Semantic Simultaneous Localization and Mapping for Augmented Reality

Bing Yu, Yang Li, Chao Ping Chen*, Nizamuddin Maitlo, Jiaqi Chen, Wenbo Zhang, and Lantian Mi

Smart Display Lab, Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China

Email: ccp@sjtu.edu.cn

Abstract

We propose a semantic SLAM for augmented reality, which combines SLAM-based navigation and YOLO-based object detection. Our solution is able to simultaneously create sparse map points and annotate them with detected objects. Our experiments are implemented with Nvidia's GTX 1080Ti on KITTI's datasets.

Keywords

SLAM; augmented reality; ORB; YOLO; navigation; semantic; object detection.

1. Introduction

Augmented reality (AR) is a technology that could merge the virtual world with the real world in real time [1]. To seamlessly merge the above two worlds, simultaneous localization and mapping (SLAM) is regarded as an enabling technology to meet this end. The traditional SLAMs are usually employed for the navigation. However, for AR applications, semantic SLAMs are more advantageous as they could not only tell where it is but also what it is. J. Civera et al. proposed a semantic SLAM system, which runs EKF monocular SLAM in parallel with an object detection thread in search of SURF correspondences and checking geometric compatibility [2]. SLAM++ [3] detects real-world objects in RGB-D data by matching 3D models of known object classes. However, both of these two semantic SLAM systems perform well only in a highly-controlled environment due to the computational complexity and the localization accuracy of visual SLAM. The Dense Planar SLAM system [4] seamlessly maps an environment using planar and non-planar regions while tracking the sensor pose in real-time. SemanticFusion [5] combines Convolutional Neural Networks (CNNs) and a state-of-the-art dense SLAM system, ElasticFusion to create a dense semantic map. All of the four semantic SLAM systems mentioned above are focusing on the dense map, though semantic SLAM on the sparse map is fairly enough for some applications in fact.

In this paper, by combining ORB-SLAM2 [6] and YOLO [7], we proposed a semantic SLAM system that can annotate map points in sparse maps with detected objects, which performs well on KITTI's datasets when processing monocular or stereo input. Firstly, ORB-SLAM2 is chosen as the backbone of our system for feature extraction, pose tracking and raw map creating because it is a feature-based SLAM system and it can calculate the trajectory of the camera in real time and generate the sparse 3D reconstruction of the scene. Then, YOLO is chosen for object detection because YOLO is a state-of-the-art, real-time object detection system. On a Titan X, it processes images at 40-90 FPS and has a mAP on VOC 2007 of 78.6% and a mAP of 48.1% on COCO test-dev [7].

2. Proposed Solution

Architecture: The architecture of the proposed SLAM system is outlined in Fig. 1. ORB-SLAM2 is used as the backbone of the system. Inspired by the idea of the parallel processing used in Ref. [2], ORB-SLAM2 is run in parallel with the detecting thread, in which YOLO is implemented to detect objects in keyframes. Instead of using the whole image sequence, keyframes are adopted to detect objects because in ORB-SLAM2, only map points in keyframes will be stored for the final sparse map. When the tracking thread finishes inserting a keyframe, it will send this keyframe to the local mapping thread. It is designed that the keyframe, which is sent to local mapping thread, will be as well sent to the detecting thread, which will detect objects with YOLO on the received keyframe. When the proposed system is run in stereo mode, the detecting thread is designed for the object detection on the left image of the keyframe.

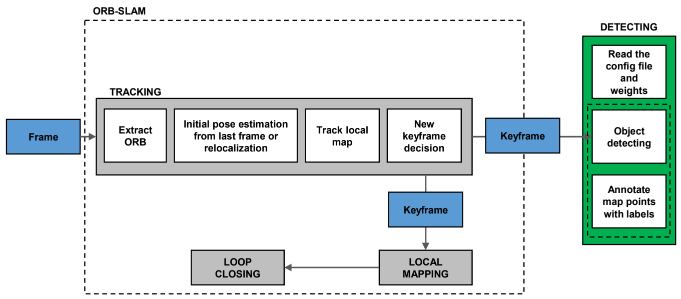


Figure 1. Architecture of the proposed semantic SLAM system. Threads of ORB-SLAM2 and the designed detecting thread are shown in the middle and right, respectively. Created by ORB-SLAM2's tracking thread, a new keyframe will be sent to the detecting thread which is designed to utilize YOLO to perform object detection on the received keyframe and annotate map points in the received keyframe with detected objects.

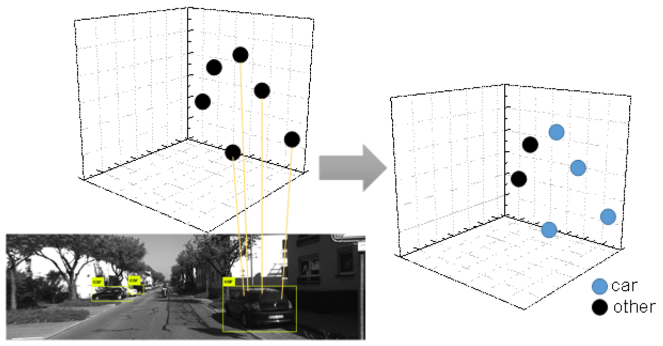
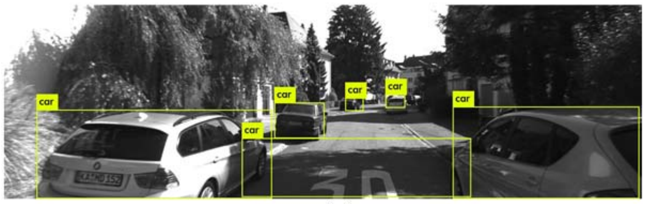


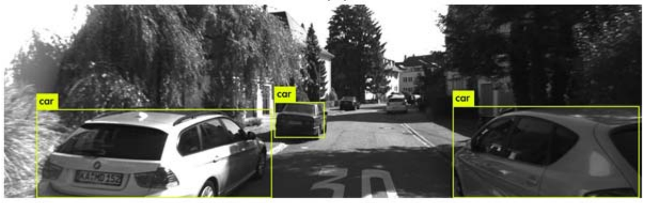
Figure 2. Principle of the proposed system. 6 ORB features are assumed in this keyframe, which are corresponding to the six map points in the world coordinate system. After YOLO finishes object detection, all the ORB features belong to the 'car' bounding box are found and the corresponding map points will be labeled as 'car'.

Algorithm: As shown in Fig. 2, 6 ORB features are assumed in this keyframe, which correspond to 6 map points in the world coordinate system. After YOLO finishes object detection, all the ORB features belong to the 'car' bounding box are labeled as 'car' and colored by blue. If the procedure mentioned above is done in all the keyframes, all the map points in the sparse map belong to cars will be found and the goal of adding semantic content to the sparse is achieved.

YOLO predicts *x*, *y*, *w*, *h* and confidence for each bounding box [7]. It calculates the class-specific confidence scores for each box by multiplying the conditional class probabilities and the individual box confidence predictions [7]. These class-specific confidence scores encode both the probability of that class appearing in the box and how well the predicted box fits the object [7]. If the class-specific confidence scores are larger than the threshold, 0.24 (default value), YOLO will output the results of these bounding boxes [7]. However, YOLO does not perform well on KITTI's datasets when using the default value 0.24 of the threshold. When the system uses the experienced value 0.5, it will seldom come across a case of mistaken identity at the cost of missing the distant object, as shown in Fig. 3. But the risk of missing the distant object can be compensated by the following keyframes which may captures the same object.



(a)



(b)

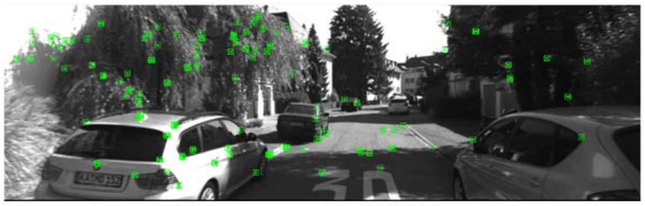
Figure 3. (a) Detecting results when YOLO uses the default value 0.24 as the threshold of the class-specific confidence score. (b) detecting results when YOLO uses the experienced value 0.5 as the threshold of the class-specific confidence score. According to the result of (a) and (b), it can be concluded that when the system uses the experienced value 0.5, it will seldom come across a case of mistaken identity at the cost of missing the distant object.

3. Results and Discussion

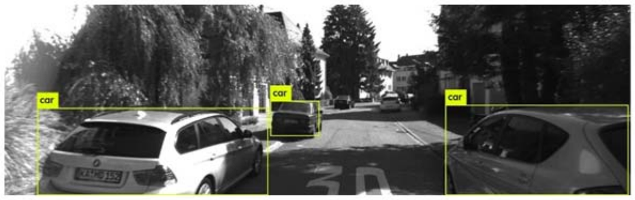
Experimental Conditions: Intel Core i7-7700K, Nvidia GTX 1080Ti, Corsair 16 GB RAM, Samsung 250 GB SSD, Ubuntu 16.04 LTS, KITTI's datasets.

Since there are a large number of cars within KITTI's datasets, we decide to run the SLAM system to annotate all the map points with detected cars in order to achieve the best experimental results. As shown in Fig. 4, the map points annotated with detected cars in the image of the 63th keyframe are definitely in the view of the 63th keyframe. Thus, it can be concluded that when the system is tested on KITTI's datasets and uses monocular sequences as input, it can successfully annotate map points that belong to cars, although the profile of cars is not recognizable.

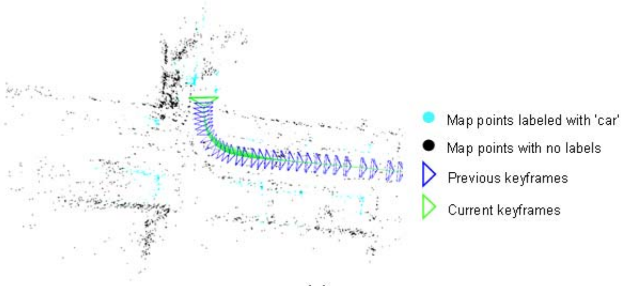
As listed in Table 2, when the system is processing monocular input, the time cost for object detection is less than the tracking time. When it comes to the stereo input, the detecting time is almost 1/3 of the tracking time. Since the detecting thread can always catch up with the tracking thread, it can be concluded that the system can perform SLAM and object detection simultaneously when processing monocular or stereo input.



(a)



(b)



(c)

Figure 4. Simulated results (monocular) of the proposed semantic SLAM system on KITTI's datasets. (a) image of the 63th keyframe; (b) object detection result of the 63th keyframe; (c) corresponding sparse map created by the system. A great number of ORB features are on the white car (left-bottom) and some on the car (right-bottom). The corresponding blue map points are definitely in the view of the current keyframe (green triangle), as shown in (c).

Table 1. Tracking time of ORB-SLAM2 on KITTI's datasets

Type of input	Median tracking time (ms)	Mean tracking time (ms)
Monocular	23.6523	29.1592
Stereo	50.0219	53.808

Table 2. Tracking time and detecting time of the proposed SLAM system on KITTI's datasets

Type of input	Median tracking time (ms)	Mean tracking time (ms)	Detecting time (ms)
Monocular	24.075	29.3642	<19
Stereo	50.784	54.4569	<19

Besides, since YOLO is implemented on another thread to perform object detection, it has few side effects to the ORB-SLAM2's threads. Fig. 5 and Fig. 6 demonstrate the difference between the sparse map created by ORB-SLAM2 and the sparse map created by the proposed system, respectively. According to Table 1 and Table 2, there is only a small increase on the tracking time due to the addition of the detecting thread.

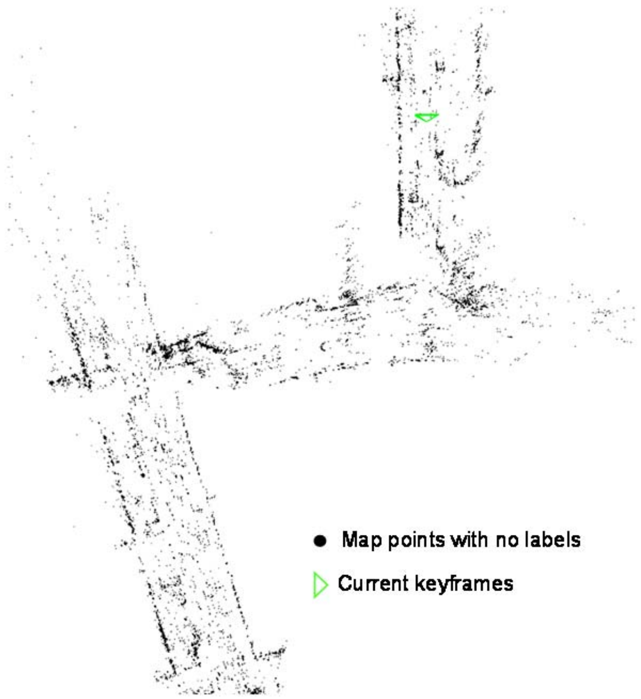


Figure 5. Sparse map created by ORB-SLAM2 (monocular) on KITTI's datasets.

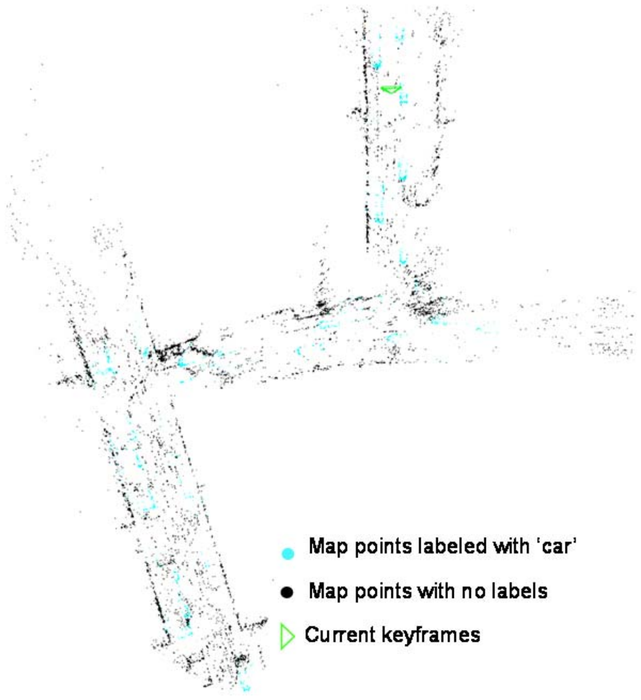


Figure 6. Sparse map created by the proposed semantic SLAM system (monocular) on KITTI's datasets. Both Fig. 5 and Fig. 6 demonstrate the same area, but in Fig. 6 map points belong to cars are annotated with detected car.

4. Conclusions

A semantic SLAM system has been proposed to apply on the sparse map and it performs well on KITTI's datasets. Based on the simulations, its key performance including tracking time and detecting time has been studied. Since the time cost for object detection is less than the tracking time, it can be concluded that it can perform SLAM and object detection simultaneously when processing monocular or stereo input. Plus, due to the parallel processing, the proposed system has few side effects to the SLAM's performance. By supporting both navigation and object detection, our semantic SLAM could offer some new possibilities that might be difficult with the conventional SLAMs. For example, when equipped with a near-eye display [8-10], semantic SLAM would allow a user to see not just a three-dimensional map, but the labeling of objects that are recognizable.

5. Acknowledgements

Science and Technology Commission of Shanghai Municipality (1701H169200); Shanghai Jiao Tong University (AF0300204, WF101103001/085); Shanghai Rockers Inc. (15H100000157).

6. References

[1] W. Barfield, *Fundamentals of Wearable Computers and Augmented Reality 2nd Edition* (CRC Press, 2015).

[2] J. Civera, D. Galvez-Lopez, L. Riazuelo, J. D. Tardos, and J. M. M. Montiel, "Towards semantic SLAM using a monocular camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011), pp. 1277–1284.

[3] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: simultaneous localisation and mapping at the level of objects," in *IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 1352–1359.

[4] R. F. Salas-Moreno, B. Glocker, P. H. J. Kelly, and A. J. Davison, "Dense planar SLAM," in *IEEE International Symposium on Mixed and Augmented Reality* (2014), pp. 157–164.

[5] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: dense 3D semantic mapping with convolutional neural networks," in *IEEE International Conference on Robotics and Automation* (2017), pp. 4628–4635.

[6] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," IEEE Transactions on Robotics **33**(5), 1255–1262 (2017).

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 779–788.

[8] L. Zhou, C. P. Chen, Y. Wu, Z. Zhang, K. Wang, B. Yu, and Y. Li, "See-through near-eye displays enabling vision correction," Opt. Express **25**(3), 2130–2142 (2017).

[9] Y. Wu, C. P. Chen, L. Zhou, Y. Li, B. Yu, and H. Jin, "Design of see-through near-eye display for presbyopia," Opt. Express **25**(8), 8937–8949 (2017).

[10] C. P. Chen, L. Zhou, J. Ge, Y. Wu, L. Mi, Y. Wu, B. Yu, and Y. Li, "Design of retinal projection displays enabling vision correction," Opt. Express **25**(23), 28223–28235 (2017).