SEMANTIC-RTAB-MAP (SRM): A SEMANTIC SLAM SYSTEM WITH CNNS ON DEPTH IMAGES

MINGYUAN MAO¹, HEWEI ZHANG¹, SIMENG LI¹ AND BAOCHEANG ZHANG¹,²*

¹Beihang University, Beijing, China
²Shenzhen Academy of Aerospace Technology, Shenzhen, China

(Communicated by Dapeng Tao)

ABSTRACT. SLAM (simultaneous localization and mapping) system can be implemented based on monocular, RGB-D and stereo cameras. RTAB-MAP is a SLAM system, which can build dense 3D map. In this paper, we present a novel method named SEMANTIC-RTAB-MAP (SRM) to implement a semantic SLAM system based on RTAB-MAP and deep learning. We use YOLOv2 network to detect target objects in 2D images, and then use depth information for precise localization of the targets and finally add semantic information into 3D point clouds. We apply SRM in different scenes, and the results show its higher running speed and accuracy.

1. Introduction. In recent years, SLAM (simultaneous localization and mapping) has been attracting increasing attention in the field of computer vision. Beginning with the monocular camera, researchers have achieved great improvement in SLAM. However, limited by the amount of computation, researchers generally think that real time and dense maps cannot be built only using CPUs. Namely, most real time SLAM system can only build semi-dense maps. Additionally, the maps built by the SLAM projects do not include semantic information, so robots can only localize themselves but cannot really understand the scenes. Therefore, two main challenges of SLAM are the dense 3D map reconstruction and the understanding of scenes, so-called semantic SLAM. We build a dense map with semantic information so that robots could understand the scenes as human-beings do.

There are mainly two different ways to realize a semantic SLAM system. One is to segment the image first and then use the 2D-3D transfer to produce a 3D point cloud. For instance, Li et al. [8] made a mono semantic SLAM by combining LSD-SLAM with deep learning methods. Another is to implement segmentation based on the whole 3D point cloud by using deep learning methods, such as the PointNet proposed by Qi et. al [1].

In this paper, we combine a matured SLAM system named RTABMAP with a convolutional neural network named YOLO and implement some methods in image processing to build up a semantic SLAM system. In our method, the 2D image acquired by the camera will be first processed and then be transferred to a 3D point cloud based on the RTAB-MAP [4, 5, 6, 7] because it proves to be capable of constructing dense 3D point clouds. Our main idea is reflected in Fig. 1. We

2010 Mathematics Subject Classification. Primary: 58F15, 58F17; Secondary: 53C35.
Key words and phrases. Semantic SLAM, YOLO, real-time, depth-image, region growing.
* Corresponding author: bczhang@buaa.edu.cn.
first need to roughly localize the objects in Keyframes by using the bounding boxes. Considering that SLAM has to be real-time, we decided to use the network of YOLO (you only look once) [13] to do object detection and localization (with bounding boxes). Since generally there will be a depth discrepancy between the target object and the surrounding objects, we utilize the corresponding depth information of the image to precisely localize the target objects. Specifically, we use the Canny operator to detect edges of the target object on the depth image and then process it with the region growing algorithm, and we finally colorize the target object in the 3D point cloud.

The rest of the paper is organized as follows: section II summarizes the long-term and large-scale work of SLAM and the work of utilizing the deep learning methods to do image segmentation and object detection; section III describes our method in detail, including its principles and flow; section IV shows our results of experiments and comparisons with other methods qualitatively and quantitatively; section V concludes the main idea of this paper, including its advantages and disadvantages.

2. Related work. In recent years, many researchers have made explorations and attempts in the field of semantic SLAM. For most of the probabilistic approaches, 2D images are processed first and then convert into 3D information, while others are based on segmenting the 3D point cloud directly.

In 2016, Li et. al [8] utilized the convolutional neural network (CNN) to segment images on the basis of LSD-SLAM and saved the result in the 3D point cloud, achieving impressive results (Fig. 2).

However, LSD-SLAM utilizes the Direct Method to realize tracking, making it very susceptible to camera intrinsics and exposure. In addition, LSD-SLAM is mainly used on monocular cameras and can only build semi-dense 3D point clouds, so that the semantic map will be relatively incomplete.

McCormac et. al [10] also built a CNN to do image segmentation based on ElasticFusion [18] and then saved the results in the 3D point cloud, realizing good results. Nevertheless, ElasticFusion can only be used in small-scale scenes, which will not be useful when being utilized on large-scale scenes involved in SLAM. Moreover, it is not accurate to directly segment the 2D image because if the boundary of the segmented target goes beyond the scope of the real target, there will be a situation in which two objects far from each other are labeled as one object.
Qi et al proposed the PointNet [1] to realize the direct segmentation on 3D point clouds. However, segmenting the point cloud directly is much more complicated than 2D image segmentation and usually brings about higher network complexity. Besides, the input of the network needs to be a previously built 3D point cloud. Therefore, although this method can segment a 3D map that has already been constructed, it cannot meet the requirement of SLAM’s being real-time.

Sunderhauf et al [17] built a semantic point cloud based on ORB-SLAM2 [11]. They proposed to do object detection by Single-shot Multi-box Detector (SSD) [9] first, and then add semantic information by image segmentation utilizing the depth information. Nevertheless, ORB-SLAM2 can only extract sparse feature points. A map with sparse feature points cannot help us with navigation and collision avoidance. Additionally, they chose to extract the edges in the 3D space, increasing the algorithm complexity, and yet the results are not better than those produced by segmenting 2D images directly.

3. Semantic-RTAB-MAP (SRM). Our idea resides in using the CNN to process the 2D image and then add the semantic information into the 3D point cloud. In this way, we can establish a complete system of semantic SLAM which would do localization while moving and thus construct the semantic 3D map. However, to avoid the error caused by segmentation and simplify the structure of our neural network and improve the real-time performance, we decide to detect objects and roughly localize the objects first rather than only do segmentation. We will add semantic information based on the RTAB-MAP because it can build dense point clouds and realize long-term and large-scale SLAM. For the neural network, we select YOLOv2 [14] to recognize the objects in 2D images. In the following paragraphs, we will briefly introduce the RTAB-MAP and YOLO that has been used in our system. After that, we will detailly explain the algorithm flow of SRM.

3.1. RTAB-MAP. In the field of RGB-D SLAM, RTAB-MAP is a relatively traditional project which is mainly applied on RBD-D sensors and can also be used on stereo cameras. It includes all of the parts that a typical RGB-D SLAM project should have: the feature-based visual odometry, the loop closure detection based
on bag-of-words, the pose optimization and mapping. Now, we are allowed to directly get the binary program from ROS. In addition, RTAB-MAP builds Working Memory (WM) and Long-Term Memory (LTM) by using the method of Memory Management (shown in Fig. 3) proposed by Labb’ et. al [7] and keeps the number of Keyframes involved in the loop closure detection under a fixed limit by sifting the frames. In this way, long-term and large-scale constraint can be satisfied.

Besides, RTAB-MAP is also capable of constructing dense real-time point clouds. We will process keyframes RTAB-MAP received to add semantic information.

3.2. YOLO. Because of the disadvantages of image segmentation mentioned in section II, we decide to detect objects first to roughly predict where objects are before image segmentation and localize the objects more precisely later. In recent years, many major breakthroughs have been achieved in the field of object detection. Generally, the detection systems can be classified into two categories. The first one includes algorithms implementing R-CNN [2] on the basis of Region Proposal while the second one includes one-state algorithms which only utilize one CNN to directly predict the classes and localizations of objects, such as YOLO [13] and SSD [9]. Although systems in the first category can achieve higher accuracy on detection, they usually work at a lower speed. On the opposite, systems in the other category have higher running speed and yet have lower accuracy. YOLO is a real-time network which can run very fast, fully satisfying the requirements of SLAM. Moreover, we will segment the depth image after rough predictions on bounding boxes to precisely localize the objects. The structure of YOLOv1 is shown in Fig. 4, which was first proposed by Redmon et. al in 2015 [13].

However, YOLOv1 struggles with objects that are small or very close to each other, reflecting its low generalization capacity. To address these shortcomings, the softmax layer and a pooling layer were removed in YOLOv2 [14] network, proposed in 2017. Some results produced by YOLOv2 are shown in Fig. 5. So far, YOLOv2 network is one of the state-of-the-art detection systems in the field of object detection which is faster than many other detection networks, such as FasterR-CNN [16], ResNet [3] and SSD. Consequently, we will use YOLOv2 to detect objects.
3.3. Method. Since RTAB-MAP is capable of constructing the local 3D point cloud based on keyframes with depth information and constructs the global point cloud by calculating the camera pose, we only need to process Keyframes to label the objects and save the labels in the 3D point cloud. The flow chart of our method is shown in Fig. 6.

Object Detection Using YOLOv2. We input keyframes into the YOLOv2 network to recognize the objects in the images and draw corresponding bounding boxes. The classes and localizations of these objects are recorded in the vector denoted as $\bar{r}$ which is defined as follows:

$$\bar{r} = (r_l, r_r, r_t, r_b, r_c)$$

(1)

Where $r_l$ represents the distance from the left edge of the image to the left edge of the bounding box; $r_r$ represents the distance from the left edge of the image to the right edge of the bounding box; $r_t$ represents the distance from the top edge of the image to the top edge of the bounding box; $r_b$ represents the distance from the top edge of the image to the bottom edge of the bounding box; $r_c$ represents the class of the target object. We deliberately assign different colors to different classes of objects on the purpose of visualizing the semantic results in the final 3D point cloud.
Processing the depth image with TRIAL RESTRICTION —— precise localization. In the case of RGB-D camera, when the Keyframe is collected, the corresponding depth image will be simultaneously received. In the case of stereo camera, the depth image can be generated by OpenCV based on the image captured by the left camera. After that, in the depth image, we can localize the target objects. Next, we will do segmentations in these target areas one by one.

For example, if we have detected a handbag in the original image and draw the corresponding bounding box, its localization and label can be recorded in the vector $\vec{r}$ with which we can find the corresponding area in the depth image, denoted as DepthROI by using (implementing) the ROI tool in OpenCV. In this process,
we would notice that the position of the objects in depth image don not accu-
ately match the objects' in original image, but we don’t need to worry about it
because the original RTAB-MAP project will address this problem automatically
when building the 3D point cloud. However, up to now we only have the rough
localization of the handbag by predicting the bounding box, so we need to localize
it more precisely. Therefore, we implement the region growing algorithm to fur-
therly process DepthROI, separating the handbag from the background. We do
not employ another neural network to accurately extract the edge of the handbag
because in that way the real-time performance improved by YOLO network will be
compromised.

Since the target object and the surrounding object usually have very different
depths, pixel values of DepthROI will change drastically on the edges of the hand-
bag. If we implement edge detection, we will be able to acquire the outline of the
handbag. Then, we will select a start point for region growing which will be stopped
as soon as it reached the edge of the handbag. Consequently, we can separate the
handbag from the background more precisely.

Canny operator is implemented here to extract the edges because it can generate
almost closed edges, which is of great importance for the region growing process. On
the opposite, if the edges are not closed, the region growing process will not come to
an end until it reached the boundary of the whole image. For Canny operator, the
high threshold determines the sensitivity to edges while the low threshold determines
the continuity of the edges. These two thresholds could be adjusted based on the
specific situation. In this article, we set the low threshold to be 2 and the high threshold to be a quarter of the threshold calculated by Otsu method [12].

For choosing the start point of the region growing process, we usually meet with two kinds of situations. In general, the geometrical center is within the area of the target object. Therefore, we set the start point to be the geometrical center of DepthROI. Nevertheless, if some special objects are needed to be detected, especially for some objects with big holes in the center, then it is necessary to choose the start point with weights determined by the value of pixels in DepthROI. Since the pixels of the DepthROI in the target area are usually darker than those in the background, to make sure that the start point is located within the area of the target, we propose a method as follows.

\[
\begin{align*}
M &= \sum D_{i,j} \\
m_0 &= \sum (255 - D_{i,j}) \times i \\
m_1 &= \sum (255 - D_{i,j}) \times j
\end{align*}
\]

(2)

Where \(D_{i,j}\) is the value of the pixel whose coordinate is \((i, j)\). Therefore, the coordinate of the start point is \((m_0/M, m_1/M)\). However, if the camera has comparatively poor configurations and fails to detect the depth of relatively further distant objects, this method cannot work well.

After the start point is determined, we can start the region growing process. In our method, if the difference of pixel values between the current pixel and the pixels in its 4-neighborhood is bigger than a threshold, which is set to be 2 in this article, this current pixel will be directly removed and the pixels in its 4-neighborhood will not be labeled as the target area. Although this condition is a bit strict and we will probably separate only part of the target from the background, the remaining pixels of the target area will be labeled gradually as the camera moves. In addition, we cannot accept the overgrowth which means other class of objects are mislabeled as the current class. After region growing, we implement corrosion on DepthROI and finally achieve very good results.

**Add semantic information into Keyframes.** We utilize DepthROI in which the target has been separated from the background to add semantic information into the RGB-D image. We are just supposed to process the pixels of targets in the RGB-D image, denoted as RGBROI, since their localizations and labels have been recorded. Then we change the pixel values using (3):

\[
I_{i,j} = \begin{cases} 
I_{i,j} & D_{i,j} = 0 \\
I_{\text{label}} & D_{i,j} = 255 
\end{cases}
\]

(3)

Where \(I_{i,j}\) is the pixel value consisting of three channels in RGBROI; \(D_{i,j}\) is the pixel value in DepthROI; \(I_{\text{label}}\) represents the color assigned to the corresponding label. As a result, we can get a new RGB-D image with semantic information and avoid the disadvantages of direct image segmentation at the same time. The color assigned to the corresponding label. The result is shown in Fig. 9.

Finally, we utilize the semantic RGB-D image and its depth image to generate 3D point cloud where different classes of targets will be painted with different colors.

4. **Result.** To evaluate the capabilities of our SRM system, we run a Demo named Robot Mapping on the official website. We also use Kinect2(Xbox One) to test SRM in various scenes with different target objects and build corresponding 3D point clouds. Considering a global point cloud cannot clearly show the edges of the
target objects we labeled, we just show some typical local point clouds including semantic information in this paper, see Fig. 10, Fig. 11 and Fig. 12.

4.1. RTAB-MAP Demo-Robot Mapping.

4.2. Other scenes and target objections. According to the images above, we can conclude that point clouds including semantic information can be built with SRM by these steps: 1) implement object detection with YOLO on 2D Keyframes; 2) extract edges of targets in the depth image with Canny operator; 3) segment the target precisely with regional growing algorithm; 4) add semantic information into the point cloud.

4.3. Comparison of SRM with other similar ones in the community.
   • Comparison with the method proposed by Li et. al
Figure 10. The left is the original local point cloud of RTAB-MAP. The right is the corresponding point cloud of SRM. Two refrigerators are recognized by YOLOv2 and painted with green (we assume that green represents refrigerators).

Figure 11. (First Column) Original RGB image. (Second Column) Edges of targets extracted in depth image by Canny operator. (Third Column) Corresponding local point clouds. The blue sticks in the point clouds in the first three rows are axes. Because we detected different targets separately, some different targets are painted the same color. When we detect them at the same time, we just need to assign different colors to different classes of objects.
Figure 12. The left is the original RGB image. The right is the corresponding point cloud. We paint the bottle red, the laptop blue and the handbag green. We don’t show Edges of targets extracted in depth image here because SRM process different objects one by one, which means we don’t have an image including all of their edges. The handbag is not shown completely in the point cloud because the Kinect2 didn’t get the depth data of that area.

They built semantic maps based on LSD-SLAM [8] which could only build semi-dense point cloud, so the maps would be less complete than those built by RTAB-MAP. In addition, some points of buildings relatively distant from the cars are mislabeled as cars, which may lead to big errors when robots localize objects or understand this scene when they use this map, see Fig. 13. We can see that buildings are relatively far to the cars, which may lead to big errors when robots do localization or other tasks when they use this map. This situation is mostly common in the methods that directly segment 2D images.

Because Li et.al did not mention where they get that image, we can only use our own images to make a comparison (Fig. 14).

We can see that red points are all within the boundary of handbag because of the restriction of the edges extracted by Canny operator. However, when the depths of target objects and surroundings are almost the same, SRM also make some small mistakes (Fig. 15).

We can see that the area in the green bounding box is mislabeled. However, this kind of mistakes is relatively trivial because inaccurate points will only be built on the objects very close to the target, which means relatively small errors when the robot uses this map.

5. Conclusion. In this paper, we propose a method named SRM to realize semantic SLAM by implementing YOLOv2 to detect objects in 2D images and utilizing corresponding depth images to add semantic information into the 3D point clouds. It can avoid big errors in the point cloud and gain higher running speed.

Limitations: SRM relied too much on the quality and accuracy of depth images so that the results may not be good if the depth image is in poor conditions. In addition, two fixed threshold values of Canny operator and the end condition of regional growing process cannot be guaranteed to suit all objects. In future work we will develop a more stable and precise SRM system that can select those parameters dynamically according to the characteristic of specific objects.

Acknowledgment. The work was supported by Shenzhen Peacock Plan KQTD2016112515134654 and Shenzhen Academy of Aerospace Technology, Shenzhen, China. Baochang Zhang is the corresponding author.
**Figure 13.** The performance of semi-dense 3D semantic mapping. [8] (Top) Original image. (Bottom) Corresponding point cloud. Red represents buildings. Purple represents cars. Green bounding boxes are added by us. They are not included in the original image.

**Figure 14.** The left is the original RGB image. The right is the point cloud.

**REFERENCES**

[1] R. Q. Charles, H. Su, K. Mo and L. J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2017).

[2] R. Girshick and J. Donahue, Trevor Darrell and Jitendra Malik, Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, (2013), 580-587.

[3] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2016), 770-778.

[4] M. Labbé and F. Michaud, Long-term online multi-session graph-based slam with memory management, Autonomous Robots, 3 (2017), 1-18.
Figure 15. The left is the original RGB image. The right is the point cloud.

[5] M. Labbe and F. Michaud, Online global loop closure detection for large-scale multi-session graph-based SLAM, IEEE/RSJ International Conference on Intelligent Robots and Systems, (2014), 2661–2666.
[6] M. Labbé and F. Michaud, Appearance-based loop closure detection for online large-scale and long-term operation, IEEE Transactions on Robotics, 29 (2013), 734–745.
[7] M. Labbe and F. Michaud, Memory management for real-time appearance-based loop closure detection, IEEE/RSJ International Conference on Intelligent Robots and Systems, (2011), 1271–1276.
[8] X. Li and R. Belaroussi, Semi-dense 3d semantic mapping from monocular slam, 2016.
[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed and C. Y. Fu, et al, SSD: Single Shot MultiBox Detector. European Conference on Computer Vision, Springer International Publishing, (2016), 21–37.
[10] J. McCormac, A. Handa, A. Davison and S. Leutenegger, Semanticfusion: dense 3d semantic mapping with convolutional neural networks, 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017.
[11] R. Mur-Artal and J. D. Tardós, Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM, Robotics: Science and Systems, (2015), 1–9.
[12] N. Otsu, A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man, and Cybernetics, 9 (1979), 62–66.
[13] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, You only look once: Unified, real-time object detection, Computer Vision and Pattern Recognition, (2016), 779–788.
[14] J. Redmon and A. Farhadi, YOLO9000: Better, faster, stronger, IEEE Conference on Computer Vision and Pattern Recognition, (2017), 6517–6525.
[15] J. Redmon and A. Farhadi, Yolov3: an incremental improvement, 2018.
[16] S. Ren, K. He, R. Girshick and J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Transactions on Pattern Analysis and Machine Intelligence, 39 (2017), 1137–1149.
[17] N. Sünderhauf, T. T. Pham, Y. Latif, M. Milford and I. Reid, Meaningful maps with object-oriented semantic mapping, Ieee/rsj International Conference on Intelligent Robots and Systems, IEEE, (2017), 5079–5085.
[18] T. Whelan, S. Leutenegger, R. S. Moreno, B. Glocker and A. Davison, ElasticFusion: Dense SLAM Without A Pose Graph. Robotics: Science and Systems, 2015.

E-mail address: maomingyuan@buaa.edu.cn
E-mail address: heweizhang@buaa.edu.cn
E-mail address: lisimeng@buaa.edu.cn
E-mail address: bczhang@buaa.edu.cn