2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

# Moving object detection for visual odometry in a dynamic environment based on occlusion accumulation

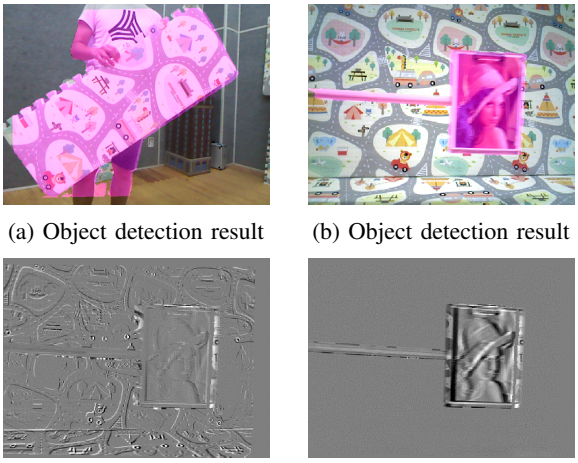
Haram Kim[1] , Pyojin Kim[2] and H. Jin Kim[1]

***Abstract*— Detection of moving objects is an essential capability in dealing with dynamic environments. Most moving object detection algorithms have been designed for color images without depth. For robotic navigation where real-time RGB-D data is often readily available, utilization of the depth information would be beneficial for obstacle recognition. Here, we propose a simple moving object detection algorithm that uses RGB-D images. The proposed algorithm does not require estimating a background model. Instead, it uses an occlusion model which enables us to estimate the camera pose on a background confused with moving objects that dominate the scene. The proposed algorithm allows to separate the moving object detection and visual odometry (VO) so that an arbitrary robust VO method can be employed in a dynamic situation with a combination of moving object detection, whereas other VO algorithms for a dynamic environment are inseparable. In this paper, we use dense visual odometry (DVO) as a VO method with a bi-square regression weight. Experimental results show the segmentation accuracy and the performance improvement of DVO in the situations. We validate our algorithm in public datasets and our dataset which also publicly accessible.**

## I. Introduction

Dynamic environments still pose a challenge in robotics. A capability to detect moving objects allows extending results of various studies performed for static scenes to dynamic environments. Although much research such as the background subtraction method has been carried out to recognize moving objects, it is ineffective in scenes where moving objects occupying a major portion of the image, (which we refer to as "moving objects *dominate* the scene"). Such situations happen frequently when moving objects are nearby and should be treated as a top priority.

To address the above-mentioned problems arising from dynamic environments, we propose an occlusion accumulation method. The proposed method utilizes both depth and camera pose information obtained from VO to detect moving objects without a particular background model. Some direct VO methods using robust regression weight [1] are not able to track the accurate camera pose by falling into a local minimum when a moving object dominates the scene. In this work, we use the estimated camera pose up to the previous image to distinguish the moving objects and exclude them from the images to find the next camera pose. If only the robust regression in the optimizer (bi-square regression in this paper) is used, the camera pose is likely to be estimated with respect to the moving object when the moving object dominates the scene. In the proposed method that estimates the camera pose with robust regression, the method excludes the moving object before it dominates the scene. Unlike the other mapping methods where memory usage increases each time an image is processed, the memory requirement of the proposed method is not heavy regardless of a long duration of image sequences, because it continuously updates the single occlusion information.



(a) Object detection result (b) Object detection result

(c) Residual image from the robust DVO (d) Residual image from the proposed method

Fig. 1: Results of moving object detection. (a), (b) are the results of detecting moving objects, and (a) contains a large moving object which dominates the scene with a freely moving camera. (c) Typical DVO only approach reaches a local minimum whereas (d) the proposed method demonstrates that the global optimum to the background is reached.

We warp the previous depth image using the estimated camera pose and subtract with the current depth image. Accumulating those subtracted values shows a similar result to the background subtraction without relying on background modeling. Our method is robust to the effect of the moving object which dominates the scene and avoids other problems that occur when the background is incorrectly modeled. It is possible to detect a moving object which was originally static and regarded as the background.

The purpose of our research is to detect moving objects and to improve the performance of arbitrary VO in dynamic environments. Our contributions are summarized as follows.

- We suggest a new method of the occlusion accumulation which can detect moving objects that dominate the scene, without a background model.

The SNU-Samsung smart campus research center (0115-20190023) at Seoul National University provides research facilities for this study.

[1] Lab for Autonomous Robotics Research (LARR), Seoul National University, Seoul, South Korea

[2] Computer Graphics and Vision Lab (GrUVi), Simon Fraser University, Burnaby, BC

- Our algorithm can improve the performance of any VO algorithm in a dynamic environment by augmenting it.
- The proposed algorithm can detect objects which have similar texture to the background and can be further improved by using an externally obtained camera pose.

## II. Related Work

There are several types of moving object detection algorithms [2] using background modeling, trajectory classification, etc. The methods in [3], [4], [5], [6], belonging to the category of background modeling, commonly extract feature points and clustered features to distinguish the background and the foreground. They warp the previous image with homography transform attained from features in the dominant cluster. Such methods show a quite accurate object segmentation result in the scenes dominated by the background, but they assume that the dominant cluster is from a background, and they take a few seconds to process.

In [7], the background and foreground are distinguished by a dual-mode single Gaussian model (SGM). The adaptive adjustment of the SGM learning rate reduces computation time, but some moving objects are perceived as background if the background is similar in color to the object.

In [8], trajectory classification is conducted. They collect points that are tracked for several frames and find the dominant camera motion by applying RANSAC on the trajectories for tracked points. The authors defined conditional probability on trajectory models and applied the maximum a posterior rule to distinguish the points included in the moving object. The authors of [9], [10] also propose a trajectory classification method by defining a low-dimensional descriptor for describing trajectory shape or spectral clustering with spatial regularity. They detect the outlier trajectories whose sum of the Euclidean distance to trajectories of the nearest neighbors exceeds a specified threshold or by using K-means clustering. The methods of [8], [9], [10] require an appropriate feature extraction method, and they may show sparse segmentation results due to utilization of the tracked features.

Because the aforementioned methods aimed to detect moving objects or tracking objects far from the camera, they used datasets where the background dominates the scene such as a surveillance system or sports relay. On the other hand, in robotic applications, because it is important to process in real-time and detect nearby objects which may occupy the scene, these studies can become less effective.

With the growing popularity of RGB-D cameras such as Kinect and ASUS Xtion PRO, moving object detection algorithms have begun to utilize them. The method in [11] estimates camera ego-motion using depth values, and then it uses a homogeneous transformation to warp the previous frame as in [3], [4], and [5]. The segmentation is achieved by the particle filter and vectorization method. Accurate image warping is performed using depth values, but the problem of objects occupying a major portion of the image is not addressed in these methods. In [12], they calculated a spatiotemporal graph Laplacians and spatial smoothness among the background pixels. Then, they adapt RPCA to incorporate two approaches for classifying foreground from background. Due to the high computational complexity of RPCA, computation time per frame is several seconds.

In the field of visual odometry, many studies have investigated camera pose estimation in dynamic environments by reducing the effect of the moving objects. In [1], [13] and [14], the camera pose is estimated by assigning the t-distribution or Huber norm weights, so that the dense visual odometry (DVO) method performs properly in some dynamic environments. The method of [15] labels the background with a non-parametric model and depth images obtained from the RGB-D camera, and estimates the camera pose by modifying the energy-based DVO. The algorithm would be ineffective when used for object detection, because they focus on labeling the background to estimate pose stably and does not label the moving object.

The method suggested in [16] performs rigid motion segmentation on RGB-D camera by a grid-based optical flow. The process of flow calculation and spatial, temporal segmentation was conducted in real-time, but the resulting segmentation [16] was a low resolution that is not as clear as the silhouette-shaped segmentation.

In [17], the authors propose a robust camera pose estimation in dynamic environments and 3D scene-flow estimation on the moving object by applying depth geometric clustering and checking whether clusters match the camera motion of images. They labeled the foreground and background as probability, and utilize the probability label as a weighing factor on pose estimation to perform well in dynamic environments. The algorithm does not clearly distinguish moving objects and the specified number of clusters would affect the performance.

## III. Moving Object Segmentation

Various factors such as illumination change, moving object, camera ego-motion induce image changes. Assuming that the photo-consistency assumption is not violated, in a static scene that involves moving objects with fixed camera, image change indicates the moving objects. In a dynamic scene which has arbitrary camera motion, we can attain a motionless image by warping the previous image with the estimated camera pose. We use the camera pose estimated from DVO with robust regression weight in this paper. We will refer to DVO with robust regression weight as robust DVO in this paper and more details about robust regression weight will be covered in Section IV. After attaining a motionless image, the dynamic scene can be treated as a static scene. We focus on this feature and detailed explanation will follow.

### A. Occlusion Accumulation

When a moving object roams in the scene, depth can be changed. The depth change can be classified into two kinds. The first one is occlusion: the object will occlude the background in the head of the direction of movement. The second one is reappearance: the background will appear in

the tail of the direction of movement. We assume that the depth of background is larger than that of the moving object. Object which pass the rear of the static background such as pillar, tree, etc. do not affect the depth value. We consider the pixel whose depth value changes to a smaller value as a pixel of the moving object.

The occlusion map and the warping function are defined as

$$\Delta Z_i(u, \xi_i^{i+1}) = Z_i(\mathrm{w}(u, \xi_i^{i+1})) - Z_{i+1}(u) \tag{1}$$

and

$$\mathrm{w}(u, \xi_i^{i+1}) = \pi(\exp(\xi_i^{i+1})\pi^{-1}(u, Z_i(u))) \tag{2}$$

where $Z_i(u)$ is the *i*th depth image and $u$ is a pixel on image $\Omega(640 \times 480)$. $\xi_i^{i+1} \in \mathbb{R}^6$ represents the 6-DOF camera pose parameter between the *i*th frame and the (*i*+1)th frame. The project function $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ maps a 3D point into a 2D image pixel. The term $\exp(\xi_i^{i+1}) \in \mathrm{SE}(3)$ represents the transformation matrix for the corresponding camera pose parameter $\xi_i^{i+1}$.

If $\Delta Z_i > 0$, it means that the depth has become smaller, i.e. the occlusion has occurred. Otherwise, $\Delta Z_i < 0$ means that the depth has increased, i.e. the reappearance. We define the occlusion accumulation map by

$$\begin{aligned} A_{i+1}(u) &= \Delta Z_i(u, \xi_i^{i+1}) + \tilde{A}_i(\mathrm{w}(u, \xi_i^{i+1})) \\ &\approx \Delta Z_i(u, \xi_i^{i+1}) + \sum_{k=1}^{i} \Delta Z_k(\mathrm{w}(u, \xi_{k+1}^{i}), \xi_k^{k+1}) \\ &\approx Z_{k_u}(\mathrm{w}(u, \xi_{k_u}^{i+1})) - Z_{i+1}(u) \end{aligned} \tag{3}$$

The initial occlusion accumulation map is set as $A_1(u) = 0$ for $\forall u \in \Omega$ and $k_u$ is the index of the frame which first observed the pixel $u$ of the current frame. The $\tilde{A}(u)$ is truncated $A(u)$ and described later in Eqs.(5) and (6),

The warped pixels before the index at $k_u$ would be out of the observation window $\Omega$ and there is nothing to calculate. The value of $\Delta Z$ would reveal zero values for these indices.

When the truncation steps for $A(u)$ have not been triggered, the second and third approximate equalities in Eq.(3) would be satisfied by $A(u) = \tilde{A}(u)$. One can interpret the occlusion accumulation map as the result of subtraction between $k_u$th frame and current frame. In contrast to the background subtraction, we utilize the occlusion map, $\Delta Z_i(u, \xi_i^{i+1})$. Thus, our method does not need to depend on recognition of the background or 3D mapping.

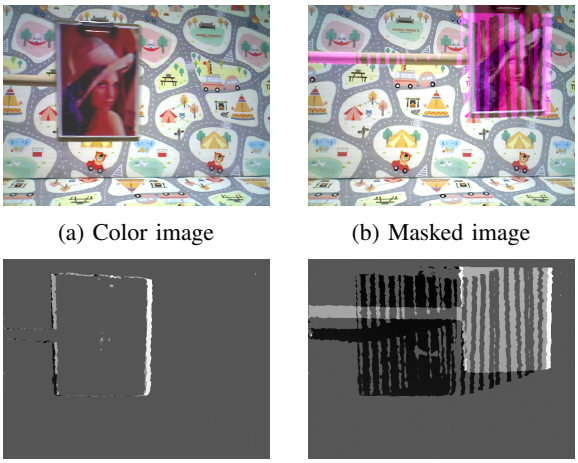
Existing depth sensors have the depth error, which quadratically grows along the depth value in [17]. To deal with such measurement uncertainty, we set a threshold as

$$\tau_\alpha(u) = \alpha \cdot Z(u)^2 \tag{4}$$

Although we set the threshold to be larger than zero, unwanted error values can exceed the threshold through the accumulation. Furthermore, negative values can be accumulated, so that occlusion could not exceed the threshold at all. In order to reduce their effect, we truncate the occlusion accumulation map $A_i(u)$ to $\tilde{A}_i(u)$ by the following:

$$\tilde{A}_i(u) = 0 \quad \text{for } A_i(u) \leq \tau_\alpha(u) \tag{5}$$

$$\tilde{A}_i(u) = 0 \quad \text{for } \Delta Z_i(u, \xi_i^{i+1}) \leq -\tau_\beta(u)\ . \tag{6}$$



(a) Color image (b) Masked image

(c) $\Delta Z(u)$ (d) $A(u)$

Fig. 2: Illustration of the occlusion accumulation method. $A(u)$ represents the accumulation of $\Delta Z(u)$ over time. A bright area is a region where $A(u)$ shows a positive value and is considered as a moving object, and a dark area is a region where $A(u)$ shows a negative value and the background is revealed again. The middle gray represents a value of zero. The moving object detection results from Eq.7 is shown in (b). The area considered as a moving object is painted purple.

The truncation step in Eq.(5) could disturb detecting the moving object which slowly approaches toward the camera. However, once the moving object is detected, the small values of $\Delta Z_i(u, \xi_i^{i+1})$ will be reflected. When the background reappears, $A_i(u)$ would not be lower than the threshold $\tau_\alpha(u)$ due to depth error accumulation. The truncation step in Eq. (6) helps to detect the background reappearance. The moving object that moves away from the camera hardly exceed the threshold. Even so, if it exceeds the threshold, the object would soon fade away in the scene. The result of the occlusion accumulation is shown in Fig. 2. In order to help understanding, $A(u)$ has not been truncated using Eq. (5) through every iteration yet. Since the depth value is not measured near the object, a stripe-like result appears in Fig. 2d.

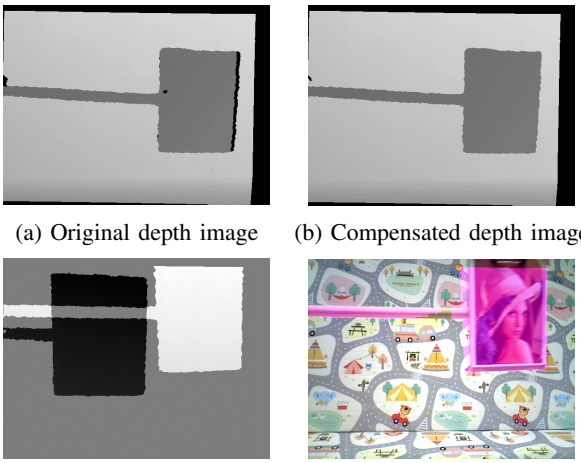
After calculating $A_i(u)$, we truncate the occlusion accumulation map to distinguish moving objects and background. The background map $B_i(u)$ would reveal 0 if a moving object shows proper occlusion. Otherwise, if the moving object has passed and the background reappears again, $B_i(u)$ would reveal 1, i.e.,

$$B_i(u) = \begin{cases} 0 & \text{if } A_i(u) > \tau_\alpha(u) \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

The object mask can be calculated by inverting $B_i(u)$.

### *B. Depth Compensation*

Often, the depth images have invalid depth on an edge of the object or near the border of an image window. Its effect is shown in Fig. 2d. The segmentation in Fig. 2b has empty spaces over the moving object. Some of those invalid depth



(a) Original depth image (b) Compensated depth image

(c) $A(u)$ (d) Masked image

Fig. 3: Result of depth compensation. A raw depth image (a) is compensated as (b). Compared to Fig.1, the unwanted stripe-like result is corrected.

values in the current frame can be compensated by

$$Z_i(\bar{u}) = Z_{i-1}(\mathrm{w}(\bar{u}, \xi_i^{i-1})) \tag{8}$$

where the unmeasured depth on the pixel $\bar{u}$ satisfies $Z(\bar{u}) = 0$. The compensated depth is used in the next depth image. The result of the depth compensation is shown in Fig. 3. The original depth image has the unmeasured area that is colored as black, near the border and the boundary of the object. The unmeasured area on boundary of the object is filled with the previous depth image as in Fig. 3b. From the compensated depth image, $A(u)$ is now calculated reasonably (Fig. 3c), so that the object segmentation could cover the entire moving object (Fig. 3d).

### C. Occlusion Prediction on Newly Discovered Area

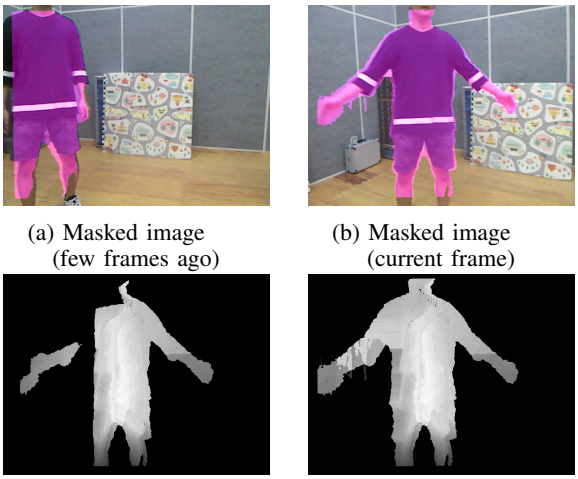
Suppose that there is a newly discovered area which has to be classified as the background or the moving object in a dynamic scene. The occlusion map $\Delta Z_i(\tilde{u}, \xi_{i-1}^i)$ cannot be correctly calculated for $\tilde{u}$, i.e. the pixels of the newly discovered area, because the warped pixel onto the previous image $\mathrm{w}(\tilde{u}, \xi_i^{i-1})$ is not in the image window $\Omega$. We define the newly discovered area as

$$\tilde{\Omega} = \{\tilde{u} | \mathrm{w}(\tilde{u}, \xi_i^{i-1}) \notin \Omega\} \tag{9}$$

We suggest a method of predicting $A(\tilde{u})$ with the fast marching method as the following:

$$A_i(\tilde{u}) = \frac{\sum_{\delta u \notin \tilde{\Omega}} \{A_i(\delta u) + \nabla_{\delta u} Z_i(\tilde{u})\}}{\sum_{\delta u \notin \tilde{\Omega}} 1} \tag{10}$$

The symbol $\delta u$ means the nearest pixels of $\tilde{u}$ whose accumulated value $A(\delta u)$ has been calculated, and $\nabla_{\delta u} Z_i(\tilde{u}) = Z_i(\tilde{u}) - Z_i(\delta u)$ is the gradient of depth map with respect to $\delta u$. Because $A(u)$ has correspondence with the depth map, we predict $A(\tilde{u})$ with the gradient of the depth map and the occlusion accumulation map for the nearest pixel $\delta u$ adjoining the known area. In other words, we interpolate $A(\tilde{u})$ with the average of $A(\delta u) + \nabla_{\delta u} Z_i(\tilde{u})$. After that,



(a) Masked image (few frames ago) (b) Masked image (current frame)

(c) $A(u)$ without prediction (d) $A(u)$ with prediction

Fig. 4: Example of occlusion prediction on the newly discovered area. When a new area is detected while the camera moves, there is no depth information in a few frames ago. Occlusion map has zero value on such area. (c) shows the effect of the unmeasured depth values due to a newly discovered area. After executing the occlusion prediction method, $A(u)$ has proper occlusion map values on the newly discovered area shown in (d) and the moving object detection result obtained from (d) is shown in (b).

we update the newly discovered area by $\tilde{\Omega} \leftarrow \tilde{\Omega} - \{\delta u\}$, and repeat Eq. (10) until $\tilde{\Omega} = \phi$. This process operates on two consecutive frames. Since the effect of the newly discovered area is not noticeable in two consecutive frames, we compared the masked image with some frame interval in Fig. 3d.

In the positive area of the predicted $A(u)$ which is considered as a moving object, the background depth has never been detected, because the background was continuously occluded by the moving object until the current image. The algorithm could fail to recognize such appearance of the background. Even if the background appears, $A(u)$ may not be lower than the threshold in Eq. (7). Then the truncation of Eq. (6) is triggered so the background recognition works properly. The result of the prediction is depicted in Fig. 4. When there is no nearest moving object outside the predicted area, we remove $A(u)$ of the predicted area to prevent error propagation. Also, if there are small-sized moving object labels, we suppress them as they are usually negligible.

## IV. Robust Pose Estimation

In this paper, we use DVO for camera pose measurement. Initially, we estimate the camera motion using only the sequence of RGB-D images. After detecting moving objects, the camera motion is estimated considering background mask $B(u)$. The method using robust regression [1] effectively estimate ego-motion over a small region of moving object, but it reaches a local minimum solution when the region of moving object is large as shown in Fig.1, because the background is confused with the moving object. On the other hand, the remaining moving object area after excluding the


Fig. 5: Segmentation result for tested sequences in the order shown in Table.I. The first row shows the masked image and the second row represents the ground truth. The result of the proposed method with VO and external pose estimation is presented orderly in the third and fourth rows (except the TUM dataset for the reasons described in text). Since the depth is not measured on the boundary of the image, moving objects are not labeled in those areas.

moving object detected up to the current frame is small enough to find upright camera pose. Thus, our method does not confuse the background and the moving object thanks to the accurate camera pose. The camera pose estimation is achieved by minimizing the cost function as shown in Eq. (11). In the cost function, we use the bi-square norm shown in Eq. (13) which completely ignores the effect of the outliers. This property makes it possible to obtain the camera pose $\xi$ based on background pixels which have a relatively small residual.

$$\xi_i^{i+1} = \underset{\xi}{\operatorname{argmin}} \sum_{u \in \Omega} B_i(\mathrm{w}(u,\xi)) J_i(u,\xi) \tag{11}$$

where

$$J_i(u,\xi) = \rho_{k_I}(\Delta I_i(u,\xi)) + \gamma \cdot \rho_{k_Z}(\Delta Z_i(u,\xi)) \tag{12}$$

$$\rho_k(e) = \begin{cases} \frac{k^2}{6}\left\{1 - \left[1 - (\frac{e}{k})^2\right]^3\right\} & \text{for } |e| \le k \\ \frac{k^2}{6} & \text{for } |e| > k \end{cases} \tag{13}$$

The symbol $I_i$ in Eq. (12) is the $i$th color image and $k$ is the user-defined bi-square threshold. The residual values $\Delta I$ and $\Delta Z$ that are larger than $k$ do not influence the optimization process. In this paper, we use the levenberg-marquardt optimizer, and the parameter values $k_I = 48/255$, $k_Z = 0.5$, and $\gamma = 0.001$ are used.

## V. Experimental Results

To evaluate our algorithm, we need RGB-D datasets in the presence of camera motion, ground truth data for moving objects, and camera trajectory. The dynamic object sequences of TUM dataset [18] contain the ground truth camera pose and RGB-D images, but the ground truth segmentation for moving object was not provided because the moving object such as a sitting person whose arm is moving could not be segmented clearly. Thus, we used TUM dataset only to evaluate relative pose error. Additionally, we evaluate the performance of the proposed moving object detection and the pose estimation method with the dataset in [16] and our dataset. The data sequences from [16] are named with the prefix 'ICSL' in table I.

We collected dataset with an RGB-D camera of Kinect V1, and obtained the ground truth pose of the camera using VICON. In addition, we manually obtained the ground truth segmentation of moving objects in pixel-wisely.

It contains 5 sequences for static camera, whose name starts with '*static*' and 4 sequences for the dynamic environment whose name starts with '*dynamic*' where camera motion and moving objects exist simultaneously. The sequences '*static_tree*','*static_man*' respectively contain a tree and a man as a moving object. The sequence of '*static_board*' contains a large-sized moving object which could be misrecognized as the background. In '*static_destruct*', '*static_construct*' sequences, there are situations where a building-shaped object is brought in and then removed. These sequences induce background model changes. In the sequence '*dynamic_toss*', two people toss a doll to each other. It shows that multiple objects and fast objects can also be detected with our algorithm. The dataset and detailed information can be found here:

https://haram-kim.github.io/LARR-RGB-D-datasets/

The results of the object segmentation are shown in Table I and Fig. 5. We used the F1-score as a criterion for the moving object classification evaluation. The precision refers to how many of the pixels identified as moving objects are ground truth moving objects pixels and the recall refers to how many of the moving objects pixels in the image were correctly identified as moving objects. The F1-score is the harmonic mean of the precision and the recall. We calculated F1-score for each frame, and we averaged it in each sequence. The closer the score is to 1, the better the performance of moving object segmentation is. We use the symbol '$\times$' when quantitative evaluation is meaningless due to extremely poor performance, and '-' for the algorithm that failed due to very few inliers in the corresponding sequence.

We evaluated the segmentation performance of proposed method not only with robust DVO, but also with VICON data as an externally obtained camera pose. The proposed method with camera pose of VICON data mostly performs better than the method with VO.

In general static situations, all the compared algorithms perform well, but the performance of [5], [11] deteriorates in '*static_destruct*' and '*static_construct*' sequences, where the moving object appear at the beginning or the

TABLE I: Object segmentation result (F1-score) & relative pose error (RMSE)

| Sequences | [5] | [11] | Proposed method | Ours with ext. pose | Robust DVO | | Proposed method | | Joint VO-SF [17] | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | tr.(m) | rot.(°) | tr.(m) | rot.(°) | tr.(m) | rot.(°) |
| static_tree | **0.9385** | 0.8873 | 0.8821 | 0.8892 | 0.102 | 3.313 | 0.061 | **1.856** | **0.032** | 3.969 |
| static_board | 0.9044 | **0.9256** | 0.9182 | 0.9177 | × | × | **0.292** | **4.839** | 0.584 | 10.916 |
| static_man | 0.7350 | **0.8975** | 0.8739 | 0.8755 | 0.280 | 4.024 | 0.173 | 4.944 | **0.072** | **1.045** |
| static_destruct | 0.5267 | 0.5565 | 0.8391 | **0.9038** | 0.535 | 4.818 | 0.336 | 4.495 | **0.131** | **3.014** |
| static_construct | 0.3948 | 0.8362 | **0.8868** | 0.8396 | 0.809 | 10.433 | 0.153 | 3.821 | **0.078** | **1.272** |
| ICSL_place_items | 0.7139 | 0.6494 | 0.7204 | **0.7366** | 0.041 | 1.082 | **0.030** | **0.893** | 0.094 | 5.379 |
| ICSL_two_objects | 0.5073 | 0.8256 | **0.8583** | 0.8580 | 0.170 | 6.493 | **0.013** | **0.344** | - | - |
| dynamic_board | × | 0.6527 | **0.9264** | 0.9247 | × | × | **0.111** | **1.939** | 0.135 | 4.644 |
| dynamic_man1 | × | 0.4704 | 0.8123 | **0.8287** | 0.255 | 9.027 | **0.157** | **4.108** | 0.223 | 12.949 |
| dynamic_man2 | × | × | **0.8975** | 0.8955 | 0.646 | 11.286 | **0.166** | **2.165** | 0.206 | 6.180 |
| dynamic_toss | × | 0.4395 | 0.7259 | **0.7975** | 0.635 | 8.546 | **0.324** | **2.312** | 0.378 | 4.028 |
| ICSL_fast_object | × | × | 0.8639 | **0.8779** | 0.318 | 13.543 | **0.092** | **3.603** | - | - |
| ICSL_slow_object | × | 0.7424 | 0.8971 | **0.9142** | 0.407 | 17.585 | **0.091** | **3.779** | - | - |
| TUM_sitting_static | | | | | 0.037 | 0.972 | **0.035** | **0.961** | 0.045 | 1.699 |
| TUM_sitting_xyz | | | | | 0.078 | 2.027 | **0.073** | **1.860** | 0.205 | 4.066 |
| TUM_walking_static | | | | | 0.370 | 2.581 | **0.217** | **0.197** | 0.249 | 4.173 |
| TUM_walking_xyz | | | | | 0.974 | 15.871 | **0.259** | **4.069** | 0.659 | 12.370 |

end of the video. The performance of [5] degrades even if the background and the moving object are different in color. The sequences of dynamic environments contain a difficult situation in that some newly discovered area found by the camera is covered by moving objects, thus the background of that area has never been revealed. The algorithms in [5], [11] suffer in those situations so that the results were greatly affected. Since we do not use the background model and predict $A(u)$ for the newly discovered area, our method distinguished moving objects stably in '$static\_destruct$', '$static\_construct$' and other dynamic environment sequences. In the dataset of [16], our method correctly distinguishes the moving object from the background which have similar texture.

To evaluate the performance improvement of the VO when augmented with our algorithm, the translational and rotational relative pose error metric in [18] is adopted. We set the time parameter $\Delta = 150$, which means that we evaluate the drift per 5 seconds recorded at 30 Hz. The results are shown in Table. I, Fig. 6. Even though RMSE is calculated with the frames that do not have a moving object, our algorithm improves the estimation result of the robust DVO significantly in various datasets, especially in dynamic environments.

As a result of applying the joint VO-SF in [17], an algorithm based on DVO, it mostly performs better than DVO and shows smaller error than our proposed method in a *fixed* camera. For the $static\_board$ sequence where the moving object is dominant in the scene, the joint VO-SF method show large relative pose error than our method. Interestingly, the results of $TUM\_sitting$ sequences show that the bi-square weight method is more robust than the Cauchy weight based method, which can be explained by their difference in outlier rejection intensity. The proposed method, compared to robust DVO and Joint VO-SF in dynamic environment, performs better in both rotational and translational motion regardless of a moving object which dominates a scene.

As can be seen in Fig. 5, although the moving objects are well separated, error exists due to the lack of texture in the background scene. We expect that more accurate pose estimation in a dynamic environment can be achieved by applying other VO algorithms which show better performance on pose estimation in a static environment.

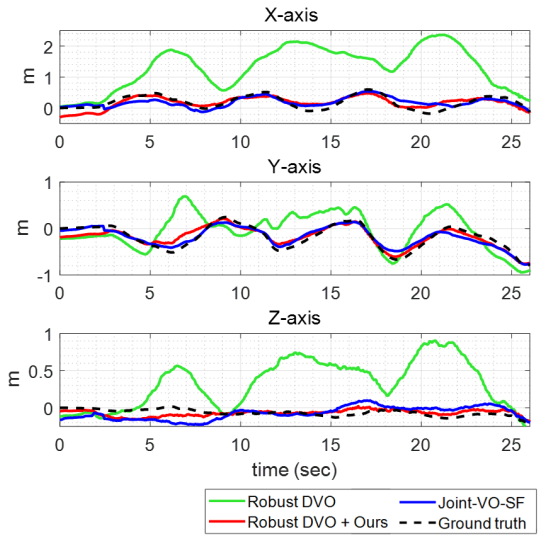



Fig. 6: Translation estimation results on the sequence '$dynamic\_man2$'. The robust DVO fails to estimate the ego-motion due to a moving object. When combined with our algorithm, the performance improves significantly.

## VI. Conclusions

We proposed a novel moving object detection method which utilizes occlusion accumulation and camera pose instead of a background model. To do this, we also presented the depth compensation on the unmeasured area and the occlusion prediction on the newly discovered area. Our method could detect the moving object which dominates the scene and improved the performance of robust DVO. In future work, we will combine the proposed method for robotic navigation tasks, which is designed to easily integrate with SLAM algorithms or obstacle avoidance algorithms.

## REFERENCES


[1] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3748–3754.

[2] M. Yazdi and T. Bouwmans, "New trends on moving object detection in video images captured by a moving camera: A survey," *Computer Science Review*, vol. 28, pp. 157–177, 2018.

[3] A. Viswanath, R. K. Behera, V. Senthamilarasu, and K. Kutty, "Background modelling from a moving camera," *Procedia Computer Science*, vol. 58, pp. 289 – 296, 2015, second International Symposium on Computer Vision and the Internet (VisionNet'15).

[4] S. Minaeian, J. Liu, and Y.-J. Son, "Effective and efficient detection of moving targets from a uav's camera," *IEEE transactions on intelligent transportation systems*, vol. 19, no. 2, pp. 497–506, 2018.

[5] W.-C. Hu, C.-H. Chen, T.-Y. Chen, D.-Y. Huang, and Z.-C. Wu, "Moving object detection and tracking from video captured by moving camera," *Journal of Visual Communication and Image Representation*, vol. 30, pp. 164–180, 2015.

[6] D. Zamalieva and A. Yilmaz, "Background subtraction for the moving camera: A geometric approach," *Computer Vision and Image Understanding*, vol. 127, pp. 73–85, 2014.

[7] K. Moo Yi, K. Yun, S. Wan Kim, H. Jin Chang, and 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.

[8] Y. Sheikh, O. Javed, and T. Kanade, "Background subtraction for freely moving cameras," in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 1219–1225.

[9] X. Yin, B. Wang, W. Li, Y. Liu, and M. Zhang, "Background subtraction for moving cameras based on trajectory-controlled segmentation and label inference." *Ksii Transactions on Internet & Information Systems*, vol. 9, no. 10, 2015.

[10] T. Brox and J. Malik, "Object segmentation by long term analysis of point trajectories," in *European conference on computer vision*. Springer, 2010, pp. 282–295.

[11] Y. Sun, M. Liu, and 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.

[12] S. Javed, T. Bouwmans, M. Sultana, and S. K. Jung, "Moving object detection on rgb-d videos using graph regularized spatiotemporal rpca," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 230–241.

[13] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense rgb-d mapping," 2013.

[14] C. Kerl, "Odometry from rgb-d cameras for autonomous quadrocopters," *Master's Thesis, Technical University*, 2012.

[15] D.-H. Kim and J.-H. Kim, "Effective background model-based rgb-d dense visual odometry in a dynamic environment," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1565–1573, 2016.

[16] S. Lee and H. J. Kim, "Real-time rigid motion segmentation using grid-based optical flow," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 1552–1557.

[17] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers, "Fast odometry and scene flow from rgb-d cameras based on geometric clustering," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3992–3999.

[18] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and 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.