Detect-SLAM: Making Object Detection and SLAM Mutually Beneficial

Fangwei Zhong[1] Sheng Wang[2] Ziqi Zhang[1] Chen Zhou[1] Yizhou Wang[1]
[1]Nat'l Engineering Laboratory for Video Technology
Cooperative Medianet Innovation Center
Key Laboratory of Machine Perception (MoE)
Sch'l of EECS, Peking University, Beijing, 100871, China
[2]Sch'l of ECE, Shenzhen Graduate School, Peking University, Shenzhen, 518055, China
{zfw, sheng.wang, zzzq, zhouch, yizhou.wang}@pku.edu.cn

Abstract

Although significant progress has been made in SLAM and object detection in recent years, there are still a series of challenges for both tasks, e.g.*, SLAM in dynamic environments and detecting objects in complex environments. To address these challenges, we present a novel robotic vision system, which integrates SLAM with a deep neural network-based object detector to make the two functions mutually beneficial. The proposed system facilitates a robot to accomplish tasks reliably and efficiently in an unknown and dynamic environment. Experimental results show that compare to the state-of-the-art robotic vision systems, the proposed system has three advantages: i) it greatly improves the accuracy and robustness of SLAM in dynamic environments by removing unreliable features from moving objects leveraging the object detector, ii) it builds an instance-level semantic map of the environment in an online fashion using the synergy of the two functions for further semantic applications; and iii) it improves the object detector so that it can detect/recognize objects effectively under more challenging conditions such as unusual viewpoints, poor lighting condition, and motion blur, by leveraging the object map.*

1. Introduction

In recent years, tremendous progress has been made in areas of simultaneous localization and mapping (SLAM) and image-based object detection. As an extensively investigated topic, many visual-based SLAM systems have been proposed, whose localization precision down to a few centimeters and could build a large-scale 3D map in real time [5, 12, 18]. With the recent advancement of deep convolutional neural networks (CNN), the performance of image-based object detection [14, 24, 26] has been boosted.

However, the performance and application of both functions, while running separately, are limited by a series of thorny problems. For instance, the SLAM systems are usually prone to failure in dynamic environments, and the detectors can be sensitive to changing viewpoints [22], occlusion, *etc*. Nonetheless, these two tasks can be complementary: SLAM aims to estimate the ego-motion and geometry of the environment from a video; object detectors represent the semantic information of an image by placing a bounding box with a pre-defined object class around the instance. Intuitively, a question is raised that is it possible to integrate SLAM with object detector in a system to share the geometry information and the semantic understanding with each other to make them mutually beneficial?

Inspired by the recent success of combing SLAM with object detection [2, 3, 7, 21, 32], we propose a novel framework, Detect-SLAM, which integrates visual SLAM with a deep neural network (DNN) based object detector to make them mutually beneficial.

In Detect-SLAM, we utilize the semantic information to eliminate the negative effects caused by moving objects in the SLAM pipeline. To overcome the delay of semantic information mainly caused by communication and detection, we propose a method that propagates moving probability of each keypoint in real-time. In the SLAM pipeline, we also build an *object map*, a semantic map composed of all detected static objects in the mapping thread. Such object map can be regarded as a database containing the knowledge about object class and location. Using this map, a robot can execute commands such as 'deliver the book from the nearest desk to me' or answer queries about the semantics of the scenario, like 'How many monitors do we have in the room?'. To enhance the object detector to work under more challenging conditions such as unusual viewpoints, poor lighting conditions, and severe occlusion, we utilize the object map as prior knowledge for the detector and project the object map into 2D image plane for spatio-temporal consistent object region proposal. Such SLAM enhanced detector can be used to mine hard examples in

978-1-5386-4886-5/18/$31.00 ©2018 IEEE
DOI 10.1109/WACV.2018.00115

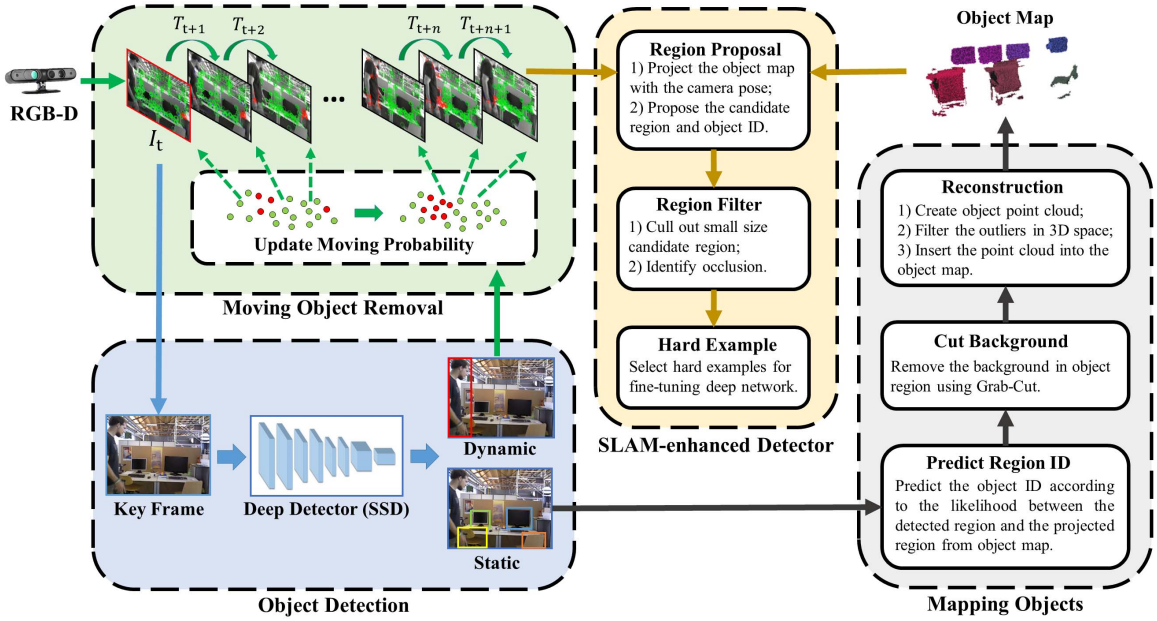


Figure 1. **The framework of Detect-SLAM.** It is composed of *moving object removal*, *object detection*, *SLAM-enhanced detector* and *mapping objects*. The input data are RGB-D images. Moving object removal contains the tracking and local mapping thread of ORB-SLAM. The deep detector (SSD) is running on the GPU, the others are running on the CPU.

those challenging conditions, and these hard examples can be subsequently used as training data to improve/fine-tune the original deep detector.

To the best of our knowledge, this is the first work that couples SLAM and the DNN-based detector to simultaneously accomplish three vision tasks: improving the robustness of SLAM in dynamic environments, building a semantic map, and boost the performance of object detection, as illustrated in Fig. 1.

We employ the ORB-SLAM [18] and Single Shot Multibox Object Detector(SSD) [14] in Detect-SLAM. The ROS [23] interface is utilized for communication between system components, which makes it straightforward to distribute the Detect-SLAM across mobile devices and GPU devices in real-time.

In the rest of the paper, we will discuss related work in Section 2, present the technical details of our system in Section 3, demonstrate our system with experimental result in Section 4 and end with conclusions in Section 5.

2. Related Work

Simultaneous Localization and Mapping: For most previous SLAM systems [36, 11, 34], the core assumption is that the environment is largely static and the camera is the only moving object in the scene. However these ideally cases are hardly met indoors or outdoors. [9, 37] attempted to treat those moving objects, mostly people, as outliers and remove them from the environment representations. Some other SLAM systems like [13, 16, 17, 35] use the framework that combines a SLAM system with object tracking and detection to improve positioning accuracy in dynamic environments.

Such frameworks combining SLAM and detection are still used in recent years. The works most closely related to ours are Paschalis *et al*. [20] and Sun *et al*. [33]. Both works detect moving objects frame by frame without building models for moving object like our system, where Paschalis *et al*. [20] focus on 3D point cloud but Sun *et al*. [33] on 2D patches. They both include segmentation and get moving objects' mask additionally, but our system omits these operations and filter features belonging to moving objects by updating the moving probability of features, which means our method is based on feature-level representation and could be more robust and efficient.

Deep Neural Network Based Object Detection: With the advancement of deep-neural-network [8, 14, 24, 25, 26], the accuracy of image-based object detection has been boosted [1, 10]. Faster R-CNN [26] is one of the most accurate deep neural networks with more than 80% mAP in the PASCAL VOC dataset, depending on region proposal algorithms [6]. Redmon *et al*.designed an unified architecture for YOLO [24] and its improved model YOLOv2 [25], making YOLOv2 one of the fastest networks that could process images at 91 FPS with 69% mAP or 40 FPS with 78.6%

mAP on PASCAL VOC dataset [6]. Single Shot Multibox Object Detector (SSD) [14] is the first DNN-based real-time object detector that achieves above 70% mAP in PASCAL VOC dataset [6] with 40 FPS in TitanX. This detector balances speed and accuracy well, hence we deploy SSD as the detector module in our Detect-SLAM.

However, it is hard to reach real-time performance while communicating with SLAM or deploying it in embedded system. To overcome the latency between the SLAM and the detector, we avoid detecting frame by frame and consider the spatio-temporally consistency of successive frames.

Combining SLAM and Object Detection: Previous works [3, 21, 29] have combined SLAM or SfM technology with object detection to solve issues in SLAM and recognition. Pillai *et al*. [21] designed a SLAM-supported object recognition system earning outperformance. McCormac *et al*. [15] combine SLAM with CNN to produce a semantic 3D map efficiently. Bowman *et al*. [2] integrate discrete recognition and data association problems with continuous SLAM optimization into an optimization problem, which result in more accurate trajectories. Duncan *et al*. [7] created a monocular-based method with object detector to solve the scale ambiguity and drift in conventional monocular SLAM system. Sucar *et al*. [32] combine a kalman filtering-based monocular SLAM with semantic information provided by detector to estimate the global scale of the 3D models in a Bayesian scheme.

However, all of these combining systems are designed to solve single problems. To do more tasks in the combining system, Chhaya *et al*. [3] combine SLAM with shape priors to detect and reconstruct vehicles from multiple view, improving 3D shape estimation and robust camera trajectory estimation in the pipeline. Unfortunately, this pipeline has to run off-line. By contrast, our Detect-SLAM framework runs in real-time, accomplishing three vision tasks: improving the robustness of SLAM in dynamic environments, building a semantic map and boosting the performance of object detection simultaneously.

3. Detect-SLAM

In this section, we present the technical details about Detect-SLAM. In Detect-SLAM, we incorporate a DNN-based object detector into the SLAM system, as illustrated in Fig. 1. Detect-SLAM is built on ORB-SLAM2 [18], which has three main parallel threads: Tracking, Local Mapping, and Loop Closing. Compared to ORB-SLAM2, the Detect-SLAM includes three new processes:

- Moving objects removal, filtering out features that are associated with moving objects.
- Object Mapping, reconstructing the static objects that are detected in the keyframes. The object map is composed of dense point clouds assigned with object ID.
- SLAM-enhanced Detector, exploiting the object map as prior knowledge to improve detection performance in challenging environments.

3.1. Moving Objects Removal

For moving object removal, we modify the Tracking and Local Mapping thread in ORB-SLAM2 to eliminate the negative impact of moving objects, shown in Fig. 3. Note that the moving objects in this process belong to movable categories which are likely to move currently or in times to come, such as people, dog, cat, and car. For instance, once a person is detected, no matter walking or standing, we regard it as a potentially moving object and remove the features belonging to the region in the image where the person was detected.

One key issue concerning detection-enhanced SLAM is the efficiency of object detection. The object detection process should be fast enough to enable per-frame detection in real-time, so that the unreliable regions can be removed during the per-frame tracking process of the SLAM. However, naively applying the detector in each frame is not a viable option, as even the state-of-the-art SSD [14] method runs only at about 3 FPS in our preliminary experiments.

In this section, we propose two strategies to effectively overcome this issue: 1) detect moving objects in keyframes only and then update the moving probability of points in the local map to accelerate the tracking thread; 2) propagate the moving probability by feature matching and matching points expansion in the tracking thread to remove the features extracted on moving objects efficiently before camera pose estimation.

We call the probability of a feature point belonging to a moving object as **moving probability**. Shown as Fig.2, We distinguish these keypoints into four states according to the moving probability. Both high-confidence points are used in matching point expansion to propagate the moving probability to those neighbouring unmatched points. After every point gets the moving probability via propagating, we remove all of dynamic points and using RANSAC to filter other outliers for the pose estimation.

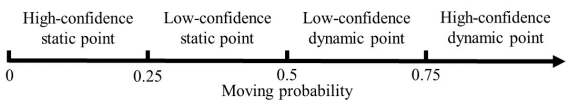


Figure 2. Four states of keypoints distinguished by the moving probability.

Updating Moving Probability. Considering the delay of detection and the spatio-temporal consistency of successive frames, we only select the color images of keyframes to detect, shown as in Fig. 3. The rule of keyframe selection is the same as ORB-SLAM2. Then we pre-process and forward-propagate the color image through the deep

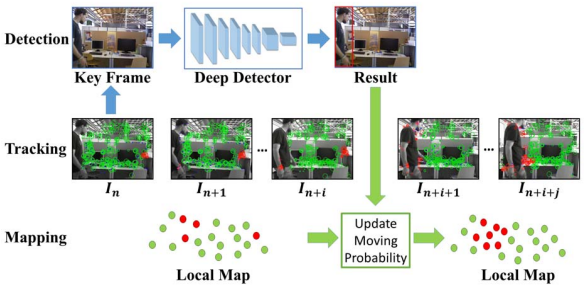


Figure 3. The process of moving object removal.

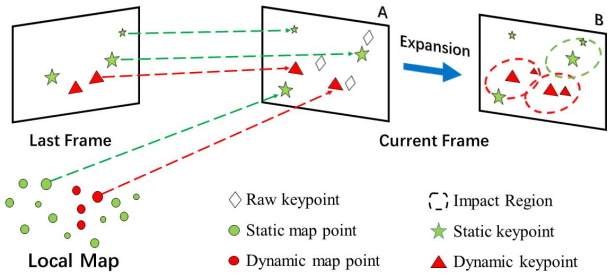


Figure 4. The process of propagating the moving probability frame-by-frame. The size of the points reflects the confidence.

neural network, meanwhile propagating moving probability frame-by-frame in the tracking thread. Once the detection result is obtained, we insert the keyframe into the local map and update the moving probability in the local map. We update the probability of 3D points that has found matching keypoints in the keyframe according to the following equation:

$$P_t(X^i) = (1-\alpha)P_{t-1}(X^i) + \alpha S_t(x^i), \qquad (1)$$

where $P_{t-1}(X^i)$ is the moving probability of 3D point X^i after the updating in the last keyframe I_{t-1}. If it is a new point, we set $P_{t-1}(X^i) = P_{init} = 0.5$. The state of matched keypoint x^i in keyframe I_t is $S_t(x^i)$. It depends on the detected region. If the keypoint x^i is in the bounding box of moving objects, we treat it as a determinate dynamic point, whose state value $S_t(x^i) = 1$. The others are treated as determinate static points, with state value $S_t(x^i) = 0$. α is an impact factor to smooth the immediate detection result. A higher value means more sensitivity to the immediate detection result and the lower value means that more history results from multi-view are considered. In our experiment, we set $\alpha = 0.3$ in the reason that we observed the detector sometimes provide false result in complex environments.

Moving Probability Propagation. In the tracking thread, we estimate the moving probability of every keypoint frame by frame via two operations: 1) feature matching and 2) matching point expansion. We call it as *'Moving Probability Propagation'*, for the moving probability in the current frame is propagated from keypoints in the last frame, points in local map without any knowledge of detection. Fig. 4 shows the details of moving probability propagation in symbol.

Feature matching. We use the same features as ORB-SLAM2, for feature matching to take advantage the robustness and efficiency of the ORB features [28]. During feature matching, when a keypoint x_t^i is matched to another keypoint x_{t-1}^i in the last frame, the moving probability $P_t(x_{t-1}^i)$ is propogated.

Apart from this, once a key point is matched to any 3D point X_t^i in the local map, it is also assigned a moving probability which is equal to the value of matched point $P(X_t^i)$. Note that if a point finds matched point not only in the last frame but also in the local map, the probability of local map should be prioritized. We assign an initial probability P_{init} to the other unmatched points in this frame. The initial probability $P_(init)$ is set to 0.5, as we have no prior assumption about which state these points belong to.

We summarize the operation that use feature matching to propagate the moving probability in an equation as below:

$$P(x_t^i) = \begin{cases} P(x_{t-1}^i), & \text{if } \|\phi(x_t^i) - \phi(x_{t-1}^i)\| < \theta \\ P(X_t^i), & \text{if } \|\phi(x_t^i) - \phi(X_t^i)\| < \theta \\ P_{init}, & otherwise \end{cases} \qquad (2)$$

where $\phi(x_t^i)$, $\phi(x_{t-1}^i)$, $\phi(X_t^i)$ represent the ORB-feature of point x_t^i, x_{t-1}^i and X_t^i respectively. θ is a threshold for feature matching.

Matching point expansion. This operation is designed to expand the moving probability from the high-confidence points to other neighboring points that do not correspond to any matched points in the feature matching operation. It relies on the assumption that the state of points in the neighborhood is consistent in most of case.

So after propagating via feature matching, we select high-confidence points χ_t, including static and dynamic ones. Then we expand the impact region of the high-confidence points to a round region with radius r and look for unmatched points within the region. The probability of the found points is updated following the rule as below:

$$P(x_t^i) = P_{init} + \sum_{x_t^j \in \chi_t} \lambda(d)(P(x_t^j) - P_{init}), \qquad (3)$$

Where P_{init} is the initial moving probability. If a point is impacted by more than one high-confidence point, we will sum all the impact of these neighboring high-confidence points. We formulate the impact of a high-confidence point considering the difference of moving probability $P^m(x_t^j) - P_{init}$ and a distance factor $\lambda(d)$. If the point is in the impact region of a high-confidence point($d \leqslant r$), the distance factor $\lambda(d) = Ce^{-d/r}$, C is a constant value, otherwise($d > r$), $\lambda(d) = 0$.

3.2. Mapping Objects

Reconstructing the environment in a map is the core ability of SLAM system, but most of the maps are built in pixel or low-level features without semantics. Recently, with the advancement of object detection, creating semantic map supported by object detector becomes more promising.

In this process, we reconstruct an object map containing all of the detected objects. Every 3D point in the map is assigned an object ID for recognition. The pipeline of this process is shown in Fig. 1.

The original ORB-SLAM only builds sparse map aiming to improve localization accuracy. As for semantic map, such sparse map is not enough for further applications. So we build on top of existing RGB-D-based ORB-SLAM and insert a dense point clouds mapper, similar to some other RGB-D based dense mapping solution [4].

Predicting Region ID. At the beginning of mapping process, we predict the object ID of every detected region in image space. The purpose of the region ID prediction is to find corresponding object ID in the object map or generate a new ID if it is detected for the first time. The object ID prediction is based on the geometry assumption that the area of projection and detection should overlap if both areas belong to the same object. So we compute the intersection over union(IOU) between two areas to represent the level of overlapping:

$$IoU(R_1, R_2) = \frac{R_1 \cap R_2}{R_1 \cup R_2} \tag{4}$$

where R_1 is the area from detection, R_2 is the area from the object map projection.

When we find that two areas are overlapping($IOU > 0.5$), we estimate the depth likelihood between them:

$$P(R_1|R_2) = IoU(R_1 R_2) * e^{-Err}, \tag{5}$$

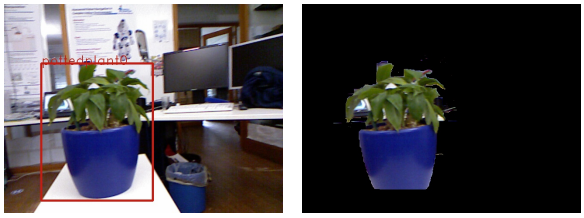
where $error(depth)$ is the MSE between the observed and the projected depth in the overlapping region.

$$Err = \frac{1}{N} \sum_{(u,v)\in R_1 \cap R_2} (D_p(u,v) - D_o(u,v))^2, \tag{6}$$

where $D_o(u,v)$ and $D_p(u,v)$ represent the depth of observed and projected in the pixel (u,v). N is the number of point clouds projected into the overlapping region $R_1 \cap R_2$.

If the depth likelihood is higher than the threshold θ_d, we assign the object ID of projected region to the detected region. On the contrary, we assign a new object ID to the region.

Cutting Background. Even though the detector provides the bounding box of object in the image, it is a rectangle box containing some unexpected background for building a clean object map. Therefore, we cut the background



(a) Get object region (b) Cut background

Figure 5. The left is the raw color image with bounding box, and the right is the result of segmentation by Grab-Cut.

utilizing $Grab - Cut$ [27] algorithm, serving points projected from previous reconstructed object in the overlapping area as seed of foreground and points outside of the bounding box as background. Then we get a segment mask of the target object from the bounding box, after three iterations.

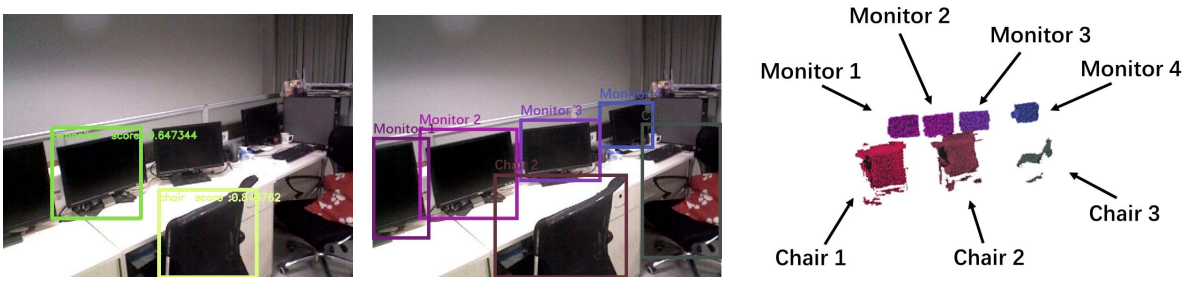
Reconstruction. Leveraging the object mask, we create object point cloud assigned with object ID and filter the noise point in 3D space. Finally, we transform the object point cloud into the world coordinate with the camera pose and insert them to the object map.

3.3. SLAM-enhanced Detector

Semantic map can be used not only for optimizing trajectory estimation [2, 15] but also improving object detector [21]. Contrary to classical per-frame object detection methodologies, robots observe the same instance of objects in its environment several times and from disparate viewpoints. It is natural to improve object detector via providing geometry information from the reconstructed 3D context to the object detection, which ensures the object detector working with spatially consistency. When the object detector is supported by the reconstructed object map and precisely estimated camera pose, it is enhanced.

Region Proposal. In the SLAM-enhanced object detector, the region of each object is proposed by projecting the 3D object map M into 2D plane with the current camera pose estimated by tracking. Using the projected image, we propose candidate regions that are likely to contain objects by clustering the pixels with the same object ID. As the point cloud in object map is labeled by object ID while building object map, every correspond pixel in the projecting image get the same object ID naturally. In this case, the object ID of every region can be recognized directly.

Region Filter. But there are still unexpected regions proposed, such as some small region caused by noisy point cloud or regions that contain occluded objects. So we cull out small candidates whose region size is less than 20×20 px and estimate likelihood between the observed depth and projected depth to detect the occluding candidates, in the region checker stage. The likelihood of the candidate in depth is similar to the region likelihood function (Equation.5), ex-



(a) The original SSD (b) The SLAM-enhanced detector (c) The object map of the scenario

Figure 6. Image **(a)** and **(b)** show the comparison of detection results from the original SSD and our SLAM-enhanced detector. The object map **(c)** of the scenario including four monitors and three chairs. The SLAM-enhanced detector detects the chairs successfully when it is only observed partially.

cept that $IoU = 1$ in constant.

Hard Example Mining. Previous works [30] have demonstrate that selecting hard examples to train or fine-tune deep neural network can yield significant boosts in detection performance. To make the deep detector perform better without SLAM, we apply the SLAM-enhanced object detector to mine hard examples, which augments the training data. Then we fine-tune the original SSD network to boost detection performance in similar scenes. The instances of the hard examples are shown in Fig. 7.

Figure 7. **Hard examples mined by the SLAM-enhanced detector.** Hard examples contain monitors observed from one side and potted plants observed with motion blur.

4. Experiment

In our system, we implement the DNN-based detector in python, the other processes in C++. We use a pretrained Inception v3 based SSD model[1] as our deep detector and the RGB-D based ORB-SLAM2 as our basic SLAM system. The communication between SSD and SLAM is implemented with the Robot Operating System [23]. All experiments are conducted in real-time.

We have evaluated the two main functions of our system, moving object removal and SLAM-enhanced detector, respectively. The object map is not evaluated for the imperfect of dataset. But the boosted performance of SLAM-enhanced detector can be regard as a positive proof of the object map. Our system runs in real-time on an Intel Core i7-4700 laptop with 16GB RAM and Nvidia GPU GTX960M. The GPU is only used for the deep detector.

[1] https://github.com/zhreshold/mxnet-ssd

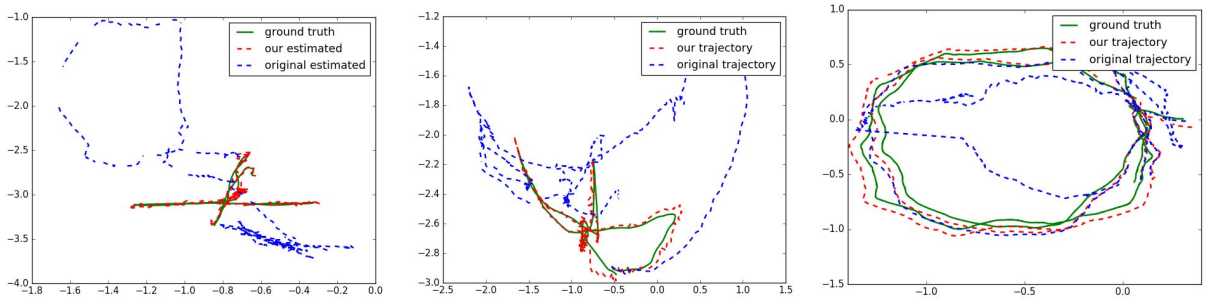
4.1. Robust SLAM in Dynamic Environments

In this section, we demonstrate our detector-based moving object removal approach on TUM RGB-D datasets. Sturm *et al.* [31] collected several video sequences with a RGB-D Kinect camera to evaluate RGB-D SLAM systems. They also provide the camera trajectories, obtained from a high-accuracy motion capture system. There are various patterns of camera ego-motion, such as moving along the x-y-z axes, rotating along the roll-pitch-yaw axes, and following a halfsphere-like trajectory. This dataset contains several typical dynamic scenes. For example, in *fr3/w/rpy* sequence, two people walk around the table while the camera rotating. Since large parts of the visible scenes are dynamic, it constitutes a very difficult task.

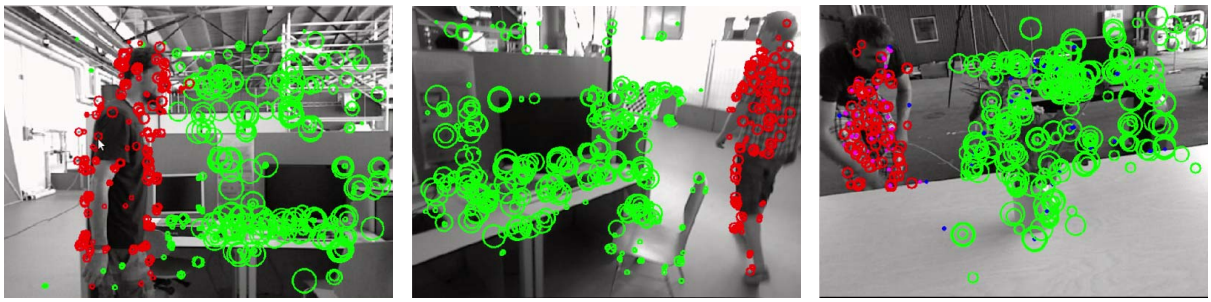
ORB-SLAM is known as the state-of-the-art method producing good results in static scenarios and low-dynamic scenarios, while our method extends its application scenarios to high-dynamic scenarios. To evaluate the effectiveness of our moving object removal method, we select seven high-dynamic sequences, two low-dynamic sequences and one static sequence in the experiment.

Fig. 4.1 shows some examples of computed trajectories of our system using the TUM dataset, with comparison to the ground-truth and the original RGB-D-based ORB-SLAM [19]. Qualitatively, we can see that our estimated trajectories are much closer to the ground-truth compared to ORB-SLAM in these dynamic environments. While a man walking past the camera, the trajectory recovered by the original ORB-SLAM drifts severely, as ORB-SLAM treats the features in the person's body as static landmarks. By contrast, our approach successfully filters those moving features before motion estimation, shown as the red points in Fig. 8(d), 8(e), and 8(f).

For a quantitative comparison we run the sequences on each system 5 times and compute the median, to account for the non-deterministic nature of the multi-threading system. The overall comparison of the results are summarized in Ta-



(a) fr3/w/xyz (b) fr3/w/half (c) fr2/flower



(d) fr3/w/xyz (e) fr3/w/half (f) fr2/flower

Figure 8. **The first row** are the estimated trajectories by our system (red), RGB-D ORB-SLAM (blue), and groundtruth (green) in fr3/w/xyz, fr3/w/half, and fr2/flower, respectively. **The second row** are the snapshots of the states of keypoints when running. The red points represent the dynamic keypoints, the green points represent the static keypoints, and the radius of each reflects the confidence of each state. These images show that using our method can classify most of key points into dynamic or static state accurately frame-by-frame.

ble 1, obtained by the original RGB-D-based ORB-SLAM, a motion removal approach for RGB-D SLAM in dynamic scenes [33](represented as *MR*), our method without last keyframe matching(represented as *Ours1*), and our complete method(represented as *Ours2*). For brevity, we use the words *fr, half, w, s, d, v* as representatives for *freiburg, halfsphere, walking, sitting, desk, validation* in the name of the sequences.

As we can see, the applicability of the original ORB-SLAM in challenging dynamic scenarios is limited. The motion removal approach [33] performs much better than the original ORB-SLAM in dynamic scene, but still unable to get ideal result comparing to those produced by ORB-SLAM in static scenarios. On the contrary, our method acquires comparable results. It verifies not only the intuition that semantic information can make the feature-based SLAM be much more robust, but also the usability and effectiveness of our practical measures in handling some challenging dynamic scenarios.

Additionally, comparing *Ours1* with *Ours2*, we can find that it is necessary and meaningful to propagate moving probability from the last frame to track the moving features frame-by-frame, in case that the delay of detection causes the moving probability of points in local map unable to update in time, especially when the object is moving largely.

Table 1. Comparison of Translation RMSE(*m*) in TUM Dataset.

	MR[1]	ORB[2]	Ours1[3]	Ours2
fr3/w/xyz	0.0932	1.0122	0.0254	**0.0241**
fr3/w/xyz/v	0.0655	0.9993	0.0268	**0.0218**
fr3/w/half	0.1252	0.5827	0.2021	**0.0514**
fr3/w/half/v	0.0811	0.3286	0.0697	**0.0522**
fr3/w/rpy	**0.1333**	0.8951	0.4559	0.2959
fr3/w/rpy/v	0.2333	0.3262	0.1050	**0.0767**
fr2/flower	-	0.5471	0.1489	**0.0569**
fr3/s/xyz	0.0470	**0.0117**	0.0210	0.0201
fr3/s/half	0.0482	**0.0178**	0.0273	0.0231
fr3/d/person	**0.0596**	0.0947	0.0825	0.0813

[1] A motion removal method for RGB-D SLAM [33]
[2] The original RGBD-based ORB-SLAM
[3] Our method but not propagating moving probability from last frame.

Table 2. The run time of each operation in moving object removal

	Propagation	Detection	Updating
Run-times(s)	≈ 0.02	≈ 0.31	≈ 0.01

Besides, we investigate the run-time performance of moving probability propagation, updating the moving probability in local mapping and object detection. Note that the run-time of detection includes sending the keyframe to detector, preprocessing the image, feed-propagating the image in the deep neural network and feed-back the result to the SLAM system. Considering that the original ORB-SLAM has taken the feature matching operation, so we only calculate the extra run-times excluding feature matching. The detection is time consuming compared to other process, but it only happens when a new keyframe inserted. The moving probability propagation is efficient to keep the tracking thread running in real-time. We summarize the run-times of our approach in Table 2.

4.2. Boosting Detection Performance

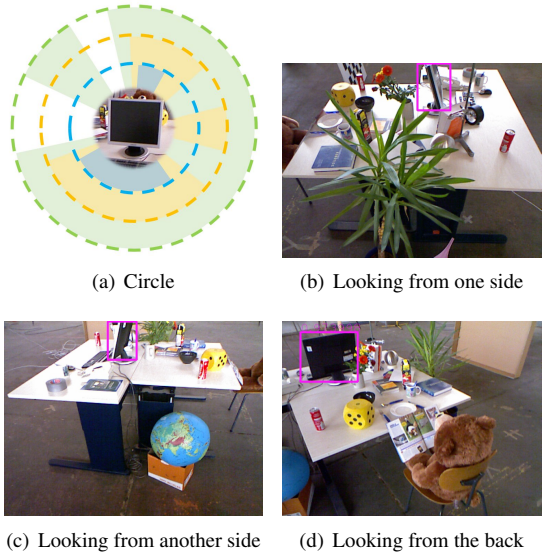
We collect 200 hard examples about monitors and 150 hard examples about potted plant from 15 sequences of TUM dataset and 5 sequences of our office to fine-tune the SSD network, aiming to make the object detector more robust to the deform of object or the motion blur while changing viewpoints. While fine-tuning network, all convolutional feature layers are fixed except box prediction layers.

To evaluate the improvement of the detector, we select and annotate two sequences(*fr2/desk* and *fr1/plant*). They film a monitor and a plant respectively over wide change of viewpoint and some of them observe partial objects. There are 2201 images captured the monitor on the desk in *fr2/desk* and 1070 images captured the plant in *fr1/plant*. In *fr2/desk*, the monitor is occluded by others occasionally. In *fr1/plant*, we can find a number of blurred images caused by the camera motion. Therefore, to boost detection performance in these sequences, the detector has to overcome the challenges including partial observation, motion blur, and occlusion.

In the experiment, we can see that most of bounding box proposed by the three method is precise when the confidence threshold is set to 0.7. But in most of challenge case, the original SSD missed the target object but our method detected it. Quantitatively, we compare the recall of target object of the three fashion detectors in both sequences, shown in Table. 3. As we can see, both our joint and fine-tuned methods extend the detectable viewpoints of monitors. The object map built simultaneously led to a huge boost in the enhanced-SLAM detector performance. The fine-tuned SSD is also improved by the collected hard examples. And the accurate bounding box proposed by our system is also a proof of the accuracy of localization and the object map.

5. Conclusions

In this paper, we present a novel coupled framework called Detect-SLAM for making object detector and SLAM mutually beneficial in one system. Detect-SLAM can localize precisely in dynamic scenes, build a semantic object map, and detect/recognize objects robustly. The experiments on TUM dataset demonstrate the effectiveness of our system in SLAM and object detection. The SLAM-enhanced detector is robust to motion blur, uncommon viewpoints by means of the object map. Moreover, we have apply our system to mining hard examples about monitors and plants appeared in TUM dataset, and then fine-tune the SSD network.



(a) Circle (b) Looking from one side

(c) Looking from another side (d) Looking from the back

Figure 9. The first image **(a)** visualizes the orientations that detect the monitor successfully in a circle. The blue, yellow and green sectors represent the original SSD, fine-tuned SSD, and our SLAM-enhanced detector, respectively. As we can see, the SLAM-enhanced detector covers most of orientations and the SSD performs better after fine-tuning. The other three images represent results looking the monitor from different orientations and positions.

Table 3. Comparision of recalls of target object in *fr2/desk* and *fr1/plant*.

	Original	Fine-tuned	SLAM-Enhanced
monitor	33.44%	44.62%	95.27%
plant	41.12%	53.56%	92.24%

6. Acknowledgement

We would like to thank the authors of ORB-SLAM2 and SSD for providing source code of their work, and the authors of the Tum Dataset for their efforts in collecting sequences with ground truth trajectories of camera in dynamic scenes. This work is supported in part by 973-2015CB351800, NSFC-61625201, NSFC-61421062, NSFC-61527804.

References

[1] I. Bogun, A. Angelova, and N. Jaitly. Object recognition from short videos for robotic perception. *arXiv preprint arXiv:1509.01602*, 2015. 2

[2] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic data association for semantic slam. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1722–1729. IEEE, 2017. 1, 3, 5

[3] F. Chhaya, D. Reddy, S. Upadhyay, V. Chari, M. Z. Zia, and K. M. Krishna. Monocular reconstruction of vehicles: Combining slam with shape priors. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5758–5765. IEEE, 2016. 1, 3

[4] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard. 3-d mapping with an rgb-d camera. *IEEE Transactions on Robotics*, 30(1):177–187, 2014. 5

[5] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. In *arXiv:1607.02565*, July 2016. 1

[6] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 2, 3

[7] D. P. Frost, O. Kähler, and D. W. Murray. Object-aware bundle adjustment for correcting monocular scale drift. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 4770–4776. IEEE, 2016. 1, 3

[8] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[9] D. Hhnel, D. Schulz, and W. Burgard. Mobile robot mapping in populated environments. *Advanced Robotics*, 17(7):579–597, 2003. 2

[10] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Optimizing deep cnn-based queries over video streams at scale. *arXiv preprint arXiv:1703.02529*, 2017. 2

[11] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. *IEEE International Conference on Intelligent Robots and Systems*, pages 2100–2106, 2013. 2

[12] M. Labbe and F. Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2661–2666. IEEE, 2014. 1

[13] K.-H. Lin and C.-C. Wang. Stereo-based simultaneous localization, mapping and moving object tracking. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3975–3980. IEEE, 2010. 2

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 2, 3

[15] J. McCormac, A. Handa, A. Davison, and S. Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4628–4635. IEEE, 2017. 3, 5

[16] D. Migliore, R. Rigamonti, D. Marzorati, M. Matteucci, and D. G. Sorrenti. Use a single camera for simultaneous localization and mapping with mobile object tracking in dynamic environments. In *ICRA Workshop on Safe navigation in open and dynamic environments: Application to autonomous vehicles*, pages 12–17, 2009. 2

[17] L. Montesano, J. Minguez, and L. Montano. Modeling dynamic scenarios for local sensor-based motion planning. *Autonomous Robots*, 25(3):231–251, 2008. 2

[18] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 1, 2, 3

[19] R. Mur-Artal and J. D. Tardos. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. *arXiv*, (October), 2016. 6

[20] P. Panteleris and A. A. Argyros. Vision-based slam and moving objects tracking for the perceptual support of a smart walker platform. In *ECCV Workshops (3)*, pages 407–423, 2014. 2

[21] S. Pillai and J. Leonard. Monocular slam supported object recognition. *arXiv preprint arXiv:1506.01732*, 2015. 1, 3, 5

[22] W. Qiu and A. Yuille. Unrealcv: Connecting computer vision to unreal engine. In *Computer Vision–ECCV 2016 Workshops*, pages 909–916. Springer, 2016. 1

[23] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009. 2, 6

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. 1, 2

[25] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. 2

[26] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 1, 2

[27] B. C. Rother. Kolmogorov v and blake a (2004) grabcut: interactive foreground extraction using iterated graph cuts. In *ACM Trans Graph*, 2010. 5

[28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *Computer Vision (ICCV), 2011 IEEE international conference on*, pages 2564–2571. IEEE, 2011. 4

[29] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. Slam++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1352–1359, 2013. 3

[30] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016. 6

[31] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. 6

[32] E. Sucar and J.-B. Hayet. Probabilistic global scale estimation for monoslam based on generic object detection. *arXiv preprint arXiv:1705.09860*, 2017. 1, 3

[33] Y. Sun, M. Liu, and M. Q.-H. Meng. Improving RGB-D SLAM in dynamic environments: A motion removal approach. *Robotics and Autonomous Systems*, (December):1–13, 2016. 2, 7

[34] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. Intelligent robotics and autonomous agents. the MIT Press, Cambridge (Mass.) (London), 2005. 2

[35] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, 2007. 2

[36] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015. 2

[37] D. F. Wolf and G. S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19(1):53–65, 2005. 2