A Shared Model Based Dense Real-Time Semantic SLAM Method Towards Repetitive Scene

Xinle Li, Wenbo Nie, Wei Zhang, Xiaobo Lin and Yao Yu

May 28, 2019

A Shared Model Based Dense Real-Time Semantic SLAM Method Towards Repetitive Scene

1st Xinle Li
School of Automation and Electrical Engineering, University of Science and Technology Beijing
Beijing, China
xinle_li@yeah.net

2nd Wenbo Nie
School of Automation and Electrical Engineering, University of Science and Technology Beijing
Beijing, China
969185900@qq.com

3rd Wei Zhang
School of Automation and Electrical Engineering, University of Science and Technology Beijing
Beijing, China
2033329616@qq.com

4th Xiaobo Lin
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, China
linxiaobo@xs.ustb.edu.cn

5th Yao Yu*
School of Automation and Electrical Engineering
University of Science and Technology Beijing
Beijing, China
yuyao@ustb.edu.cn

***Abstract*—Dense Simultaneous localization and mapping has attracted people's attention in recent years. However,it always consists a large map which led to an increase in storage space and generates incomplete map. In this paper, we designed a semantic SLAM system which reduce map storage space while improving integrity. The key idea is to segment objects from the background to individual models using deep neural network and reconstruct the models of same class with a common map storage space. We built a complete dense semantic system and propose a method to match two same objects in large distance.**

***Index Terms*—SLAM, semantic segmentation, model share, dense map**

I. Introduction

With the maturity of automation technology, demand for robotic autonomous navigation increases rapidly. In the past, control commands to robots from human were often directly controlling of the underlying actuators. Nowadays, with the development of artificial intelligence technology, computer have a better comprehension of the environment. Meanwhile, robots also meet the needs to accept complex and unclear instructions. For example, people put an order on the robot, ”give me the cup”. After receiving the command, the robot needs to locate the location of the ’water cup’ and the ’sender’ to distinguish it from the background environment, so they can plan a path from the robot to the cup and then from the cup to the sender.

There are several ways to locate the robot by themselves [1–3], and those methods have achieved good results in practice. Besides localization, in order to accomplish this task, the robots also need to fully understand the environment where they work at. And it calls for two essential conditions: firstly, building a complete map reconstruction of the surrounding environment, and secondly, semantic segment the object from background in the three-dimensional map.

To accomplish this task, we are committed to developing a system that segment objects from environment and reconstruct them into individual models, and we fuse the models which are the same class into a common shared model to reduce the storage space while improve the integrity of them.

The contributions of this paper are summered as follows:

- We build a system that process the dense SLAM and the semantic segmentation in real time.
- We make an improvement on over segmentation method, which pick out the edge of objects much precisely.
- We propose a novel method to build object models, which reduce the storage space of the map and increase integrity of models.

We will expand this paper in the following order: SectionII will introduce some relative work of our system. SectionIII overview the whole paper and introduce the framework of our slam system, explaining the running process of each module. SectionIV will introduce how to get the segmentation results we want from the output of the deep neural network and the over-segmented image. SectionV will briefly describe how to estimate the current camera pose using the existing models and the new frame. And sectionVI will illustrate how to incorporate the current frame into the model using the estimated camera pose. Finally, sectionVII will show the experiment result that we get from our system.

II. Relative work

Since the concept of SLAM was proposed, it has gradually become a popular hot topic, and many excellent researchers keep emerging. There are some efficient slam systems proposed recently, which target on building 3d maps. Each of them achieved good results in specific field. We will highlight some of the works that our sysetem is related to next.

A. Dense SLAM

Only when the environment map is fully modeled will it be possible for the machine to make its own path planning. So the dense SLAM method is essential for robot autonomous navigation. KinectFusion [4] predicted the current position and posture of the camera based on the point cloud of the current frame and the previous frame. Then, the TSDF value is updated according to the camera position and attitude, and the point cloud is fused. Finally, the surface is estimated according to TSDF value. Kintinuous [5] is an improvement of KinectFusion. The pose estimation is realized by ICP and direct method using GPU. With loop closure detection, the deformation graph is used for the first time to perform non-rigid transformation of 3D rigid body reconstruction, so that the results of the two reconstructions in the loop closure can coincide.

However, these method calls for a huge storage space, it's not practical if you want to deploy it for long-term use. ElasticFusion [6] use surfel model to construct the map, which require less storage space than others. With the form of deformation graph, the accuracy of reconstruction and pose estimation can be improved by constantly optimizing the way of rebuilding the map.

B. *Semantic Segmentation*

In recent years, deep learning is developing rapidly.The algorithm of image semantic segmentation is updated by time, and the result gets more and more precise. Faster-RCNN [7] use RPN to generate proposal Windows for each picture. Map the proposal Windows to the convolution feature map at the last layer of CNN. And use the softmax loss and smooth L1 Loss for classification and Bounding box regression joint training. Mask R-CNN [8] is an improved version of Faster-RCNN, which contain a two-stage framework. In the first stage, the image is scanned and proposals (areas that may contain a target) are generated. In the second stage, proposals are classified and boundary boxes and masks are generated.

C. *Semantic SLAM*

Semantic SLAM is a hot topic in current research, various methods are also emerging. VSO [9] take use of semantic re-projection error to derive a new cost function and minimize it using the expectation maximization (EM) scheme. Their work can maintain matching consistency under long-distance large-scale movements. But the framework they proposed cannot work independently and can only be embedded on other SLAM systems. To make full use of the semantic information of input frame, individual models of objects has become a new interest in research. Li, Peiliang etc.[10] treat different objects as different maps, and track them while reconstructing them. In terms of semantic fusion, they constructed four optimization terms and achieved satisfactory results. But the tracking method used is still the traditional Local Feature, which can not construct dense map.

Meanwhile, some researchers are also starting to look at applications of semantic SLAM. Stenborg etc.[11] solve the problem on how to use a 3D map to locate when already have a good one. Its map contains three-dimensional point cloud and object classification of each point. With such a semantic level of constraint, it achieves a good positioning effect with more stable semantics than local features.The downside, however, is that it require a rich semantic information in observations.

III. System Overview

Our SLAM system build a dense map with multiple objects in real-time.To be more specific, our system combined the out put of semantic segment pipeline and dense slam pipeline to build a dense 3D map with multiple models which can be tracked independently.

A. *surfel*

We use surfel as the basic unit of map construction. Different from other map representations, surfel contain an area unit instead of volume. It can simply represent the surface features of an object, regardless of the internal composition of the object. Each surfel in the map is consisted of a position $P \in R^3$,a normal vector $n \in R^3$,a color vector $c \in N^3$,a weight $w \in R$, a radius $r \in R$, an initialization timestamp t_0 and a last updated timestamp t.

$$P_{surfel} = \{\overrightarrow{p}, \overrightarrow{n}, (r, g, b), w, r, t_0, t\} \tag{1}$$

Required for the parallel computing, we run our system mainly on the GPU. We use OpenGL as the storage, updating, fusion and render tool, as it has powerful capabilities to process spatial point, and store them without redundant information of empty area. And we use CUDA to be the main compute tool, as it is good at processing the surface texture.

B. *Framework*

Our system is designed to contain two threads, one for slam pipeline and the other for semantic segmentation pipeline. As the semantic segmentation step for a single frame takes much more time than slam pipeline, it requires a time delay between them to make sure the output label could be fully utilized. Inspired by [12], we design a system with a queue that contain Q newest frames, as the Fig. 1 shows.

The yellow square indicates an original frame with RGB-D information.and the orange squares indicate that beside the RGB-D images it also has additional semantic information. The SLAM pipeline expressed as green rectangle always process the frame at the head of the queue Q, while semantic segmentation pipeline expressed as orange rectangle always get the input image from the back of queue. In order to facilitate the update of the queue, the SLAM pipeline take control of the movement of the queue. Due to the different rate of processing frames. The frame that put color image to semantic segmentation pipeline has moved forward several steps in the queue when the correspond label image is produced. The distance of movement at the n-th segmentation output is recorded as D_n, and apperently, D_n differs from time to time.

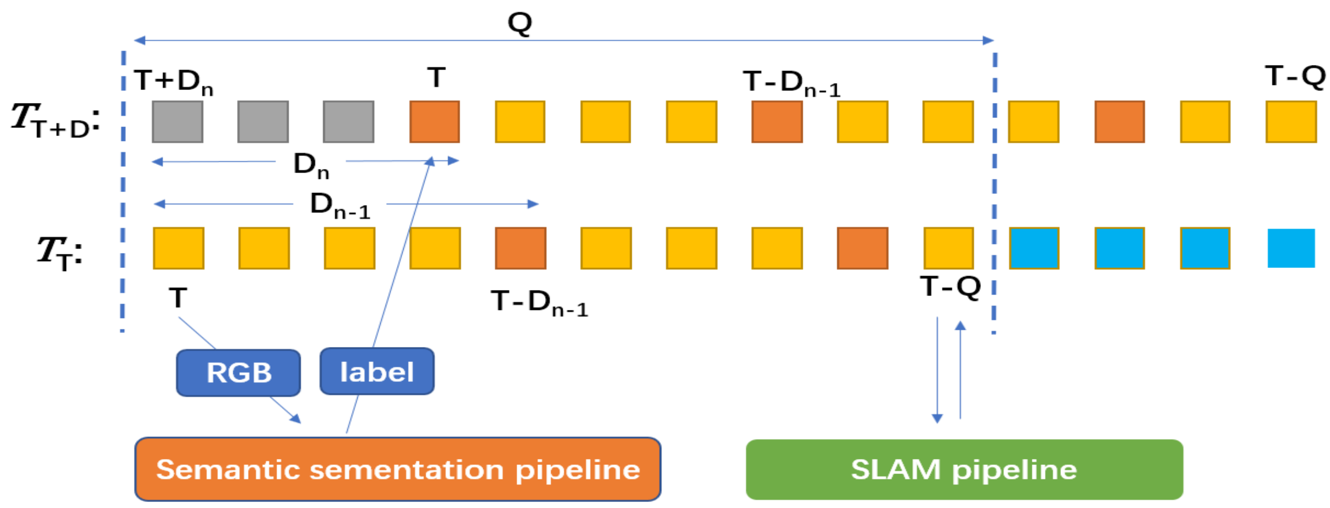

Fig. 1. Overview of running order of our system over frame queue

IV. Segmentation

We mean to build a 3D semantic model based map, in order to distinguish each other, every surfel in the map should be assigned with specific model id. To achieve this target, we need to segment each pix in each new frames before fusing it into global map.We adopt Mask-RCNN to perform the semantic segmentation task. Semantic segmentation methods like Mask-RCNN and deep-lab perform well in detecting objects, however, they share some common flaws. 1)Low process speed, SLAM require process new frames in real time, generally with high frame rate. However, the DNN pipeline which generate the mask of object usually perform slowly on most civilian computers. 2)The direct output of semantic segmentation DNN usually perform unsatisfactorily, missing object pixels or contain wrong pixels.

Inspired by Co-Fusion[13] and Mask-Fusion[12], we use two threads to process the DNN and SLAM pipeline simultaneously. After obtain the semantic segmentation of current frame, we combine it with the over segmentation image to get a preciser result. Fig. 2 is one of the frame in the dataset, which we will used to demonstrate our work.

A. Segmentation with DNN

In this section, we use the color image from the newest frame as the input of deep neural network.The DNN used here are replaceable, as long as it generates the outline of detected objects. In our paper, we adopt the state-of-art method Mask-RCNN which is pre-trained over PASCAL VOC2012 dataset. In order to eliminate the influence of other objects, we could only pick the labels of interest for output. For example, we pick out the label of the chair here, and the result is shown in Fig. 4.

B. Segmentation with raw information

For every input frame, we suppose to segment the image using RGB-D information. We use the local information to detect the edge of each object.Martin etc use deep and normal differ to segment area, which may lead to a miss segmentation if two objects close to each other. Different from their work, we combine color information with depth and normal ,which helps detect the edge of objects meanwhile keep pixs in one area if they only differ in colors (e.g. painting and texture on one object).

We set a fixed size adjacent area $\mathcal{N}$ for every pixel. We calculate the distance from each pixel in $\mathcal{N}$, and find the farthest one as ε_d.Similarly, calculate the max differ in normal ε_n and the max differ in color ε_c. Then, use the assigned weight of each w_d, $w_n$and w_c, to get the final difference value ε_f. If $\varepsilon_f > \mathcal{T}_{edge}$, we consider it as the edge of objects, where $\mathcal{T}_{edge}$ is a given threshold. The formulas are as follows:

$$\varepsilon_d = \max_{i\in\mathcal{N}}\{(v_i - v) * n\} \tag{2}$$

$$\varepsilon_n = \max_{i\in\mathcal{N}f}\{\begin{cases} 1 - n_i * n & if(v_i - v) * n > 0 \\ 0 & else \end{cases} \tag{3}$$

$$\varepsilon_c = \max_{i\in\mathcal{N}}\{\max_{c\in r,g,b}\{|c_i - c|\} - \min_{c\in r,g,b}\{|c_i - c|\}\} \tag{4}$$

$$\varepsilon_f = (w_d * \varepsilon_d + w_n * \varepsilon_n) * (w_c * \varepsilon_c + 1) \tag{5}$$

C. Fuse segmentation result

Due to the low rate of deep neural networks, it's not feasible to obtain segmentation result of the DNN every frame. While the incoming surfels increase per frame with the high rate of SLAM pipeline, it will lead to an incorrect classify of point if we don't assign the label information in time. The semantic information process is divided into two cases:

1) if current frame contain a DNN output, we combine it with the result of over segmentation in (IV-B). To be more specifically, we traverse every over segmentation area $\mathcal{S}_i$, and statistic the amount of point n_i and the point number of each

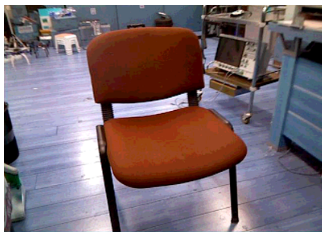
Fig. 2. Color image

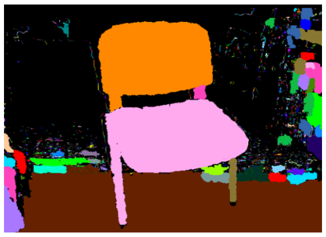
Fig. 3. Over segment image

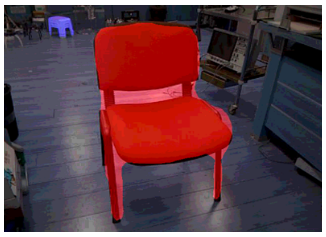
Fig. 4. DNN semantic image

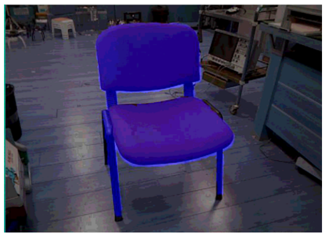
Fig. 5. Fused image

label n_i^c, and compare the proportion of them with a given threshold $\mathcal{T}_{seg}$ as (6) shows.

2) If current frame comes without DNN output, we re-project the model that already build in the map to the pixel coordinate. Then, perform a linear interpolation to fill the empty area surround by effective point. After that, we match it to the raw segmentation result same as above.

$$l_i = \begin{cases} c & \frac{n_i^c}{n_i} > \mathcal{T}_{seg} \\ 0 & else \end{cases} \tag{6}$$

After getting the label of area, we assign it to each pixel in the area. Meanwhile, comparing with the former state of the model label, we will know if there are new objects detected.

V. Tracking

In this section, we elaborated the method to track the camera position. Without keeping the history key frame, we re-project the dense 3D model to the last camera coordinate instead. When there comes a new frame, we use bilateral filter to get rid of the noise. We perform the direct model with the filtered image and the re-project texture.

The basic idea is to use the Lie algebra to represent transformation matrice between the current and previous frame, and then calculate the ICP error term (7) and photometric error term (8) between the two frames.

ICP error term: where $\mathbf{v}_t^k$ means the position of k-th surfel on the time t.

$$E_{icp} = \sum_k \left(\left(\mathbf{v}_t^k - \exp(\hat{\boldsymbol{\xi}}) \mathbf{T} \mathbf{v}_{t-1}^k \right) \cdot \mathbf{n}_t^k \right)^2 \tag{7}$$

Photometric error term: where $\mathcal{C}_t$ refer to the color texture of time t, and $\mathbf{P}_m$ refer to the surfel position in the model.

$$E_{rgb} = \sum_{\mathbf{u} \in \Omega} \left(\mathcal{C}_t (\mathbf{u}) - \mathcal{C}_{t-1} \left(\boldsymbol{\pi} \left(\mathbf{K} \exp(\hat{\boldsymbol{\xi}}) \mathbf{T} \mathbf{P}_m \right) \right) \right)^2 \tag{8}$$

Next we use the Gauss-Newton nonlinear least-squares method to iteratively minimize the joint cost function (9), and obtain the optimal Lie algebra.Finally, we use it to generate a transformation matrix, and then get the current camera position.

$$E_{track} = E_{icp} + w_{rgb} E_{rgb} \tag{9}$$

$$\mathbf{T}' = \exp(\hat{\xi}) \mathbf{T} \tag{10}$$

In the system, we use the fern method to detect the global loop closure, and then use the deformation map to update the entire Map. Since these tasks are not the main contribution of this article, we will not elaborate on it here.

VI. Maping

When a new surfel is observed, We initialize it in the background. When a new frame track the camera position successfully, we re-project every model to the camera coordinate, assign the semantic segmentation result to every visuable surfel and reclassify them to each model.

Every model own a model-id which differs form model to model and a class-id which is same with ones in common label. If there are more than one area that won common label in the segmentation image, we construct them into global map coordinate and caculate the minise distance between the surfel clouds to judge if they are the different part of a singal object or just two individual objects.

We divide the model into two types, one is the origin model, which maintains an independent surfels collection and the pose of the model, and the other is what we called shared model, which maintains an independent pose but no independent surfels space, instead of what is a tag pointing to a specific origin model.

A. *build origin model*

When detecting a new object, We first allocate a model with an independent space for it. We initialize the model pose as identity matrix,in other worlds, set the current camera position as model coordinate orign. We caculate the transformation matrix and transform the global surfels that own the specific label into model coordinate .

B. *match shared model*

Tracersing the origin model list, if there are different origin models with same class id, we supose a petential shared model may exist. We perform a match task every $\mathcal{K}$ frame (we set $\mathcal{K} = 5$ in our experiment). Considering the different lighting conditions and shadow positions, the color texture of two objects may differ a lot. To make matter worse, the direction of color gradient we used to optimize results perhaps exactly the opposite if the light source is between two objects.

In our system, we porpose a nerval method to match two same objects with a big spatial difference.The main idea is to minimizing a combined geometric and mask error function:

$$E_m = \min_{\xi_m} \left(E_m^{icp} + \lambda E_m^{mask} \right) \tag{11}$$

C. mask error term

Inspired by [9], we use label mask to calculate the mask error term instead of photometric error term. We define the Error term as:

$$\begin{aligned} E_m^{mask} &= \sum_{c \in \mathcal{C}} w_i^{(c)} \log\left(p\left(S_k | T_k, X_i, Z_i = c\right)\right) \\ &= -\sum_{c \in C} w_i^{(c)} \cdot \frac{1}{\sigma^2} DT_k^{(c)} \left(\pi\left(T_k, X_i\right)\right)^2 \end{aligned} \tag{12}$$

Where $p\left(S_k | T_k, X_i, Z_i = c\right)$ means the observation likelihood, with the known camera position T_k and the surfel location X_i. $Z_i \in \mathcal{C}$ is the label of the i-th pixel. And we define it as follows:

$$p\left(S_k | T_k, X_i, Z_i = c\right) \propto e^{-\frac{1}{2\sigma^2} DT_k^{(c)} \left(\pi\left(T_k, X_i\right)\right)^2} \tag{13}$$

$DT_k^{(c)}(p)$ means the distance to the nearest pixel which own the label c from the pixel location p at k-th frame. And $\pi\left(T_k, X_i\right)$ is the projection of the point X_i with the camera position T_k.

D. geometric error term

Similar with (7), the model geometric error term are expressed as (14).

$$E_{icp}^m = \sum_k \in m \left(\left(\mathbf{v}^k - \exp(\hat{\boldsymbol{\xi}}) \mathbf{T} \mathbf{v}_t^k\right) \cdot \mathbf{n}^k\right)^2 \tag{14}$$

E. Transformation matrix initialization

The premise of this optimization is that there are enough correct matching points. However, similar to large-scale movement, the direction and position between model to model often differ a lot. Without a good initial transformation matrix, the matching points may be incorrect, thus diverging the optimization results.Fortunately, due to the independence of the two models, we can make a simple estimation of the transformation matrix.

Firstly, we calculate the centroid position of the each model P^m and the average of the normal vectors N^m:

$$P^m = \frac{\sum_i P_i^m}{n_m} \tag{15}$$

$$N^m = \frac{\sum_i N_i^m}{n_m} \tag{16}$$

Next, we figure out the rotation Angle and the axis of rotation between the normal vectors a and b:

$$\theta = \arccos\left(\frac{N^a \cdot N^b}{|N^a||N^b|}\right) \tag{17}$$

$$\omega = N^a \times N^b = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{18}$$

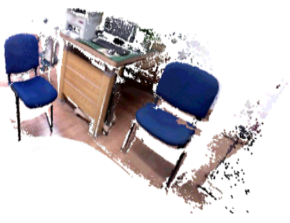
Fig. 6. Total sence

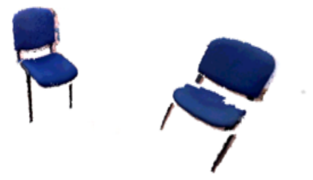
Fig. 7. Chair models

According to Rodrigues' rotation formula, we derive the rotation matrix as:

$$\mathbf{R}_t = \mathbf{I} + \widehat{\omega} \sin\theta + \widehat{\omega}^2 (1 - \cos\theta) \in \mathbb{SO}_3 \tag{19}$$

where:

$$\widehat{\omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{20}$$

the translation vector and transformation matrix can then be derived:

$$t_t = P^a - \mathbf{R}_t P^b \tag{21}$$

$$\mathbf{T}_t = \begin{bmatrix} & \mathbf{R}_t & & t_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{SE}_3 \tag{22}$$

F. model fusion

Although we do not use color texture while matching, they are supposed to combine the color information when fused into a common shared model. We develop a strategy that if there are surfels both origin models in one pixel, firstly we choose the lighter one to color the shared model. After that, we use Gaussian filter to make the color texture smoother.

VII. Experiment

We carry the experiment on a PC that have an Intel Xeon E5-2630 v3 CPU at 2.4 GHz and a Nvidia GeForce GTX 1080Ti GPU.

The application scenarios we envision need to contain numerous repetitive individuals.Since the existing public dataset does not meet our needs, we collected and built several datasets ourselves to test the performance of our system. We select the Microsoft's Kinect V2 and the ASUS Xtion pro live as image sensors, and collect images in the lab at around 30Hz. Separately, we collect a dataset for a short time(about 20 second) and another dataset for a longer period of time (about 3 minute). And for each dataset, we make sure that two chairs and a teddy bear appear in the shooting range.

TABLE I
The Dataset

	Kinect V2	**Xtion Pro Live**
short time	dataset1	dataset2
long time	dataset3	dataset4

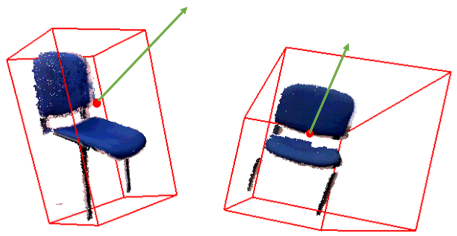

Fig. 8. The barycenter, mean normal and boundbox of models

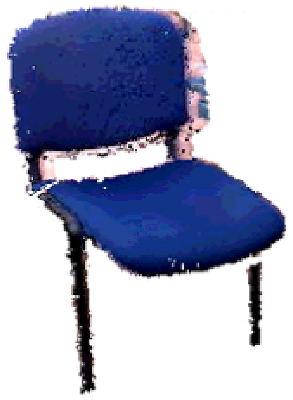

Fig. 9. The shared model

The result shown in Fig. 6-9 is run on a sence that contain two individual chair with common outlook. Fig. 6 is the final image of all models and the Fig. 7 is image of models labeled as 'Chair'. Fig. 8 visualized the barycenter and mean normal that we used to calculate the transformation matrix of two models. And Fig. 9 shows the shared model that fused with two origin models.

VIII. CONCLUSIONS

This paper introduced a dense SLAM system that can semantically segment the constructed map into object models, meanwhile, fuse the objects of the same kind into a common share model. We derive the segmentation, tracking and Mapping method and perform an experiment on our own datasets.

We get a noticeable improvement in the Repetitive Scene compared to MaskFuion which is the state-of-Art dense semantic SLAM. Further, we will try to build the independent surfels model on the basis of the shared model to distinguish different objects while keeping most of surfels of the origin models shared.

ACKNOWLEDGMENT

This work has been funded from National Natural Science Foundation of Chian, No.61520106009, No.61533008 and No.61703037.

REFERENCES

[1] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[2] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[3] X. Xie, Y. Yu, X. Lin, and C. Sun, "An ekf slam algorithm for mobile robot with sensor bias estimation," in *2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, 2017, pp. 281–285.

[4] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking." in *ISMAR*, vol. 11, no. 2011, 2011, pp. 127–136.

[5] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald, "Kintinuous: Spatially extended kinectfusion," 2012.

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

[7] 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*, 2015, pp. 91–99.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[9] K.-N. Lianos, J. L. Schonberger, M. Pollefeys, and T. Sattler, "Vso: Visual semantic odometry," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 234–250.

[10] P. Li, T. Qin *et al.*, "Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 646–661.

[11] E. Stenborg, C. Toft, and L. Hammarstrand, "Long-term visual localization using semantically segmented images," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6484–6490.

[12] M. Runz, M. Buffier, and L. Agapito, "Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects," in *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2018, pp. 10–20.

[13] M. Rünz and L. Agapito, "Co-fusion: Real-time segmentation, tracking and fusion of multiple objects," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4471–4478.