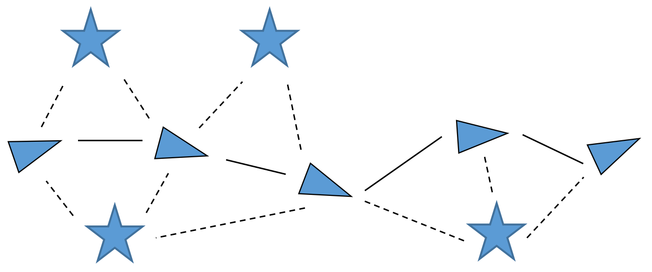
In this section, we briefly introduce the graph SLAM [31] and explain why we use motion removal to improve the SLAM performance. We illustrate the problem using the pose graph framework, because most of the current RGB-D SLAM systems adopt this framework. In addition, it is natural to express our idea in the structure of pose graph. Fig. 1 describes the pose graph in a dynamic environment. The edges linking robot poses constrain the motion model, while the edges linking robot poses and landmarks constrain the measurement model. The graph SLAM can be reduced to pose graph SLAM by removing landmarks and introducing new edges between relevant pairs of robot poses. The newly introduced edges between the pairs of robot poses constraint the nodes equivalently to the edges of the previous pose graph.

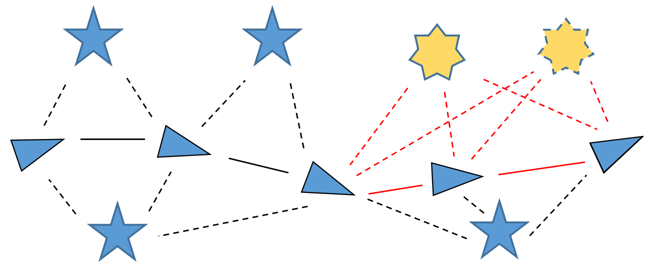
In RGB-D SLAM, the constraints between robot poses encode the transformations between key frames. Loop detection is to find the pairs of key frames which are likely to be captured at a same location. It adds an edge between the associated key frames and computes the transformation between them. The pose graph SLAM can be solved by minimizing a number of non-linear errors:

$$\begin{aligned}\mathcal{E}_{PoseGraph}(\chi) = {} & \chi_0^T \Lambda_0 \chi_0 \\ & + \sum_{i\in\mathcal{P}} [\varphi_i^{i-1} - \mathcal{H}(\chi_{i-1}, \chi_i)]^T \,\Gamma_i^{i-1} [\varphi_i^{i-1} - \mathcal{H}(\chi_{i-1}, \chi_i)] \\ & + \sum_{\{m,n\}\in\mathcal{P}} [\zeta_m^n - \mathcal{H}(\chi_m, \chi_n)]^T \,\Xi_m^n [\zeta_m^n - \mathcal{H}(\chi_m, \chi_n)]\end{aligned} \qquad (1)$$

where χ represents the robot poses that we want to find, $\mathcal{P}$ represents the node set of the graph, φ and ζ are measurement information, Γ and Ξ are the information matrices, $\mathcal{H}$ represents the model of the constraints. Note that the first term $\chi_0^T \Lambda_0 \chi_0$ encodes the anchoring constraint which is treated as a constant in the optimization problem [32]. In (1), the second term and the third term refer to the errors from the motion constraints and the loop constraints.

Moving objects can corrupt the pose estimation and bring false positives in loop detection. Fig. 1 demonstrates the negative effects caused by dynamic landmarks. We can see the pose graph is distorted in Fig. 1c. In addition, a false edge (denoted as a red line) between two poses is added due to the confused loop detection.


(a) Graph SLAM in a static environment.


(b) Graph SLAM in a dynamic environment.

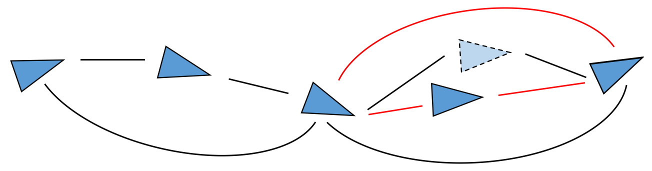

(c) The jeopardized pose graph in the dynamic environment.

Fig. 1. This figure illustrates the negative effect caused by dynamic landmarks. The triangles represent robot poses. The pentagrams and hexagrams represent static and dynamic landmarks respectively. The solid edges link robot poses. The dotted edges link robot poses and the landmarks sensed at those poses. The black edges represent the constraints obtained without dynamic objects. The red edges represent the constraints obtained with dynamic objects. The two yellow hexagrams in sub-figure (b) represent the two locations of a dynamic landmark observed at different time. In sub-figure (c), the blue triangles represent robot poses obtained in the dynamic environment. The light blue triangle represent the pose in the static environment. The jeopardized pose graph is clearly illustrated by comparing the robot poses with the light blue triangle. This figure is best viewed in color.

To explain why we use motion removal to improve SLAM, we augment (1) as follows:

$$\begin{aligned}\mathcal{E}'_{PoseGraph}(\chi, \theta) = {} & \chi_0^T \Lambda_0 \chi_0 \\ & + \sum_{i\in\mathcal{P}} \mathcal{F}_i^{i-1}(\theta) \| \left(\varphi_i^{i-1} - \mathcal{H}(\chi_{i-1}, \chi_i)\right) \|_{\Gamma_i^{i-1}}^2 \\ & + \sum_{\{m,n\}\in\mathcal{P}} \mathcal{G}_m^n(\theta) \| \left(\zeta_m^n - \mathcal{H}(\chi_m, \chi_n)\right) \|_{\Xi_m^n}^2\end{aligned} \qquad (2)$$

where $\|\cdot\|$ represents the 2-norm, $\theta \in \mathbb{R}^U$ represents the associated dynamic landmarks, U is the number of the landmarks. We introduce two tuning functions $\mathcal{F}(\theta)$ and $\mathcal{G}(\theta)$ in this paper, where $\mathcal{G}(\theta)$ has been described in [33]. They are both mappings $\mathbb{R}^U \to \mathbb{R}$. The function $\mathcal{F}(\theta)$ tunes the value of the motion constraint error. It is a non-linear function which can tune the error value close to the case without dynamic objects. Specially, there exists $\mathcal{F}(\theta) = 1$, if $U = 0$. In this case, the environment is static. The performance of $\mathcal{F}(\theta)$ will be degraded if dynamic landmarks are not correctly identified. The function $\mathcal{G}(\theta)$ outputs a binary decision. It deletes false loop constraints caused by dynamic landmarks. It is described as follows:

$$\mathcal{G}(\theta) = \begin{cases} 0, & C \in \Delta \\ 1, & otherwise \end{cases} \qquad (3)$$

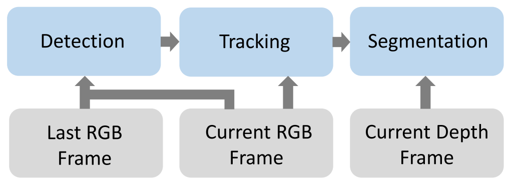


Fig. 2. The overview of the proposed motion removal approach. The approach is divided into three stages: *Detection*, *Tracking* and *Segmentation*. The arrows represent the directions of data flow. The *Detection* stage takes as input two consecutive RGB images. The *tracking* and *Segmentation* stages take as input the current RGB image and the current depth image respectively.

where C denotes the loop constraint with which the pairs of robot poses are associated, Δ represents the set of false loop closing constraints caused by dynamic landmarks. The function can remove false loop constraints if the dynamic landmarks are correctly determined. Similar to $\mathcal{F}(\theta)$, the performance of $\mathcal{G}(\theta)$ will be degraded if not all the dynamic landmarks are identified. As we can see, the tuning functions aim to improve RGB-D SLAM by eliminating the negative effects caused by dynamic landmarks. However, the prerequisite is to determine the dynamic landmarks θ. The tuning functions can work well only when θ is correctly determined. Motion removal is able to correctly determine θ. This explains why we use motion removal to improve the SLAM performance in this paper.

4. The proposed motion removal approach

4.1. Overview of approach

The idea of our approach is straightforward. It consists of three steps. The first step is to roughly detect moving-object motions based on ego-motion compensated image differencing. The second step is to enhance the motion detection by tracking motions using particle filter. The third step is to apply the Maximum-a-posterior (MAP) estimator on vector quantized depth images to precisely determine the foreground. It should be noted that what we track in our approach are motion patches but not moving objects. Our approach is different from most tracking techniques [34] which build models for moving objects and track the built models.

Fig. 2 shows an overview for our motion removal approach. We use RGB images for motion detection and tracking. The camera ego-motion is computed using RANSAC-based homography estimation with two consecutive RGB images. Note that the camera ego-motion is expressed as perspective transformations in the 2-D image space. This is because the perspective transformations are sufficient to reflect the 3-D camera motions including translation and rotation in this paper. Moving-object motions are roughly detected by subtracting the current RGB frame with the ego-motion compensated last RGB frame. The reason why we do not compensate RGB-D point-cloud frames in 3-D is that depth measurement errors are quadratically increasing when distances are increasing [35]. Subtracting 3-D point-cloud frames will severely suffer from the unstable errors. Moreover, methods for computing 3-D transformations, such as the ICP algorithm, are much slower than 2-D computation methods. In the *detection* stage, difference images are the outputs. The pixel values in the difference image serve as the measurement information for the particle filter. In our implementation, we track motions in 2-D image space instead of in 3-D Euclidean space. This is because tracking in 3-D requires deploying a great many layers of particles in the depth direction. It will severely increase the computational cost. The posterior belief computed from the particle filter serves as the likelihood for the MAP estimation in the *segmentation* stage. Depth images clustered by vector quantization are employed for the motion segmentation. The cluster that has the highest foreground probability computed by MAP is treated as the foreground.

In our approach, an implicit assumption is that static objects should dominate the scene. If this assumption does not hold true, features will be mainly taken from moving objects and the homography will be incorrectly estimated. Another implicit assumption is that scenes must contain planes. This is a general requirement for homography computation methods [36]. Our approach will fail in scenes where this assumption does not hold true, for instance, natural environments which are usually lack of planes. Moreover, in order to facilitate the sparse homography estimation method, features must be available. Our approach will fail if scenes are featureless. Fortunately, these issues are not critical in most man-made environments.

In the *detection* stage, non-zero pixels in difference images indicate the motions caused by moving objects. Noises in image differencing results are unavoidable, especially in the cases when we get incorrectly estimated homography. Thus, we use particle filter to enhance the motion detection. Note that the noises here represent the non-zero pixels that are not located on the moving object. The motion belief of the particle filter is an integral of a short-term history. It functions as a prior knowledge for moving-object locations. This prior information alleviates particle divergence caused by the false positives of motion detection. The motion belief also makes our approach robust to short-term stop of moving objects. However, our approach will fail if moving objects become motionless. This is because measurement information gradually dominates the decisions in the iterations.

We assume that the parallax between consecutive frames is negligible. The first reason is that small parallax ensures sufficient inlier feature associations to correctly estimate the homography. The second reason is that large object movements induced by large parallax can lead to tracking lost. In most cases, cameras usually provide a high image streaming rate, such as 30 fps, which ensures the parallax sufficiently small. Thus, this assumption cannot hinder the generality of our approach.

4.2. Ego-motion compensated frame differencing

For static cameras, background in consecutive frames remains not changed. All movements in image sequences are caused by moving objects. Subtracting the current frame with the last one can intuitively remove the static background. Moving-object motions can be indicated by the non-zero pixels in the differencing result. The frame differencing is described in the formula:

$$I_d(x, y, t) = |I(x, y, t) - I(x, y, t-1)|, \tag{4}$$

where $|\cdot|$ represents absolute value, x and y are pixel coordinates, I_d is the intensity value of the pixel (x, y) in the difference image. We consider the pixel is a possible foreground point if $I_d(x, y, t) \neq 0$, otherwise the pixel belongs to background.

However, the idea of frame differencing is not feasible if camera is moving. The motions in images are induced both by moving objects and camera ego-motions. Subtracting consecutive frames brings so many noises that the frame differencing results are virtually useless. Our idea is to temporarily stabilize the camera with ego-motion compensation so that the frame differencing is able to work. We use 2-D perspective transformation matrices to represent the ego-motions. We warp the last frame with the perspective matrix and subtract the current frame with the warped frame. Motions are roughly indicated by the frame differencing results.

Fig. 3 shows the schematic diagram of our idea. We use RANSAC algorithm to compute the perspective transformation. Let $T \in SE(2)$ denote the transformation matrix. Let $\boldsymbol{u}$ and $\boldsymbol{v}$ denote two

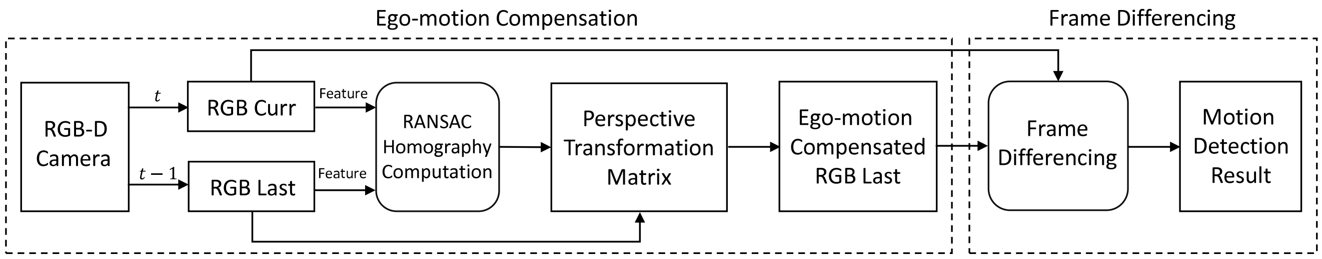


Fig. 3. The schematic diagram of the ego-motion compensated frame differencing technique. The *RGB Last* and *RGB Curr* represent the last and current RGB images captured at time $t-1$ and t. We use the RANSAC algorithm to eliminate outlier feature associations and compute the optimal homography. Camera ego-motions are represented as 2-D perspective matrices. Motions are roughly detected by subtracting *RGB Curr* with the ego-motion compensated *RGB Last*.

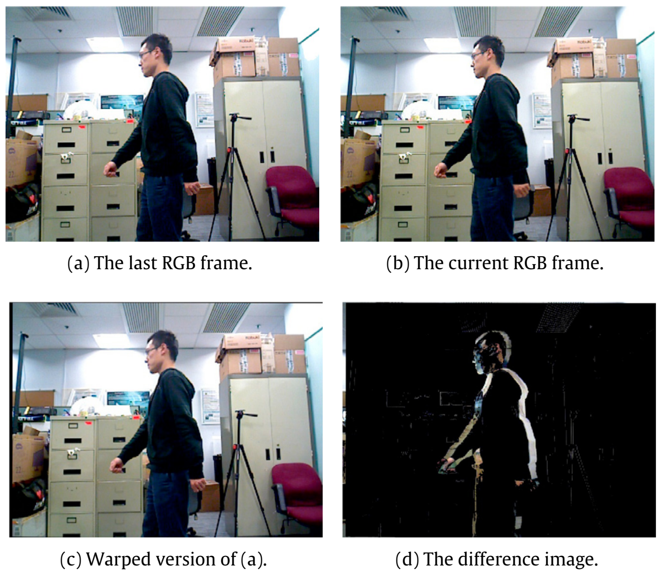

(a) The last RGB frame.
(b) The current RGB frame.
(c) Warped version of (a).
(d) The difference image.

Fig. 4. The process of the ego-motion compensated frame differencing. The shown scenario is a person walking in a room. The invalid areas in sub-figure (c) caused by image warping are filled black. As we can see in sub-figure (d), the walking person is roughly indicated by the non-zero pixels in the difference image. We can see that the noises are inevitable in the differencing results.

pixel locations of a feature pair. We warp the point $\boldsymbol{u}$ with T, and compute the reprojection error:

$$\xi = \|T\boldsymbol{u} - \boldsymbol{v}\| \tag{5}$$

where $\|\cdot\|$ represents the Euclidean distance. We consider the feature pair is an inlier if $\xi < \tau$, otherwise the pair is an outlier. The threshold τ is a pre-defined parameter in the RANSAC algorithm. The RANSAC algorithm random selects 4 feature pairs to compute T, and determines the inliers from all the feature pairs. The algorithm iterates until a set with the maximum number of inliers is achieved. Lastly, the RANSAC algorithm computes the optimal perspective matrix using the found inlier set.

With the optimal perspective matrix, we warp all pixels in the last frame to get the ego-motion compensated frame $I^T(x, y, t-1)$. Then, (4) can be modified as follows:

$$I'_d(x, y, t) = |I(x, y, t) - I^T(x, y, t-1)|. \tag{6}$$

Moving-object pixels can be roughly indicated by the non-zero pixels in $I'_d(x, y, t)$.

Fig. 4 qualitatively demonstrates the process of the ego-motion compensated frame differencing. The sequence is captured by a hand-held moving camera. As we can see, the parallax between the consecutive frames is negligible. The difference image is obtained by subtracting the warped frame with the current frame. The moving object can be roughly identified by the non-zero pixels in the differencing results.

4.3. Particle filter-based tracking

Let $\boldsymbol{x}$ denote the state variable of the particle filter. In our implementation, it is pixel coordinates. The particles here is a set of selected pixels. Particle location is the coordinates of a pixel. The state variable of particle i at time t is represented as follows:

$$\boldsymbol{x}_i^t = [x_i^t,\ y_i^t], \tag{7}$$

where x and y consist of the particle location. The posterior belief of the state variable at time t can be recursively estimated with a particle transition model and a measurement model.

Let $p(\boldsymbol{x}_i^t|\boldsymbol{u}_i^t, \boldsymbol{x}_i^{t-1})$ denote the transition probability from time $t-1$ to t, and $p(z_i^t|\boldsymbol{x}_i^t)$ denote the measurement probability at time t. We have the following Bayesian equations:

$$\begin{cases} \overline{bel}(\boldsymbol{x}_i^t) = \sum p(\boldsymbol{x}_i^t|\boldsymbol{u}_i^t, \boldsymbol{x}_i^{t-1}) bel(\boldsymbol{x}_i^{t-1}) \\ bel(\boldsymbol{x}_i^t) = \eta p(z_i^t|\boldsymbol{x}_i^t)\overline{bel}(\boldsymbol{x}_i^t) \end{cases}, \tag{8}$$

where η is a normalization constant. The belief $bel(\boldsymbol{x}_i^0)$ at $t = 0$ is initialized with a uniform distribution. We randomly deploy the particles in a uniform distribution in the first iteration.

Based on the observation that object motions are usually non-linearly distributed in image space, we model the transition probability in a 2-D Gaussian distribution. It makes our approach adapt to various types of object movements. As shown in (9), the locations of a particle at time t are predicted as follows:

$$p(\boldsymbol{x}_i^t|\boldsymbol{u}_i^t, \boldsymbol{x}_i^{t-1}) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp\left[-\frac{1}{2}((\boldsymbol{x}_i^t - \boldsymbol{\mu}_i^t)^T)\Sigma^{-1}(\boldsymbol{x}_i^t - \boldsymbol{\mu}_i^t)\right], \tag{9}$$

where $|\Sigma|$ is the determinant of the variance Σ, $\boldsymbol{\mu}_i^t$ is the previous particles that are warped to the current frame using the previously computed homography. $\boldsymbol{\mu}_i^t$ is obtained with the following formula:

$$\boldsymbol{\mu}_i^t = T_{t-1}^t \boldsymbol{x}_i^{t-1}, \tag{10}$$

where $\boldsymbol{x}_i^{t-1}$ is the particle in the last frame, T_{t-1}^t is the perspective matrix of the computed homography. Note that the warped particles which have locations beyond the image range are deleted.

The intensity value in the difference image encodes the probability of the pixel being the foreground. The measurement probability is represented as particle weight. To improve the robustness of weight computation, neighboring pixels are taken into consideration. The weight of a particle is a weighted average of the neighboring pixel values in the difference image. We calculate the weight w_i for particle i using a Gaussian kernel. We use circle shaped neighboring area. Assume there are M number of pixels in the area. The weight w_i is given by:

$$w_i^t = \sum_{j=1}^{M} I'_d(x_j, y_j, t)\frac{1}{\sqrt{2\pi}\upsilon} \exp\left(-\frac{\varrho^2}{2\upsilon^2}\right), \tag{11}$$

where x_j, y_j are the coordinates for pixel j, $I'_d(x_j, y_j, t)$ is the intensity value in the difference image, υ is the standard deviation of the Gaussian kernel, ϱ is the Euclidean distance between pixel j and particle i, which is calculated as:

$$\varrho = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}. \tag{12}$$

In the re-sampling stage of the particle filter, we use the Sequential Importance Re-sampling (SIR) strategy [37] to generate new particles. The importance is proportional to the weights of the particles. The particles with higher importance are replicated until the total number of particles reaches a pre-defined particle number.

4.4. Vector quantization-based segmentation

Vector Quantization (VQ) is a kind of data lossy compression method. It maps a vector set into a subset of itself or a set with less elements. After VQ, the original set is represented by a limited number of vectors [38]. For VQ in our approach, the vectors are composed of pixel values and pixel coordinates. The distortion error $D(\boldsymbol{v}, \boldsymbol{v}^*)$ between a vector $\boldsymbol{v} \in \mathbb{R}^Z$ in the original set and a vector $\boldsymbol{v}^* \in \mathbb{R}^Z$ in the quantized set is determined as follows:

$$D(\boldsymbol{v}, \boldsymbol{v}^*) = \frac{1}{Z}\|\boldsymbol{v} - \boldsymbol{v}^*\|, \tag{13}$$

where $\|\cdot\|$ represents the 2-norm, Z is the dimensionality of the vector. To quantize an image is to divide the vectors formed by all pixels into a set of clusters. Thus, the problem of vector quantization for an image becomes the problem of clustering pixels with locality and color information. In this paper, we employ the K-means clustering algorithm [39] to realize the quantization.

Based on the observation that objects in depth images are easier to be identified by the depth values, we use depth images for VQ in this paper. This observation is demonstrated in Fig. 5. As we can see, the performance of clustering using an RGB image and a depth image differs greatly. It is obvious that the foreground can be more easily extracted from the quantized depth image. Using depth information for foreground segmentation is a benefit here.

With the clustered depth image, we segment motions using the MAP estimation. Let $p(s_{k,t})$ denote the probability of cluster k being the foreground at time t. To get the motion segmentation result is to find the cluster that has the maximum posterior foreground probability. The MAP estimation is able to compute the posterior probability using the likelihood from the tracking results. Let $p(m|s_{k,t})$ denote the likelihood from the particle filter. The posterior probability $p(s_{k,t}|m)$ can be computed by the following equation:

$$p(s_{k,t}|m) = \frac{p(m|s_{k,t})p(s_{k,t})}{p(m)}, \tag{14}$$

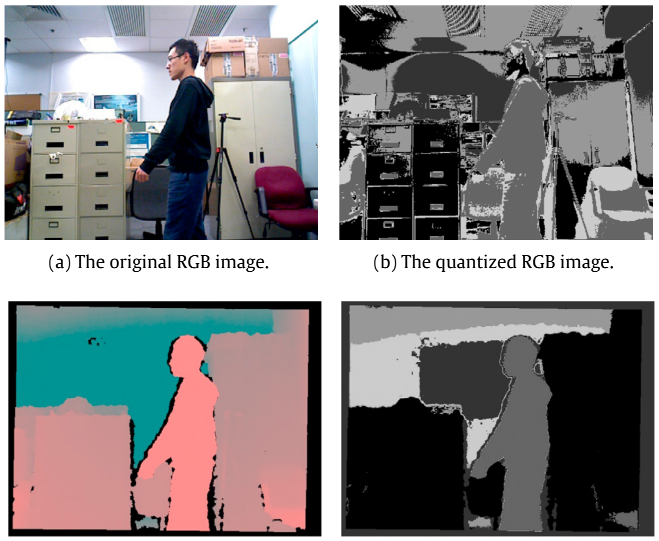
so the segmentation problem becomes to find the cluster k that has the maximum posterior probability:

$$k = \underset{k}{\operatorname{argmax}}\, p(s_{k,t}|m), \tag{15}$$

where $p(s_{k,t})$ is the prior probability for cluster k being the foreground, $p(m)$ is a normalization constant. The prior probability obeys a uniform distribution in each iteration. The proportion of particles that lie in the cluster k is employed as the likelihood:

$$p(m|s_{k,t}) = \frac{n_t}{N_t}, \tag{16}$$

where n_t is the number of particles that lie in the cluster k at time t, N_t is the total number of the particles at time t. Higher values of $p(m|s_{k,t})$ indicate higher likelihood for cluster k being the foreground.



(a) The original RGB image. (b) The quantized RGB image.

(c) The original depth image. (d) The quantized depth image.

Fig. 5. The comparison for VQ using an RGB image and a depth image. Note that the depth values in sub-figure (c) are increasing from red to green. The cluster number is set to 5 here. As we can see, the walking person is more easily identified due to the less texture information in the depth image. The benefit of using depth information is clearly demonstrated here. This figure is best viewed in color.

5. Experimental results and discussions

In this section, we present the experimental results and discussions. In the first part, we qualitatively demonstrated our motion removal approach in typical dynamic environments using our self-generated dataset. In the second part, we integrated the proposed motion removal approach into the front end of DVO (Dense Visual Odometry) SLAM [8] which is a kind of dense RGB-D SLAM system. Our approach was evaluated by examining the performance of the integrated SLAM system. Experiments were performed using the public TUM RGB-D dataset [30] and extensive quantitative evaluation results were given. A PC with an Intel i3 CPU and 4GB memory was used to run the programs. The RGB-D images were processed at the 640 ×480 resolution. The numbers of clusters was fixed to 5. For DVO SLAM, we left the open-source implementation [40] unchanged. Our approach cannot work in real time at the current stage, since our approach involves a dense image segmentation which costs about half a second per frame.

5.1. Demonstration using our dataset

In this section, we generated an RGB-D dataset with a walking person as the moving object. The person walks in a normal speed during the experiments. Image sequences was captured using a hand-held Asus Xtion Pro Live camera [41]. There were two sequences: `office` and `hallway`, which were recorded in an office room and an indoor hallway respectively.

Figs. 6 and 7 qualitatively demonstrate sample experimental results using the two sequences. As we can see, our approach produces precise motion removal results. In the tracking results, however, the particles concentrate on parts of the body but not the whole body. This is because the motions on these areas are generally larger than the motions on other parts of the body. More non-zero pixels from difference images can be found at these areas. The particles tend to converge on areas where larger measurement values are given.

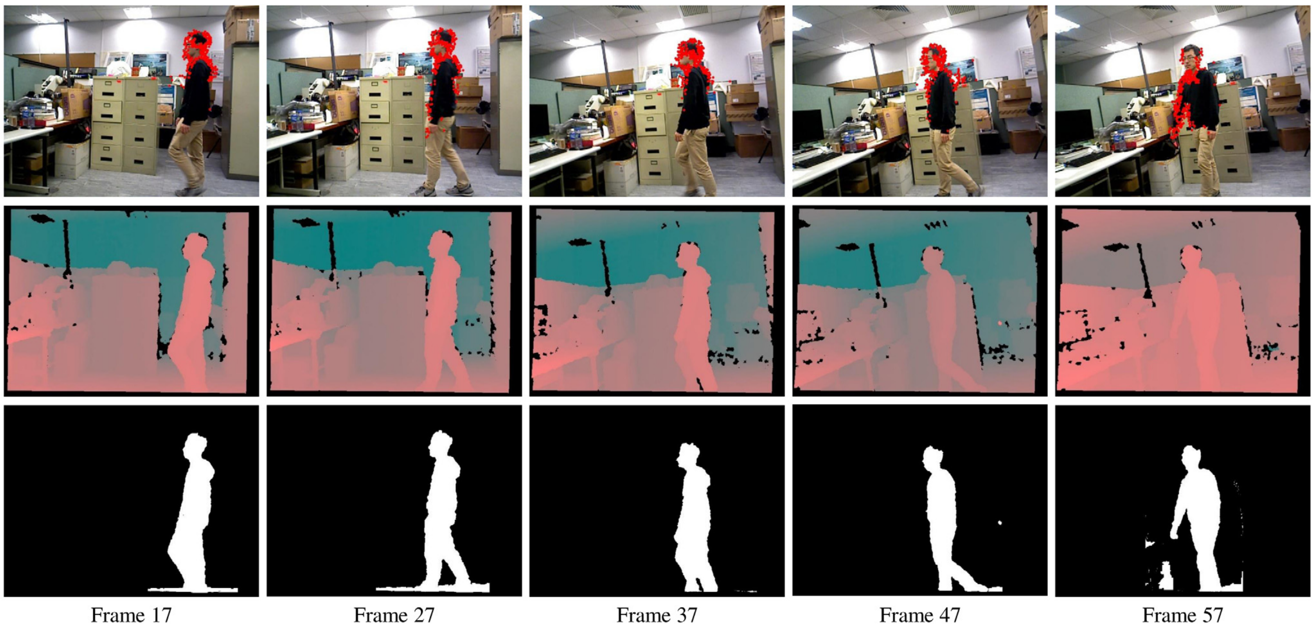


Fig. 6. Sample experimental results using the `office` sequence. The sub-figures from top row to bottom row are motion tracking results, depth images and motion removal results respectively. The red dots represent the locations of the particles. The width of the ROI is 1.4 times of the width of the rectangular contour of the particles. The height of the ROI is set to the height of the image. This figure is best viewed in color.

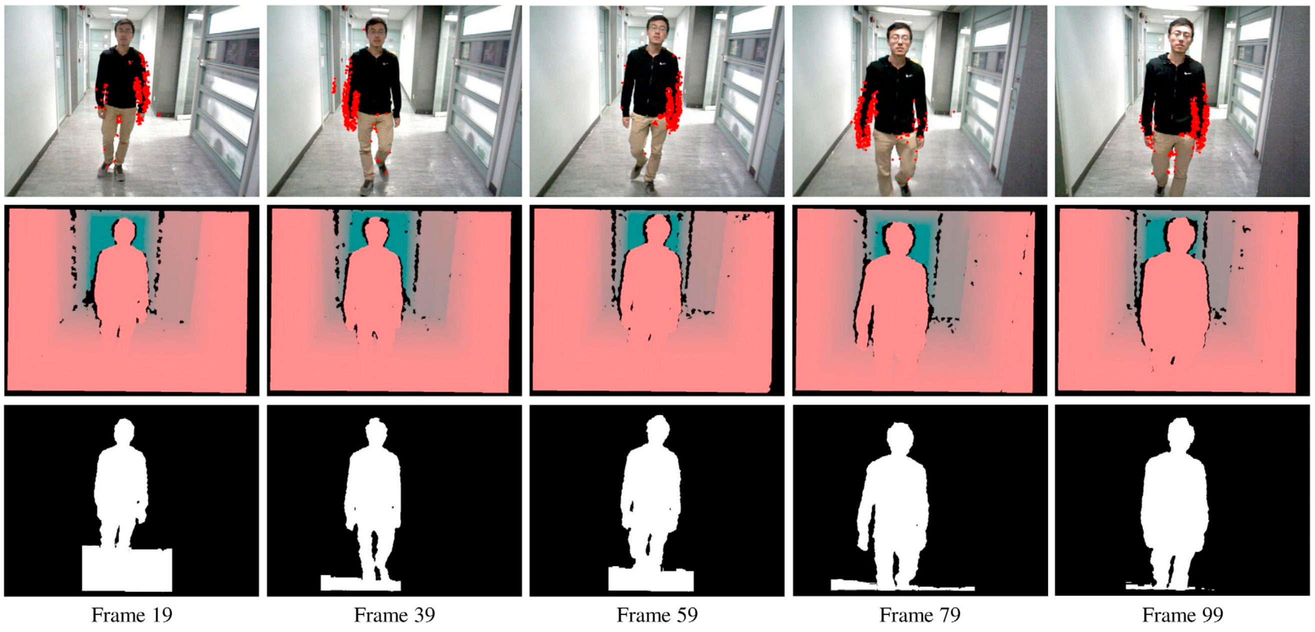
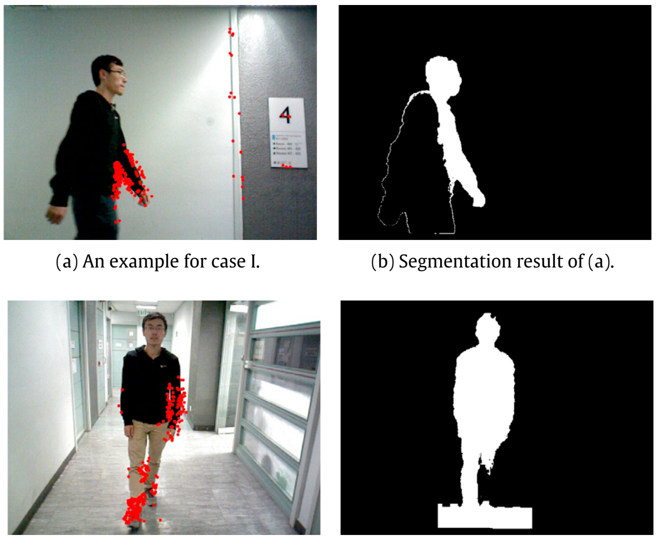


Fig. 7. Sample experimental results using the `hallway` sequence. The sub-figures from top row to bottom row are motion tracking results, depth images and motion removal results respectively. The red dots represent the locations of the particles. The width of the ROI is 1.3 times of the width of the rectangular contour of the particles. The height of the ROI is set to the height of the image. This figure is best viewed in color.

In Fig. 7, some background objects are misclassified as foreground. This is caused by the depth closeness between the foreground and the background. Background objects are wrongly classified into the foreground cluster due to the close depth values. In order to alleviate this problem, we do the image segmentation within an Region-of-interest (ROI). The ROI is adaptively determinedaccording to the rectangular contour of the particles. Note that the experimental results with the TUM dataset are obtained without using ROI. This is because we would like to present quantitative results with prior information at the minimum level. In addition, there are normally discriminative distances between the foreground and the background in the TUM dataset, so using ROI will not contribute too much.

Fig. 8 shows two typical segmentation failure cases. In Fig. 8b, the walking person is over-segmented because the number of clusters seems too large for this case. In Fig. 8d, the distance between the body and the leg is so large that the algorithm fails to classify the whole body into one cluster. These two problems can be alleviated by decreasing the number of clusters. This figure also reveals the weakness of using a fixed number of clusters.


(a) An example for case I. (b) Segmentation result of (a).

(c) An example for case II. (d) Segmentation result of (c).

Fig. 8. Two typical failure cases for motion removal. Red dots represent the locations of the particles, which shows the tracking results. The figure reveals the weakness of using a fixed number of clusters. This figure is best viewed in color.

5.2. Evaluation using TUM dataset

In this section, we integrated the proposed motion removal approach into the front end of the DVO SLAM system. Our approach acted as a pre-processing stage to filter out data that were associated with moving objects. The g2o library [42] was used for the pose graph optimization in the SLAM back end. Quantitative evaluation results were obtained using the TUM `Dynamic Objects` dataset. The TUM dataset provides ground truth captured by a precise motion capture system for each sequence. Motion removal was processed through all the frames in each sequence.

The TUM `Dynamic Objects` dataset contains several typical dynamic-environment scenarios such as `desk`, `sitting` and `walking`, etc. In the `desk` sequences, a person walks into a scene and then spends most of the time sitting at a desk. The person interacts with some objects while sitting. In the `sitting` sequences, two persons sit at a desk, and talk, gesticulate a little bit. We consider the movements in the `desk` and `sitting` sequences as *low-dynamic motions* in this paper. In the `walking` sequences, two persons walk back and forth in the scene and sit at a desk occasionally. The movements in the `walking` sequences are considered as *high-dynamic motions*. The `Dynamic Objects` dataset includes several typical types of camera ego-motions. In the `desk` sequences, the camera is steadily hold by a walking person. In the `walking` and `sitting` sequences, there are 4 types of camera ego-motions: `halfsphere`, `rpy`, `static` and `xyz`.

- `halfsphere`: The camera moved following a halfsphere-like trajectory.
- `rpy`: The camera rotated along the roll–pitch–yaw axes.
- `static`: The camera was roughly kept in place manually.
- `xyz`: The camera moved along the x-y-z axes.

For brevity, we use the words `fr, half, w, s, d, v` as representatives for `freiburg, halfsphere, walking, sitting, desk, validation` in the names of the sequences.

Fig. 9 demonstrates some selected motion removal results using the `Dynamic Objects` dataset. As we can see, our approach is able to remove moving objects effectively in various challenging scenarios. However, false positives of motion removal can be found in some frames. For instance, the chair and the table in sub-figures `fr3/w/half`, `fr3/w/half/v`, `fr2/d/person` and `fr3/w/rpy/v` are wrongly classified as foreground. This is caused by the depth closeness between the background and the foreground. Some pepper-and-salt false positives can be found in the sub-figure `fr3/s/xyz`. They are caused by the noises of the depth measurement. In the sub-figure `fr3/w/xyz/v`, only one person is segmented. This is because we use MAP to determine the foreground. Only one motion cluster can be labeled as foreground. Motions that cannot be classified into the foreground cluster are neglected. In other sub-figures, because the two persons are classified into one motion cluster, we can see they are correctly segmented.

We use the metrics Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) for the quantitative evaluation. The metric ATE measures the global consistency. Let $L \in SE(3)$ and $R \in SE(3)$ denote the ground truth and estimated camera poses. Let $\pi(R, L)$ denote the rigid transformation which aligns the estimated camera trajectory to the ground truth trajectory. The ATE J_i at pose i is defined as,

$$J_i = L_i^{-1}\pi(R, L)R_i. \tag{17}$$

The metric RPE measures the odometry drift. Let $H_{i,i+\delta}$ denote the transformation from camera pose i to camera pose $i+\delta$. We have $H^L_{i,i+\delta} = L_i^{-1}L_{i+\delta}$ and $H^R_{i,i+\delta} = R_i^{-1}R_{i+\delta}$ which represent the transformations calculated from the ground truth poses and the estimated poses. The RPE Q_i at pose i is defined as,

$$Q_i = {H^L_{i,i+\delta}}^{-1} H^R_{i,i+\delta}, \tag{18}$$

where the interval δ can be set to the value of frame rate for per-second evaluation or 1 for per-frame evaluation. Note that we use per-second evaluation in this paper. For details about ATE and RPE, we refer readers to [30] for more information.

RANSAC-based methods are generally sensitive to the reprojection error threshold τ, which has been described previously. Thus, it is necessary to choose an appropriate value for τ. We observe the SLAM performance in terms of ATE and the translational drift of RPE with different tries of τ using the TUM `fr3/w/xyz/v` sequence. The results are plotted in Fig. 10a. As we can see, the error values and the standard deviations tend to grow when τ is increasing. This shows that the SLAM performance is degraded and the instability is increasing when τ is increasing. Furthermore, we observed that the error values and the standard deviations are relatively small when τ is less than 6. Thus, we empirically choose 3 as the reprojection threshold in our experiments.

We also evaluate the SLAM performance given different values for the number of particles in the *tracking* stage. Similar to the experiments of the RANSAC parameter tuning, we observe the values of ATE and the translational drift of RPE using the TUM `fr3/w/xyz/v` sequence. The results are plotted in Fig. 10b. As we can see, the ATE and the RPE results are not sensitive to the numbers of particles. However, the values seem not stable when the number of particles is too small. Thus, we empirically set the number of particles to 1000 in this paper.

We present the quantitative evaluation results in Table 1–Table 3. The first column of the tables shows the sequence names. The term `Without Our Approach` represents the original DVO SLAM approach. The term `With Our Approach` represents the DVO SLAM with our motion removal approach incorporated in the SLAM front end. To facilitate further comparisons, we present the values of RMSE, Mean Error, Median Error and the Standard Deviation (S.D.) in this paper. We expect that the SLAM performance of the original DVO SLAM algorithm is improved by our motion removal approach. The improvement values in the tables are calculated as follows:

$$\digamma = \left(1 - \frac{\beta}{\alpha}\right) \times 100\%, \tag{19}$$

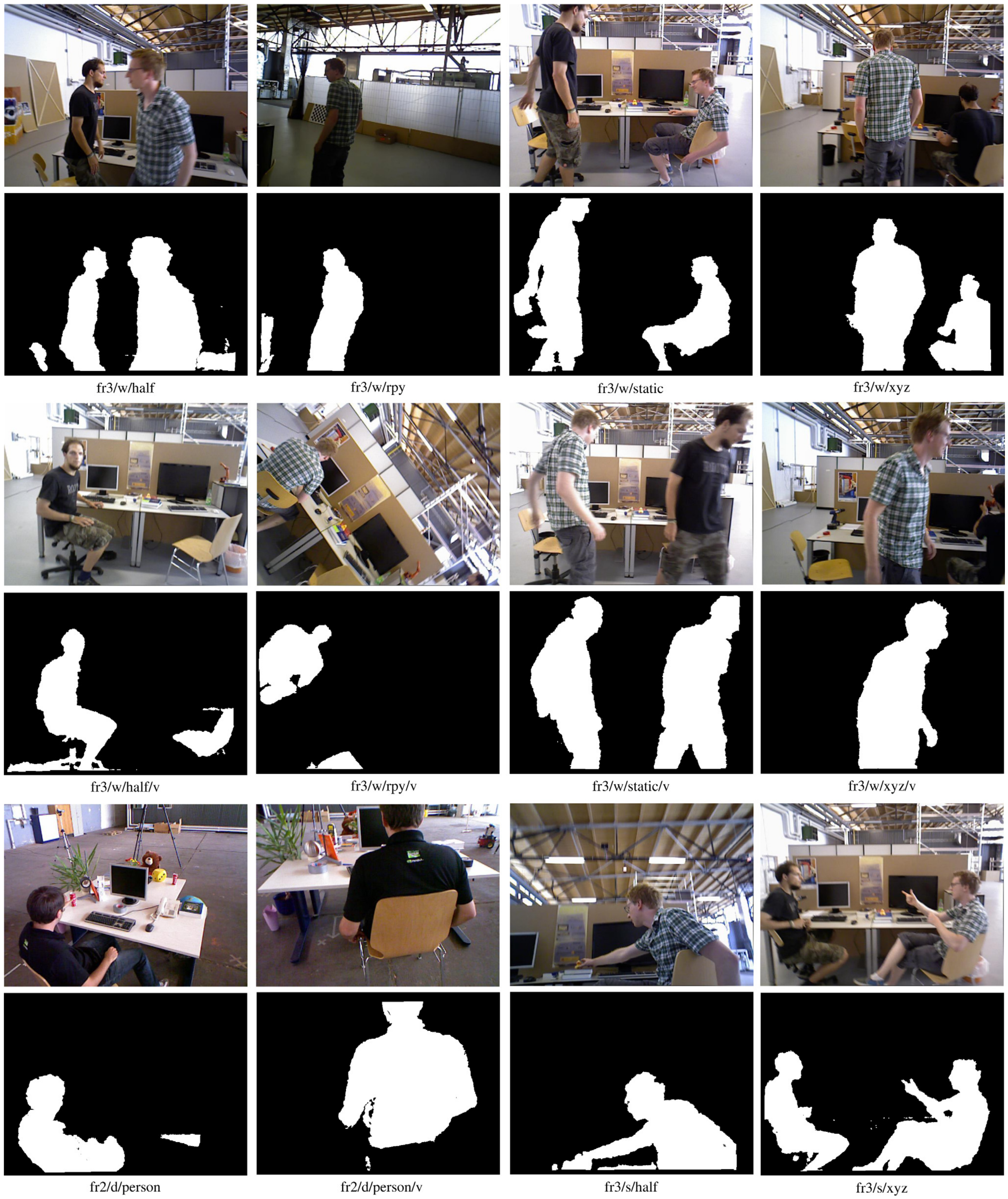
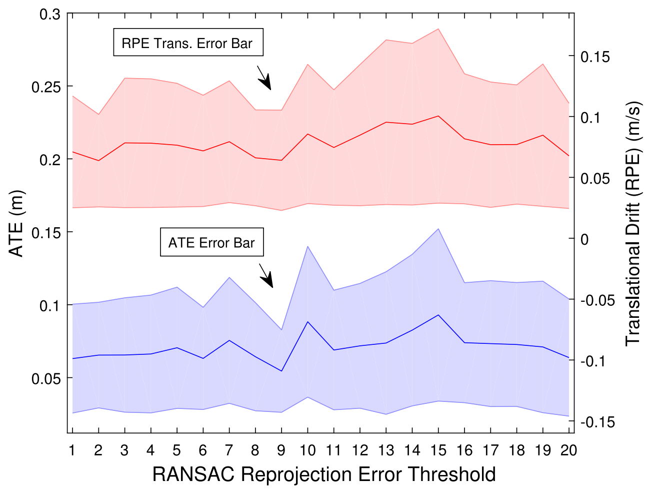


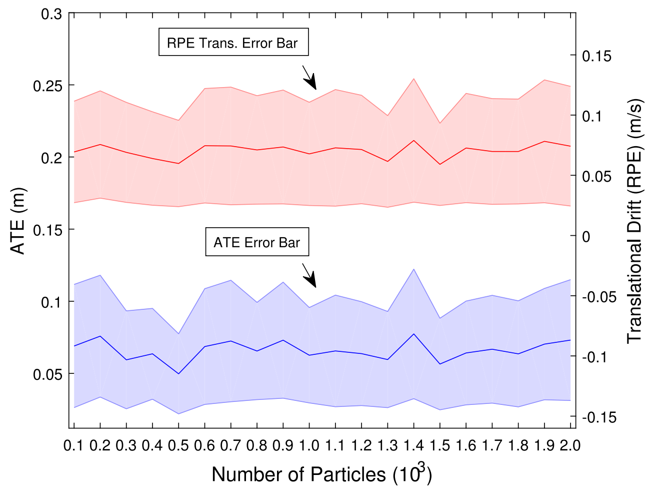
Fig. 9. Selected motion removal results using the TUM `Dynamic Objects` dataset. Corresponding RGB images are also presented. As we can see, our motion removal approach is able to effectively remove the moving objects in such challenging dynamic scenarios.

where $\mathcal{F}$ represents the improvement value, α represents the value without our approach, β represents the value with our approach. We highlight the RMSE values in the tables. The RMSE values are prone to be influenced by large or occasional errors [43]. Thus, RMSE values can better indicate the robustness compared to the mean and median values. We also highlight the S.D. values, because they encode the stability of the system.

As we can see from Table 1, the average RMSE and S.D. improvement values for the high-dynamic sequences are 80.12% and 68.89% respectively. This demonstrates that our approach is able to greatly improve the SLAM performance in terms of ATE for



(a) The parameter tuning for the error threshold.



(b) The parameter tuning for the number of particles.

Fig. 10. The parameter tunings for the RANSAC reprojection error threshold and the number of particles using the TUM `fr3/w/xyz/v` sequence. RMSE values for ATE are plotted in the blue curves. RMSE values for the translational drift are plotted in the red curves. The error bars encode the standard deviations. For the tuning of the RANSAC parameter, the number of particles is set to 1000. For the tuning of the number of particles, the RANSAC parameter is set to 3. The results from sub-figure (a) indicate that relatively good SLAM performance can be obtained when the error value is less than 6. The results from sub-figure (b) show that the SLAM performance is not sensitive to the number of particles. This figure is best viewed in color.

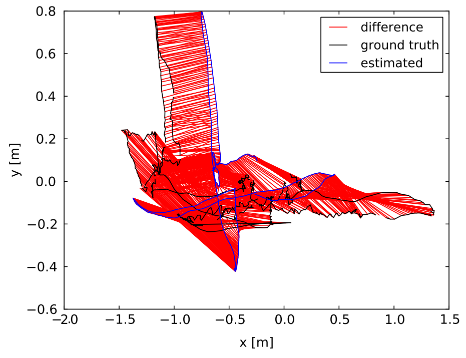
Table 1
ATE in meters for the experiments without and with our motion removal approach. Low dynamic sequences are denoted with a superscript star. Others are high-dynamic sequences. Our approach effectively improves the RGB-D SLAM performance in all scenarios in terms of ATE.

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**	**0.1252**	0.0867	0.0671	**0.0903**	**76.32%**	81.86%	83.30%	**60.04%**
fr3/w/rpy	**0.7304**	0.6730	0.6649	**0.2837**	**0.1333**	0.1035	0.0829	**0.0839**	**81.75%**	84.62%	87.53%	**70.43%**
fr3/w/static	**0.2120**	0.1628	0.1134	**0.1358**	**0.0656**	0.0377	0.0229	**0.0536**	**69.06%**	76.84%	79.81%	**60.53%**
fr3/w/xyz	**0.5966**	0.5334	0.4508	**0.2672**	**0.0932**	0.0764	0.0608	**0.0534**	**84.38%**	85.68%	86.51%	**80.01%**
fr3/w/half/v	**0.3735**	0.3142	0.2305	**0.2019**	**0.0811**	0.0619	0.0460	**0.0524**	**78.29%**	80.30%	80.04%	**74.05%**
fr3/w/rpy/v	**0.9115**	0.8740	0.8677	**0.2588**	**0.2333**	0.1420	0.0888	**0.1852**	**74.40%**	83.75%	89.77%	**28.44%**
fr3/w/static/v	**0.2016**	0.1365	0.0785	**0.1485**	**0.0319**	0.0232	0.0169	**0.0220**	**84.18%**	83.00%	78.47%	**85.19%**
fr3/w/xyz/v	**0.8778**	0.7102	0.5067	**0.5158**	**0.0655**	0.0525	0.0397	**0.0392**	**92.54%**	92.61%	92.16%	**92.40%**
fr3/s/half*	**0.0616**	0.0524	0.0431	**0.0324**	**0.0470**	0.0399	0.0315	**0.0249**	**23.70%**	23.85%	26.91%	**23.15%**
fr3/s/xyz*	**0.0505**	0.0393	0.0336	**0.0317**	**0.0482**	0.0391	0.0326	**0.0282**	**4.55%**	0.51%	2.98%	**11.04%**
fr2/d/person*	**0.0853**	0.0834	0.0868	**0.0180**	**0.0596**	0.0546	0.0530	**0.0239**	**30.13%**	34.53%	38.94%	**−32.78%**
fr2/d/person/v*	**0.1468**	0.1305	0.1170	**0.0672**	**0.0394**	0.0352	0.0333	**0.0177**	**73.16%**	73.03%	71.54%	**73.66%**

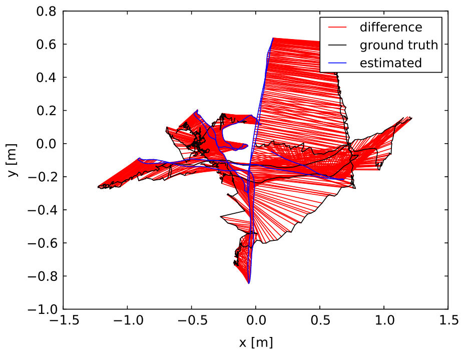
Table 2
Translational drift (RPE) in m/s for the experiments without and with our motion removal approach. Low dynamic sequences are denoted with a superscript star. Others are high-dynamic sequences. Our approach generally improves the RGB-D SLAM performance in terms of the translational drift.

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**	**0.1672**	0.1078	0.0696	**0.1278**	**49.09%**	48.02%	16.35%	**49.80%**
fr3/w/rpy	**0.4644**	0.3357	0.2175	**0.3210**	**0.1751**	0.1363	0.0973	**0.1099**	**62.30%**	59.40%	55.26%	**65.76%**
fr3/w/static	**0.2451**	0.1277	0.0231	**0.2093**	**0.0842**	0.0446	0.0178	**0.0714**	**65.65%**	65.07%	22.94%	**65.89%**
fr3/w/xyz	**0.4019**	0.2801	0.1640	**0.2882**	**0.1214**	0.0889	0.0537	**0.0827**	**69.79%**	68.26%	67.26%	**71.30%**
fr3/w/half/v	**0.2682**	0.1529	0.0532	**0.2203**	**0.0955**	0.0688	0.0436	**0.0661**	**64.39%**	55.00%	18.05%	**70.00%**
fr3/w/rpy/v	**0.3907**	0.2303	0.0909	**0.3156**	**0.3077**	0.1712	0.0879	**0.2556**	**21.24%**	25.66%	3.30%	**19.01%**
fr3/w/static/v	**0.1853**	0.0955	0.0311	**0.1589**	**0.0436**	0.0308	0.0219	**0.0309**	**76.47%**	67.75%	29.58%	**80.55%**
fr3/w/xyz/v	**0.4614**	0.2624	0.0527	**0.3795**	**0.0783**	0.0575	0.0386	**0.0532**	**83.03%**	78.09%	26.76%	**85.98%**
fr3/s/half*	**0.0466**	0.0346	0.0235	**0.0312**	**0.0458**	0.0373	0.0301	**0.0265**	**1.72%**	−7.80%	−28.09%	**15.06%**
fr3/s/xyz*	**0.0360**	0.0245	0.0165	**0.0264**	**0.0330**	0.0244	0.0172	**0.0222**	**8.33%**	0.41%	−4.24%	**15.91%**
fr2/d/person*	**0.0147**	0.0116	0.0097	**0.0091**	**0.0172**	0.0139	0.0116	**0.0101**	**−17.01%**	−19.83%	−19.59%	**−10.99%**
fr2/d/person/v*	**0.0171**	0.0145	0.0127	**0.0090**	**0.0252**	0.0193	0.0149	**0.0163**	**−47.37%**	−33.10%	−17.32%	**−81.11%**

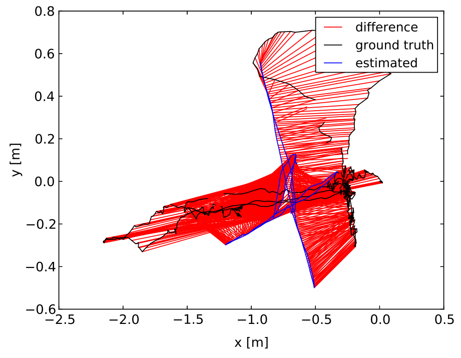
the high-dynamic scenarios. The stability is also enhanced with our approach. By examining the RMSE values, we find that our approach brings more improvements in the `static` and `xyz` sequences. The reason is that camera movements are relatively slow in these sequences. This is a benefit for our approach because slow camera motions ensures small parallax between consecutive frames. For the low-dynamic sequences, we get 7.73% average RMSE improvement and 18.77% S.D. improvement. As we can see, our approach provides less improvements for the low-dynamic cases. We think the reason is that dynamic objects can be easily identified as outliers in the low-dynamic sequences. In addition, low-dynamic objects in the tested sequences were normally kept in a fixed place. This hardly jeopardizes the loop closing process. Thus, the original DVO SLAM algorithm is able to achieve sufficiently good performance in the low-dynamic sequences, which leaves limited spaces for our improvements.



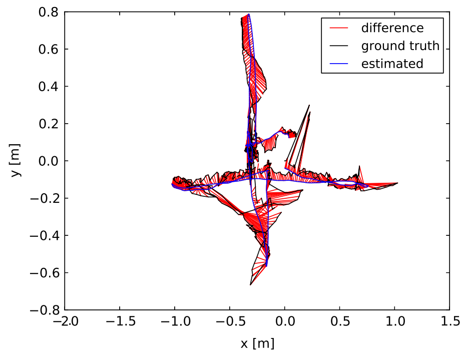
(a) fr3/w/half—without.



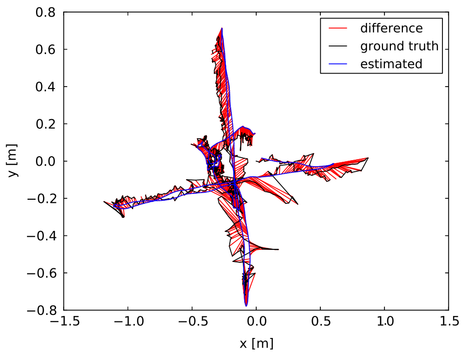
(b) fr3/w/half/v—without.



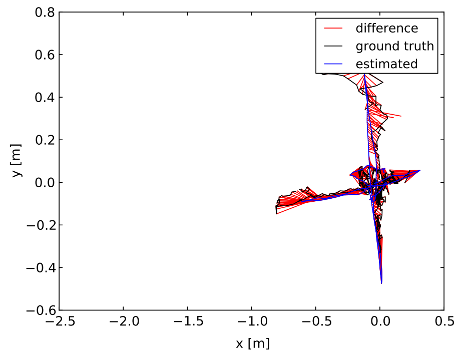
(c) fr3/w/xyz—without.



(d) fr3/w/half—with.

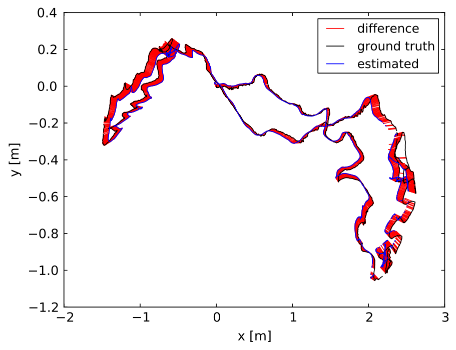


(e) fr3/w/half/v—with.

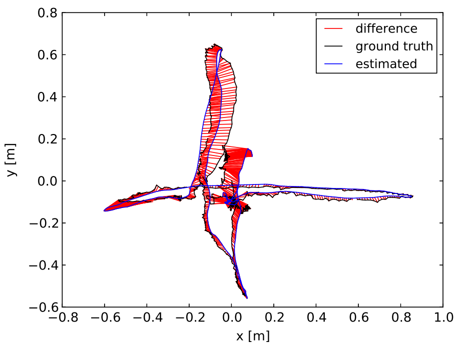


(f) fr3/w/xyz—with.

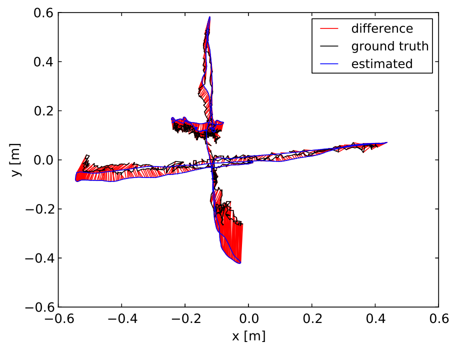
Fig. 11. Plots of ATE for the high-dynamic sequences `fr3/w/half`, `fr3/w/half/v`, `fr3/w/xyz`. The words *without* and *with* represent the experiments performed without and with our approach. This figure is best viewed in color.



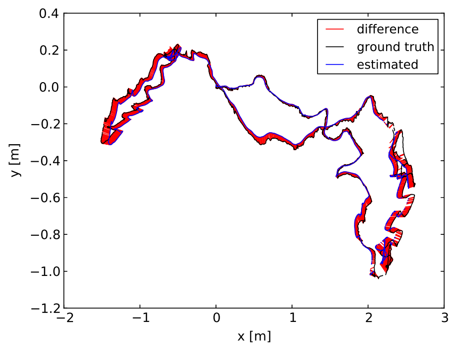
(a) fr2/d/person—without.



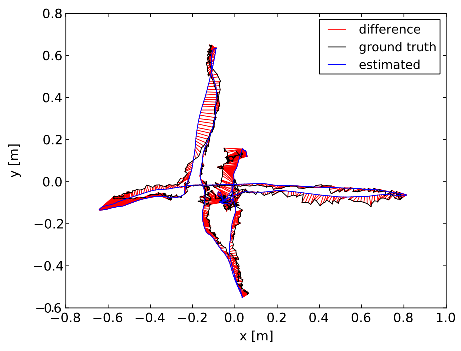
(b) fr3/s/half—without.



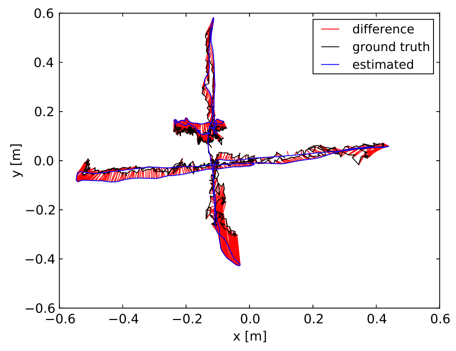
(c) fr3/s/xyz—without.



(d) fr2/d/person—with.



(e) fr3/s/half—with.



(f) fr3/s/xyz—with.

Fig. 12. Plots of ATE for the low-dynamic sequences `fr2/d/person`, `fr3/s/half`, `fr3/s/xyz`. The words *without* and *with* represent the experiments performed without and with our approach. This figure is best viewed in color.

Tables 2 and 3 show the visual odometry performance. As we can see, the results are in accordance with the above ATE analysis. The sequences with slow camera ego-motions exhibit high improvement values, and the original DVO SLAM algorithm performs

Table 3
Rotational drift (RPE) in deg/s for the experiments without and with our motion removal approach. Low dynamic sequences are denoted with a superscript star. Others are high-dynamic sequences. Our approach generally improves the RGB-D SLAM performance in terms of the rotational drift. The unit for the median values is rad/s.

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	0.0412	**5.0395**	**5.0108**	3.2882	0.0412	**3.7810**	**24.22%**	23.19%	0.00%	**24.97%**
fr3/w/rpy	**9.0292**	6.7447	0.0760	**6.0029**	**4.3755**	3.3596	0.0419	**2.8033**	**51.54%**	50.19%	44.87%	**53.30%**
fr3/w/static	**4.2761**	2.2631	0.0090	**3.6282**	**2.0487**	1.0548	0.0084	**1.7563**	**52.09%**	53.39%	6.67%	**51.59%**
fr3/w/xyz	**8.6593**	5.5484	0.0598	**6.6483**	**3.2346**	2.2587	0.0259	**2.3154**	**62.65%**	59.29%	56.69%	**65.17%**
fr3/w/half/v	**5.4066**	3.4424	0.0274	**4.1692**	**3.3035**	2.1688	0.0246	**2.4918**	**38.90%**	37.00%	10.22%	**40.23%**
fr3/w/rpy/v	**6.7382**	4.2855	0.0379	**5.1998**	**4.4968**	3.2548	0.0361	**3.1028**	**33.26%**	24.05%	4.75%	**40.33%**
fr3/w/static/v	**3.2292**	1.7556	0.0101	**2.7103**	**1.0815**	0.6967	0.0082	**0.8272**	**66.51%**	60.32%	18.81%	**69.48%**
fr3/w/xyz/v	**8.7977**	5.2338	0.0256	**7.0716**	**1.9484**	1.5431	0.0203	**1.1896**	**77.85%**	70.52%	20.70%	**83.18%**
fr3/s/half*	**2.4747**	1.8168	0.0205	**1.6802**	**2.3748**	1.8932	0.0255	**1.4336**	**4.04%**	−4.21%	−24.39%	**14.68%**
fr3/s/xyz*	**0.9956**	0.8114	0.0119	**0.5769**	**0.9828**	0.8060	0.0119	**0.5622**	**1.29%**	0.67%	0.00%	**2.55%**
fr2/d/person*	**0.5986**	0.4994	0.0077	**0.3300**	**0.7341**	0.6151	0.0093	**0.4007**	**−22.64%**	−23.17%	−20.78%	**−21.42%**
fr2/d/person/v*	**0.6271**	0.5406	0.0082	**0.3179**	**0.8843**	0.7263	0.0107	**0.5045**	**−41.01%**	−34.35%	−30.49%	**−58.70%**

well in the low-dynamic sequences. In the desk sequences, we found that our approach degrades the original performance. We think the reason is that the low-dynamic motions are usually not continuous in the desk sequences. Moving objects often become motionless in some frames. This is an unfavorable factor for motion tracking. It distracts the particles and makes our approach fail to segment the motions. Thus, more false segmentation results are generated. RPE performance is degraded due to the wrongly deletion of useful information. In addition, there is no object moving in the early part of the desk sequences. The scenarios during this period are in fact static environments, where noises of frame differencing dominate the motion detection. Our approach is susceptible to the noises in such scenarios and produces lots of false positives. We think this is also a reason for the performance degeneration.

Fig. 11 shows selected ATE plots for the high-dynamic cases. As we can see, the errors are greatly reduced with our approach. Fig. 12 shows selected ATE plots for the low-dynamic cases. We can see the original DVO SLAM algorithm provides good performance in these sequences. With our approach incorporated into the SLAM front end, the ATE values are further reduced.

6. Conclusions

We proposed here an RGB-D data-based motion removal approach. The motivation of this paper is to improve RGB-D SLAM in dynamic environments using the proposed motion removal approach. The proposed approach was divided into three stages. We tightly coupled the three stages in an on-line framework. In the experiments, we incorporated the approach into the front end of the DVO SLAM algorithm. Our approach acted as a pre-processing stage to filter out data that were associated with moving objects. Quantitative evaluations were carried out using the public TUM RGB-D dataset. The results show that our approach was able to effectively improve the RGB-D SLAM performance in various challenging scenarios. However, our approach still presents some limitations. For instance, the homography estimation will be degraded when parallax between consecutive frames is large. The tracking will fail when moving objects become motionless. As we use MAP to determine the foreground, only one motion cluster can be segmented. This limits our approach for scenarios with many moving objects. To overcome these limitations, we would like to enhance our approach with learning capability in the future. Motion appearance will be learned on the fly. In addition, we would like to extend our approach to a full RGB-D SLAM system, which can adaptively work both in static and dynamic environments.

Acknowledgments

Research presented in this paper was partially supported by the Research Grant Council of Hong Kong SAR Government, China, under project No. 21202816 and No. 16212815; National Natural Science Foundation of China No. 6140021318, Shenzhen Science, Technology and Innovation Commission (SZSTI) JCYJ20160428154842603 and JCYJ20160401100022706, awarded to Prof. Ming Liu, and partially supported by RGC GRF grants CUHK 415512 and CUHK 415613, CRF grant CUHK 6CRF13G, CUHK VC discretional fund #4930765, and Internal grant SKLRS201402B of the State Key Laboratory of Robotics and Systems (HIT), awarded to Prof. Max Q.-H. Meng. The authors would like to thank Zhe Min for acting as the moving object in our dataset.

References

[1] C. Stachniss, J.J. Leonard, S. Thrun, Simultaneous Localization and Mapping, Springer International Publishing, Cham, 2016, pp. 1153–1176.
[2] G. Younes, D. Asmar, E. Shammas, A survey on non-filter-based monocular Visual SLAM systems, 2016. arXiv preprint arXiv:1607.00470.
[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I.D. Reid, J.J. Leonard, Simultaneous localization and mapping: present, future, and the robust-perception age, 2016. arXiv preprint arXiv:1606.05830.
[4] M. Bloesch, S. Omari, M. Hutter, R. Siegwart, Robust visual inertial odometry using a direct EKF-based approach, in: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, IEEE, 2015, pp. 298–304.
[5] T. Moore, D. Stouch, A generalized extended kalman filter implementation for the robot operating system, in: Intelligent Autonomous Systems 13, Springer, 2016, pp. 335–348.
[6] J. Han, L. Shao, D. Xu, J. Shotton, Enhanced computer vision with microsoft kinect sensor: A review, IEEE Tran. Cyber. 43 (5) (2013) 1318–1334.
[7] E. Lachat, H. Macher, T. Landes, P. Grussenmeyer, Assessment and calibration of a RGB-D camera (Kinect v2 Sensor) towards a potential use for close-range 3D modeling, Remote Sens. 7 (10) (2015) 13070–13097.
[8] 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.
[9] G. Hu, S. Huang, L. Zhao, A. Alempijevic, G. Dissanayake, A robust rgb-d slam algorithm, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 1714–1719.
[10] P. Henry, M. Krainin, E. Herbst, X. Ren, D. Fox, Rgb-d mapping: Using Kinect-style depth cameras for dense 3d modeling of indoor environments, Int. J. Robot. Res. 31 (5) (2012) 647–663.
[11] 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.
[12] 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.
[13] A. Segal, D. Haehnel, S. Thrun, Generalized-icp, in: Robotics: Science and Systems, vol. 2, no. 4, 2009.
[14] G. Pandey, S. Savarese, J.R. McBride, R.M. Eustice, Visually bootstrapped generalized ICP, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 2660–2667.
[15] 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.
[16] Y. Sun, M. Liu, M. Q.-H. Meng, Motion removal from moving platforms: An RGB-D data-based motion detection, tracking and segmentation approach, in: 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), IEEE, 2015, pp. 1377–1382.

[17] A. Sobral, A. Vacavant, A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos, Comput. Vision Image Understanding 122 (2014) 4–21.
[18] R.M. Karp, On-line algorithms versus off-line algorithms: How much is it worth to know the future? in: IFIP Congress (1), vol. 12, 1992, pp. 416–429.
[19] Y. Sheikh, O. Javed, T. Kanade, Background subtraction for freely moving cameras, in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 1219–1225.
[20] S. Dey, V. Reilly, I. Saleemi, M. Shah, Detection of independently moving objects in non-planar scenes via multi-frame monocular epipolar constraint, in: European Conference on Computer Vision, Springer, 2012, pp. 860–873.
[21] F. Liu, M. Gleicher, Learning color and locality cues for moving object detection and segmentation, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 320–327.
[22] D. Zamalieva, A. Yilmaz, J.W. Davis, (2014) Exploiting temporal geometry for moving camera background subtraction, in: ICPR, pp. 1200–1205.
[23] R.K. Namdev, A. Kundu, K.M. Krishna, C. Jawahar, Motion segmentation of multiple objects from a freely moving monocular camera, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, IEEE, 2012, pp. 4092–4099.
[24] T. Lim, B. Han, J.H. Han, Modeling and segmentation of floating foreground and background in videos, Pattern Recognit. 45 (4) (2012) 1696–1706.
[25] K. Moo Yi, K. Yun, S. Wan Kim, H. Jin Chang, J. Young Choi, Detection of moving objects with non-stationary cameras in 5.8 ms: Bringing motion detection to your mobile device, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2013, pp. 27–34.
[26] A. Teichman, J.T. Lussier, S. Thrun, Learning to Segment and Track in RGBD, IEEE Trans. Autom. Sci. Engrg. 10 (4) (2013) 841–852.
[27] D. Giordano, F. Murabito, S. Palazzo, C. Spampinato, Superpixel-based video object segmentation using perceptual organization and location prior, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.
[28] Y. Wang, S. Huang, Towards dense moving object segmentation based robust dense RGB-D SLAM in dynamic scenarios, in: Control Automation Robotics & Vision (ICARCV), 2014 13th International Conference on, IEEE, 2014. pp. 1841–1846.
[29] 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.
[30] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers, A benchmark for the evaluation of RGB-D SLAM systems, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 573–580.
[31] G. Grisetti, R. Kummerle, C. Stachniss, W. Burgard, A tutorial on graph-based SLAM, IEEE Intell. Trans. Syst. Magazine 2 (4) (2010) 31–43. winter.
[32] S. Thrun, W. Burgard, D. Fox, Probabilistic Robotics, MIT Press, 2005.
[33] N. Sünderhauf, P. Protzel, Switchable constraints for robust pose graph slam, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 1879–1884.
[34] G.S. Walia, R. Kapoor, Recent advances on multicue object tracking: a survey, Art. Intell. Rev. 46 (1) (2016) 1–39.
[35] K. Khoshelham, S.O. Elberink, Accuracy and resolution of kinect depth data for indoor mapping applications, Sensors 12 (2) (2012) 1437–1454.
[36] A. Colombari, A. Fusiello, V. Murino, Segmentation and tracking of multiple video objects, Pattern Recognit. 40 (4) (2007) 1307–1317.
[37] M. Bolic, P.M. Djuric, S. Hong, Resampling algorithms and architectures for distributed particle filters, IEEE Trans. Signal Process. 53 (7) (2005) 2442–2450.
[38] H.-D. Cheng, X. Jiang, Y. Sun, J. Wang, Color image segmentation: advances and prospects, Pattern Recognit. 34 (12) (2001) 2259–2281.
[39] J.A. Hartigan, M.A. Wong, Algorithm AS 136: A k-means clustering algorithm, J. R. Stat. Soc. Ser. C (Appl. Stat.) 28 (1) (1979) 100–108.
[40] DVO SLAM website. [Online]. Available: https://github.com/tum-vision/dvo_slam/.
[41] Xtion PRO LIVE website. [Online]. Available: https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/.
[42] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, g2o: A general framework for graph optimization, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 3607–3613.
[43] 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.

Yuxiang Sun received his M.Sc. degree from the University of Science and Technology of China (USTC), Hefei, China, in 2012, and the B.A. degree from Hefei University of Technology (HFUT), Hefei, China, in 2009. He is now a Ph.D. student at the Department of Electronic Engineering, The Chinese University of Hong Kong (CUHK), Hong Kong, China. His current research interests include Visual SLAM, autonomous navigation, 3-D computer vision, motion perception, etc. He was a recipient of the Best Student Paper Finalist Award at IEEE ROBIO 2015.

Ming Liu received the B.A. degree in automation from Tongji University, Shanghai, China, in 2005. During his master study at Tongji University, he stayed for one year at Erlangen–Nurnberg University and Fraunhofer Institute IISB, Germany, as a Visiting Scholar. He received the Ph.D. degree from the Department of Mechanical Engineering and Process Engineering, ETH Zurich, Zurich, Switzerland, in 2013. He is now an Assistant Professor with the Department of Mechanical and Biomedical Engineering, City University of Hong Kong. His current research interests include autonomous mapping, visual navigation, topological mapping and environment modeling, etc. Prof. Liu is the recipient of the Best Student Paper Award at IEEE MFI 2012, the Best Paper in Information Award at IEEE ICIA 2013, the Best RoboCup Paper at IEEE IROS 2013, and twice the Winning Prize of the Chunhui-Cup Innovation Contest.

Max Q.-H. Meng received his Ph.D. degree in Electrical and Computer Engineering from the University of Victoria, Victoria, Canada, in 1992. He is currently Professor and Chairman of the Department of Electronic Engineering at The Chinese University of Hong Kong. He is affiliated with The State Key Laboratory of Robotics and Systems at Harbin Institute of Technology. He was a Professor and the Director of ART (Advanced Robotics and Teleoperation) Lab in the Department of Electrical and Computer Engineering at University of Alberta, Canada, from 1994 to 2004. His primary research interests are in medical and surgical robotics, active medical devices and wireless capsule endoscopy, medical image-based automatic diagnosis system, tele-medicine and tele-healthcare systems, biosensors and multi-sensor fusion, rehabilitation and robotic assistive technologies, adaptive and intelligent systems, and related medical and industrial applications. He is a recipient of the IEEE Third Millennium Medal and a Fellow of IEEE.