Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment

Article *in* Robotics and Autonomous Systems · April 2019

DOI: 10.1016/j.robot.2019.03.012

CITATION
1

READS
263

5 authors, including:

Linhui Xiao
Chinese Academy of Sciences
1 PUBLICATION **1** CITATION

SEE PROFILE

Xudong Zou
Chinese Academy of Sciences
30 PUBLICATIONS **177** CITATIONS

SEE PROFILE

Robotics and Autonomous Systems 117 (2019) 1–16

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment

Linhui Xiao [a,b], Jinge Wang [a,b], Xiaosong Qiu [a,b], Zheng Rong [c], Xudong Zou [a,b,*]

[a] State Key Laboratory of Transducer Technology, Institute of Electronics, Chinese Academy of Sciences, Beijing 100190, China
[b] School of Electronic, Electrical and Communication, University of Chinese Academy of Sciences, Beijing 100049, China
[c] Science and Technology on Microwave Imaging Laboratory, Institute of Electronics, Chinese Academy of Sciences, Beijing 100190, China

HIGHLIGHTS

- An SSD object detector is constructed to detect dynamic objects with prior knowledge in the newly detection thread.
- Missed detection compensation algorithm based on the speed invariance in adjacent frames is proposed.
- Selection tracking algorithm is proposed to eliminate the dynamic objects and improve the robustness and accuracy of the system.
- A feature-based Dynamic-SLAM system combined semantic information is constructed to realize the detection of dynamic environment in robot localization and mapping.

ARTICLE INFO

Article history:
Received 9 October 2018
Received in revised form 19 January 2019
Accepted 21 March 2019
Available online 2 April 2019

Keywords:
Simultaneous localization and mapping (SLAM)
Semantics
Object detection
Dynamic environment

ABSTRACT

When working in dynamic environment, traditional SLAM framework performs poorly due to interference from dynamic objects. By taking advantages of deep learning in object detection, a semantic simultaneous localization and mapping framework named Dynamic-SLAM is proposed, in order to solve the problem of SLAM in dynamic environment. First, based on the convolutional neural network, an SSD object detector which combines prior knowledge is constructed to detect dynamic objects in the newly detection thread at semantic level. Then, in view of low recall rate of the existing SSD object detection network, a missed detection compensation algorithm based on the speed invariance in adjacent frames is proposed, which greatly improves the recall rate of detection. Finally, a feature-based visual SLAM system is constructed, which processes the feature points of dynamic objects through a selective tracking algorithm in the tracking thread, to significantly reduce the error of pose estimation caused by incorrect matching. The recall rate of the system is increased from 82.3% to 99.8% compared with the original SSD network. Several experiments show that the localization accuracy of Dynamic-SLAM is higher than the state-of-the-art systems. The system successfully localizes and constructs an accurate environmental map in real-world dynamic environment by using a mobile robot. In sum, our experimental demonstrations verify that Dynamic-SLAM shows improved accuracy and robustness in robot localization and mapping comparing to the state-of-the-art SLAM system in dynamic environment.

1. Introduction

Simultaneous Localization and Mapping (SLAM) [1] has been developed for about thirty years and has become a crucial technology in the field of robotics, automation and computer vision. Among the different sensor modalities, cameras are cheaper and provide rich visual and environmental information, which has tremendous room for growth in the future. Currently, applications such as Micro Aerial Vehicle (MAV), Unmanned Ground Vehicles (UGV), autonomous driving, Virtual Reality (VR), and Augmented Reality (AR) require the reliable localization and mapping result provide by SLAM technology. However, the pose estimation of robustness and accuracy in visual SLAM, especially in the dynamic environment (visual SLAMIDE), still have many crucial challenges.

In order to make the visual SLAM system work normally in dynamic environment, the usual approach is to avoid using the feature points on dynamic objects. Therefore, it is necessary to calculate the position of the dynamic object in advance. Meanwhile, object detection and semantic segmentation based on deep learning have achieved remarkable results in this respect. Applying the deep learning technology to SLAM is essential to solve the

* Correspondence to: Room 1121, Kedian Building, No. 9 North Section, Zhongguancun, Haidian District, Beijing, 100190, China.
E-mail addresses: xiaolinhui16@mails.ucas.ac.cn (L. Xiao), xdzou@mail.ie.ac.cn (X. Zou).

https://doi.org/10.1016/j.robot.2019.03.012
0921-8890/

problem of robot localization and mapping in the real dynamic environment.

In this study, Dynamic-SLAM which constructed on the base of ORB-SLAM2 [2] is a semantic monocular visual SLAM system based on deep learning in dynamic environment. Dynamic-SLAM mainly includes a visual odometry frontend, which includes two threads and one module, namely tracking thread, object detection thread and semantic correction module; and a SLAM back-end, mainly including local mapping thread and loop closing thread. The framework is illustrated in Fig. 1, which will be introduced in detail in Section 3. Dynamic-SLAM can successfully run in real-time in complex indoor and outdoor dynamic environments. It solves the problem of localization, BA, loop closing, sparse reconstruction and other issues in the presence of pedestrians and vehicles. It can satisfy the localization and mapping assignment in the normal real environment. The main contributions of this paper are as follows:

- In view of the low recall rate of the existing SSD object detection network, a missed detection compensation algorithm based on the speed invariance in adjacent frames is proposed for SLAM system, which greatly improves the recall rate for detection and provides a good basis for the following module.
- A selection tracking algorithm is proposed to eliminate the dynamic objects in a simple and effective way, which improves the robustness and accuracy of the system.
- A feature-based visual Dynamic-SLAM system is constructed. Based on the SSD convolutional neural network, deep learning technology is constructed to a newly object detection thread, which combines prior knowledge to realize the detection of dynamic objects at semantic level in robot localization and mapping.

In the rest of this paper, SLAM and deep learning related work are discussed in Section 2. The main work is described and demonstrated in Section 3 in detail, and a series of evaluation results and real-world test for the robot platform are presented in Section 4. Finally, the conclusion is made in Section 5.

2. Related work

2.1. Visual SLAM

Visual SLAM has achieved rapid development in the last decade. It has drawn the attention of researchers because of its low cost and small size. Davison et al. is the pioneer of visual SLAM. In 2007, he proposed Mono-SLAM [3] for the first time to implement a monocular real-time SLAM system. Subsequently, the PTAM [4] (Parallel Tracking and Mapping) proposed by Klein et al., creatively divided the entire system into two threads: tracking and mapping, which became the benchmarks for the follow-up SLAM researchers. In 2014, the LSD-SLAM proposed by Engel et al. [5] clarified the relationship between the pixel gradient and the direct method. At the same time, Forster et al. [6,7] proposed a fast Semi-direct monocular Visual Odometry (SVO), which combines the feature point and direct tracking optical flow method. Later, frameworks such as DSO [8], VINS-Mono [9] also used the direct method. This method saves computing resources in tracking and matching, however, the insensitivity to features is a fatal weakness for its further application in SLAM.

To build a complete and robust SLAM framework, the use of feature points is essential. On the one hand, feature extraction and matching can ensure the accuracy of pose estimation in the SLAM tracking. On the other hand, feature method can extract more effective information from visual images, such as semantics, object recognition, feature localization, etc. In fact, the OKVIS [10] visual inertial odometry framework proposed by Leutenegger, ORB-SLAM [11] and ORB-SLAM2 [2] proposed by Mur-Artal et al., are the successful applications of SLAM based on the feature tracking.

The ORB-SLAM2 framework is the successor of the classic SLAM thread model. To complete the SLAM system, it uses the ORB [12] feature point and three main parallel threads: tracking thread for tracking feature points in real time, local mapping thread for constructing local Bundle Adjustments (BA) map, and loop closing thread for correcting the accumulated drift and performing a pose-graph optimization. It can enable the system to run for a long time under large scenes and large loops, thus ensuring the global consistency of the trajectory and the map. ORB-SLAM2 is one of the best performing frameworks for localization and mapping. However, it is necessary to make further exploration for it still has many shortcomings in dealing with dynamic environment problems.

2.2. SSD object detection network

At present, object detection and semantic segmentation methods based on the convolutional neural network have made continuous breakthroughs in speed and accuracy. The performances of deep learning on related tasks have gradually surpassed other methods and become mainstream.

Compared with image recognition, object detection is more difficult because it not only needs to classify different objects in the picture, but also requires giving the position of each object. The Neural Network methods headed by R-CNN [13], such as SPP-Net [14], Fast R-CNN [15], Faster R-CNN [16] etc., use the convolutional neural network to automatically learn features and avoid the limitation of manual design features. They obtain the object detection results through several operations such as candidate frame extraction, feature extraction and classification. Although these methods have high precision, it is very time-consuming because candidate frame extraction and object recognition are performed in two steps. In contrast, the YOLO (You Only Look Once) [17] algorithm discards the intermediate step of generating candidate region, which directly processes the regression problem for each bounding box through a single convolutional neural network and predicts the probability of the corresponding category, thus achieves a fast speeds and maintain a good accuracy.

SSD (Single Shot MultiBox Detector) proposed by Liu et al. [18], draws on the Anchor boxes in the Faster-RCNN to adapt the objects of different shapes. It is the recognizer of feature point in Dynamic-SLAM, which uses the basic network structure of VGG16, leaving the first five layers unchanged, while converting the *fc6* and *fc7* layers into two convolutional layers by using *à trous* [19] algorithm, and then adding three convolutional layers and an average pooling layer behind. Finally, the final detection results are obtained through Non-Maximal Suppression (NMS). By omitting the generation step of initial candidate box, it enables the entire object detection process to be completed in a single network, thereby achieving high detection efficiency (46 fps, Titan X) and detection accuracy (77.2%). In the premise of guarantee speed, SSD has obtained enough accuracy to surpass the method based on the candidate box.

2.3. SLAMIDE problem and deep learning

As early as 2003 [20], there have been some studies on SLAM in Dynamic Environments (SLAMIDE) problem. The conventional SLAM methods in dynamic environment can be classified into two categories: one is the detection of dynamic object and tracking [20]; the other is the detection of dynamic objects and filtering [21]. The former is summarized as the Detection and Tracking

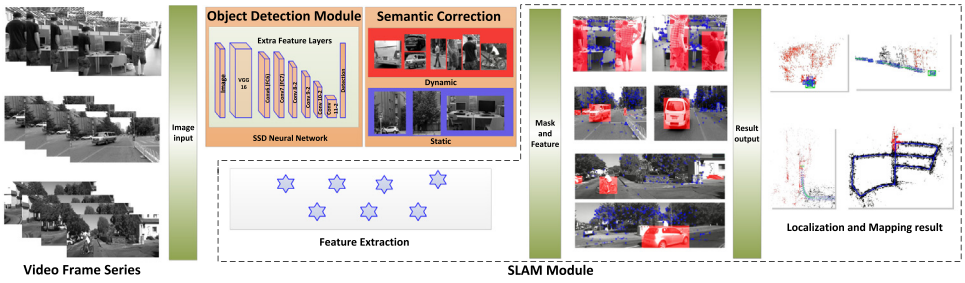


Fig. 1. System overview of Dynamic-SLAM. Images captured by monocular camera serve as the input data for the SLAM module and the object detection module. The output data of the object detection module is fed back to the SLAM module in real time after passing through the semantic correction module.

of Moving Objects (DATMO) [22] method. Some researchers [23] suggested that there are at least four related data association techniques in dealing with SLAMIDE issues after obtaining a reasonable set of detected features, including GNN [24], JPDA [25], MHT [26] and RANSAC [27]. Charles et al. [28] proposed a framework that combines least-squares with sliding window optimization and generalized expectation maximization. Chen et al. [29] proposed a SLAM in Dynamic Environment system that only relies on landmarks and without priori information. Walcott-Bryant et al. [30] proposed a Dynamic Pose Graph (DPG) model for dynamic environments. In 2018, Bahraini et al. [31] proposed an approach to segment and track multiple moving object by using MultiLevel-RANSAC. In general, most visual researchers still focus on how to extract dynamic objects from adjacent frames by using feature detection methods. A potential drawback of feature detection method is that it will fail when the dynamic object is moving too slow or fast. Muhamad et al. [32] made an survey for the problems of visual SLAM and Structure from Motion (SfM) in dynamic environments. According to the categorization, Dynamic-SLAM is a robust visual SLAM.

SLAMIDE has always been an insurmountable problem in the SLAM field. The reason is that traditional SLAM theory is completely based on the assumption of static environment. Its challenges mainly come from two aspects: first, it is difficult to define a dynamic object from planar pixel; the second is dynamic objects are not easily detected and tracked.

Since deep learning has achieved favorable performance in object detection, many researchers have combined deep learning with SLAMIDE issues [32]. Zhang et al. [33] integrate a deep CNN model to improve the accuracy of terrain segmentation and make it more robust against wild environments. Some similar RGB-D SLAMIDE studies [34–37] have been developed by combining the state-of-the-art SLAM framework with deep learning network and achieved considerable results. Most of the SLAMIDE researchers focused on the Lidar SLAM and RGB-D SLAM because these sensors can directly get depth information to estimate the position of dynamic objects [38]. In contrast, the study of SLAMIDE in monocular visual system is very limited. Barnes et al. [39] present a self-supervised approach to ignoring the distractors in monocular camera images so that robustly estimating vehicle motion in urban dynamic environments. Bescos et al. [40] used a similar approach with ours. They combined ORB-SLAM2 with Mask RCNN [41] to implement monocular and stereo system in dynamic environment, and combined multi-view geometry models and deep learning algorithms to implement RGB-D system.

3. System overview and approach

In this study, Dynamic-SLAM is constructed on the base of ORB-SLAM2 [2] by adding a dynamic object decision module that incorporate semantics and optimizes the visual odometry algorithm using feature point. Robustness of Dynamic-SLAM is achieved by segmenting the static and dynamic features in the image and regarding the dynamic parts as outliers. Pose estimation and nonlinear optimization is computed based on the static feature points which calculated by the selective tracking algorithm, so as to avoid the interference of the dynamic environmental objects. In view of the low recall rate of the existing SSD object detection network, a missed detection compensation algorithm based on the basic motion model is proposed, which greatly improves the accuracy in the object detection module. Fig. 2 plots the brief framework of Dynamic-SLAM.

3.1. Missed detection compensation algorithm

In the course of experiment, it is found that the detection accuracy of the neural network is not enough for SLAM and can be further improved. Since the pictures do not have a significant correlation, the detection accuracy cannot be improved by the context information in the conventional object detection task. However, in SLAM, video frames arrive in time series, the detection results of the previous frames can be used to predict the next detection result, so as to avoid the missed or false detections in the next keyframe.

In the accuracy assessment of object detection tasks, two aspects are usually more concerned by researchers: Precision and Recall. For dynamic object detection task in the SLAMIDE problem, the latter is more important. In the event of missed or false detection, the difference between two adjacent images will cause sharp changes in the number of correct matching of feature points, thus leading to instability of the system. According to the work of Davis et al. [42], the Recall rate and Miss rate can be defined as follows:

$$Recall\ Rate = True\ Positive\ Rate = \frac{TP}{TP + FN} \tag{1}$$

$$Miss\ Rate = 1 - Recall = \frac{FN}{TP + FN} \tag{2}$$

In SSD, donate $x_{ij}^{p} = \{1, 0\}$ be the indicator for matching the default box i to the ground truth box j of object category p. When missed detection occurs, $\hat{x}_{ij}^{p}$ is equal to 0, and the Smooth L1 localization loss [15] between predicted box (l) and ground truth box (g) will increase:

$$L_{loc}(x, l, g) = \sum_{i \in Pos,\ m \in \{cu, cv, w, h\}}^{N} \sum x_{ij}^{k} smooth_{L1}(l_i^m - \hat{g}_j^m) \tag{3}$$

where $D(cu, cv, w, h)$ donate the center point coordinate (cu, cv) of bounding box (d) and its width and height.

The missed detection compensation model for adjacent frames is proposed, which is based on a reasonable assumption: *The moving speed of dynamic objects tends to remain constant in a short period (i.e. the acceleration tends to 0).* The velocity of the

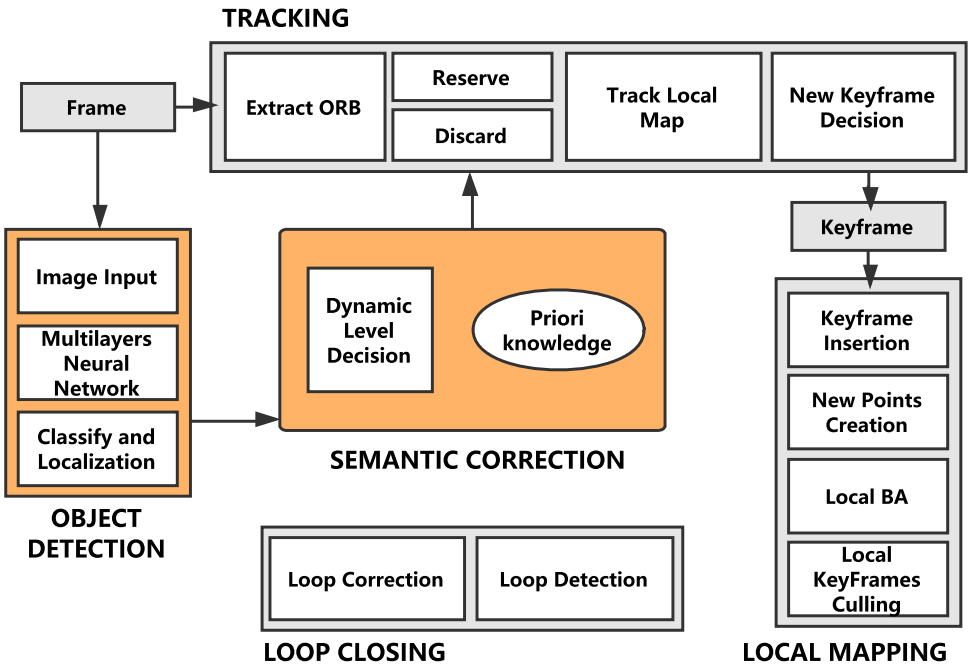


Fig. 2. Brief framework of Dynamic-SLAM. It shows the relationship between the newly added module and the existing three threads of ORB-SLAM2. An Object Detection thread and a Semantic Correction module is used to eliminate the interference of dynamic object.

dynamic object in the pixel plane is denoted by $\vec{v}$, and the $a_{\max}$ represents the threshold value of velocity change rate of the dynamic object in pixel plane as well. The following relationship should be satisfied between them:

$$|\Delta\vec{v}| \leq a_{\max} \tag{4}$$

The moving speed is used to describe the motion displacement of dynamic object between the several frames. In practice, the time difference between two adjacent frames in the video sequence is very short, and the change amount of dynamic object in motion displacement does not alter much. This model can be used to determine whether a missed detection is occurred, and also provides a compensation strategy in the event of missed detection. The corresponding bounding boxes between the previous frames and the current frame K in a short period of time can be confirmed by $\Delta\vec{v}$. If it is larger than $a_{\max}$, it is considered to be a mismatch, that is, a missed detection. The current keyframe K enters the SSD network and outputs the detected object list. Each item in the list is the localization of bounding box, that is the detected object position coordinates ${}^{K}D_i(0<i<N^K$, where N^K is the bounding box number in frame K). For each detection result of the previous frame ${}^{K-1}D_i$, if it is not detected in the area ${}^{K}A_i({}^{K}a_{i,u}, {}^{K}a_{i,v}, {}^{K-1}\hat{w}, {}^{K-1}\hat{h})$ determined by $a_{\max}$ in the current frame, it is considered as a missed detection, and ${}^{K}\hat{D}_i({}^{K}\hat{c}_{i,u}, {}^{K}\hat{c}_{i,u}, {}^{K-1}\hat{w}, {}^{K-1}\hat{h})$ needs to be added to the detection list in frame K, where:

$${}^{K}a_{i,(u,v)} = {}^{K-1}c_{i,(u,v)} + \frac{1}{k}\sum_{f=K-1}^{K-k}\Delta^{f}c_{i,(u,v)} \pm \frac{1}{2}a_{\max}(u,v) \tag{5}$$

$${}^{K}\hat{c}_{i,(u,v)} = {}^{K-1}c_{i,(u,v)} + \frac{1}{k}\sum_{f=K-1}^{K-k}\Delta^{f}c_{i,(u,v)} \tag{6}$$

where k is the number of the previous frames that used to perform missed detection and compensation on the current frame. The algorithm processing is as follows:

Algorithm 1: Missed detection compensation algorithm

Data: *frame Ki, bounding box list* ${}^{Ki}D_i$, *include cu, cv, w, h.*

```
Begin
    for i in K-1D_i, in current frame K, do
        calculate KA_i,
        for l in bounding box KD_l in current frame K, do
            if ∃ KD_l ∈ KA_i then t=l,
                calculate Δ Kc_t,u, Δ Kc_t,v;
            else if ∀ KD_l ∉ KA_i then t=i,
                KD_t = (Kĉ_t,u, Kĉ_t,v, K-1w, K-1h);
                add KD_t in KD_l;
                update KD List;
        end
    end
end
```

The effect of missed detection compensation algorithm is demonstrated in Fig. 3. It plays a vital role in the selection and tracking of dynamic objects.

The selection of k and $a_{\max}$ in the above formula affects the sensitivity of missed detection compensation. It will lead to oversensitivity and error compensation if k is too small. Usually 3–5 frames are selected before the frame K is determined. At the same time, if it detects the object are missing in more than two consecutive frames, the compensation will be abandoned. This may introduce a small amount of error compensation, but it can significantly reduce the occurrence of missed detection. In addition, the correct detection result may be regarded as missed detection when $a_{\max}$ is too small. If the value is too large, the system will be insensitive, where multiple dynamic object detection areas may overlap.

3.2. Dynamic object determination based on prior knowledge

The semantics of environmental objects are people's interpretation of the environment based on experience. In the process of remembering unfamiliar surroundings, people will automatically ignore dynamic objects such as vehicles and pedestrians

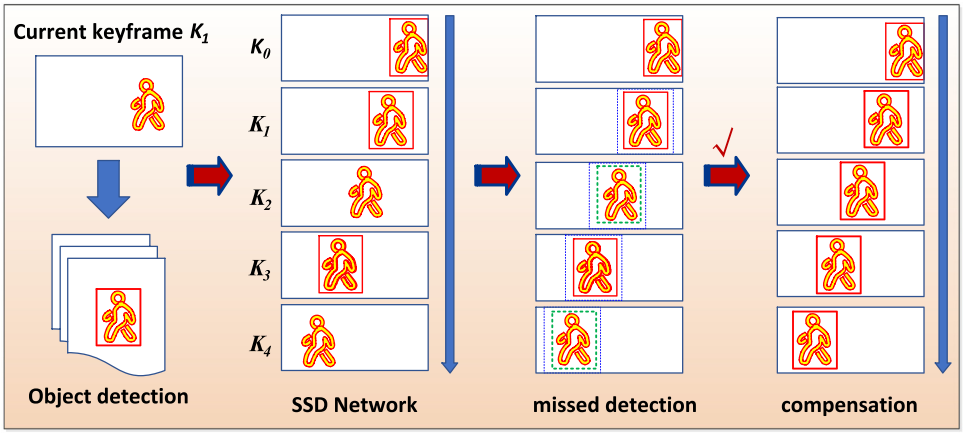


Fig. 3. Demonstration diagram of missed detection compensation algorithm. The detection results of SSD Network represented by the red bounding box. The missed detection can be determined based on the threshold $a_{\max}$ (blue dashed box), and then the result of compensation is calculated based on the speed invariant assumption (green dashed box) . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

with the help of prior knowledge, while remembering static objects such as buildings and vegetation. If the SLAM system does not understand the surrounding environment from the semantic level, it cannot truly distinguish between dynamic and static ones. It is just able to find moving objects in a short time, thus fail to guarantee long-term consistency. Therefore, the results of object detection combined with prior knowledge can perform a dynamic object decision model at the semantic level. For Dynamic-SLAM, output information from the front-end SSD object detection module is redundant. It only needs to know whether the detected feature points are static or dynamic. According to the prior knowledge of human, the dynamic characteristics of object can be scored from 0 point (static) to 10 point (dynamic). The score should be compared with a predefined threshold. Thus, static and dynamic distinctions are made to feature points. The approximate score of common objects in this interval is shown in Fig. 4.

In fact, this is a Maximize a Posterior problem. When the bounding box of the previous frame is detected as a dynamic object, the box of the current frame will also be judged as the same point and re-detected, then enter the missed detection compensation loop. Donate $\mathbb{C}=\left\{{}^{K}c_i : K = 1, 2, \ldots, K, \ldots, N\right\}$ and $\mathbb{Z}=\left\{{}^{K}d_i : K = 1, 2, \ldots, K, \ldots, N\right\}$ as the bounding box real condition and measurements, according to Bayes' rule,

$$P(\mathbb{C}|\mathbb{Z}) = \frac{P(\mathbb{Z}|\mathbb{C})\mathrm{P}(\mathbb{C}|)}{P(\mathbb{Z})} \propto P(\mathbb{Z}|\mathbb{C})\mathrm{P}(\mathbb{C}|) \tag{7}$$

and then,

$$\mathbb{C}^{*}_{MAP} = \arg\ \max P(\mathbb{C}|\mathbb{Z}) = \arg\ \max P(\mathbb{Z}|\mathbb{C})\mathrm{P}(\mathbb{C}|) \tag{8}$$

In Dynamic-SLAM framework, the determination result of the frame$K$ will be interfered by the past multiple frames. Conditional probability expansion according to frame $K-1$, there is:

$$P({}^{K}c|{}^{0}c, d_{1:k-1}) = \int P({}^{K}c|{}^{K-1}c,\ d_{1:k-1})P({}^{K-1}c|{}^{0}c, d_{1:k-1})d^{K-1}c \tag{9}$$

The dynamic object determination base on prior knowledge solves the redundant semantic information obtained from the neural network, which is more practical in feature point processing.

3.3. The object detection module and selective tracking

Feature tracking is not cumbersome, but not all features can be used for tracking, and not all features of dynamic objects are not available for tracking. There are two special cases here: 1. one object is classified as a dynamic object but it is static in the scene (for example, the cars in a car park) 2. the dynamic object occupies most of the view field of the camera (for example, in a very crowded shopping mall). Differentiating the foreground dynamic feature points and the background static feature points in different scenarios is crucial.

The specific system structure of Dynamic-SLAM is shown in Fig. 2. The object detection thread uses SSD object detection network to calculate the category and location of the object. After that, the object is further divided into dynamic object or static object by the semantic correction module, and then the position of the dynamic object is provided to the tracking thread. The tracking thread performs ORB feature extraction on each keyframe image, and then matches the result with the reference frame to obtain the corresponding relationship between the feature points of the two images, which is used to estimate the camera pose.

When the initialization is completed, the pose estimation is PnP (Perspective-n-Point) problem. The nonlinear optimization method represented by Bundle Adjustment [43] is used to solve the problem. This method can fully utilize all the matching results and get the optimal estimation of poses. The cost function of the nonlinear optimization is defined as follows:

$$\xi^{*} = \arg\min_{\xi} \frac{1}{2}\sum_{i=1}^{n}\left\|u_i - \frac{1}{S_i}K\exp(\xi^{\Lambda})P_i\right\|_2^2 \tag{10}$$

The minimum reprojection error is the difference between the observed pixel coordinates u_i and the reprojection coordinates of the 3D point P_i in the current pose ξ. The goal of optimization is to find a pose that minimizes the reprojection errors.

In the dynamic environment, the reprojection error of the feature points on dynamic object will be at an excessively high level, which will cause the camera pose fail to converge an optimal value. The decision result of the semantic correction module of Dynamic-SLAM is expressed as a mask image $mask(u, v) = \{0, 1\}$, which is a two-dimensional matrix with the same width and height as the original image, and the point with the element value of 0 represents the static pixel point, while the point of 1 is expressed as the dynamic pixel point.

$${}^{K}mask(u, v) = \begin{cases} 1 & p(u, v) \in {}^{K} D_i \\ 0, & p(u, v) \notin {}^{K} D_i \end{cases} \tag{11}$$

At the same time, the front-end visual odometry will reads the mask image at any time and performs the selective tracking operation for feature point according to the mask image value.

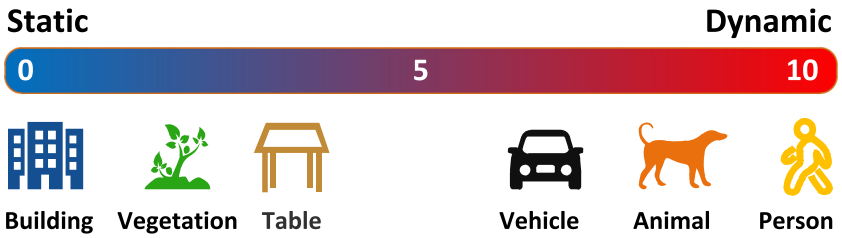


Fig. 4. Dynamic characteristic score of common objects based on life experience. Only a few common objects are listed in the figure. When the detection score is higher than the threshold, it is determined as a dynamic object. If it is lower, it is determined as a static object. The size of the threshold can usually be set as 5.

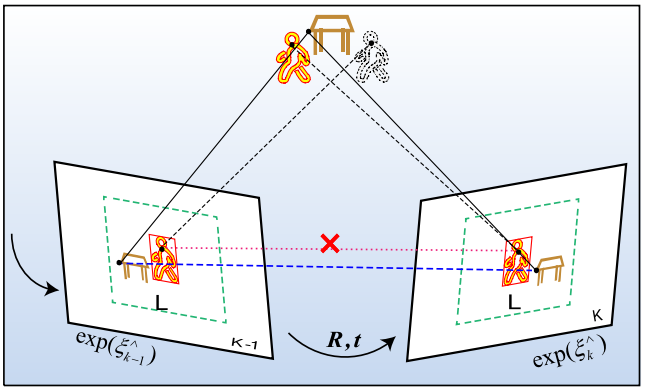


Fig. 5. Graphical representation of the selective tracking algorithm.

Fig. 5 illustrate the process of selective tracking algorithm. The pixel coordinates ${}^{K}p_{d_i}$ of dynamic feature points are obtained from the semantic correction module. Then, calculating the average pixel displacement $\bar{S}_L(u, v)$ of the static feature points in the pixel region L, which with a distance of l around the bounding box ${}^{K}D_i$. Finally, calculating the pixel displacement of the dynamic feature point and performing a determination:

$$\forall d_i \in {}^{K} D_i, \left| {}^{K}p_{d_i}(u, v)mask - {}^{K-1}p_{d_i}(u, v)mask \right| \leq \rho \bar{S}_L(u, v) \quad (12)$$

where ρ is the decision coefficient and $\bar{S}_L$ is the decision threshold, which is calculated as follows:

$$\bar{S}_L(u, v) = \frac{1}{N_L} \sum_{i \in L} \left| \frac{1}{Z_{s_i}} K \exp(\xi_K^{\wedge}) P_{s_i} - \frac{1}{Z_{s_i}} K \exp(\xi_{K-1}^{\wedge}) P_{s_i} \right| \quad (13)$$

If the relative displacement between the dynamic feature point and the static feature point is within an acceptable range, it is allowed to use for tracking, otherwise it is culled.

Algorithm 2: Selective tracking algorithm

Data: *Keyframe Ki, bounding box list* ${}^{Ki}D_i(c_x, c_y, w, h)$, ${}^{Ki}p_i$, *l, mask(u,v)*

```
Begins
  for ^K D_i in keyframe K, do
    for ^K p_i ∈ ^K D_i(^K c_{i,u}, ^K c_{i,v}, ^K w+l, ^K h+l), do
      calculate S̄(u,v) = (1/N_L) Σ(^K p_i mask − ^{K−1} p_i mask);
      mask=0, p is static point
    end
    then calculate Δ^K c_{i,u}, Δ^K c_{i,v}
    if |Δ^K c_i(u,v)|_1 > ρ S̄(u,v) then
      eliminate p_si in ^K D_i
  end
end
```

The selective tracking algorithm can make full use of the detection results of deep learning. It can well solve the feature point problem when the dynamic object is at static or the dynamic object occupies most of the view field, thus ensuring the robustness of the system. It sacrifices dimensional information to some extent, but the strategy is effective when the robot is at a low speed.

In order to ensure the real-time performance of the Dynamic-SLAM system, the object detection and tracking are divided into two threads. At the same time, a new data structure class named *Detection* is designed for its safe and efficient and supports concurrent operations to pass the detection results. Furthermore, a mutex lock named *unique_lock* is used to ensure that two threads do not have an access conflict. Since the processing speeds of the object detection thread and the tracking thread are not synchronous, asynchronous read and write of shared variables is used to achieve inter-thread communication and make maximum use of CPU time.

4. Experiments and results

Six experiments from different aspects is designed and implemented to verify the robustness and accuracy of the Dynamic-SLAM system in dynamic environment, including SSD object detection verification, anti-dynamic object interference test, initialization test under dynamic object interference, indoor accuracy test by using TUM dynamic dataset, outdoor large-scale mapping test and real-world environment test by using mobile robot.

The priori knowledge of SSD neural network used in the experiment is the mainly 20 classes of objects (i.e. airplane, bike, bird, boat, bottle, bus, car, cat, chair, cow, table, dog, horse, motorbike, person, plant, sheep, sofa, train and tv) that can be recognized in the network. The model with prior knowledge is derived from the trained network SSD300*VOC07+12+COCO,[1] which is trained through the PASCAL VOC2007, VOC2012 [44] and MS COCO [45] datasets. In the missed detection compensation module, $a_{\max}$ as a priori knowledge, its value is set as 20 pixels per square frame for the sampling rate is 30 fps.

The experimental operating environment is a laptop computer configured as an Intel Core i5-7300HQ CPU (4-core 2.5 GHz), 8 GB of RAM, and NVIDIA GeForce GTX1050Ti GPU with 4 GB of graphic memory for detection module.

4.1. SSD object detection verification

In order to reduce the miss rate in the SLAM system and increase the recall rate, a missed detection compensation algorithm for adjacent frames is proposed in Section 3. To verify the effectiveness, the new model is compared with the original SSD network.

The TUM RGB-D benchmark [46] is a series of datasets provided by the TUM computer vision group. The content covers various aspects of SLAM research and now has become one of the standards for performance and accuracy testing of the SLAM

[1] https://github.com/weiliu89/caffe/tree/ssd.

Fig. 6. Results of the SSD object detection test. Above: original detection results of SSD (recall rate 82.3%), Below: detection results after the compensation (recall rate 99.8%). The red box shows the position of the detected object and the corner tag indicates the object category. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

algorithm. In the sequence *walking_rpy,* the detection results of SSD network are successfully detected 401 times in the continuous 487 frame images, with failed 86 times, and the recall rate is 82.3% (i.e. the miss rate is 17.3%). After motion compensation, the test succeeded 486 times and failed once, and the detection rate reached 99.8% (i.e. the miss rate dropped to 0.2%). Fig. 6 shows the results of object detection sampling every 30 frames. Experiments show that the missed detection compensation model greatly improves the recall rate of object detection, which provides a good basis for the following localization and mapping module.

4.2. Anti-dynamic object interference test

It is often the case that dynamic objects are in the camera view for a long time, which is common in robotic applications. When the dynamic objects have a relative velocity to the surrounding environment, the traditional SLAM system cannot determine which features should be used to estimate its own motion.

Thus, a dataset is designed for the situation where people are always present in the camera's field. The operational effects of ORB-SLAM2 and Dynamic-SLAM are tested respectively. Fig. 7 shows the feature extraction of ORB-SLAM2 and Dynamic-SLAM. Fig. 8 shows the localization and mapping results of the two frameworks. Fig. 9 is the number of dynamic and static feature points throughout the process. In order to highlight the proportion of dynamic feature points, the average number of feature points in each keyframe is about 3090 in this experiment, with an average of 2290 dynamic feature points and an average of 809 static feature points. The proportion of dynamic feature points is 74.11%, with 47.93% more than static feature points. In the left of Fig. 8, the camera completely stops after a certain period of movement, the scale is drifting seriously, and the red map points are no longer updated. It can be found that in this extreme scenario, the ORB-SLAM2 system uses the dynamic object as the background environment, resulting in the localization result dependent on the relative movement between the dynamic object and the camera. In contrast, Dynamic-SLAM can automatically select the features of the static environment. Its localization result is a coherent route, and the constructed map is also consistent with the actual scene. The experiment tested the SLAM localization and mapping ability under the extreme condition, with respect that Dynamic-SLAM successfully responded to this environment. Experiments show that the dynamic point selective tracking strategy of Dynamic-SLAM is more effective in dynamic environment.

4.3. Initialization test under dynamic object interference

Initialization is an important step in visual SLAM. It is prone to erroneous initialization under the interference of dynamic objects. Since there is no map created beforehand at the initial moment, the scale factor and the pose can only be determined by the initial adjacent frame matching.

In Fig. 10, the camera is fixed to test the impact of dynamic objects on initialization. It can be seen that ORB-SLAM2 cannot distinguish between foreground objects and background objects when working in a dynamic environment. In contrast, Dynamic-SLAM successfully detects the position of the dynamic object and exclude the interference, thus avoiding false initialization.

Next, the camera is kept in a small swing condition while the person's distance from the camera changed constantly and repeated for 20 times. The result is that the ORB-SLAM2 is initialized successfully for 15 times with a success rate of only 75%, and the Dynamic-SLAM2 initialization succeed for 19 times that the success rate is 95%. This strategy improves the initialization process in dynamic environment because of its dynamic object determination based on prior knowledge and the trained detection network.

4.4. Indoor accuracy test by using TUM dynamic dataset

Localization and mapping in dynamic environment always have a worse performance in various frameworks. In order to accurately evaluate the performance of the framework, the dynamic sequences in the TUM RGB-D Benchmark are used to perform dynamic environmental testing. It is a scene in which two people working around the desk, with a large range of motion and a significant proportion in the field of vision. The accuracy and performance of the algorithms is tested in this scenario.

A complete comparison of the state-of-the-art monocular visual SLAM framework is made by using TUM dataset. Under the condition that the scale factor is undetermined, the Absolute Pose Error (APE) and Relative Pose Error (RPE)[2] of the keyframe trajectories is used to evaluate the localization accuracy for each framework. Sim(3) Umeyama alignment is used in the calculation. The RPE contains both relative translation error and relative rotation error. For measure the relative rotation error is more meaningful than the absolute one, the APE is only compared with translation part. Considering the validity of the data, we mainly use the Root Mean Square Error (RMSE), Mean error, Median

2 github.com/MichaelGrupp/evo.

Fig. 7. Feature extraction of ORB-SLAM2 (above) and Dynamic-SLAM (below) for anti-dynamic object interference.

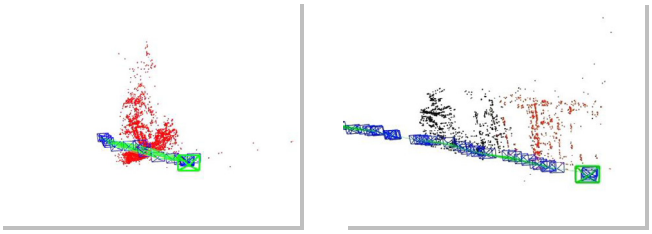


Fig. 8. Localization and mapping results. Left: ORB-SLAM2 (fail) and Right: Dynamic-SLAM (success). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

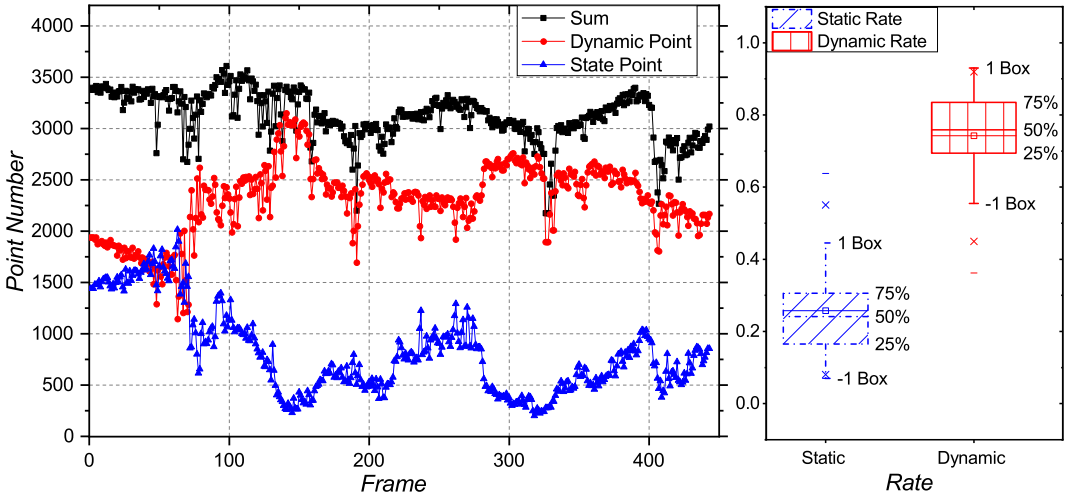


Fig. 9. Number of dynamic and static feature points throughout the process in this extreme dynamic scenario. The proportion of dynamic feature points is 76.33%, with 47.93% more than static feature points.

error, and Standard deviation (Std.) to evaluate the result. The calculation method of Improvements (Imp.) is as follows:

$$\kappa = (1 - \frac{\lambda}{\gamma}) \times 100\% \tag{14}$$

where κ is the improvement value, γ is value obtained from ORB-SLAM2, λ is the value obtained from Dynamic-SLAM. The comparison of the trajectory RMSE error over 3 executions in each sequence.

In Table 1, the main comparison objects are the absolute pose error of ORB-SLAM2 (Mono) [2], SVO [7], LSD-SLAM [5] and PTAM [4]. We also compared with DynaSLAM [40], a similar SLAMIDE method that uses ORB-SLAM2 combined with semantic segmentation strategy. The data for DynaSLAM and the last two framework results are obtained from the reported results [11,40], which are processed in the similar method. In Tables 2 and 3, the main comparison objects are relative pose error of ORB-SLAM2 and Dynamic-SLAM. Since the scale is too small, the unit of the RPE rotation part uses the degrees per frame. Fig. 11 shows the real-time image of ORB-SLAM2 and Dynamic-SLAM in the sequence *fr3_walking_xyz*. In the case where the operating parameters are identical, the localization results for all keyframes is recorded to compare with the ground truth values and calculate the errors.

The Absolute Pose Error results are shown in Table 1 and Fig. 12. The Relative Pose Error are listed in Tables 2 and 3. As can be seen from the represented results of Tables 1–3, the result of Dynamic-SLAM is better than other frameworks in most cases. Taking RMSE as the standard, the accuracy of Dynamic-SLAM is 7.48%∼62.33% higher than ORB-SLAM2. In terms of operational

Fig. 10. Results of initialization test under dynamic object interference. Above: ORB-SLAM2, Right: Dynamic-SLAM. In ORB-SLAM2, the feature points are mostly gathered on the dynamic object and initialized incorrectly in the fourth picture. When the feature point is changed from the green to the blue and the feature is accurate and stable at the same time, it indicates that the initialization is successful in ORB-SLAM2. Generally, the initialization time is within 5 s.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 11. Real-time images of ORB-SLAM2 (above) and Dynamic-SLAM (below) in the sequence *fr3_walking_xyz*.

Table 1
TUM dynamic dataset. Comparison of localization accuracy, Absolute Pose Error (APE) for translation part (Unit: cm).

Dynamic sequence	Length (m)	PTAM	LSD-SLAM	SVO (+BA)	DynaSLAM	ORB-SLAM2 (Mono)				Dynamic-SLAM				Imp.
		RMSE	RMSE	RMSE	RMSE	RMSE	Median	Mean	Std.	RMSE	Median	Mean	Std.	RMSE
fr2_desk_ps	17.044	×	31.73	17.14	**0.8**	4.971	1.829	2.857	4.068	1.873	0.952	1.277	1.370	**62.32%**
fr3_sit_half	6.503	×	5.87	12.06	–	1.992	1.451	1.687	1.059	**1.461**	1.230	1.290	0.686	**26.66%**
fr3_sit_rpy	1.110	–	–	7.89	–	5.811	4.709	5.363	2.237	**3.448**	3.087	3.228	3.015	**40.66%**
fr3_sit_xyz	5.496	0.83	7.73	6.03	1.3	0.836	0.547	0.731	0.405	**0.601**	0.565	0.570	0.190	**28.11%**
fr3_walk_half	7.686	×	×	11.25	2.1	3.714	2.348	2.807	2.432	**2.139**	1.507	1.823	1.118	**42.41%**
fr3_walk_rpy	2.698	–	–	18.91	–	10.528	6.794	8.764	5.833	**6.025**	4.689	5.385	2.702	**42.77%**
fr3_walk_xyz	5.791	×	12.44	9.30	1.4	1.431	1.082	1.225	0.519	**1.324**	0.836	1.080	0.765	**7.48%**

Results for Dynamic-SLAM, ORB-SLAM2, SVO are the median over 3 executions in each sequence. Results for PTAM, LSD-SLAM, DynaSLAM are obtained from the reported results [11,40]. Imp. represents the improvement of Dynamic-SLAM compared to the original ORB-SLAM2. '×' denotes tracking failure, '–' denotes no available data.

performance, the running time of the two algorithms in sequence *walking_xyz* is record in Table 4. Performance of Dynamic-SLAM is 10% higher than ORB-SLAM2. The other sequences are almost the same performance.

Although Dynamic-SLAM adds an object detection thread, it does not slow down the entire system but shortens the running time because the object detection is placed in a separate thread and uses GPU acceleration. The improvement is due to the reduction of invalid feature points, which enables the system calculated

Table 2
TUM dynamic dataset. Comparison of localization accuracy, Relative Pose Error (RPE) for translation part (Unit: cm/frame).

Dynamic sequence	ORB-SLAM2 (Mono)			Dynamic-SLAM			Imp.
	RMSE	Mean	Std.	RMSE	Mean	Std.	RMSE
fr2_desk_ps	2.584	0.975	2.393	**1.958**	0.946	1.714	**24.23%**
fr3_sit_half	1.694	1.254	1.139	**1.451**	1.236	1.002	**19.38%**
fr3_sit_rpy	6.495	3.910	5.185	**4.303**	3.555	3.092	**33.75%**
fr3_sit_xyz	**0.948**	0.847	0.424	0.998	0.901	0.427	−5.27%
fr3_walk_half	3.795	2.761	2.604	**2.192**	1.852	1.172	**42.24%**
fr3_walk_rpy	6.776	4.702	4.879	**5.605**	4.184	3.729	**17.28%**
fr3_walk_xyz	**1.772**	1.535	0.885	1.796	1.457	1.050	−1.35%

Table 3
TUM dynamic Dataset. Comparison of localization accuracy, Relative Pose Error (RPE) for rotation part (Unit: degrees/frame).

Dynamic Sequence	ORB-SLAM2 (Mono)			Dynamic-SLAM			Imp.
	RMSE	Mean	Std.	RMSE	Mean	Std.	RMSE
fr2_desk_ps	0.891	0.496	0.740	**0.833**	0.516	0.778	**6.51%**
fr3_sit_half	0.581	0.458	0.358	**0.551**	0.496	0.240	**5.16%**
fr3_sit_rpy	1.368	0.999	0.935	**0.991**	0.807	0.576	**27.56%**
fr3_sit_xyz	**0.557**	0.501	0.243	0.613	0.514	0.335	−10.05%
fr3_walk_half	0.763	0.623	0.439	**0.666**	0.555	0.368	**12.71%**
fr3_walk_rpy	1.278	0.967	0.835	**1.149**	0.886	0.732	**10.09%**
fr3_walk_xyz	1.825	1.002	1.526	**0.598**	0.533	0.272	**67.23%**

Table 4
Operational performance in sequence *walking_xyz* by taking the tests 3 times to get the average value.

Time (s)	Median	Mean	Total
ORB-SLAM2	0.050	0.050	42.90
Dynamic-SLAM	**0.044**	**0.045**	**38.40**
Improvements	**12%**	**10%**	**10.49%**

just with the effective feature points, thereby saving the time of pose estimation and nonlinear optimization.

4.5. Outdoor dynamic large-scale mapping in KITTI dataset

Dynamic environment is more practical in large-scale scenarios. KITTI dataset [47] is currently the largest computer vision algorithm evaluation dataset for autonomous driving in large-scale scenarios. It contains 11 sequences of outdoor real scene, which also provides accurate ground truth collected by GPS and *Velodyne* laser scanner. This dataset is used to further verify the effectiveness of the algorithm in the real environment (except sequence 01 for few trackable objects in a highway). Since ORB-SLAM2 is already a favorable framework in large-scale scenarios, its monocular version is used to compare and measure the improvement of Dynamic-SLAM. The ORB-SLAM and DynaSLAM is also compared in the experimental results.

Table 6
KITTI dataset. Comparison of localization accuracy, Relative Pose Error (RPE) for translation part (Unit: m/frame).

Sequence	ORB-SLAM2 (Mono)			Dynamic-SLAM			Imp.
	RMSE	Mean	Std.	RMSE	Mean	Std.	RMSE
00	7.295	3.763	6.207	**6.077**	2.744	5.422	**16.70%**
02	4.719	3.718	2.906	**4.241**	3.278	2.692	**10.13%**
03	3.777	3.284	1.866	**3.702**	3.194	1.871	**1.99**
04	0.134	0.122	0.055	**0.099**	0.091	0.037	**26.12%**
05	10.263	3.178	9.759	**8.881**	3.035	8.346	**13.47%**
06	1.106	0.529	0.971	**0.756**	0.446	0.610	**31.65%**
07	**2.408**	1.786	1.617	2.622	1.949	1.754	−8.89%
08	3.861	3.029	2.395	**3.819**	2.975	2.395	**1.09%**
09	4.152	3.649	1.981	**4.107**	3.583	2.007	**1.08%**
10	3.564	3.282	1.389	**3.517**	3.247	1.351	**1.32%**

Table 7
KITTI dataset. Comparison of localization accuracy, Relative Pose Error (RPE) for rotation part (Unit: degree/m).

Sequence	ORB-SLAM2 (Mono)			Dynamic-SLAM			Imp.
	RMSE	Mean	Std.	RMSE	Mean	Std.	RMSE
00	15.518	6.599	14.046	**10.659**	4.749	9.542	**31.31%**
02	8.681	4.508	7.419	**7.897**	4.003	6.911	**9.03%**
03	**6.452**	2.577	5.916	7.178	2.982	7.616	−11.25%
04	0.481	0.394	0.276	**0.407**	0.338	0.227	**15.38%**
05	8.05	3.531	7.235	**8.008**	3.432	7.236	**0.52%**
06	7.143	2.385	6.733	**7.113**	2.344	6.715	**0.42%**
07	9.663	4.304	8.652	**9.758**	4.448	8.686	−0.98%
08	11.192	5.077	9.974	**10.901**	4.924	9.725	**2.60%**
09	8.148	4.321	6.908	**7.977**	4.266	6.740	**2.10%**
10	7.825	3.089	7.189	**7.760**	3.043	7.139	**0.83%**

APE and RPE are also used to evaluate the accuracy of the results. Due to the large scale of the KITTI dataset, the unit of the RPE rotation part adopts degrees per meter. The other part of evaluation method is the same as TUM dataset test. Fig. 13 shows the Real-time image of Dynamic-SLAM. Table 5 and Fig. 14 show the Absolute Pose Error result. Tables 6 and 7 show the Relative Pose Error w.r.t translation part and rotation part. The data of ORB-SLAM and DynaSLAM is derived from the reported results [11,40]. It can be seen from Table 5 that the performance of Dynamic-SLAM is 0.10%∼53.69% better than ORB-SLAM2. At the same time, 70% of the results are better than DynaSLAM. Tables 6 and 7 also show the result of Dynamic-SLAM is better than ORB-SLAM2 in 90% cases.

In the ORB-SLAM2, there are a number of feature points on these dynamic objects, which have an impact on localization and mapping. It can be seen that the monocular SLAM has increasingly angle drift and scale drift as the motion scene becomes more complicated. Both the Dynamic-SLAM and the ORB-SLAM2 suffered severe scale drift in Sequence 02 and 08. In contrast, Dynamic-SLAM leaves the scale drifts lighter due to the elimination of dynamic interference, and all the performance is better

Table 5
KITTI dataset. Comparison of localization accuracy, absolute Pose Error (APE) for translation part (Unit: m)

Sequence	Dimension (m × m)	ORB-SLAM (Mono)	DynaSLAM (Mono)	ORB-SLAM2 (Mono)				Dynamic-SLAM					Improvements
		RMSE	RMSE	RMSE	Mean	Median	Std.	RMSE	Mean	Median	Std.	Scale	RMSE
00	564 × 496	6.68	7.55	9.579	8.072	6.104	5.157	**4.436**	4.094	4.031	1.708	17.16	**53.69%**
02	599 × 946	21.75	26.29	24.842	19.600	16.887	15.263	**20.367**	15.587	11.642	13.109	21.71	**18.01%**
03	471 × 199	1.59	1.81	1.078	0.923	0.840	0.556	**0.828**	0.602	0.427	0.568	11.42	**23.19%**
04	0.5 × 394	1.79	0.97	1.233	1.098	1.072	0.559	**1.109**	0.983	1.006	0.513	35.44	**10.06%**
05	479 × 426	8.23	**4.60**	5.730	5.308	5.456	2.158	5.724	5.365	5.422	1.995	21.70	**0.10%**
06	23 × 457	14.68	14.74	15.241	13.172	13.884	7.687	**12.455**	10.911	11.774	6.007	23.55	**18.28%**
07	191 × 209	3.36	2.36	1.962	1.675	1.425	1.022	**1.823**	1.652	1.676	0.772	11.23	**7.08%**
08	808 × 391	46.58	40.28	30.29	21.72	14.63	21.11	**27.487**	20.754	16.367	18.022	16.09	**9.25%**
09	465 × 568	7.62	**3.32**	10.374	9.473	10.395	4.229	9.285	8.416	10.028	3.923	21.91	**10.50%**
10	671 × 177	8.68	**6.78**	9.515	8.157	6.879	4.899	8.909	6.889	5.137	5.649	16.75	**6.37%**

Results for Dynamic-SLAM, ORB-SLAM2 are the median over 3 executions in each sequence. Results for ORB-SLAM and DynaSLAM are obtained from the reported results [11,40]. Improvements/Imp. represents the improvement of Dynamic-SLAM compared to the original ORB-SLAM2.

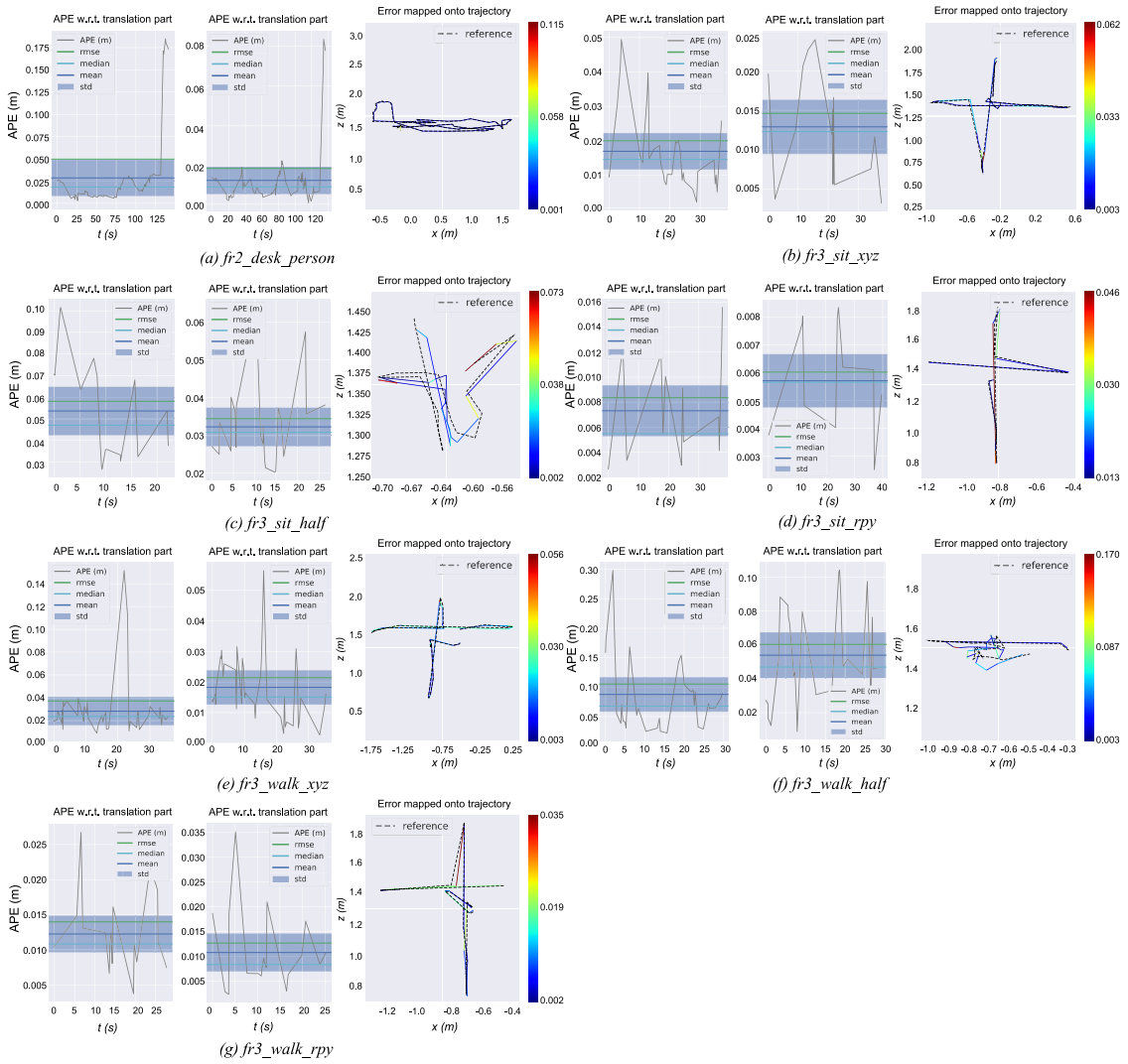


Fig. 12. The Absolute Pose Error results (with Sim(3) Umeyama alignment) and the Error mapped onto trajectory for each dynamic sequence in TUM dataset. For each sequence part of the dataset, there are 3 subgraphs, left: APE w.r.t translation part (m) for ORB-SLAM2, middle: APE w.r.t translation part (m) for Dynamic-SLAM, right: APE Error mapped onto trajectory (m) by compared with ground truth (dotted line) for Dynamic-SLAM, and the color bar on the right is the error range map. It has not made a comparison in the trajectory map since there is no obvious difference. As for the APE in the left and middle subgraphs, the error of Dynamic-SLAM is smaller than ORB-SLAM2.

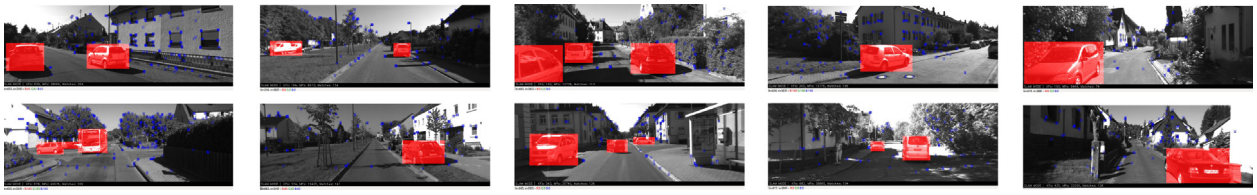


Fig. 13. Real-time images of Dynamic-SLAM by testing KITTI dataset. The objects covered by red mask in the figure are dynamic objects, including cars, pedestrians, etc. There are a number of feature points on these dynamic objects, which have an impact on localization and mapping. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

than ORB-SLAM2. Although most of the dynamic objects are in a static state in this dataset, the selective tracking algorithms is helpful for improving the accuracy of the entire system.

4.6. Real-world environment test by using mobile robot

In fact, the TUM and KITTI datasets are not very suitable for testing dynamic environment algorithms. In the KITTI dataset, most of the dynamic objects are at static, which is quite different from the real scene and is less difficult. For the monocular, the TUM dynamic dataset has some difficulty, because its motion blur is too severe and the scale is too small, which affects the algorithm's evaluation. Thus, Dynamic-SLAM system is further test in a physical robot in real-word dynamic environment.

4.6.1. Mobile robot platform

The various application scenarios of SLAM require the support of embedded platforms to achieve miniaturization and real-time operation. The robot platform is mainly composed of an embedded development board and a mechanical component. A custom

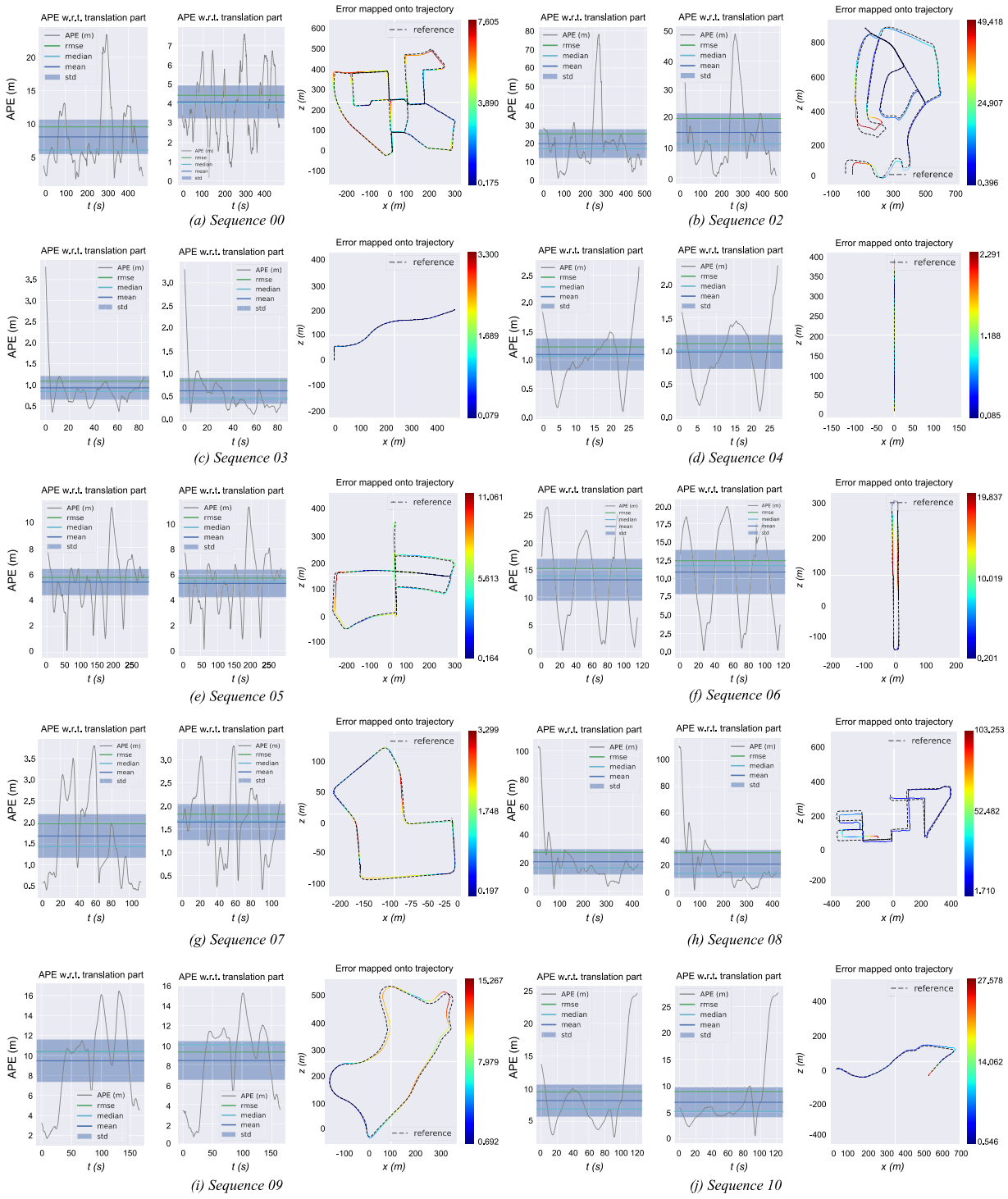


Fig. 14. The Absolute Pose Error results (with Sim(3) Umeyama alignment) and the Error mapped onto trajectory for each dynamic sequence in KITTI dataset. For each sequence part of the dataset, there are 3 subgraphs, left: APE w.r.t translation part (m) for ORB-SLAM2, middle: APE w.r.t translation part (m) for Dynamic-SLAM, right: APE Error mapped onto trajectory (m) by compared with ground truth (dotted line) for Dynamic-SLAM, and the color bar on the right is the error range map. It has not made a comparison in the trajectory map since there is no obvious difference. As for the APE in the left and middle subgraphs, the error of Dynamic-SLAM is smaller than ORB-SLAM2.

firmware is developed to run at the embedded level to address the feedback control and evaluation needs. As shown in Fig. 15, the robot is an Unmanned Ground Vehicles (UGV) with a volume of 65 cm × 55 cm × 50 cm and a weight of 40 kg. The core board unit is a NVIDIA Jetson TX2[3] development board, which carries Dynamic-SLAM system (ROS package) and UGV control module. The related configuration includes an NVIDIA Pascal architecture GPU, 256 CUDA cores, and quad core CPU (1.4 GHZ normal) based on ARM architecture, 8 GB RAM. The visual sensor is a mvBlueFox-MLC200w monochrome global shutter camera produced by *MatrixVison* company, with a resolution of 752 × 480 and a pixel size of 6 μm × 6 μm. A 32 bit MCU is used as the control center in UGV and controlled by Jetson TX2. All algorithm

3 https://www.nvidia.com/en-us/autonomous-machines/.

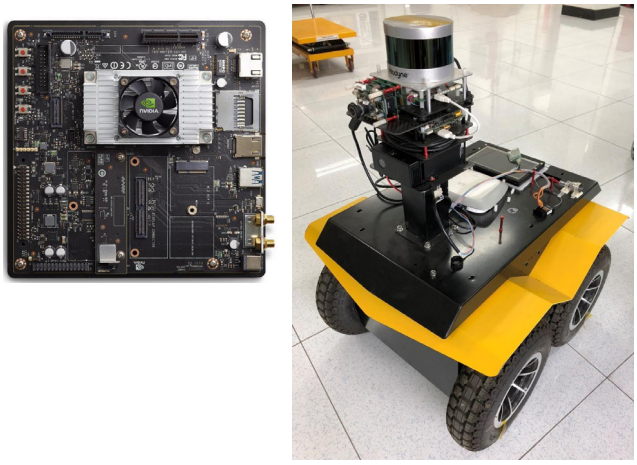


Fig. 15. The NVIDIA Jetson TX2 and UGV robot.

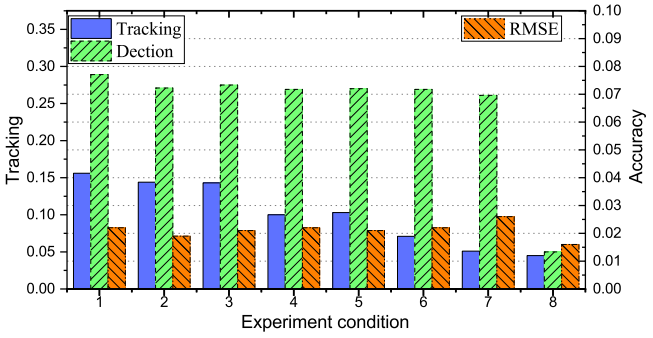


Fig. 16. The value change of tracking, detection, and root mean square error with the upgrading of optimization level by using dynamic environment sequence *walking_xyz* in the TUM RGB-D benchmark dataset. GPU optimization (Detection) did not make. In each optimization, the experiment is repeated 3 times and the average is taken as the final result.

Table 8
Experimental results after optimization.

Experiment condition	Tracking (s)	Detection (s)	RMSE (m)
G1	0.156	0.289	0.022
G2	0.144	0.271	0.019
G3	0.143	0.335	0.021
G4	0.100	0.329	0.022
G5	0.103	0.330	0.021
G6	0.071	0.330	0.022
G7	0.051	0.307	0.026
G8	0.045	0.050	0.016

From experiment 1 to experiment 7, the optimization level is increased in turn. *G1* is the initial result after transplantation. *G2* is the result in Release mode and with O3 optimization level of compiler. *G3* is the result of reducing the feature point from 1000 to 800. *G4* is further to reduce the feature points to 500. *G5* uses 500 feature points while reducing the feature matching threshold from 100 to 50. *G6* is to reduce the feature points to 300, while reducing the number of image pyramid layers from 8 to 4. In *G7*, the feature points are further reduced to 200, the number of image pyramid layers is reduced to 2, the feature matching threshold is reduced to 30, and the map point threshold is reduced from 50 to 30. *G8* is the raw comparison data in laptop.

development is performed in Ubuntu 16.04 and ROS Kinetic[4] with C++.

4.6.2. Performance optimization and evaluation after transplant

When transplanted to the robot platform, the average processing time per frame is three times the previous one due to the lower performance CPU and GPU (0.156 s after transplantation in Table 8 and 0.045 s before in Table 4). In order to test whether the performance of the algorithm after transplant meets real-time requirements, dynamic environment sequence *walking_xyz* in the TUM RGB-D benchmark dataset is already used to evaluate the transplant results after taken improvement measures. The evaluation results are shown in Table 8. The results of the experiments are shown in Fig. 16. With the upgrading of the optimization level, the tracking time is gradually shortened, and ultimately the performance is similar to the laptop. At the same time, although the accuracy is lost slightly, there is no significant effect on the results. In general, the performance of the transplantation is basically the same as the original laptop and the accuracy is lost slightly.

4.6.3. Real-world experiment

An outdoor large-scale scene experiments on embedded platforms is conducted. Experimental scene is a community in Beijing. The entire scene spans about 350 m from north to south and 370 meters from east to west. The entire route forms a closed loop. The maximum speed of the UGV in the test was limited to 2 m/s and experiment lasted 15 min. The experiment was repeated three times. The localization and mapping effects of Dynamic-SLAM and ORB-SLAM2, which use the same optimization conditions, is tested in this scenario by using the UGV robot. In addition, an APX-15 UAV and a 16-line 3D lidar sensor VLP-16 produced by *Velodyne* is used to collect ground truth.

Fig. 17 Shows the Dynamic-SLAM feature extraction and dynamic object elimination effects in real-time. The localization and mapping results shows in Figs. 18 and 19.

Fig. 17 shows that people and vehicles are successfully detected in Dynamic-SLAM and interference from dynamic object are eliminated. Because of incorrect association of dynamic objects, the global optimization of ORB-SLAM2 cannot be performed normally. As a result, it cannot correct the previous error after the loop closing and finally fails in Fig. 18. In contrast, Dynamic-SLAM achieved a reliable result after the loop closing. In comparison with ground truth in Fig. 19, the trajectory RMSE error of Dynamic-SLAM is 2.29 m, which is an acceptable localization accuracy.

The experiments show that the Dynamic-SLAM algorithm can be competent for simultaneous localization and mapping work on the embedded mobile robot platform in real dynamic environment.

5. Conclusion

In this paper, a complete SLAM framework named Dynamic-SLAM is constructed, which is a semantic monocular visual simultaneous localization and mapping system using deep learning to improve the performance in dynamic environment. This framework has three major contributions. Firstly, in view of the low recall rate of the existing SSD object detection network, a missed detection compensation algorithm based on the speed invariance in adjacent frames is proposed for SLAM system, which greatly improves the recall rate for detection and provides a good basis for the following module. Secondly, a selective tracking algorithm is proposed to eliminate the dynamic objects in a simple and effective way, which improves the robustness and accuracy of the system. Finally, A feature-based visual Dynamic-SLAM system is constructed. Based on the SSD convolutional neural network, deep learning technology is constructed to a newly object detection thread, which combines prior knowledge to realize the detection of dynamic objects at semantic level in robot localization and mapping.

[4] http://www.ros.org/.

Fig. 17. Outdoor large-scale and real-world dynamic environment experiment with embedded platform for Dynamic-SLAM.

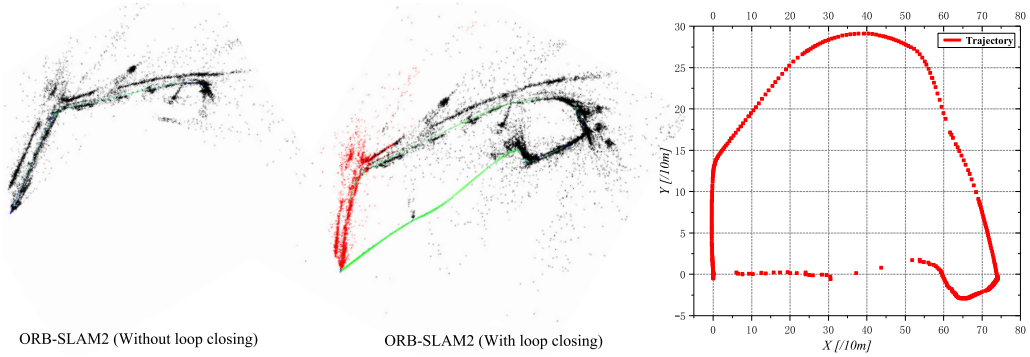


Fig. 18. Localization and mapping results of ORB-SLAM2 (failed).

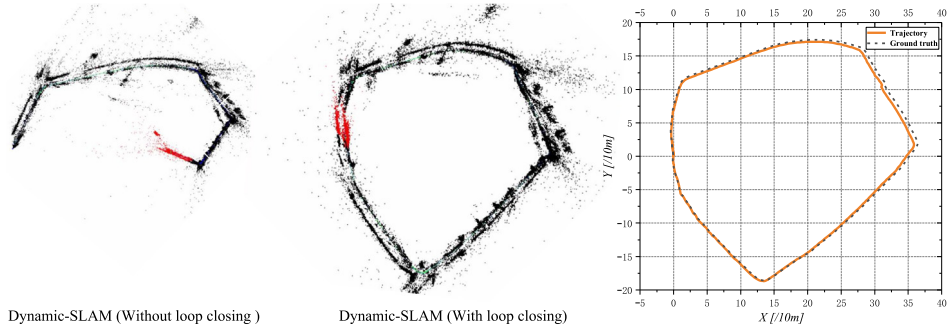


Fig. 19. Localization and mapping results of Dynamic-SLAM (succeeded).

To evaluate the framework, six experiments are designed to verify the superiority, accuracy, robustness and portability of Dynamic-SLAM. Compared with the original SSD detection network, the system increased the recall rate of detection from 82.3% to 99.8% in the TUM dataset. In the test of TUM indoor dynamic environment dataset, the localization accuracy of Dynamic-SLAM is higher 7.48%∼62.33% than the current state-of-the-art ORB-SLAM2 system and the operation performance is improved about 10%, and also better than the PTAM, LSD-SLAM, SVO, DynaSLAM framework. In the KITTI outdoor large-scale dynamic environment test, Dynamic-SLAM successfully localized and constructed a more accurate environmental map, and the performance is better than ORB-SLAM2 and DynaSLAM. In order to further verify the practicality of the algorithm, Dynamic-SLAM is transplanted to the embedded robot platform and successfully implemented the localization and mapping in real-world dynamic environment, while ORB-SLAM2 failed. Experiments result shows that Dynamic-SLAM has a reliable performance in superiority, accuracy and robustness in dynamic environment.

In summary, this paper is an exploration of SLAM technology in the real dynamic environment and successfully demonstrated the broad prospects for the integration of artificial intelligence based on deep learning and SLAM. In further research, deep learning can be applied more widely in SLAM, not just the frontend. At the same time, more complex environments should be challenged.

Acknowledgment

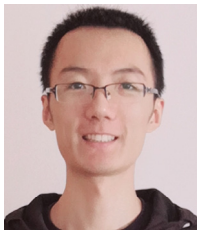
National Key R&D Program of China (2018YFB20020301).

References

[1] R. Smith, P. Cheeseman, On the representation and estimation of spatial uncertainty, Int. J. Robot. Res. 5 (4) (1986) 56–68.
[2] R. Mur-Artal, J.D. Tardós, ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras, IEEE Trans. Robot. 33 (5) (2017) 1255–1262.
[3] A.J. Davison, I.D. Reid, N.D. Molton, O. Stasse, MonoSLAM: Real-time single camera SLAM, IEEE Trans. Pattern Anal. Mach. Intell. 29 (6) (2007) 1052–1067.
[4] G. Klein, D. Murray, Parallel tracking and mapping for small AR workspaces, in: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007, pp. 225–234.

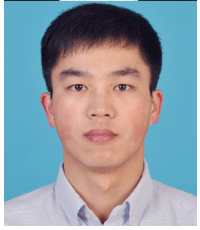
[5] J. Engel, T. Schöps, D. Cremers, LSD-SLAM: Large-Scale Direct Monocular SLAM, Springer International Publishing, Cham, 2014, pp. 834–849.
[6] C. Forster, M. Pizzoli, D. Scaramuzza, SVO: Fast semi-direct monocular visual odometry, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 15–22.
[7] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, D. Scaramuzza, SVO: Semidirect visual odometry for monocular and multicamera systems, IEEE Trans. Robot. 33 (2) (2017) 249–265.
[8] J. Engel, V. Koltun, D. Cremers, Direct sparse odometry, IEEE Trans. Pattern Anal. Mach. Intell. 40 (3) (2018) 611–625.
[9] T. Qin, P. Li, S. Shen, VINS-Mono: A robust and versatile monocular visual-inertial state estimator, IEEE Trans. Robot. (2018) 1–17.
[10] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, P. Furgale, Keyframe-Based Visual-Inertial Odometry Using Nonlinear Optimization, 2014.
[11] R. Mur-Artal, J.M.M. Montiel, J.D. Tardós, ORB-SLAM: A versatile and accurate monocular SLAM system, IEEE Trans. Robot. 31 (5) (2015) 1147–1163.
[12] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: An efficient alternative to SIFT or SURF, in: 2011 International Conference on Computer Vision, 2011, pp. 2564–2571.
[13] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[14] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1904–1916.
[15] R. Girshick, Fast R-CNN, 2015.
[16] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2015.
[17] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You.Only.Look.Once:. Unified, Real-Time.Object. Detection, You only look once: Unified, real-time object detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.
[18] W. Liu, et al., SSD: Single Shot MultiBox Detector, Springer International Publishing, Cham, 2016, pp. 21–37.
[19] M. Holschneider, A Real-Time Algorithm for Signal Analysis with Help of the Wavelet Transform, 1989, pp. 289–297.
[20] W. Chieh-Chih, C. Thorpe, S. Thrun, Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas, in: 2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422), Vol. 1, 2003, pp. 842–849.
[21] D.F. Wolf, G.S. Sukhatme, Mobile robot simultaneous localization and mapping in dynamic environments, Auton. Robots 19 (1) (2005) 53–65, journal article.
[22] A. Petrovskaya, et al., Awareness of road scene participants for autonomous driving, in: E. Azim (Ed.), Handbook of Intelligent Vehicles, Springer, 2012, pp. 1383–1432.
[23] C. Qiu, Z. Zhang, H. Lu, H. Luo, A Survey of Motion-Based Multitarget Tracking Methods, 2015, pp. 195–223.
[24] A. Asvadi, P. Peixoto, U. Nunes, Detection and tracking of moving objects using 2.5D motion grids, in: 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 2015, pp. 788–793.
[25] R.H. Wong, J. Xiao, S.L. Joseph, A robust data association for simultaneous localization and mapping in dynamic environments, in: The 2010 IEEE International Conference on Information and Automation, 2010, pp. 470–475.
[26] T.-D. Vu, J. Burlet, O. Aycard, Grid-based localization and local mapping with moving object detection and tracking, Inf. Fusion 12 (1) (2011) 58–69.
[27] P.C. Niedfeldt, R.W. Beard, Multiple Target Tracking Using Recursive RANSAC, 2014, pp. 3393–3398.
[28] C. Bibby, I.D. Reid, Simultaneous Localisation and Mapping in Dynamic Environments (SLAMIDE) with Reversible Data Associa, 2007.
[29] B.F. Chen, Z.X. Cai, Research on mobile robot SLAM in dynamic environment, Appl. Mech. Mater. 130–134 (2012) 232–238.
[30] A. Walcott-Bryant, M. Kaess, H. Johannsson, J.J. Leonard, Dynamic Pose Graph SLAM: Long-Term Mapping in Low Dynamic Environments, 2012, pp. 1871–1878.
[31] M.S. Bahraini, M. Bozorg, A.B. Rad, SLAM In dynamic environments via ML-RANSAC, Mechatronics 49 (2018) 105–118.
[32] M.R.U. Saputra, A. Markham, N. Trigoni, Visual SLAM and Structure from Motion in Dynamic Environments: A Survey, 2018, pp. 1–36.
[33] W. Zhang, Q. Chen, W. Zhang, X. He, Long-range terrain perception using convolutional neural networks, Neurocomputing 275 (2018) 781–787.
[34] Y. Sun, M. Liu, M.Q.H. Meng, Motion removal for reliable RGB-D SLAM in dynamic environments, Robot. Auton. Syst. 108 (2018) 115–128.
[35] C. Yu, et al., DS-SLAM: A semantic visual SLAM towards dynamic environments, in: presented at the 2018 IEEE International Conference on Intelligent Robots and Systems (IROS), 2018.
[36] F. Zhong, S. Wang, Z. Zhang, C. Chen, Y. Wang, Detect-SLAM: Making object detection and slam mutually beneficial, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, pp. 1001–1010.
[37] R. Scona, M. Jaimez, Y. Petillot, M. Fallon, D. Cremers, Staticfusion: background reconstruction for dense rgb-d slam in dynamic environments, in: presented at the 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018.
[38] Z. Lu, Z. Hu, K. Uchimura, Slam estimation in dynamic outdoor environments, Int. J. Humanoid Robot. 07 (02) (2010) 315–330.
[39] D. Barnes, W. Maddern, G. Pascoe, I. Posner, Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments, in: presented at the 2018 IEEE International Conference on Robotics and Automation (ICRA), 2018.
[40] B. Bescós, J.M. Fácil, J. Civera, J.J.I.R. Neira, A. Letters, DynSLAM: Tracking, mapping and inpainting in dynamic scenes, IEEE Robot. Autom. Lett. (2018) 1-1.
[41] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988.
[42] J. Davis, The relationship between precision–recall and roc curves, in: ICML '06 : Proceedings of the 23rd international conference on Machine learning, New York, NY, USA, 2006.
[43] B. Triggs, P.F. McLauchlan, R.I. Hartley, A.W. Fitzgibbon, Bundle Adjustment — A Modern Synthesis, Berlin, Heidelberg, 2000, pp. 298–372.
[44] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes (voc) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.
[45] T.-Y. Lin, et al., Microsoft COCO: Common objects in context, in: 2014 European Conference on Computer Vision (ECCV), Springer International Publishing, Cham, 2014, pp. 740–755.
[46] 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, 2012, pp. 573–580.
[47] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision Meets Robotics: The KITTI Dataset, 2013, pp. 1231–1237.

Linhui Xiao received his B.E. degree in Physics from Nanchang University, Nanchang, China, in 2016 and is currently pursuing the Master's degree in Microelectronics and Solid State Electronics from University of Chinese Academy of Sciences (UCAS), Beijing, China. His current research interests include SLAM, robotics, computer vision, etc. xiaolinhui16@mails.ucas.ac.cn.



Jinge Wang received his Master's degree in Signal and Information Processing from University of Chinese Academy of Science, Beijing, China in 2018, and the bachelor's degree in Electrical Engineering and Automation from Beijing Institute of Technology, Beijing, China, in 2015. His current research interests include computer vision, simultaneous localization and mapping, robotics, control and planning, deep learning, etc. wjg172184@163.com.

Xiaosong Qiu received her master degree in information and signal processing from the University of Chinese Academy of Sciences, Beijing, China, in 2018, and received her B.E. degree in information engineering from Jilin University, Changchun, China, in 2015. Her research interests include computer vision and simultaneous localization and mapping. qiuxiaosong15@mails.ucas.ac.cn.



Zheng Rong received his Ph.D. degree in Electronics Science and Technology from Beijing Institute of Technology (BIT), China, in 2017. During his Ph.D. study, he stayed one year in Robotics Institute of Carnegie Mellon University (CMU), USA, as a visiting scholar. He received the BA degree in Information Engineering from Beijing Institute of Technology, China in 2010. He is now an assistant researcher in Institute of Electronics, Chinese Academy of Science (CAS) Beijing, China. His current research interests include visual odometry/SLAM, multi-sensor fusion, perception for robotics, autonomous vehicles, etc. Zheng.Rong.14@gmail.com.

Xudong Zou received the B.Sc. degree in microelectronics from Peking University, China, in 2009, and the Ph.D. degree in mechanics, materials, and design from the University of Cambridge, U.K., in 2013. He was at the Nanoscience Center, Department of Engineering, University of Cambridge, as a Research Associate, and the Churchill College as a PostDoctoral By-Fellow. He is currently a Professor with the State Key Laboratory of Transducer Technology, Institute of Electronics, Chinese Academy of Sciences and the School of Electronics, Electrical and Communication Engineering, University of Chinese Academy of Sciences. His main research interest lies in the area of integrated micro and nano electromechanical systems, with a focus on resonators and inertial sensors, including the design and optimization of high-resolution MEMS resonant inertial sensors, studying the nonlinear effects on the noise processes in MEMS oscillators, and the coupled MEMS resonators in sensing applications. He also works on the sensor fusion technologies such as SLAM, VIO using MEMS sensors for autonomous navigation. xdzou@mail.ie.ac.cn.