Neurocomputing 345 (2019) 3–14

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

ObjectFusion: An object detection and segmentation framework with RGB-D SLAM and convolutional neural networks

Guanzhong Tian[a], Liang Liu[a], JongHyok Ri[a], Yong Liu[b,*], Yiran Sun[a]

[a] State Key Laboratory of Industrial Control Technology, Institute of Cyber-systems and Control, Zhejiang University, Hangzhou 310027, China
[b] Institute of Information Technology, Kim Il Song University, Pyongyang 190016, Republic of Korea

ARTICLE INFO

Article history:
Received 4 July 2018
Revised 10 December 2018
Accepted 3 January 2019
Available online 3 February 2019

Keywords:
Intelligent Sensing
CNNs
SLAM
Object detection
Segmentation

ABSTRACT

Given the driving advances on CNNs (Convolutional Neural Networks) [1], deep neural networks being deployed for accurate detection and semantic reconstruction in SLAM (Simultaneous Localization and Mapping) has become a trend. However, as far as we know, almost all existing methods focus on design a specific CNN architecture for single task. In this paper, we propose a novel framework which employs a general object detection CNN to fuse with a SLAM system towards obtaining better performances on both detection and semantic segmentation in 3D space. Our approach first use CNN-based detection network to obtain the 2D object proposals which can be used to establish the local target map. We then use the results estimated from SLAM to update the dynamic global target map based on the local target map obtained by CNNs. Finally, we are able to obtain the detection result for the current frame by projecting the global target map into 2D space. On the other hand, we send the estimation results back to SLAM and update the semantic surfel model in SLAM system. Therefore, we can acquire the segmentation result by projecting the updated 3D surfel model into 2D. Our fusion scheme privileges in object detection and segmentation by integrating with SLAM system to preserve the spatial continuity and temporal consistency. Evaluation performances on four datasets demonstrate the effectiveness and robustness of our method.

1. Introduction

Understanding the content and the meaning of a perceived scene is a crucial capability for a robot to execute more intelligent behavior. To solve the core of this problem, the high accuracy of object recognition in 3D scene [1] and the inclusion of rich semantic information within a dense map is inevitable. As a specific example, suppose a user want his robot to “fetch the slippers from the shoe rack”. This simple fetching task requires knowledge of both what the target is and what’s the shape of it, as well as where it is located. However, even simple as it seems, this fetching task is still challenging for most robots since current SLAM systems are full of geometric information and lack of semantic information. In order to address this challenge, we fuse the high-level features from CNNs with geometric information from SLAM system in order to build a framework which enables a much greater range of functionality and a better understanding of the scene for robots.

Another challenge when adapting CNNs into robots is that although CNNs are widely used in the field of computer vision [2], most of the existing CNN-based object detection and semantic segmentation methods are processing of the single 2D frame [3,4] which lacks the spatial continuity and temporal consistency of objects. Different pixels of the same object should be continuous in space and the same object appears in the consecutive frames should be close in location. What we need to do is to adjust the CNNs to handle the consecutive frames and preserve the consistency of space and time. Integrating SLAM system into Convolutional Neural Networks provides a possible solution to preserve time-space consistency.

The SLAM system beginning as a technique to enable real-time robotic navigation has achieved great success in self-localization, scene reconstruction, and other robotic fields. However, most vision-only SLAM solutions [5–7] which employ simple sparse corner-like features as well as edges, planes, often perform poorly on object detection and reconstruction. For the next level of robot intelligence, maps need to extend beyond geometry they need to contain semantics [8]. The semantics information of objects can open a new perspective for the SLAM system.

In this paper, we propose a novel framework for object detection and segmentation task based on SLAM system. The foundation of our approach is built upon a general 2D object detection

* Corresponding author.
E-mail address: cckaffe@hotmail.com (Y. Liu).

https://doi.org/10.1016/j.neucom.2019.01.088
0925-2312/

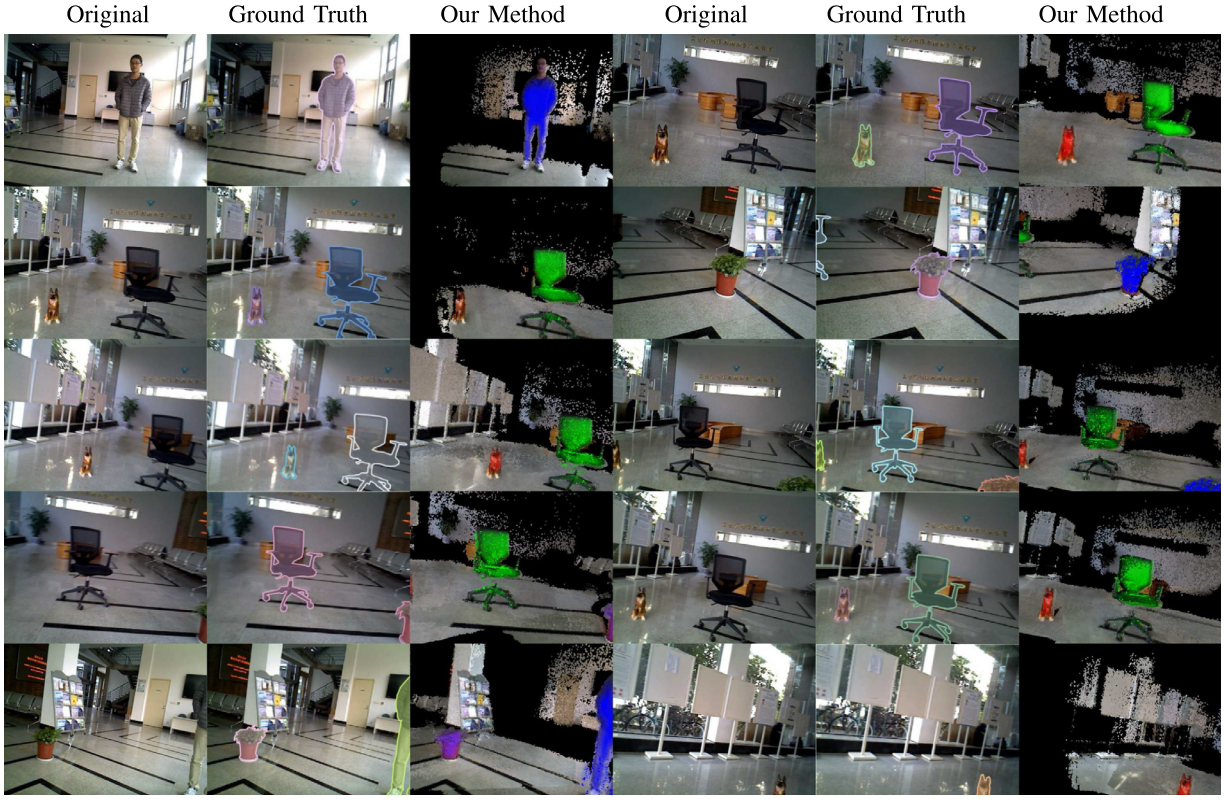

Fig. 1. A sketch of our performance on semantic segmentation task. The first and third rows are input images while the second and fourth rows are projection of 3D semantic surfel model of our method.

Convolutional Neural Networks and a visual SLAM system based on surfel model. All common detection neural networks which output bounding boxes as proposals such as YOLO [9], SSD [10], Fast RCNN [11], Faster RCNN [12] can be easily transplanted into our framework to be combined with SLAM systems. Objects detected on 2D images are represented as the proposals with its confidence scores. We use a general object detection framework for each new frame image to obtain the 2D object detection proposals. Then we use SLAM system to predict the estimation proposal for the next frame according to proposals from previous frame. This process will optimize inter-frame spatial continuity of the object by updating the confidence scores in each frame. The proposals for current frame and estimation results from SLAM are projected into a global 3D surfel model [13] and update the global target map by pose transformation matrix between new frame and historical frames, which is obtained by the ICP algorithm for the point clouds registration. Meanwhile, we project the global target map into 2D space and predict new object detection proposals. Therefore, proposals both from CNNs and SLAM are able to fuse together to achieve a better performance on detection task. On the other hand, after filtering the outliers, the fused proposals are used to update the 3D surfel model in SLAM system to label every surfel element semantically. When projecting the filtered surfel model into current 2D perspective, we can obtain the semantic segmentation results (Fig. 1).

In general, the contribution of this paper can be summarized as follows:

- By taking advantage of spatial continuity and temporal consistency of objects from SLAM system, we combine knowledge from single frame (extracted by Convolution Neural Networks) with knowledge between frames to achieve remarkable progress in accuracy.
- By combining with SLAM system, we use only one single neural network to accomplish both the detection and the segmentation.
- We build more grounded and appropriate datasets, evaluate our method on these datasets by comparing with the baseline, which indicates an impressive progress of our method.
- We experiment our proposed framework on benchmarks, compared with state-of-the-art methods on the task of detection and segmentation and prove the effectiveness and the robustness of our approach.

2. Related work

The field of fusing convolutional neural networks with SLAM system towards building 3D semantic map or obtaining better segmentation results is a hot topic and has yield quite a few works. Among these works, the most related ones are John McCormac et al. [14] and Niko Sünderhauf et al. [15]. John McCormac et al. [14] aims to build a dense, semantically annotated 3D map, they combined a specific segmentation neural network by Noh et. al. to provide semantical result and then fuse it with the SLAM system. Unlike them, our framework explore the inherent redundancy among object task and segmentation task, using bounding boxes as intermediate products to segment objects in surfel model and finally build a framework to finish the object detection and segmentation at the same time with one single neural network. Niko Sünderhauf et al. [15] combined SSD [10] with SLAM system towards obtain better segmentation results. Unlike us, They build the semantic map based on point cloud and employ depth image to segment objects to construct an adjacency graph.

Then we discuss related work with respect to the several relevant field that we incorporate within our method, i.e., SLAM, object detection and semantic segmentation.

SLAM. SLAM has always been a hot topic in the feild of robotics and there exists a mass of literature on it. Depending on the different type of input data being processed, SLAM can be divided into either monocular camera-based [16–18] or depth camera-based [6,19,20]. On the other hand, from the view of methodology, it can be classified as straightforward [16,18,21] or feature-based [17,22,23]. Most feature-based SLAM solutions are mainly based on simple sparse corner-like features such as edges, which tend to perform poorly on semantic level object localization and reconstruction. Under this circumstances, Salas-Moreno et al. [24] come up with a novel approach to create a semantic annotated SLAM system, SLAM++, which maps indoor scenes at semantically labeled objects level. However, SLAM++ is limited since it only map objects that exist in a pre-defined database. Additionally, the features it use to match template models are hand-craft.

As for the fusion of object recognition and SLAM, Sudeep Pillai et al. [25] from MIT propose to combine monocular SLAM with traditional object recognition approach for the first time. They reconstruct the scene in a semi-dense way based on existing monocular SLAM (ORB-SLAM) and import object proposals which are consistent across multiple views into BoVW(Bag-of-Visual-Words) to be classified by traditional well-trained object classifiers. Recently developed deep Convolutional Neural Networks methods [9,11,12] has significantly outperformed former traditional methods in terms of object recognition accuracy. Therefore, combining SLAM with deep neural networks seem to be a promising solution.

Object detection. The last decade has witness the success of HOG [26] and SIFT [27] due to their achievement in recognition accuracy. These methods are based on block-wise orientation histograms, model the shape of objects via oriented-edge templates, which seems to be less informative when comparing with multi-stage features extracted by CNNs. In CNN-based state-of-the-art object detection area, there exists two mainstream: models based on region proposals [11,12] and models trained end to end [9,10]. End-to-end models like YOLO resize all images into 448 $\times$ 448, utilize features from the whole image to predict bounding boxes and unify separate parts of object detection into one single network. The grid cell proposals is constrained in YOLO, which indicates that the processing time has dropped. But the negative effect is that the detection accuracy for targets, especially small targets, has also declined. Another state-of-the-art techniques in large-scale object detection task is Faster RCNN, which divides the detection process into two procedures: first use a region proposal network to take in input images while outputting rectangular object proposals and then the proposed regions are classified by the Faster RCNN detector network. By sharing computation and using neural networks to propose regions instead of Selective Search, Faster RCNN get a good balance between accuracy and time consumed. However, both models above focus on single frame instead of consecutive frames, thus get a ordinary performance when apply to detection task in consecutive frames. In this paper, we choose Faster RCNN as our baseline to be compared and it is also the fundamental neural network to be fused into SLAM system.

Semantic segmentation. Before the arise of deep learning, approaches such as N-cut [28] and GrabCut [29] based on graph partitioning take the lead in the area of semantic image segmentation. GrabCut proposed by Carsten Rother et.al uses separate Gussian mixture models to model the foreground and background of a image and seek for the optimal parameters for the models by iteration. But Grubcut need artificial bounding box or scribbled line as auxiliary information for the sake of achieving a good performance. When it comes to the era of DL (deep learning), Long et al. [3] first adopt the concept of fully convolutional network (FCN), adapt it on image semantic segmentation and bring us a excellent end-to-end solution. Another cutting-edge innovation is CRF-RNN [4], which brings in the Conditional Random Field to optimize the prediction result after feature extraction. One weakness for DL-based segmentation is that: Both FCN [3] and CRF-RNN [4] need a mass of images and semantic labeled pixels to trained the network. Taking that into consideration, our method costs much less by updating the semantic labels in the 3D map when fusing object proposals from the CNNs. Besides, we save quite a few computation resources and improve the efficiency by using one single neural networks instead of two, because that, as we all known, CNNs are data driven and need plenty of GPU resources to train.

3. Our approach

In this section, we demonstrate our proposed framework for robot on object detection and segmentation task, where the proposals predicted by CNNs are fused together with the estimation results from the SLAM system. The flow diagram in Fig. 2 sketches the pipeline of our framework. First of all, the CNNs take in frames to obtain the 2D object detection proposals to form the local target map. Then estimation results from SLAM and the local target map are fused to update the global target map. At the same time, the global target map is projected to 2D to acquire the detection result for current frame. On the other hand, modified RANSAC is adopted to remove the outliers and estimation results from SLAM is sent back to SLAM to update the surfels. In the end, the updated 3D surfel model is projected to 2D to obtain the segmentation result.

The whole framework are mainly connected by three fundamental units: a CNN-based object detection module, a surfel-based SLAM system and a Fusion-Update scheme. The role of CNN module in our framework is to process RGB frames from camera and output a set of candidate bounding boxes with its corresponding probabilities of the target objects. Separately, The SLAM system we used in the framework are ElasticFusion [5], which provides long-term dense correspondences between frames of RGB-D video. These correspondences allow the object proposals generated by CNNs from multiple views to be fused into a global consistent sufel-based map. The Fusion-Update scheme uses the correspondences provided by SLAM system to fuse the estimation results from SLAM with the proposals from CNNs and updates the parameters for each target object in both the global target map and the surfel model.

3.1. CNN architecture

We adopt the classic Faster RCNN by Shaoqing Ren et al. [12] as our baseline detection neural network in the framework and the CNN is implemented on caffe [30]. As is depicted in Fig. 4, the original frames are passed through a pre-trained CNN up until an intermediate layer, ending up with a convolutional feature map. We use VGG-16 as a feature extractor in this part. Next, using the features that the CNN computed, the Region Proposal Network (RPN) is adopted to find up to a predefined number of regions (bounding boxes), which may contain objects. Using the features extracted by the CNN and the bounding boxes with relevant objects, Region of Interest (RoI) Pooling is used to extract those features which would correspond to the relevant objects into a new tensor. Finally, comes the R-CNN module, which uses that information to classify the content in the bounding box and adjust the bounding box coordinates.

The Faster RCNN is pre-trained on Pascal VOC2012. For the training of RPN we adopt standard stochastic gradient descent, with a learning rate of 0.001, momentum of 0.9, and weight decay of 0.0005. The decay policy for learning rate is 'step', and the step size is 30000. As for the training of RCNN, the learning rate is 0.001, momentum is 0.9, and weight decay is 0.0005. The decay

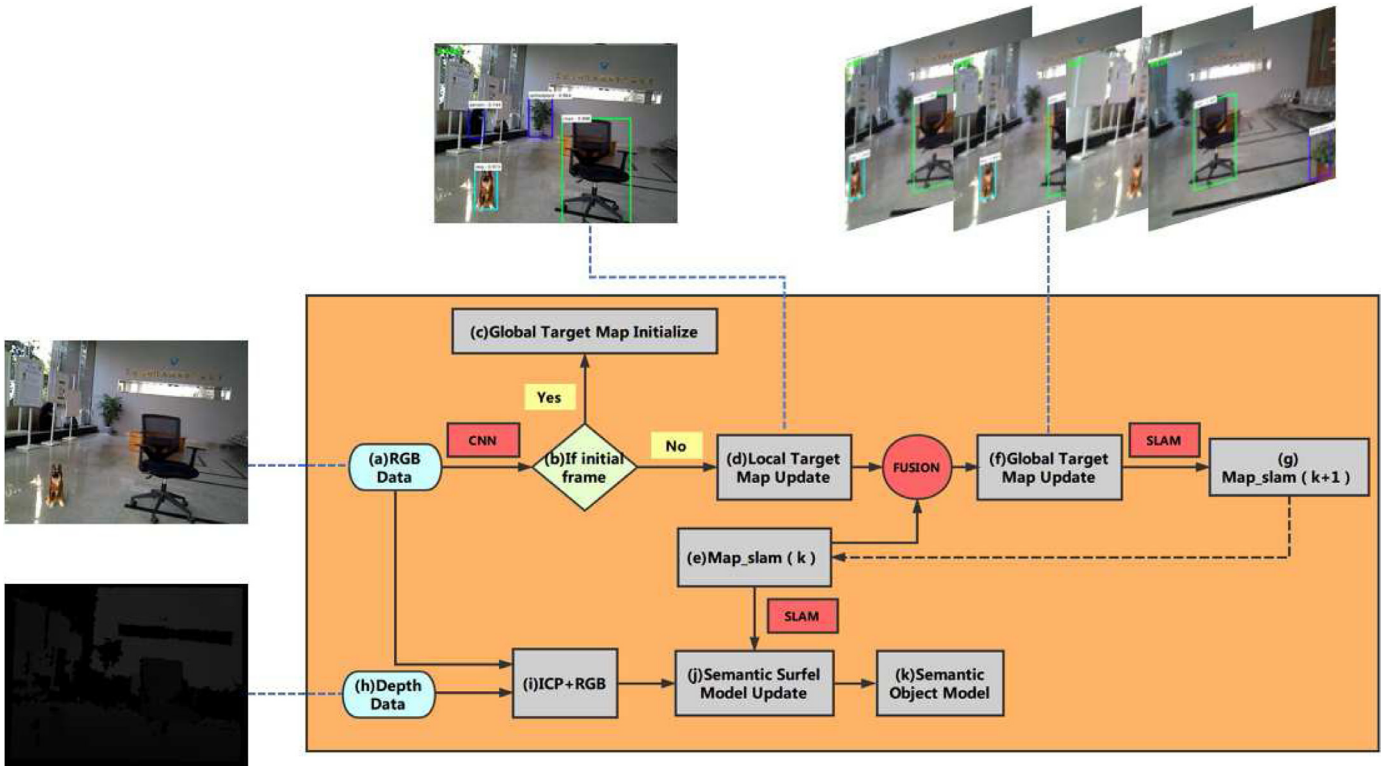

Fig. 2. Pipeline of our proposed 3D Object Detection and Segmentation Framework. (a) The CNNs takes in RGB frame as input (b) First, our algorithm judge if the input frame is initial frame. (c) If the result is yes, the Global Target Map (Map_{global}) is initialized $Map^0_{slam} = Map_{global} = boxes$. (d) If the answer is no, bounding boxes output from CNNs is used to update the Local Target Map. (e) Target Map from SLAM system (Map^k_{slam}) at the current moment. (f) Target Map from SLAM system (Map^k_{slam}) are fused with Local Target Map to update the Global Target Map. (g) Then we use the updated Global Target Map to predict Target Map from SLAM system in the next moment (Map^{k+1}_{slam}). (h) The SLAM system also takes depth data as input. (i) ICP+RGB is used for camera attitude estimation. (j) (Map^{k+1}_{slam}) is used to update the surfel model in the SLAM system. (k) The final semantic map is based on the updated surfel model.

policy for learning rate is 'exp'. We choose a batch size of 64, and train the network for a total of 20k iterations.

3.2. Surfel model

Surfel model is a method of rendering objects with rich shapes and textures at interactive frame rates and rendering presented in surfel model is based on using simple surface elements (surfels) as rendering primitives [31]. Surfels are point samples of a graphics model which is defined as a zero-dimensional n-tuple with shape and shade attributes that locally approximate an objects surface. We choose surfel model because it is quite appropriate for the modeling of dynamic geometric models which do not need to compute topological information.

Normally, surfel model is applied on the data representation of medical scanners, real-time particle system rendering and so on. In this work, each surfel in the model stores the following attributes: color information (R, G, B), coordinates in 3D space (x, y, z), label (l), normal vector ($\vec{n}$) and radius (r), initialization timestamp (t_0) and last updated timestamp (t):

$$P_{surfel} = \{x, y, z, (r, g, b), l, r, \vec{n}\} \tag{1}$$

The radius of each surfel represents the local surface area around a given point (which is the optical center of the camera in this work) while minimizing visible holes.

In the following dynamic modeling process, new surfels are added into the 3D surfel model. Therefore the color, label, location, radius and normal vector are updated in a weighting fusion way.

3.3. Global target map

We introduce the global target map in a scene in this subsection. In general, the global target map is the universal set of bounding boxes, which stores the information of each target at each position and one main goal of the fusion scheme is to get the updated global target map. As can be seen in the pipeline, after receiving RGB frames by CNN our framework need to judge if the input frame is the initial one. And different answers will result in different procedures. But both procedures involve the target map: If the answer is yes, our algorithm will initialize the global target map, otherwise we will use the proposals from the CNN to form the local target map. Acquiring and updating the global target map is a crucial stage of our framework.

The global target map, denoted as Map_{global}, is defined as the universal set for each object and its corresponding parameters in the whole scene. Correspondently, the local target map Map_{local} is the set of objects and its corresponding parameters for the current frame. Typically, when projecting the global target map into the current camera perspective, we can get the detection result. The output of the CNNs are bounding boxes, denoted as $bdbox_i$, and its confidence, denoted as $score_i$, with i represents the ith bounding box. We could use four parameters to locate bounding box: x_{left}, y_{top}, $width$, $height$. Then, we can get an example element (denoted as boxI) of the global target map:

$$boxI = \{x_{leftI}, y_{topI}, widthI, heightI, scoreI\} \tag{2}$$

As a result, the four endpoints of a bounding box can be described as follows:

$$(x_{left}, y_{top}) = (x_{left}, y_{top}) \tag{3}$$

$$(x_{left}, y_{bottom}) = (x_{left}, y_{top} + height) \tag{4}$$

$$(x_{right}, y_{top}) = (x_{left} + width, y_{top}) \tag{5}$$

$$(x_{right}, y_{bottom}) = (x_{left} + width, y_{top} + height) \tag{6}$$

As can be seen from Fig. 2, the update of global target map is a dynamic process which involves the local target map (Map_{local}) derived from CNNs and estimation results from SLAM system (Map_{slam}). Both of them are fused with each other continuously to update the global target map: Simply speaking, if proposals for a target from CNNs are consistent with estimation results from SLAM, the confidence score for the target will increase, vice versa. The whole update procedures for the global target map are described detailedly in Algorithm 1.

Algorithm 1 Procedures of updating the global target map.

Input: The target bounding boxes for each frame, denoted as *boxes*;
Output: The final global map after fusion and update; Map_{global};

if *Initial frames* **then**
Initialize the global target map: $Map^0_{slam} = Map_{global} = boxes$;
else
(1) Get Map_{local} by projecting Map_{global} into current perspective, preserve other targets beyond the scope of current perspective as: Map_{tmp};
(2) Remove the boxes under the threshold by function *MIX*: $Map_{local} = MIX(boxes, Map_{local})$;
(3) Put the updated local target map into the global target; map: $Map_{global} = MIX(Map_{tmp}, Map_{local})$;
(4) Fuse the current global target map with the predicted target map from SLAM: $Map_{global} = MIX(Map^k_{slam}, Map_{gloabl})$;
(5) Get the prediction for the next frame from current global target map: $Map^{k+1}_{slam} = predict(Map_{global})$;
end if

The Mix function in Algorithm 1 is used to fuse the results from both inputs, which are consist of two subsidiary functions: *reward* and *punish*. The definition of MIX is shown in Algorithm 2. Label A,

Algorithm 2 Function MIX.

Input: Map_A, Map_B
Output: Map_C

Suppose $Map_C = MIX(Map_A, Map_B)$
for all *box A* in Map_A **do**
  if IOU(box A, box B) > threshold and label A = label B **then**
    $Map_C = reward(box\,A, box\,B)$
  else
    $Map_C = punish(box\,A, box\,B)$
  end if
end for
return Map_C;

label B are labels of elements in Map_A, Map_B, respectively and *IOU* means Intersection over Union:

$$IOU(A, B) = (A \cap B)/(A \cup B) \tag{7}$$

Function *mix* and *punish* are defined as follows:

$$\left.\begin{aligned} &reward(box\,A, box\,B) = \left\{\frac{x_{leftA} + x_{leftB}}{2}, \frac{y_{topA} + y_{topB}}{2},\right. \\ &\frac{width_A + width_B}{2}, \frac{height_A + height_B}{2}, \\ &\left. frac(score_A + score_B)\theta_{conf}2\right\} \end{aligned}\right\} \tag{8}$$

$$\left.\begin{aligned} &punish(box\,A, box\,B) = decay(box\,A \text{ or } box\,B) \\ &= \{x_{leftA}, y_{topA}, width_A, height_A, \theta_{punA} score\} \\ &or\ \{x_{leftB}, y_{topB}, width_B, height_B, \theta_{punB} score\} \end{aligned}\right\} \tag{9}$$

θ_{conf} is a parameter to raise the scores when both sets find the same target. θ_{conf} is taken to be 1.06 by cross validation and the maximum for the score of a box is 1. Correspondingly, θ_{punA} and θ_{punB} are parameters to punish the scores due to the mismatching of two sets. Normally, the appropriate value for θ_{pun} is 0.94 by cross validation. However, when mixing Map_{tmp} with Map_{local}, the value is set as 1 since we need to combine the local target map with new targets to complete the global target map.

3.4. SLAM prediction map (Map_slam)

In our object detection and segmentation Fusion-Update scheme, there are two kinds of information need to be fused together: proposals from CNNs (in the form of local target map) and predictions from SLAM (in the form of Map_{slam}). During the fusion process, we update the global target map and obtain the Map^{k+1}_{slam} by SLAM. Then Map^{k+1}_{slam} is sent back to be fused with the next local target map, as is shown as dotted portion in Fig. 2. Since proposals from CNNs are regular bounding boxes with coordinates, we focus on how to get the predictions for the $(k+1)th$ frame from the *kth* frame in this subsection.

Suppose the current frame is k and the SLAM systems' prediction of the target map for the next frame is denoted as Map^{k+1}_{slam}, we can get the 3D coordinates for a target point of the *kth* frame in the world coordination by:

$$z_k = z_w \tag{10}$$

$$x_k = \frac{(u_k - c_x)z_k}{f_x} \tag{11}$$

$$y_k = \frac{(v_k - c_y)z_k}{f_y} \tag{12}$$

where u_k, v_k is the 2D coordinates of the target bounding box's central point in the image coordination. As the depth data obtained from depth image contains holes, to eliminate the instability by adopting data from one single point, we use the mean depth value of a small patch around the target bounding box's central point to replace the original depth data.

The coordinate relationship between the kth frame and the pth frame is defined as:

$$\begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} = inv(tanrsf_p/transf_0)(tanrsf_k/transf_0) \begin{bmatrix} x_k \\ y_k \\ z_k \\ 1 \end{bmatrix} \tag{13}$$

where $transf_0$ is the initial pose matrix while $transf_p$ and $transf_k$ is the corresponding pose matrix for the kth frame and the pth frame. Then we can project the *pth* frame's 3D coordinates into 2D:

$$u_p = \frac{f_x x_p}{z_p} + c_x \tag{14}$$

$$v_p = \frac{f_y y_p}{z_p} + c_y \tag{15}$$

By these transformations we can get the predicted bounding box of a target object in the pth frame from the kth frame, as can be shown in fig. 3.

3.5. 3D surfel model update and semantic label fusion

Previous segmentation architectures such as SemanticFusion [14] mostly use deep models that are tailored to solve one

Fig. 3. Prediction from *k*th frame to *p*th frame. Left: Bounding box of the target chair in the *k*th frame. Right: The predicted bounding box of the target chair in the *p*th frame.

specific task, i.e., semantic segmentation. They do not explore the inherent redundancy among different tasks, which can be formulated by geometry regularities via the nature of 3D scene understanding. Besides, large corpus of labels for pixel are needed to train a traditional deep segmentation neural network in order to preserve high performance with more general scenarios. Nevertheless, Our ObjectFusion framework can finish the segmentation task by processing the detection results from the object detection deep neural network in the framework and label the elements in surfel model [13] from SLAM system, which will eventually reconstruct a 3D semantic surfel model for the scene. The whole update procedures for surfel model is presented in Algorithm 3.

In consideration of the spatial continuity of objects in a frame, we filter the surfel model based on the spatial location. More specifically, for each object proposal, we use RANSAC for removing the outliers by sampling. For the clusters on the location of surfel model elements, we search and discard the largest plane, which is mostly likely to be the background. The left largest cluster in the remaining model of this object proposal is what the filtered object proposal. The filtered surfel model preserves the spatial coherence and improves the accuracy of segmentation by segment the bounding box of objects. The estimation error of the camera pose will accumulate over time. Therefore, we calculate the residual loss of the two viewpoints for the new frame and the history frames to determine whether or not to take the local loop closure detection. The local loop closure detection is based on deformation graph [32] which has been widely used in mesh deformation and non-rigid reconstruction. In addition, We encode raw RGB-D data and the pixel level semantic labels as 5 channels features into binary codes with Random Ferns method [33] for the global loop closure detection. The accuracy of the estimation of camera pose can benefit from the local loop closure detection and global semantic loop closure detection.

Before removing the outliers, we need to update the label and probabilities for each element in surfel model. Let's consider $P(O = l_i \mid I_k)$ denotes the probabilities for target object from Map_{slam}, and former parameters from sufel model, denoted as $P(l_i \mid I_{1,\ldots,k-1})$.

Algorithm 3 Procedures of updating surfel model.

Input: surfel model; Map_{slam}
Output: 3D semantic surfel map
1: **if** *Initial frames* **then**
2: Initialize the surfel model
3: **else**
4: (1) Project Map_{slam} into surfel model to update the parameters(label, probability) of each corresponding surfel elements in the model;
5: (2) Searching the maximal cluster in these surfel elements;
6: (3) Label the surfel elements in this maximal cluster as the target object;
7: (4) Project the surfel model in to 2D and we can get the semantic segmentation result.
8: **end if**

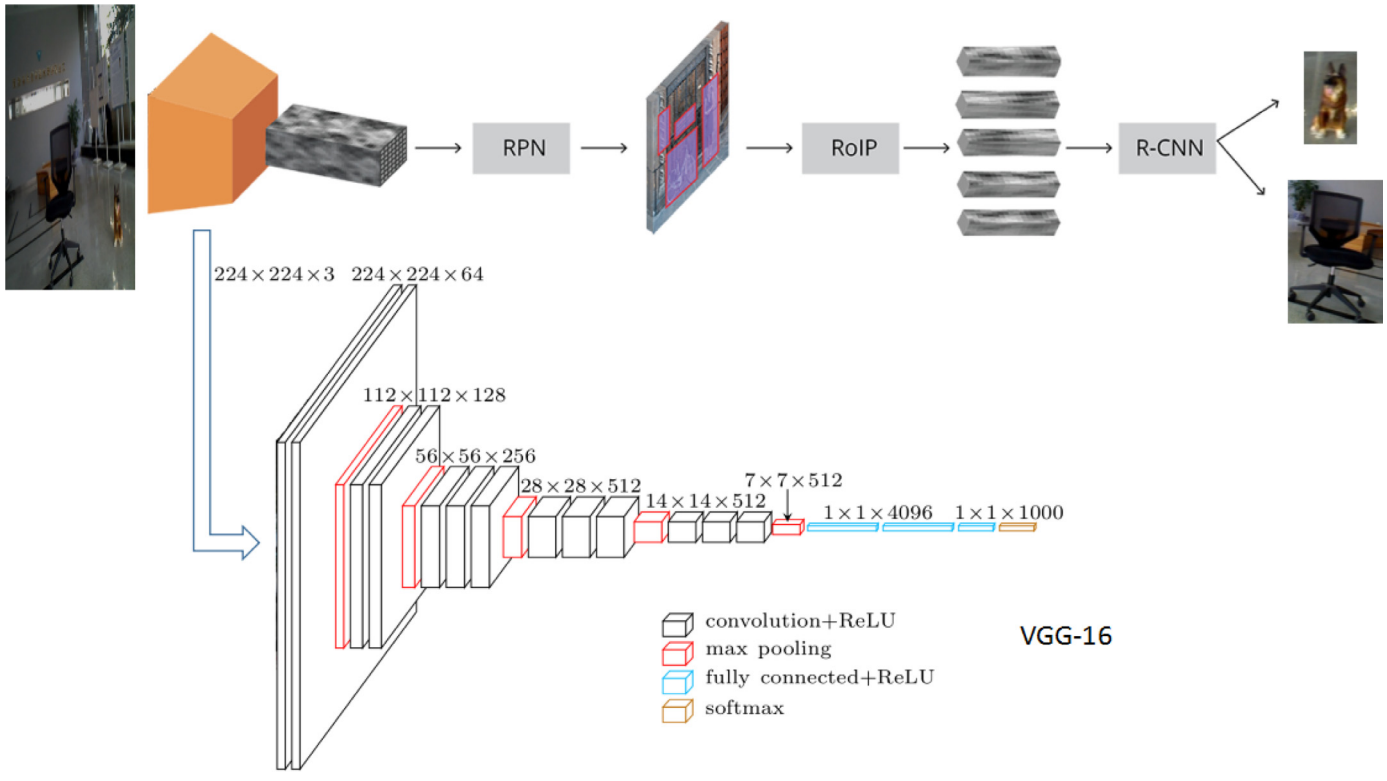

Fig. 4. Faster RCNN architecture: Object detection architecture of Shaoqing Ren et al. [12] used in our framework.

They are fused together to obtain the updated prediction results:

$$P(l_i \mid I_{1,\ldots,k}) = \frac{1}{X} P(l_i \mid I_{1,\ldots,k-1}) P(O = l_i \mid I_k) \tag{16}$$

which is applied to all label probabilities per surfel, finally normalizing with constant Z to yield a proper distribution According to the above equation, the label of surfel element is based on probability which updates every frame. The final label is the one with the maximum probability. If the new-coming frame has the same label with the history record, the probability increases and vice versa:

$$P\left(\hat{M}^{surfel}_{labeli}\right) = \lambda P\left(M^{surfel}_{labeli}\right) \tag{17}$$

$$\lambda = \lambda\left[1 - \left(2 \times \left|sgn\left(label^{p}_{old} - label^{p}_{new}\right)\right| - 1\right)\right] \times \theta_{\lambda}] \tag{18}$$

When the current label match the historical one, the output of function *sgn* is 0 and $\lambda = \lambda \times (1 + \theta_{\lambda})$ otherwise *sgn* would output 1 and $\lambda = \lambda \times (1 - \theta_{\lambda})$. θ_{λ} is a constant, normally is taken to be 0.96 by cross validation.

We take advantage of depth data, history record as well as Map_{slam} to update the surfel model's parameters (location, color, label) in SLAM. Since the SLAM prediction map (Map_{slam}) includes planes such as ground and other objects besides target object, we should remove these disturbed elements in the surfel model to make sure the label for each surfel element is accurate. We adopt RANSAC(Random Sample Consensus) [34] to remove the ground. Unlike common RANSAC, we tend to choose three point (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3) around the bounding box as the initial set instead of choosing randomly in the surfel database, which improves the efficiency of finding the background. These three points can establish a plane. By computing the distance between other points and this plane we can count the points near the plane. By update the initial set we can find the plane that have most points near it. That's the target plane which we called outliers to be removed.

After removing the outliers by RANSAC we still need to filter the surfel model in order to discard disturbed objects. Since each proposal from CNNs contain only one target object, we can accomplish the filtering process by finding the maximal cluster. The searching procedures are presented in Algorithm 4.

Algorithm 4 Searching the maximal cluster in surfel model.

1: Using Kd-tree to store Set P;
2: Initialize Set C and Set Q as Null Set;
3: **for** $p_i \in$ P **do**
4: Add p_i into Q;
5: **end for**
6: **for** $p_i \in$ Q **do**
7: Find the neighbourhood of Set P_i^k for p_i; if d(p_i, q_i) <threshold: $q_i \in P_i^k$; (d(p_i, q_i) is the Euclidian Distance)
8: **end for**
9: **for all** q_i in P_i^k **do**,
10: if not processed, add into Q;
11: After every element in Q is processed, count the number of Q and C, replace C with the bigger one and reset Q as Null Set;
12: **end for**
13: After every element in P is processed, return C.

In the end, Our system reconstruct the 3D semantic map based on surfel model and depth camera in an incremental way. By projecting the 3D semantic map to 2D image, we can get the segmentation result.

4. Experiments

In this section, we carry out experimental evaluation to validate the contributions of our proposed approach in terms of detection and semantic segmentation, by means of both qualitative and quantitative comparison against the state-of-the-art methods on four datasets.

Our experiments are carried on desktop PC with an Intel Xeon E5-2620 v3 CPU at 2.4 GHz with 16 GB of RAM and a Nvidia Tesla K40C GPU with 12 GB of VRAM. As for the implementation of the Convolutional Neural Networks, we choose Faster RCNN as our fundamental detection neural framework and the CNNs are implemented on caffe [30] and pre-trained on PASCAL VOC2012 [35] dataset.

4.1. Dataset

Considering most relative existing datasets such as NYUv2 [36] do not provide significant variations in viewpoints for a given scene and the correspondence between depth image and RGB image is not always consistent, we choose to build our own test datasets by collecting RGB-D images using the Xtion live pro depth camera, which aims for a relatively reconstruction of an indoor scene. The trajectory we choose to build the datasets are notably more loopy, both locally and globally. We believe the trajectory in our dataset is more representative of the scanning motion when inspecting a scene. Examples from the four datasets are shown in Fig. 5.

The first dataset (Dataset I) is a single target dataset, collecting in indoor scene with enough light. It consists 1801 successive frames. The second dataset (Dataset II) is multiple targets dataset, which contains multiple objects such as chairs, dogs, pot plants and so on. Dataset II is collected with changing light and composed of 1625 successive frames. Dataset III and IV are more challenging: Dataset III contains multiple objects with occlusion while dataset IV add moving obstacles. We also annotate each image in the four datasets manually.

4.2. Frame selection and convergence verification

There is no need to process every frame in actual practice, since the detection results between adjacent frames are analogous. In order to maintain a high processing efficiency, not all frames are processed by the framework, we use reasonable strategy to select target frames in a reasonable way: we design an experiment to evaluate the relationship between accuracy and frames skipped. We evaluate the performance of our CNN framework when performing a prediction on every 2^n frames, n $\in$ 0,1,...,7. As is shown in Fig. 7, when n $\leq$ 2, the accuracy almost remain unchanged. By deploying Tesla K40, the average time for processing a image is 0.237s. Therefore, we can improve the efficiency by 4 $\times$ when choosing $n = 2$.

The SLAM system we used depends on surfel model to label the elements. Each label can not generate immediately since the existing surfel database need evidences from new surfels to update the informations such as location, label, probability and so on. However, the surfels will be updated to the global map only when the confidence score has surpassed the threshold. As a result, the first few frames are not able to label all of the surfels that belong to the object when first recognizing it. Therefore, we need to measure how many frames are necessary before an acceptable semantic surfel model map can be reconstructed. Fig. 8 is the process of reconstruct a chair in the indoor scene. We can figure out that when it comes to the 10th frame, the object's reconstruction map is on the edge of convergence. Therefore, the first ten frames are

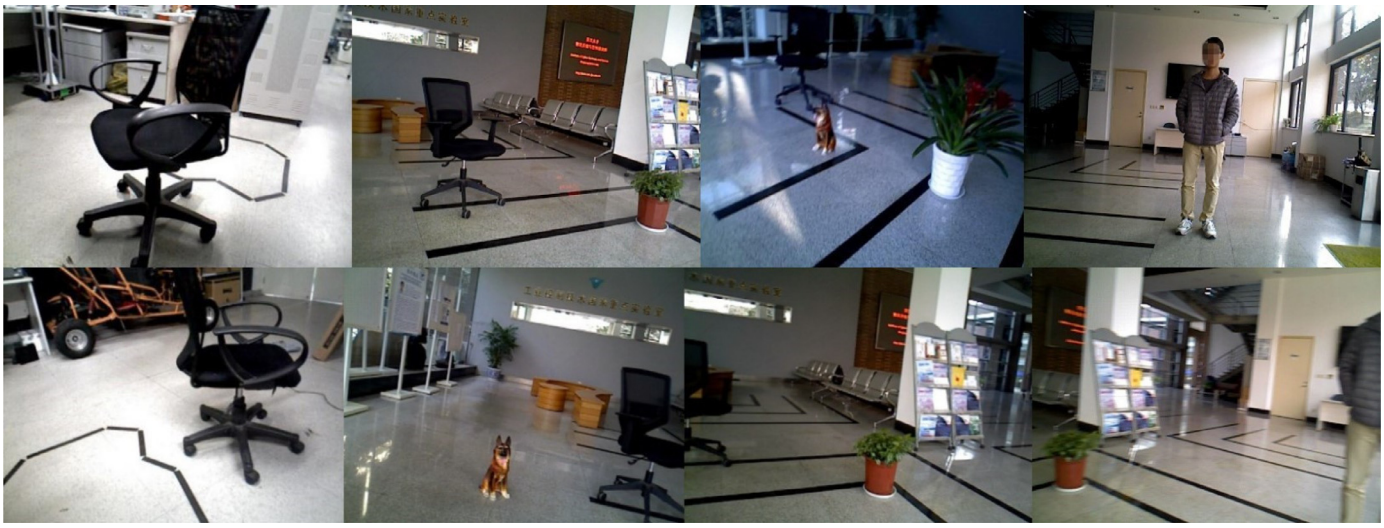

Fig. 5. Examples frbbom self-build datasets. From Left row to Right row: I, II, III, IV.

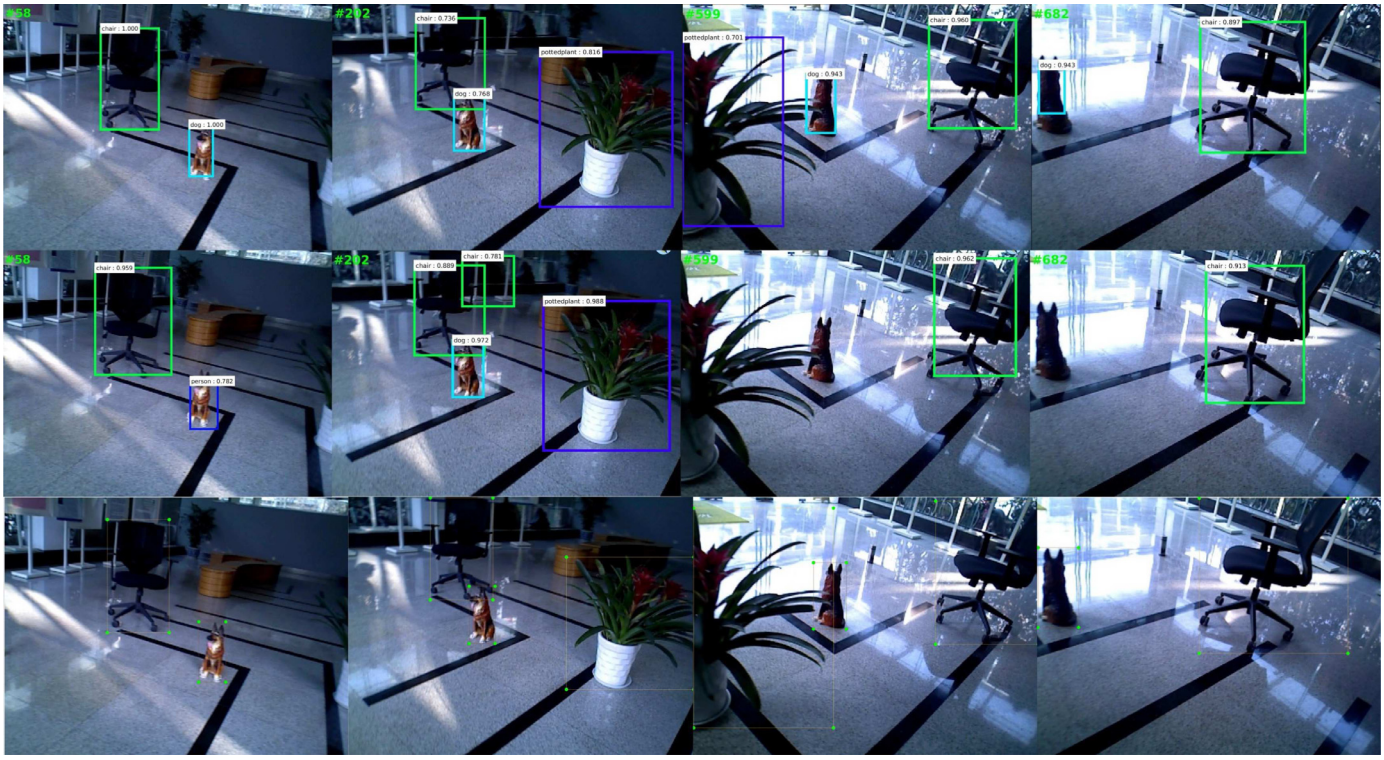

Fig. 6. Detection bounding boxes results in both methods. The first row is our method, the second row is Faster RCNN and the third row is Ground Truth.

removed in the following experiments to guarantee the comparability between our framework and the compared methods.

4.3. Object detection performance

Based on the same criterion with Faster RCNN [12], when *IOU*(Intersection over Union) $\geq$ 0.7, the test result is defined as hit the true value. Under this circumstances, we use the *precision, recall* and $F-measure$ as an evaluation criteria for different methods:

$$precision = \frac{n_{tp}}{n_{tp} + n_{fp}} \tag{19}$$

$$recall = \frac{n_{tp}}{n_{tp} + n_{fn}} \tag{20}$$

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \tag{21}$$

n_{tp} means the number of predictions that hit the true value, while n_{fp} means the number of predictions that hit the wrong value. n_{fn} represents the number of predictions fail to detect the target. In order to take into account the possible imbalance among object classes, we compute the *precision, recall* and $F-measure$ in a micro way. We evaluate the accuracy of our ObjectFusion pipeline against that of achieved by a single frame CNN framework. The quantitative results can be seen in Table 1.

From Table 1, we can draw a conclusion that our proposed method remarkably improve the precision at the cost of a slight decline of recall rate. The reason why the recall rate drop is that the fused detection system will remove some detected objects that only exist on a single view. Nevertheless, from the result of

Table 1
Comparison with faster RCNN on different data sets.

Datasets	Algorithm	Micro-Precision	Micro-Recall	Micro-F-measure
Dataset I (Single Object)	2D Faster RCNN baseline	0.6553	**0.9898**	0.7885
	Our proposed method	**0.8694**	0.9025	**0.8856**
Dataset II (Multiple Objects)	2D Faster RCNN baseline	0.7632	**0.8056**	0.7838
	Our proposed method	**0.9032**	0.7778	**0.8358**
Dataset III (Multiple Objects with occlusion)	2D Faster RCNN baseline	0.8032	**0.8492**	0.8256
	Our proposed method	**0.907**	0.8058	**0.8534**
Dataset IV (Multiple Objects with moving obstacles)	2D Faster RCNN baseline	0.7727	**0.7391**	0.7555
	Our proposed method	**0.9459**	0.7309	**0.8246**

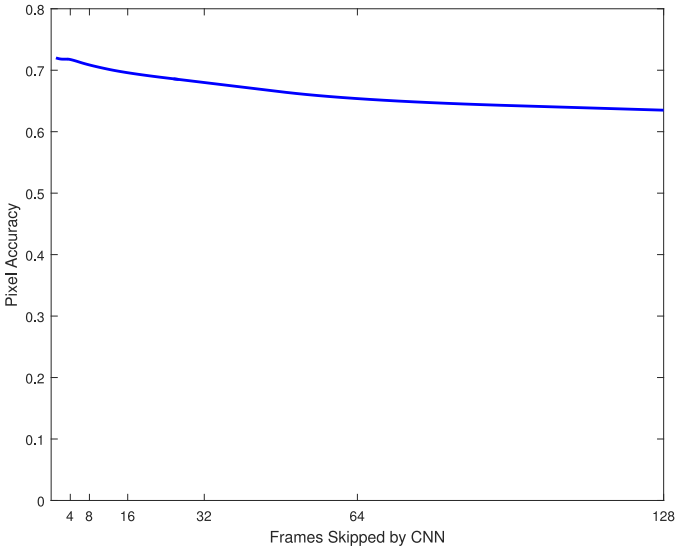

Fig. 7. The average class accuracy of our RGB-D based method on the single object dataset against the number of frames skipped between fusing semantic predictions.

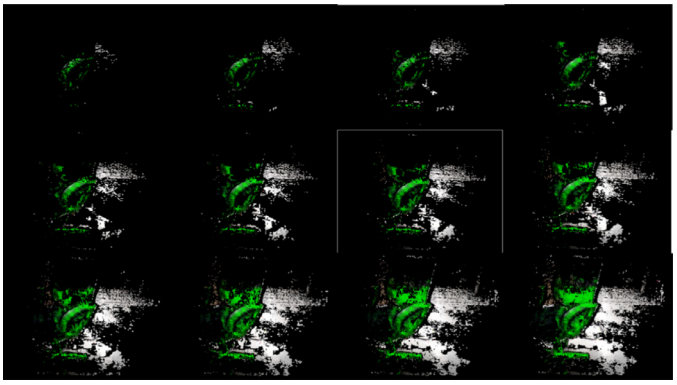

Fig. 8. Object reconstruction map convergence vertification test.

F-measure, a more comprehensive evaluation criterion that balance the *precision* and *recall*, we can draw a conclusion that our framework performs steadily better than the baseline (Faster RCNN).

The qualitative performance is shown in Fig. 6. It can be seen that even for object with part of which exists in the image, our method can successful recognize it because of predictions from the global target map. Besides, evidence from Fig. 6 shows that even for object that breaks into the image our system can still detect it smoothly.

4.4. Object semantic segmentation evaluation

We adopt the classic evaluation criteria in the semantic segmentation field to evaluate our framework : *pixel accuracy*, *precision*, *IOU*, *mean accurary*, *mean IOU* and *mean precision*. *IOU* in this section is based on the number of the pixels instead of bounding boxes in the previous section.

Table 2
Comparison with FCN,CRF-RNN, Mask RCNN, and Deeplabv3+ on Dataset I.

Category	Algorithm	IOU	Pixel Accuracy	Precision
Chair	FCN-voc8s	0.5211	0.6239	0.7923
	CRF-RNN	0.5925	0.6373	0.9386
	Mask RCNN	0.596	0.6451	0.8954
	Deeplabv3+	0.5845	0.6354	0.8562
	SORS(global)	**0.713**	**0.726**	**0.954**
	SORS(active)	0.7022	0.724	0.9366

$$pixel\ accuracy = \frac{\sum_i n_{ii}}{\sum_i t_i} \tag{22}$$

$$mean\ accuracy = \frac{1}{n_{cl}} \frac{\sum_i n_{ii}}{\sum_i t_i} \tag{23}$$

$$IOU = \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{24}$$

$$mean\ IOU = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{t_i + \sum_j n_{ji} - n_{ii}} \tag{25}$$

$$precision = \frac{\sum_i n_{ii}}{\sum_j n_{ji}} \tag{26}$$

$$mean\ precision = \frac{1}{n_{cl}} \frac{\sum_i n_{ii}}{\sum_j n_{ji}} \tag{27}$$

n_{ij} is the number of pixels which classified as j while the true value is i. n_{cl} is the total number of all classes. $t_i = \sum_j n_{ij}$ is the number of pixels that belongs to class i.

In this section, our approach, denoted as SORS(SLAM-based Object Recognition and Segmentation), is compared with two existing state-of-the-art 2D semantic segmentation methods: CRF-RNN [4] and FCN [3]. In the architecture of ElasticFusion [5], the system separate the scene into active area and inactive area according to the interval between current frame and other frames. We adopt the conception of active areas to update the surfel model. In the quantitative experiments, we perform our algorithm with two separate strategies: “global” and “active”. “Global” means updating the global surfel-based 3D map based on each frame and project the results to 2D planes while “active” only reconstruct the 3D map for current active areas and project the results to 2D images. Obviously, the “global” strategy concerns the whole scene while the “local” one focus more on adjacent areas. Comparisons on all four datasets can be seen in Tables 2–5.

Discussion Tables 2–5 show that the semantic segmentation become increasingly challenging as the scene tends to be more complicated. Nevertheless, we can see that our method prevails in all four datasets, especially in the results of pixel accuracy and IOU, which are more important because the precision only take samples that are recognized as positive into account.

Table 3
Comparison with FCN, CRF-RNN, Mask RCNN, and Deeplabv3+ on Dataset II.

Category	Algorithm	IOU	Pixel Accuracy	Precision
Chair	FCN-voc8s	0.4596	0.5893	0.4656
	CRF-RNN	0.4921	0.5065	0.5219
	Mask RCNN	0.5670	0.5971	0.7865
	Deeplabv3+	0.5235	0.5214	0.6924
	SORS(global)	**0.6396**	**0.7113**	0.794
	SORS(active)	0.5486	0.5789	**0.8561**
Dog	FCN-voc8s	**0.49**	**0.5025**	**0.8549**
	CRF-RNN	0.2827	0.2859	0.2706
	Mask RCNN	0.3213	0.3315	0.4535
	Deeplabv3+	0.3542	0.3653	0.5634
	SORS(global)	0.4727	0.4881	0.7577
	SORS(active)	0.4382	0.4513	0.8398
Potplant	FCN-voc8s	0.6012	0.6981	0.8104
	CRF-RNN	**0.656**	0.6781	0.8974
	Mask RCNN	0.5683	0.5778	0.8976
	Deeplabv3+	0.5771	0.5914	**0.9013**
	SORS(global)	0.6545	**0.7319**	0.8449
	SORS(active)	0.6035	0.6994	0.8918

Table 4
Comparison with FCN, CRF-RNN, Mask RCNN, and Deeplabv3+ on Dataset III.

Category	Algorithm	IOU	Pixel Accuracy	Precision
Chair	FCN-voc8s	0.4199	0.5299	0.3887
	CRF-RNN	0.4609	0.4788	0.3242
	Mask RCNN	0.416	0.4326	0.3079
	Deeplabv3+	0.4235	0.4654	0.3412
	SORS(global)	**0.5159**	**0.5534**	0.9015
	SORS(active)	0.4267	0.4465	**0.9347**
Dog	FCN-voc8s	**0.4306**	0.4301	0.8545
	CRF-RNN	0.3675	0.3715	0.2675
	Mask RCNN	0.3957	0.4012	0.2965
	Deeplabv3+	0.3658	0.3704	0.2567
	SORS(global)	0.4087	**0.4768**	0.8001
	SORS(active)	0.3635	0.3871	**0.8683**
Potplant	FCN-voc8s	0.6545	0.7854	0.7009
	CRF-RNN	0.612	0.7134	0.6989
	Mask RCNN	**0.6722**	0.7854	0.7422
	Deeplabv3+	0.6715	0.7753	0.7394
	SORS(global)	0.6669	**0.7904**	0.9303
	SORS(active)	0.6166	0.7499	**0.9385**

One detail we can see from Tables 2–5 is that our SORS make a big progress in the segmentation of chair while only improved within limits when it comes to dog. Apparently, chairs are more difficult to be recognized and segmented due to the wheels and discrete shape. But our method could handle these details quite well and get a better performance by combining with SLAM. In addition, our SORS algorithm is able to segment more detailedly such as the branches and leaves of pot plant, human arms and legs, etc. Qualitative comparisons are presented in Fig. 9.

Another phenomenon from Tables 2–5 is that sometimes the performance of "active" is better than "global", especially in Dataset IV. That's because the global map is designed to fuse with each frame. If our CNNs framework fail to detect target in several successive frames, the global map will lower the probability for certain label, which will cause the ineffectiveness of semantic segmentation. However, Average performance in Table 6 proves that the global SORS algorithm is still more robust and could be a more promising choice.

4.5. Run-time performance

To evaluate the run-time performance of our framework we perform experiments on a sample of random 20 sequences from the test set. Each sequence last for 5 s. The experiments were performed on an Intel Xeon(R) CPU E5-2620 v3 @ 2.40 GHz 16 CPU and an Nvidia Tesla K40C GPU. It requires 45.6 ms on average to process one frame and update the map in the SLAM system. Besides, it need an additional 1.5 ms on average to update the surfel model for each chosen frame. As is illustrated above, the other deployments in the framework do not need to be applied for each frame. The average time for a forward pass of our CNN and the update of our global target map on one frame is 121.7 ms. And our standard scheme performs this every 4 frames.

Table 5
Comparison with FCN, CRF-RNN, Mask RCNN, and Deeplabv3+ on Dataset IV.

Category	Algorithm	IOU	Pixel Accuracy	Precision
Chair	FCN-voc8s	0.3994	0.452	0.8317
	CRF-RNN	0.327	0.3452	0.6331
	Mask RCNN	0.3557	0.3817	0.7112
	Deeplabv3+	0.3326	0.3774	0.6875
	SORS(global)	0.4591	0.4727	0.7597
	SORS(active)	**0.469**	**0.4776**	**0.835**
Dog	FCN-voc8s	0.2845	0.3048	**0.8236**
	CRF-RNN	0.154	0.165	0.6147
	Mask RCNN	0.2301	0.2564	0.7074
	Deeplabv3+	0.2236	0.2602	0.7012
	SORS(global)	**0.2993**	**0.3064**	0.7855
	SORS(active)	0.2652	0.2739	0.7521
Potplant	FCN-voc8s	0.3747	0.4937	0.5441
	CRF-RNN	0.3381	0.3866	0.5387
	Mask RCNN	0.3167	0.4114	0.6002
	Deeplabv3+	0.3564	0.4611	0.7124
	SORS(global)	**0.4453**	**0.4991**	**0.7965**
	SORS(active)	0.3841	0.412	0.7758
Person	FCN-voc8s	0.8048	0.8782	0.7383
	CRF-RNN	0.8066	0.8593	0.3553
	Mask RCNN	0.8306	**0.8856**	0.8452
	Deeplabv3+	0.8295	**0.8774**	0.8395
	SORS(global)	**0.8338**	0.8848	0.8561
	SORS(active)	0.8042	0.8596	**0.8666**

Table 6
Mean performance on different datasets.

Category	Algorithm	Mean IOU	Mean Pixel Accuracy	Mean Precision
Dataset I	FCN-voc8s	0.5211	0.6239	0.7923
	CRF-RNN	0.5925	0.6373	0.9386
	Mask RCNN	0.5960	0.6451	0.8954
	Deeplabv3+	0.5845	0.6354	0.8562
	SORS(global)	**0.713**	**0.726**	**0.954**
	SORS(active)	0.7022	0.724	0.9366
Dataset II	FCN-voc8s	0.5169	0.5966	0.7103
	CRF-RNN	0.4769	0.4899	0.5633
	Mask RCNN	0.4855	0.5021	0.7125
	Deeplabv3+	0.4849	0.4927	0.7190
	SORS(global)	**0.5889**	**0.6438**	0.7989
	SORS(active)	0.5301	0.5765	**0.8626**
Dataset III	FCN-voc8s	0.5775	0.6559	0.6708
	CRF-RNN	0.5618	0.6058	0.4115
	Mask RCNN	0.4946	0.5397	0.4489
	Deeplabv3+	0.4869	0.5370	0.4458
	SORS(global)	**0.6063**	**0.6764**	0.872
	SORS(active)	0.5528	0.6106	**0.902**
Dataset VI	FCN-voc8s	0.3529	0.4168	0.7361
	CRF-RNN	0.273	0.2989	0.5955
	Mask RCNN	0.3433	0.3938	0.716
	Deeplabv3+	0.3484	0.3952	0.7351
	SORS(global)	**0.4012**	**0.4261**	0.7806
	SORS(active)	0.3728	0.3878	**0.7873**

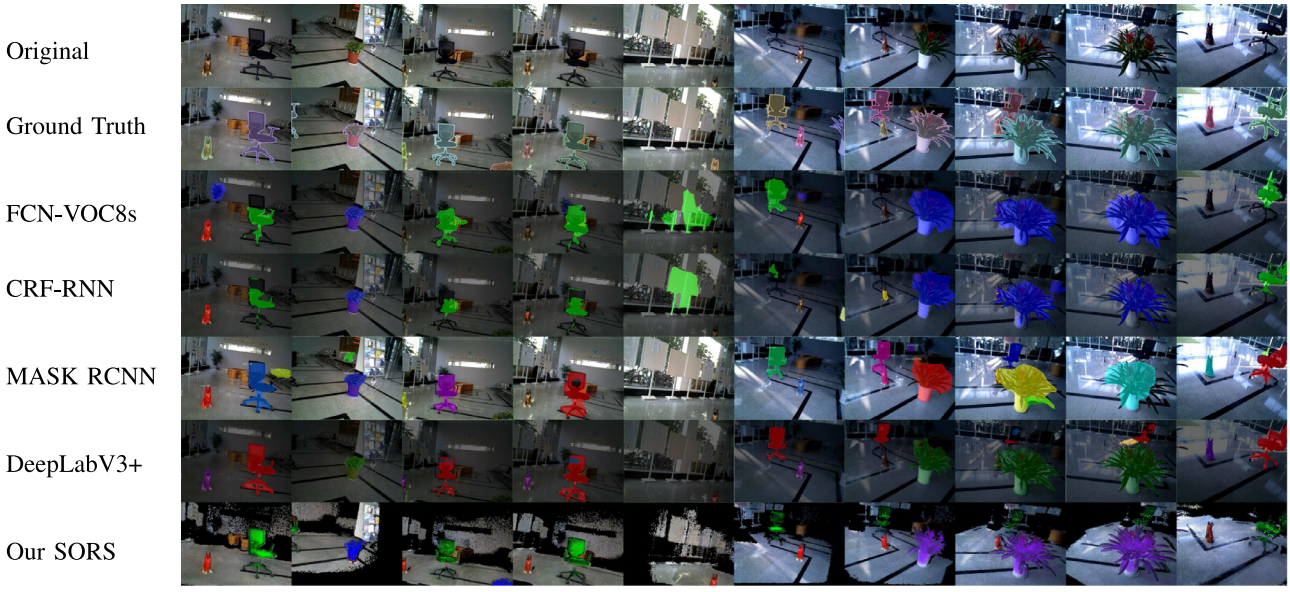

Fig. 9. Comparation with FCN, CRF-RNN, Mask RCNN and DeepLabv3+ on segmentation task.

5. Conclusion

We have shown that the integration of SLAM with 2D proposals via a deep neural network is a effective direction to solve the problems of object detection and semantic segmentation. We perform experiments on single object, multiple objects, occlusion, illumination changes, exist moving obstacle environments with the Asus Xtion Pro RGB-D sensor and show that our model obviously improves the robustness for object detection problem in the rich variety of environments. For the semantic segmentation task, our method is compared to previous 2D methods such as FCN-8s [3], CRF-RNN [4], Mask RCNN [37] and DeepLabv3+ [38] with all of the models trained on Pascal VOC 2012 dataset [39]. The results demonstrate that our model can obtain a high-quality object semantic segmentation performance. Evaluations reveal our framework is capable of incrementally reconstructing the semantic labeled scene as well as fusing proposals from CNNs, which is a new perspective towards scene understanding and semantic segmentation with RGB-D camera.

A future research schedule going further into exploit more efficient CNN architectures, combining high-level representations from CNNs and low-level representations such as edges, corners from SLAM to reconstruct a more accurate and detailed 3D semantic map.

Acknowledgment

The project is supported in part by the National Key R&D Program of China under Grant 2017YFB1302003 and in part by the National Natural Science Foundation of China under Grant U1509210.

References

[1] W. Luan, Y. Yang, C. Fermller, J.S. Baras, Fast task-specific target detection via graph based constraints representation and checking, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3984–3991, doi:10.1109/ICRA.2017.7989458.

[2] W. Liu, S. Li, D. Cao, S. Su, R. Ji, Detection based object labeling of 3D point cloud for indoor scenes, Neurocomputing 174 (2015).

[3] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431–3440.

[4] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, P.H. Torr, Conditional random fields as recurrent neural networks, in: Proceedings of the NIPS, 2015, pp. 1529–1537.

[5] T. Whelan, S. Leutenegger, R.F. Salas-Moreno, B. Glocker, A.J. Davison, Elastic-Fusion: dense SLAM without a pose graph, in: Proceedings of the Robotics: Science and Systems (RSS), Rome, Italy, 2015.

[6] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J.J. Leonard, J. Mcdonald, Real-time large-scale dense RGB-D slam with volumetric fusion, Int. J. Robot. Res. 34 (4–5) (2015) 598–626, doi:10.1177/0278364914551008.

[7] X. Zuo, X. Xie, Y. Liu, G. Huang, Robust visual slam with point and line features, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 1775–1782, doi:10.1109/IROS.2017.8205991.

[8] A. Kundu, Y. Li, F. Dellaert, F. Li, J. Rehg, Joint semantic segmentation and 3D reconstruction from monocular video, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Proceedings of the Computer Vision ECCV, Lecture Notes in Computer Science, Vol. 8694, Springer International Publishing, 2014, pp. 703–718, doi:10.1007/978-3-319-10599-4_45.

[9] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, SSD: single shot multibox detector, Springer International Publishing, Cham, pp. 21–37, 10.1007/978-3-319-46448-0_2.

[11] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015.

[12] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in: Proceedings of the NIPS, 2015, pp. 91–99.

[13] J. Stückler, S. Behnke, Multi-resolution surfel maps for efficient dense 3D modeling and tracking, J. Vis. Commun. Image Rep. 25 (1) (2014) 137–147.

[14] J. McCormac, A. Handa, A. Davison, S. Leutenegger, Semanticfusion: dense 3D semantic mapping with convolutional neural networks, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 4628–4635, doi:10.1109/ICRA.2017.7989538.

[15] N. Sünderhauf, T.T. Pham, Y. Latif, M. Milford, I.D. Reid, Meaningful maps with object-oriented semantic mapping, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 5079–5085.

[16] J. Engel, D. Cremers, LSD-SLAM: large-scale direct monocular slam, in: Proceedings of the ECCV, 2014.

[17] R. Mur-Artal, J.M.M. Montiel, J.D. Tards, ORB-SLAM: a versatile and accurate monocular slam system, IEEE Trans. Robot. 31 (5) (2015) 1147–1163, doi:10.1109/TRO.2015.2463671.

[18] R.A. Newcombe, S.J. Lovegrove, A.J. Davison, Dtam: dense tracking and mapping in real-time, in: Proceedings of the International Conference on Computer Vision (ICCV), IEEE Computer Society, Washington, DC, USA, 2011, pp. 2320–2327, doi:10.1109/ICCV.2011.6126513.

[19] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, A. Kolb, Real-time 3D reconstruction in dynamic scenes using point-based fusion, in: Proceedings of the International Conference on 3D Vision (3DV), IEEE Computer Society, Washington, DC, USA, 2013, pp. 1–8, doi:10.1109/3DV.2013.9.

[20] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges, A. Fitzgibbon, Kinectfusion: real-time dense

surface mapping and tracking, in: Proceedings of the Tenth IEEE International Symposium on Mixed and Augmented Reality (ISMAR), IEEE Computer Society, Washington, DC, USA, 2011, pp. 127–136, doi:10.1109/ISMAR.2011.6092378.
[21] J. Engel, J. Sturm, D. Cremers, Semi-dense visual odometry for a monocular camera, in: Proceedings of the ICCV, IEEE Computer Society, Washington, DC, USA, 2013, pp. 1449–1456, doi:10.1109/ICCV.2013.183.
[22] G. Klein, D. Murray, Parallel tracking and mapping for small ar workspaces, in: Proceedings of the Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality, (ISMAR), IEEE Computer Society, Washington, DC, USA, 2007, pp. 1–10, doi:10.1109/ISMAR.2007.4538852.
[23] G. Klein, D. Murray, Improving the agility of keyframe-based slam, in: Proceedings of the European Conference on Computer Vision (ECCV), 2008.
[24] R.F. Salas-Moreno, R.A. Newcombe, H. Strasdat, P.H.J. Kelly, A.J. Davison, Slam++: simultaneous localisation and mapping at the level of objects, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 1352–1359, doi:10.1109/CVPR.2013.178.
[25] S. Pillai, J.J. Leonard, Monocular slam supported object recognition, in: Proceedings of the Robotics: Science and Systems (RSS), 2015.
[26] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005, pp. 886–893.
[27] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110, doi:10.1023/B:VISI.0000029664.99615.94.
[28] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905, doi:10.1109/34.868688.
[29] C. Rother, V. Kolmogorov, A. Blake, "GrabCut"–Interactive foreground extraction using iterated graph cuts, ACM Trans. Gr. 23 (3) (2004) 309–314.
[30] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: Proceedings of the Twenty Second ACM International Conference, ACM, New York, NY, USA, 2014, pp. 675–678, doi:10.1145/2647868.2654889.
[31] H. Pfister, M. Zwicker, J. van Baar, M. Gross, Surfels: surface elements as rendering primitives, in: Proceedings of the Twenty Seventh Annual Conference on Computer Graphics and Interactive Techniques, (SIGGRAPH), ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000, pp. 335–342, doi:10.1145/344779.344936.
[32] R.W. Sumner, J. Schmid, M. Pauly, Embedded deformation for shape manipulation, in: ACM Trans. Gr. (TOG), 26, ACM, 2007, p. 80.
[33] M. Ozuysal, M. Calonder, V. Lepetit, P. Fua, Fast keypoint recognition using random ferns, IEEE Trans. Pattern Anal. Mach. Intell. 32 (3) (2010) 448–461.
[34] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395, doi:10.1145/358669.358692.
[35] D. Hoiem, S.K. Divvala, J.H. Hays, Pascal VOC 2008 challenge, PASCAL challenge workshop in ECCV (2009).
[36] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from RGBD images, in: Proceedings of the Twelfth European Conference on Computer Vision (ECCV), Springer-Verlag, Berlin, Heidelberg, 2012, pp. 746–760, doi:10.1007/978-3-642-33715-4_54.
[37] K. He, G. Gkioxari, P. Dollr, R. Girshick, Mask R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988, doi:10.1109/ICCV.2017.322.
[38] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder–decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the ECCV, 2018.
[39] M. Everingham, S.M.A. Eslami, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The pascal visual object classes challenge: a retrospective, International Journal of Computer Vision (IJCV) 111 (1) (2015) 98–136.

Guanzhong Tian received his B.S. degree in automation from Harbin Institute of Technology in 2010. He is currently a Ph.D. Candidate of the institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. His latest research interests include deep learning, robotics vision and SLAM systems.

Liang Liu received his B.S. degree in communications engineering from Zhejiang University of Technology in 2015. He is currently a Ph.D. candidate of the institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. His latest research interests include machine learning, robotics vision.

Jong Hyok Ri received the B.S. and M.S. degrees in information mathematics and computer science from Kim Il Song University, Pyongyang, D.P.R. of Korea. He is currently a Ph.D. degree candidate of the Institute of Cyber-Systems and Control, Zhejiang University, Zhejiang, China. His research interests include pattern recognition, machine learning and data mining.

Yong Liu received his B.S. degree in computer science and engineering from Zhejiang University in 2001, and the Ph.D. degree in computer science from Zhejiang University in 2007. He is currently a professor in the institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. He has published more than 30 research papers in machine learning, computer vision, information fusion, robotics. His latest research interests include machine learning, robotics vision, information processing and granular computing. He is the corresponding author of this paper.

Yiran Sun received her degree of Bachelor of Engineering in Automation of Zhejiang University in 2018. She is currently a Master Degree candidate of the institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. Her latest research interests include deep learning and computer vision.