

# DetectFusion: Detecting and Segmenting Both Known and Unknown Dynamic Objects in Real-time SLAM

Ryo Hachiuma<sup>1</sup>

ryo-hachiuma@keio.jp

Christian Pirchheim<sup>2</sup>

pirchheim@icg.tugraz.at

Dieter Schmalstieg<sup>2,3</sup>

schmalstieg@tugraz.at

Hideo Saito<sup>1</sup>

hs@keio.jp

<sup>1</sup> Keio University

Yokohama, Japan

<sup>2</sup> Graz University of Technology

Graz, Austria

<sup>3</sup> VRVis Research Center

Vienna, Austria

## Abstract

We present DetectFusion, an RGB-D SLAM system that runs in real time and can robustly handle semantically known and unknown objects that can move dynamically in the scene. Our system detects, segments and assigns semantic class labels to known objects in the scene, while tracking and reconstructing them even when they move independently in front of the monocular camera. In contrast to related work, we achieve real-time computational performance on semantic instance segmentation with a novel method combining 2D object detection and 3D geometric segmentation. In addition, we propose a method for detecting and segmenting the motion of semantically unknown objects, thus further improving the accuracy of camera tracking and map reconstruction. We show that our method performs on par or better than previous work in terms of localization and object reconstruction accuracy, while achieving about 20 fps even if the objects are segmented in each frame.

## 1 Introduction

Simultaneous localization and mapping (SLAM) is an important enabling technology for autonomous driving, robotics, augmented reality and virtual reality. Contemporary SLAM systems jointly estimate a map of an unknown environment and the pose of a camera, using either RGB [1, 2] or RGB + depth (RGB-D) [3, 4]. Most SLAM systems build purely geometric maps and assume a static scene.

However, the inherent assumption that the observed scene is static can be problematic in practice. If the system has no notion of moving objects, any observed motion must be treated as an outlier and be ignored by tracking and mapping. Thus, a SLAM system designed with the assumption of a static scene will suffer from map corruption in case of small amounts of motion. Ultimately, tracking will fail if the dominant observed motion does not

Method	Static map	Object maps	Semantic segmentation (known objects)	Motion segmentation (unknown dynamic objects)	System overall performance
ElasticFusion [23]	✓				frame-rate
StaticFusion [20]	✓			frame-rate	frame-rate
Co-Fusion [10]	✓	✓		frame-rate	frame-rate
MaskFusion [18]	✓	✓	keyframe-rate (5Hz) (Mask R-CNN)		frame-rate (on two GPUs)
MID-Fusion [24]	✓	✓	offline (Mask R-CNN)	frame-rate	not real-time
Ours	✓	✓	frame-rate (20Hz) (YOLOv3 + geom.segm.)	frame-rate	frame-rate

Table 1: Comparison of approaches and runtime performance of object-level dynamic dense SLAM methods (ElasticFusion serves as reference for static dense SLAM).

come from the camera’s ego-motion, but from large moving objects, such as walking people. Overcoming these difficulties is currently at the front of research into SLAM systems.

## 1.1 Previous work

To make SLAM more robust, recent research has tackled dynamic scenes. We can discriminate three research directions in dynamic SLAM: static background reconstruction, non-rigid object reconstruction, and reconstruction of multiple dynamic rigid objects. The first approach, static background reconstruction, concentrates solely on mapping the motionless part of the scene and on accurate camera tracking [19, 20], while explicitly detecting and excluding dynamic foreground objects. The second approach, non-rigid object reconstruction, focuses on objects undergoing deformation, such as humans [2, 8, 13]. The third approach, dynamic multi-object reconstruction, aims to explicitly model individual rigid objects by tracking every moving object in the scene individually, creating a sub-map for it, and fusing observations only into the correct sub-map [10, 18, 24].

Another opportunity to improve SLAM comes from adding semantic information to the reconstructed maps [8, 9] to aid scene understanding. Since object detection and classification can now be performed at interactive rates using convolutional neural networks (CNN), SLAM substantially benefits from robust detection and segmentation of object instances. Once objects are detected, they can be tracked and reconstructed independently.

Three recent systems which provide dynamic multi-object SLAM in the above sense are Co-Fusion [10], MaskFusion [18] and MID-Fusion [24]. The most important distinction of these systems is in how they perform segmentation. Co-Fusion segments dynamic image pixels by their motion, computed via geometric and photometric residuals during ICP-based tracking [8]. The geometric segmentation of Co-Fusion can be assisted by applying an instance segmentation algorithm, SharpMask [14]. However, SharpMask does not run in real time, and the Co-Fusion map does not store any semantics. In contrast, MaskFusion attacks instance segmentation by applying Mask R-CNN [8] to a subset of frames, effectively decoupling tracking and segmentation. Even with this decoupling, MaskFusion is expensive: It requires two Nvidia TITAN X GPU cards, one for tracking at 20Hz (for three objects), one for segmentation at only 5Hz, requiring frame skipping and re-synchronization. Similarly, MID-Fusion uses pre-computed Mask R-CNN and runs at a reported framerate of 2-3Hz. Consequently, these system cannot handle fast motions well (or at all).

Moreover, MaskFusion, MID-Fusion, and Co-Fusion (with semantics) assume that objects can be detected using pre-trained categories, such as from MS COCO [2] or PASCAL VOC [4]. Moving objects which do not belong to a trained category remain undetected and are wrongly incorporated into the background map, resulting in reduced localization accuracy. This restriction to learned categories is an important practical limitation, as real environments are full of unknown (or undetectable) objects. Similar to Co-Fusion, we explicitly segment unknown dynamic regions using ICP residuals within our two-pass frame tracking.

## 1.2 Contribution

Overall, there is currently no method which can perform instance segmentation at full frame rate, nor is there a method which can handle all kinds of dynamic objects, including undetected ones. In this paper, we present *DetectFusion*, a method that aims to fill these gaps (Table 1). It employs instance segmentation in real-time (about 20 fps) on a single GPU. We handle all kinds of dynamic objects, of both the known-detected and the unknown-undetected variety. Therefore, we make the following contributions:

**Instance segmentation at full frame rate** By detecting and segmenting known object instances in each incoming RGB-D frame in real-time (we reach about 20 fps including tracking and mapping), we can create new object maps faster (just-in-time), update multiple maps more accurately, and track all maps robustly. To the best of our knowledge, all previous work uses Mask R-CNN (or one of its predecessors) for the detection and segmentation of rigid and non-rigid objects. Mask R-CNN is a two-stage instance segmentation which is impressively accurate (40mAP on the MS COCO dataset) and also reasonably fast (5Hz). However, the single-stage detector YOLOv3 is much faster (30Hz), while still very accurate in comparison (30mAP). For each detected instance, YOLOv3 only returns a 2D bounding box and not a per-pixel mask. The key idea of our method is to intersect the box with a very fast geometric segmentation algorithm [21], resulting in pixel-accurate instance segments at a similar level of quality as delivered by Mask R-CNN. In summary, our combined detection and segmentation method is must faster than Mask R-CNN, while being similarly accurate – as we will demonstrate.

**Handling of all dynamic objects** In addition to known rigid and non-rigid objects, we also aim to detect and segment all remaining unknown dynamic objects, including objects which are undetectable (*i.e.*, object categories on which the object detector has not been trained), or has simply been spuriously undetected. This further improves tracking and mapping performance substantially. Our detection and segmentation method is based on the analysis of the geometric ICP error similar to Keller *et al.* [8] and Ruenz *et al.* [10]. In comparison, our method is more efficient, while being similarly accurate.

In our experiments, we demonstrate the performance of our system on sequences from the TUM RGB-D dataset [20] as well as self-captured sequences. For quantitative evaluation, we first evaluate the camera tracking accuracy using dynamic image sequences. Second, we evaluate the reconstruction accuracy. Third, we evaluate the computation timings of our proposed system. Please also refer to our accompanying video<sup>1</sup>.

---

<sup>1</sup>[https://www.youtube.com/watch?v=Ys3FXEP3A\\_4](https://www.youtube.com/watch?v=Ys3FXEP3A_4)

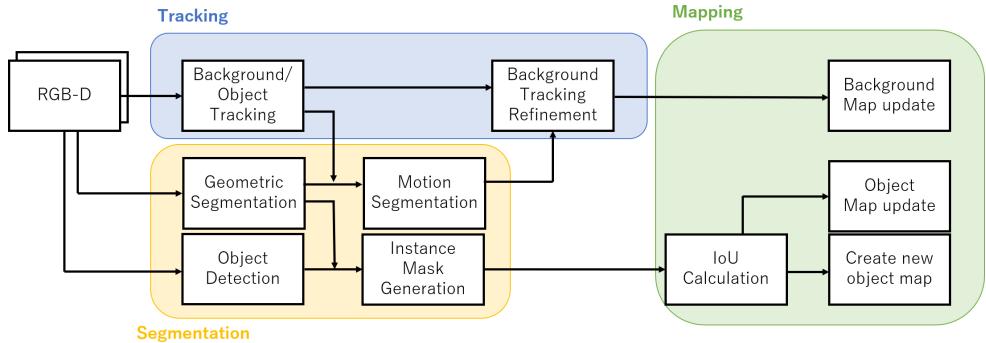


Figure 1: The system architecture of DetectFusion. The RGB-D frames of a monocular camera are segmented into object instances, which are mapped into one or more semantic maps (background and object maps). These maps can be tracked separately, allowing for dynamically moving objects.

## 2 Method

The main components of our system are tracking, mapping, and instance segmentation (consisting of object detection and segmentation) as depicted in Fig. 1. Our tracking and mapping components take great inspiration from ElasticFusion [23] and MaskFusion [18]. In particular, our system also maintains dense surfel maps.

We reconstruct one or more maps, each consisting of dense geometry and object semantics. Per default, we reconstruct a *static map* of the background. The background map only contains static scene objects. In addition, we reconstruct a variable number of semantic *object maps* which are created and updated when corresponding object instances are detected and tracked.

We can track the monocular RGB-D camera with respect to all available maps individually. Per default, we estimate the camera pose with respect to the static map. In addition, we can estimate separate camera poses for each object map which is visible in the current frame. This allows the mapped objects to move dynamically and independently of each other.

For creating the semantic object maps, we employ an object detector which is trained on a configurable (application-specific) set of semantic object categories, which we denote as *known objects*. In contrast, we denote the complement set of untrained (and thus undetectable) object categories as *unknown objects*.

Among the known objects, we furthermore distinguish between *rigid* and *non-rigid* object categories. In particular, we focus on detecting, mapping and tracking rigid object instances. Non-rigid object instances cannot be mapped and tracked by our system, and are thus explicitly ignored to not disturb tracking and mapping.

Furthermore, we aim to detect and ignore any remaining dynamically moving objects. These objects may include *undetectable unknown objects* (*i.e.*, objects of untrained categories) as well as *undetected known objects* (*i.e.*, false negatives of the detector). After detection and segmentation, these objects are explicitly ignored in tracking and mapping.

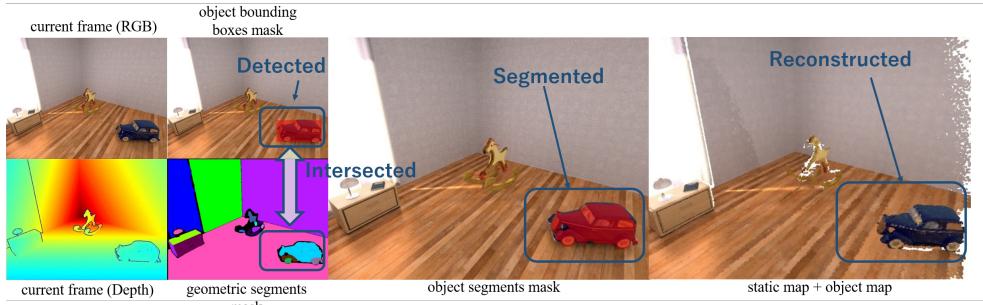


Figure 2: Instance segmentation. In this example, the moving *known object* "car" is detected and segmented and reconstructed into a separate object map. The object map is tracked separately, allowing the object to move independently from the static background.

## 2.1 Instance segmentation

We detect and segment both rigid and non-rigid objects in the current frame and leverage this knowledge for semantic mapping, as depicted in Fig. 2. Our method generates object instance segmentation masks by combining 2D object detection with geometric segmentation. In contrast to SLAM methods based on Mask R-CNN [18, 24], which compute object instance masks in two detect-then-segment stages, our method generates instance masks in a single detect-while-segment stage, which can be executed in parallel on CPU and GPU and is much faster.

**Object Detection** Detection with YOLOv3 [10, 16] results in a object bounding boxes mask localizing semantically labeled object instances. YOLOv3 can detect a variable number of object categories depending on the training dataset. Two excellent options are MS COCO [1] and PASCAL VOC [2], providing 81 and 20 object categories, respectively.

**Geometric Segmentation** We segment the 3D geometry of the current depth frame with the algorithm of Tateno *et al.* [21]. This unsupervised incremental segmentation method segments the depth frame by calculating normal and distance discontinuities in the neighborhood of the pixel. The geometric segmentation results in a geometric edges mask indicating depth discontinuities and a geometric normals mask indicating image regions with similar normals, likely belonging to the same physical object. The geometric normals mask and geometric edges mask are combined to also segment objects that have similar normals but are located at different depth levels. Finally, we apply a connected component analysis algorithm, delivering an geometric segments mask where each pixel is associated with a segment label.

**Object instance segmentation** For each detected object, we intersect the object bounding boxes mask and geometric segments mask, resulting in an object segments mask, which refines the bounding box to a pixel-accurate semantic segmentation. For each bounding box, we calculate the Intersection-over-Union (IoU) of each segment in the geometric segments mask. We handle segments which have multiple overlapping bounding boxes by sorting the bounding boxes according to their area in ascending order and assigning the object category

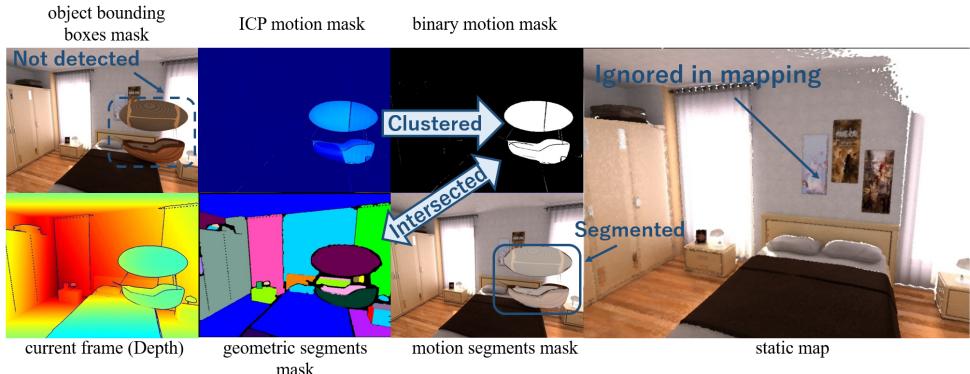


Figure 3: Motion segmentation. In this example, the moving *unknown object* "ship" cannot be detected, but is segmented using its relative motion with respect to static map. The segmented image regions are ignored in tracking and mapping.

to the segment recursively. If the IoU value is lower than a threshold, we do not assign the object category to the segment. This results in a object segments mask, which we split into a rigid object segments mask and a non-rigid object segments mask for later processing in tracking and mapping.

## 2.2 Motion Segmentation

We detect and segment the remaining undetected and undetectable dynamic objects within the current frame using ICP residual masks similar to Keller *et al.* [8] and Co-Fusion [17], as depicted in Fig. 3. We start by registering the static reference frame (rendered from the static map using the viewpoint of the previous frame) with the current frame to obtain an ICP motion mask consisting of geometric residual values of ICP pixel correspondences. High residual values highlight mismatches between the static map and the current frame, indicating dynamic image regions.

The ICP motion mask is furthermore clustered into static regions (inliers) and dynamic regions (outliers), resulting in a binary motion mask. We compute the binary motion mask by applying K-means onto the residual values of ICP motion mask, receiving two clusters and its corresponding centroid residual values. The cluster with the larger centroid residual value is selected as the dynamic cluster. Dynamic regions resulting from different moving objects are generally clustered correctly because in practice their peaks are sufficiently far from the static peak in the residual histogram.

Since the measurable object movement between two successive frames may be small, and, thus, the ICP motion mask often highlights only parts of the moving object, the binary motion mask may not contain all moving object pixels. For this reason, we compute the IoU between the binary motion mask and the geometric segments mask, resulting in a motion segments mask that covers the entire moving object, or, at least, larger parts of it. In case of no moving objects, the binary motion mask may contain spurious dynamic pixels located on object edges that are removed by the intersection since the geometric segments mask explicitly does not contain these edge pixels. The actual IoU computation is as same as for instance segmentation (see Sec. 2.1).

## 2.3 Tracking

We track the camera with respect to all maps which are visible in the current frame. For each visible map, we create a reference frame using the camera pose of the previous frame. Each reference frame is aligned with the current frame via a method similar to ElasticFusion [23]. Given the 6DoF camera pose of the previous frame, we iteratively minimize a cost function consisting of photometric and geometric error terms over the unknown relative 6DoF transformation. We optimize with a Gauss-Newton non-linear least-squares algorithm on a three-level coarse-to-fine image pyramid.

We perform camera tracking in two stages. In the first stage, we align the current frame with reference frames rendered from both the static map and all visible object maps. In the second stage, we refine the camera pose of the static map by leveraging calculations between the first and the second stage.

In particular, we calculate an *invalid mask* which marks dynamic objects which have previously been detected in the current frame, including (1) image regions of known non-rigid objects (see Sec. 2.1) and (2) image regions of unknown dynamic objects (see Sec. 2.2). *I.e.*, the invalid mask is the union of motion segments mask and non-rigid object segments mask, and marks outlier correspondences between current frame and reference frame of the static map stemming from dynamic objects. We omit these regions during tracking to improve the camera pose estimate in the second stage.

## 2.4 Mapping

We reconstruct and maintain multiple surfel maps similar to ElasticFusion [23]. In particular, we adopt their strategies for creating and updating the surfel maps, including the rules for initializing and updating the attributes (*i.e.*, position, normal, color, radius, weight) and the state (*i.e.*, active, inactive) of individual surfels.

Our multi-map reconstruction method is similar to MaskFusion [18]. We fuse image pixels of the current frame into their corresponding maps. We project the visible object map onto the current frame using the estimated camera pose, resulting in a map mask. By intersecting each map mask with the rigid object segments mask, we establish matches between object maps and the detected rigid objects.

We distinguish the following cases when fusing image pixels with our maps: For each map-object match, we update and potentially extend the object map with the corresponding object pixels. For each unmatched object, we create a new object map, resulting in an initial set of surfels. For each unmatched map, we update and potentially shrink the corresponding object map. Finally, we remove pixels identified by the invalid mask (see Sec. 2.3) and fuse the remaining pixels into the static map.

## 3 Evaluation

We conducted an extensive set of experiments to evaluate our system, including quantitative and qualitative results on tracking, reconstruction, segmentation and runtime performance, which we generated using synthetic and real, public and self-made datasets. Our experiments were performed on a computer equipped with an Intel Core i7-6950X 3.0GHz CPU, a GeForce GTX 1080Ti GPU, 64GB RAM, and running Ubuntu 16.04. YOLOv3 was trained

Setting	Sequence	ATE RMSE (cm)				
		EF	SF	CF	MF	DF
Slightly Dynamic	f3s_static	<b>0.9</b>	1.3	1.1	2.1	1.5
	f3s_xyz	<b>2.6</b>	4.0	2.7	3.1	5.2
	f3s_halfsphere	13.8	4.0	<b>3.6</b>	5.2	4.1
Highly Dynamic	f3w_static	6.2	<b>1.4</b>	55.1	3.5	3.6
	f3w_xyz	21.6	12.7	69.6	10.4	<b>8.5</b>
	f3w_halfsphere	20.9	39.1	80.3	10.6	<b>7.2</b>

Table 2: Comparison of Absolute Trajectory Error (lower is better) on TUM-RGB-D dataset.

on the MS COCO dataset using the default weights<sup>2</sup>.

We tested our method with our own sequences which we recorded using a Kinect v1 with  $640 \times 480$  resolution. For reasons of limited paper space, we would like to refer our readers to the accompanying video for these results.

### 3.1 Tracking performance

We evaluated the tracking accuracy of our system and compared with related systems on the widely used TUM RGB-D SLAM dataset [20]. This dataset provides six image sequences showing dynamically changing environments, which can be further divided into three slightly dynamic (*fr3/sitting*) and three highly dynamic (*fr3/walking*) sequences, each having different camera motion trajectories (static, xyz: up-down-left-right, spherical).

We compare our method with several state-of-the-art dynamic and static SLAM systems (see Table 1): ElasticFusion (EF) [23], StaticFusion (SF) [19], Co-Fusion (CF) [10], MaskFusion (MF) [18] and DetectFusion (DF, ours). We focus on dense SLAM systems which are known to run in real-time, and thus omitted non-real-time systems such as MID-Fusion, and sparse systems such as DynaSLAM [11].

Table 2 summarizes the Absolute Trajectory Error (ATE) when tracking the camera with respect to the static maps reconstructed by the respective systems. Overall, our system delivers results that are comparable to previous work or better.

On the highly dynamic sequences *fr3/walking\_xyz* and *fr3/walking\_halfsphere*, our system performs best, although not by a large margin. We assume that our unique combination of instance and motion segmentation makes the difference. In contrast to related systems, our system can not only detect and segment known rigid and non-rigid object instances, but also segment dynamic image regions which are covered by unknown (*i.e.*, undetectable or spuriously undetected) objects. Since we either explicitly ignore or reconstruct the segmented objects into separate object maps, the reconstruction of our static map contains less outliers and thus allows for more accurate camera tracking.

### 3.2 Segmentation and reconstruction performance

We conducted a comparison of the instance segmentation and reconstruction performance between our system and the closely related SLAM systems Co-Fusion [10] and MaskFusion [18], using a synthesized image sequence (*room4-noise*) from the Co-Fusion dataset<sup>3</sup>.

<sup>2</sup><https://pjreddie.com/darknet/yolo/>

<sup>3</sup><https://github.com/martinruenz/co-fusion>

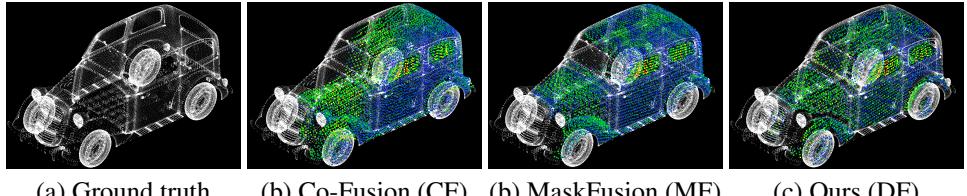


Figure 4: Ground truth model and reconstructed SLAM object maps.

Metric	Co-Fusion (CF)	MaskFusion (MF)	Ours (DF)
Accuracy	0.792	<b>0.896</b>	0.810
Completeness	0.525	0.457	<b>0.591</b>

Table 3: Comparison between ground truth model and SLAM object maps on completeness and accuracy ratios in the range [0, 1] (higher values are better).

This image sequence contains a moving *car* object to which we also have the corresponding ground truth 3D model available. Each SLAM system received the image sequence as input, delivering a set of segmented and reconstructed object maps. All systems delivered an object map of the desired *car*, depicted in Fig. 4.

For the comparison of the SLAM maps with the ground truth 3D model, we calculated *completeness* and *accuracy* ratios. Table 3 shows the results. While MaskFusion reconstructed the most accurate map, ours reconstructed the most complete map of the *car* object.

### 3.3 Runtime performance

Table 4 shows the average time spent on each processing stage of our system. The numbers have been measured using RGB and depth input images with  $640 \times 480$  resolution. MaskFusion had an average frame time of 33ms (30fps) using two cards with Nvidia Titan X GPU, while performing segmentation only every 12<sup>th</sup> frame. DetectFusion achieved an average frame time of 46ms (22fps) with one 1080 Ti, while performing segmentation in every frame. Note that this is possible because geometric segmentation and object detection do not depend on each other, allowing to run these processing steps in parallel.

Component	Runtime [ms]
Initial Tracking	3.80 / model
Geometric segmentation *	6.16
Object Detection *	19.1
Motion Segmentation	2.72
Object Mask Generation	0.57
Camera Pose refinement	6.30
Mapping	2.05 / model
Total	$46.1 + 5.16 / \text{model}$

Table 4: Average time (ms) spent on each processing stage of our system (steps marked with \* are processed simultaneously).

## 4 Discussion and Conclusions

In this paper, we presented an efficient approach for object-level dynamic SLAM. It robustly tracks the camera pose in a highly dynamic environment and continuously estimates semantics and object geometry. Experimental results show that our method outperforms previous work in terms of camera tracking accuracy in highly dynamic scenes, while being computationally much lighter than comparable approaches.

There are several important steps left for future work. First, better object detection would help both tracking and mapping. We inherit not only the impressive strengths, but also the limitations of YOLOv3. Even if properly trained and configured, YOLOv3 is not always able to correctly detect the known objects in each frame. We found in our experiments that cluttered scenes and the presence of occluded objects are particularly problematic. However, since YOLOv3 is known to have almost the same accuracy compared to Mask R-CNN, these limitations similarly affect the related SLAM systems. Our system also provides some robustness to occasional false (*i.e.*, missing or wrong) detections. Missing detections (*i.e.*, false negatives) are either, in case of dynamic objects, gracefully ignored via motion segmentation, or, in case of static objects, mitigated by the built-in robustness of the mapping and tracking processes. Wrong detections (*i.e.*, false positives) may result in redundant or duplicate object maps that our system currently cannot detect, delete or merge.

Moreover, intersection of object bounding boxes with the geometric segmentation is not always robust. Our current heuristic greedily assigns segments to instances and sometimes delivers wrong assignments or fails, in particular, if complex occlusion occurs. In addition, detection and segmentation of small and non-convex objects, such as hands or arms, can be challenging. While we favourably compared our system with SLAM based on Mask R-CNN, a direct comparison of our instance segmentation method with Mask R-CNN remains to be done.

With faster computing, it may become easier to integrate per-pixel semantic segmentation into SLAM systems at real-time rates. Nonetheless, our approach of combining fast detection with fast geometric segmentation will remain relevant, since it can quickly and inexpensively uncover the majority of dynamic object configurations. This can help to concentrate the computational power required for instance segmentation (on top of already obtained detection results) on the more difficult parts of the scene.

**Acknowledgements** This work was enabled by the Japan Science and Technology Agency (JST) under grant CREST-JPMJCR1683, and the Austrian Research Promotion Agency (FFG) under grant RSA-859208 (MATAHARI). We would like to thank the reviewers for their valuable comments.

## References

- [1] Berta Bescos, Jose M. Facil, Javier Civera, and Jose Neira. DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, Oct. 2018.
- [2] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, Pushmeet Kohli, Vladimir Tankovich, and Shahram Izadi. Fusion4D: Real-time

- Performance Capture of Challenging Scenes. *ACM Transaction on Graphics*, 35(4):114:1–114:13, 2016.
- [3] Mingsong Dou, Philip Davidson, Sean Ryan Fanello, Sameh Khamis, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, and Shahram Izadi. Motion2Fusion: Real-time Volumetric Performance Capture. *ACM Transaction on Graphics*, 36(6):246:1–246:16, 2017.
- [4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun. 2010.
- [5] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision*, pages 2980–2988, Oct. 2017.
- [6] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion. In *IEEE International Conference on 3D Vision*, pages 1–8, Jun. 2013.
- [7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *European Conference on Computer Vision*, pages 740–755, 2014.
- [8] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric Object-Level SLAM. In *International Conference on 3D Vision*, pages 32–41, Sep. 2018.
- [9] John McCormac, Ankur Handa, Andrew J. Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. *IEEE International Conference on Robotics and Automation*, pages 4628–4635, 2017.
- [10] Raul Mur-Artal and Juan D. Tardos. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, (5):1255–1262, Oct. 2017.
- [11] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, Oct. 2011.
- [12] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense tracking and mapping in real-time. In *IEEE International Conference on Computer Vision*, pages 2320–2327, Nov. 2011.
- [13] Richard A. Newcombe, Dieter Fox, and Steven M. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 343–352, 2015.
- [14] Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer International Publishing, 2016.

- 
- [15] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6517–6525, 2017.
  - [16] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.
  - [17] Martin Rünz and Lourdes Agapito. Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects. In *IEEE International Conference on Robotics and Automation*, pages 4471–4478, May 2017.
  - [18] Martin Rünz, Maud Buffier, and Lourdes Agapito. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 10–20, Oct. 2018.
  - [19] R. Scona, M. Jaimez, Y. R. Petillot, M. Fallon, and D. Cremers. StaticFusion: Background Reconstruction for Dense RGB-D SLAM in Dynamic Environments. In *IEEE International Conference on Robotics and Automation*, pages 1–9, May 2018.
  - [20] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A Benchmark for the Evaluation of RGB-D SLAM Systems. In *IEEE/RSJ International Conference on Intelligent Robot Systems*, pages 573–580, Oct. 2012.
  - [21] K. Tateno, F. Tombari, and N. Navab. Real-time and scalable incremental segmentation on dense SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4465–4472, Sep. 2015.
  - [22] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. J. Kelly, and S. Leutenegger. Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping. *IEEE Robotics and Automation Letters*, 3(2):1144–1151, Apr. 2018.
  - [23] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. ElasticFusion: Real-time dense SLAM and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.
  - [24] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew J. Davison, and Stefan Leutenegger. MID-Fusion: Octree-based Object-Level Multi-Instance Dynamic SLAM. *CoRR*, abs/1812.07976, 2018.