IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 27, 2018, accepted September 18, 2018, date of publication October 5, 2018,
date of current version December 27, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2873617

Semantic SLAM Based on Object Detection and Improved Octomap

LIANG ZHANG[1], LEQI WEI[1], PEIYI SHEN[1], WEI WEI[2], (Senior Member, IEEE), GUANGMING ZHU[1], AND JUAN SONG[1]

[1]School of Computer Science and Technology, Xidian University, Xi'an 710071, China
[2]Shaanxi Key Laboratory for Network Computing and Security Technology, School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

Corresponding author: Wei Wei (weiwei@xaut.edu.cn)

This work was supported in part by the Key Research and Development Program of Shaanxi Province under Grant 2018ZDXM-GY-036, in part by the Scientific Research Program Funded by the Shaanxi Provincial Education Department under Grant 2013JK1139, and in part by the Specialized Research Fund for the Doctoral Program of Higher Education of China under Grant 20136118120010.

ABSTRACT Due to the development of the computer vision, machine learning, and deep learning technologies, the research community focuses not only on the traditional SLAM problems, such as geometric mapping and localization, but also on semantic SLAM. In this paper, we propose a Semantic SLAM system which builds the semantic maps with object-level entities, and it is integrated into the RGB-D SLAM framework. The system combines object detection module that is realized by the deep-learning method, and localization module with RGB-D SLAM seamlessly. In the proposed system, object detection module is used to perform object detection and recognition, and localization module is utilized to get the exact location of the camera. The two modules are integrated together to obtain the semantic maps of the environment. Furthermore, to improve the computational efficiency of the framework, an improved Octomap based on the Fast Line Rasterization Algorithm is constructed. Meanwhile, for the sake of accuracy and robustness of the semantic map, conditional random field is employed to do the optimization. Finally, we evaluate our Semantic SLAM through three different tasks, i.e., localization, object detection, and mapping. Specifically, the accuracy of localization and the mapping speed is evaluated on TUM data set. Compared with ORB-SLAM2 and original RGB-D SLAM, our system, respectively, got 72.9% and 91.2% improvements in dynamic environments localization evaluated by root-mean-square error. With the improved Octomap, the proposed Semantic SLAM is 66.5% faster than the original RGB-D SLAM. We also demonstrate the efficiency of object detection through quantitative evaluation in an automated inventory management task on a real-world data sets recorded over a realistic office.

INDEX TERMS CRF, Octomap, semantic messages, SLAM.

I. INTRODUCTION

In order to interact intelligently with its surrounding, intelligent robots should not only possess the geometric map, but also have the ability to understand the semantic meaning. However, most recent researches in the field of SLAM only focus on the geometric mapping, instead of both geometric and semantic mapping. Therefore, maps built by SLAM only can tell us where obstacles are and cannot supply semantic meaning. In such conditions, it is difficult to let the robot do some high level tasks, such as: ''go to the desk and pick up the green cup''. Thus, the map created by traditional SLAM can only be useful in simple missions, such as navigation and path planning. Obviously, it cannot meet our expected intelligent demands.

In this paper, we present a RGB-D semantic SLAM framework, which not only construct the semantic maps based on the geometric SLAM, but also improve the localization accuracy according to the semantic maps. Maps built by our approach contain information about the obstacles and their location. In terms of implementation, our system has two major modules: one module is the RGB-D SLAM based on sparse feature, which provides information about the locations of objects and also builds the 3D map. The other one is object detection realized by deep learning method. According to results of these two modules, we design the integrated RGB-D semantic framework, which provides the semantic map, and improves the localization accuracy.

Specially, our system creates a point clouds map of an environment with semantic meanings, which contains separate object models with semantic and geometric information. Our map not only maintains 3D point clouds by projecting semantic messages to 3D models, but also separates object entities independently. Because it can provide more advanced understanding of environment, a robot can do more impressive mission with a semantic map, for instance, finding a red cup.

Our contribution can be summarized as follows: The proposed system can detect and classify 80-200 object classes using deep-learning based detection algorithm, while the existing semantic mapping systems [41]–[42] can only detect less than 20 classes. Furthermore, when our system builds maps, it can create 3D object models without requiring a-priori known 3D models. Because 3D object entities of a semantic class, such as cup, have many kinds of shapes, it can limit the environment understanding as the robot needs to know the 3D object model of an object before identification.

In addition, our system mainly focuses on object-level entities, however, some other semantic segmentation methods, such as [1] and [2], focus only on pixel-level entities. Maps generated by such methods are less usable, because in this condition objects are modeled offline and maintained all the time.

In the rest of this paper, first we will discuss the related works. Then, our proposed system is described in detail. Three evaluations demonstrating the semantic mapping system are provided in the evaluation part. Finally, the paper is concluded in the last section.

II. RELATED WORK

Nowadays, SLAM has reached a level of maturity where maps can be built nearly in real time. However, maps built by traditional SLAM system only contain geometric entities (points, planes, surfaces etc) instead of semantic information. Although maps with geometric information are useful for the task of mapping, localization and navigation, robots cannot perform more advanced tasks, such as finding a cup, because maps don't contain any semantic information. As a result, the semantic representation of the environment is still an open research problem.

Semantic messages and SLAM can help each other interactively. Semantic messages can supply useful information for semantic map, and can also improve the accuracy of localization. At the same time, semantic maps can become more accurate according to the objects geometric information provided by the original SLAM system.

A. SLAM

Recently, many SLAM algorithms based on different sensors have been proposed. These SLAM algorithms can be categorized into two types according to the type of sensors: one is built based on Laser Sensor and the other is constructed by camera. In the first type, gmapping [3] is a typical SLAM algorithm based on Rao-Blackwellized Particle Filters. Google's Cartographer [4] is the newest SLAM algorithm based on Lidar input, which provides a good loop closure detection. The second type of SLAM algorithm can be divided into two different types as follows:

1) FEATURE-BASED METHODS

In these methods, firstly, the algorithm uses some traditional feature representation methods to extract a sparse set of features in each image, and then matches them in near frames and recovers camera poses from the matched features. Specifically, KinectFusion [5] uses RGBD camera to generate dense point clouds, recovers camera poses and scene structure with ICP algorithm, and accelerates tracking by CUDA. RGBD_SLAM [6] can also generate dense point clouds, it tracks ORB features and optimizes camera poses by G2O algorithm. However, the algorithm in RGBD_SLAM uses every frame to optimize camera poses instead of KeyFrames, and therefore it is computationally inefficient. PTAM [7] utilizes KeyFrames to optimize camera poses, it therefore works fast and stable, but it lacks loop closure detection, relocalization and auto initialization, and it can only generate sparse point clouds. ORB-SLAM [8], which is the state of the art in this field, not only supports RGBD camera, stereo camera and mono camera, but also contains loop-closing, relocalization, and auto initialization. It can work well both in small and large scale environments.

2) DIRECT METHODS

In these methods, camera pose is estimated directly from intensity values of the image. DSO_SLAM [9] uses Direct Methods to estimate poses and maintains 5 to 7 keyframes through sliding windows, but it lacks loop closure, which leads to more errors over time. LSD_SLAM [10] can generate semi-dense depth image, and it is used to match next frame in order to estimate camera poses. However, it is sensitive to light change. SVO_SLAM [11] belongs to half Direct Methods, because only sparse model-based image alignment uses Direct Methods, while pose estimation and bundle adjustment depend on features matching.

B. OBJECT DETECTION AND SEMANTIC SEGMENTATION

In order to build semantic map, object detection is needed to generate the semantic information of the environment. With the current development in the field of machine learning, several object detection methods have been proposed. Some of the methods can only detect few objects through features stored in databases with traditional computer vision algorithms [34]. However, some researches based on deep learning can detect many objects, even the objects belonging to the same class but having different shapes.

Specifically, we are interested in the task of object detection that can be utilized to isolate object instances in the map, for instance the output of [13]–[16] and [35] are in the form of Bounding Box. R-CNN [13] generates object proposals by selective search [37]. This network extracts features through AlexNext [38] and realizes classification by SVM [36], but

it takes several seconds to process one image. In order to improve R-CNN, Fast R-CNN [14] maps feature map to feature vector, and it is used as an input to fully connect layer by ROI-pooling, and replaces SVM with softmax. It is therefore faster than R-CNN, but it is still too slow for a real-time requirement in SLAM. Faster R-CNN [35] utilizes the Region Proposal Network (RPN) to generate object proposals and adds anchor and shared features to promote the speed of detection, its speed can reach up to 5fps. Yolo[15] is the fastest object detection algorithm, which replaces object proposals with S × SS × S grids, and realizes the final detection through classification of these grids.

C. SEMANTIC SLAM

Semantic information is very useful for the robots to understand their surrounding environment. Semantic SLAM is used to calculate the motion and position, and object detection and semantic segmentation are utilized to generate semantic map. Semantic SLAM can be categorized into two types based on the object detection methods.

The first type uses traditional methods to detect object. Real-time Monocular Object SLAM [17] is the most common one, which employs Bags of Binary Words and a database with 500 3D object models to provide a real-time detection. But it limits a lot because 3D object entities of a semantic class like cup having many different kinds of shapes. Reference [18] generates object proposals through multi-view images, then extracts dense SIFT descriptors from these proposals and predicts their classes. Reference [19] employs DPM [12], in which Hog feature is used to describe the object.

The other kind of SLAM is using deep-learning methods to do the object recognition, such as method proposed in [20], however, the semantic information is built based on pixels instead of object entities. In fact, this approach is too complex and not practical due to two reasons: (1) robot wants to understand the major semantic meaning of the environment in mission execution, which means it does not care about every pixel's semantic information, (2) computational speed is not sufficient to perform pixel level semantic classification in robot SLAM system.

D. MAPPING

Several approaches have been proposed to build 3D environments, and 3D maps can be represented with point clouds, elevation maps [21], multi-level surface maps [22] and so on. In the realistic SLAM system, map's size, speed, and accuracy are all important and we should make balance in different application. Unfortunately, Point clouds store large number of points and consume a lot of memory. Furthermore, it cannot easily differentiate between cluttered and free spaces. Elevation maps and multi-level surface maps cannot represent unmapped areas, although they are efficient. More importantly, these methods can not represent arbitrary 3D environments.

In our system, Octomap [23] is adopted which is used widely in the field of mapping. OctoMap has advantages of taking measurement uncertainty into account, being space efficient and implicitly representing free and occupied space. However, it still takes too much time to build the maps. Therefore, the work of [24] and multi-threads are used in our system to accelerate the mapping process.

III. SYSTEM OVERVIEW

A. SLAM ANALYSIS

Nowadays, the geometric aspect of the SLAM problem is well understood, and has reached a level of maturity where city level maps can be built precisely and even in real time. But they can only work well in static environments or the one with small dynamic objects. In the scene with small dynamic objects, as only few feature points are situated at dynamic objects, the SLAM can therefore still work well. However, Feature-Based SLAM is easy to be effected by large moving objects. Therefore, most Feature-Based SLAM systems are built based on a strong assumption that the number of features on moving objects is much smaller than those on static objects. Besides, most SLAM system can only provide simple geometric information for robots to do simple missions like navigation and obstacles avoidance. For more advanced missions, SLAM system should provide not only geometric information, but also semantic messages. Based on these two different information aspects, a robot can have more advanced understanding of the surrounding environment. For example, if a robot wants to grasp a cup, primary, it needs to identify the cup first, then locate its position. After getting these informations, the robot can find an optimal path based on the SLAM mapping to go to the position of a cup and grasp it correctly. At the same time, maps generation speed is also another important factor in the SLAM system. Octomap is based on OcTree structure which is good for searching and building, while point clouds only store each points without any structures. Octomap can carry not only the RGB and position information but also the semantic messages. In contrast, the point clouds only store the original messages from RGB-D camera. In addition to this, Octomap uses a probabilistic model to build more accurate maps. Therefore, Octomap is suitable for navigation and more advanced missions. Unfortunately, it is still too slow for maps building with a higher resolution because it uses some ineffective algorithms to compute the empty voxels [23].

B. ORB-SLAM ANALYSIS

ORB-SLAM2 is a typical and famous Featured-Based SLAM. As Fig.1 shows, it contains three threads which run in parallel: (1) Tracking, (2) Local Mapping and (3) Loop Closing. **Tracking thread** is in charge of localizing the camera with every frame in real time and deciding when to insert a keyframe. In tracking thread, it performs an initial feature matching with the previous frame and optimizes the pose by Bundle Adjustment (BA) algorithm. If tracking is lost, it performs a global re-localization with Bag of Word, then searches map points by re-projection and optimizes the pose

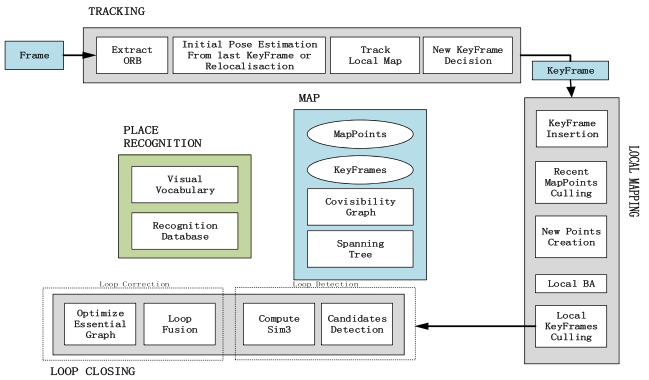

FIGURE 1. ORB-SLAM system overview.

with local map points. Finally, the tracking thread can decide if a new keyframe can be generated.

After getting a new keyframe, the **Local Mapping thread** will triangulate new map points through its relative keyframes. Then, it optimizes the pose of relative keyframes and map points with BA. Finally, redundant keyframes and low quality map points are removed.

The **Loop Closing thread** responses to loop closure with every keyframe. If a loop is detected, the similarity transformation is computed which represents the drift accumulated in the loop. Then both sides of the loop are aligned and duplicated points are fused. Finally, a pose graph optimization over similarity constraints is performed in order to achieve global consistency.

Although ORB-SLAM2 is a very practical algorithm, it still faces some questions, such as how to work well in dynamic environments, how to supply semantic information and maps and so on. The destination of our proposed system is to designing an enhanced semantic SLAM system based on ORB-SLAM2, which is not only suitable for dynamic environments, but also can supply semantic maps for robot executing in high level missions.

C. OVERVIEW OF SEMANTIC SLAM SYSTEM

Fig.2 is an overview of our proposed Semantic SLAM system, which is based on ORB-SLAM2 and integrated with some other processes. In the proposed Semantic SLAM system, ORB-SLAM2 is in charge of camera localization and mapping with every RGB-D frame. **Tracking thread** is responsible for tracking by keyframes instead of reference frames, in order to decrease the effect of moving objects. **Local Mapping thread** adds a few keyframes to create semantic messages, because semantic messages extraction cannot fulfill the requirement of real-time performance.

After getting keyframes from ORB-SLAM2, **YOLO**[15] is used to detect objects in each keyframe to get semantic message. In our implementation, we use the tiny-weight version to detect objects, because this version is trained on MS-COCO Dataset, which contains 80 different kinds of objects.

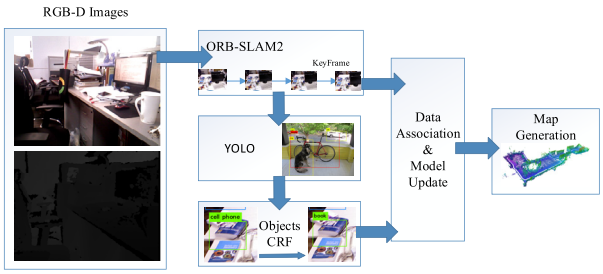

FIGURE 2. An overview about our system.

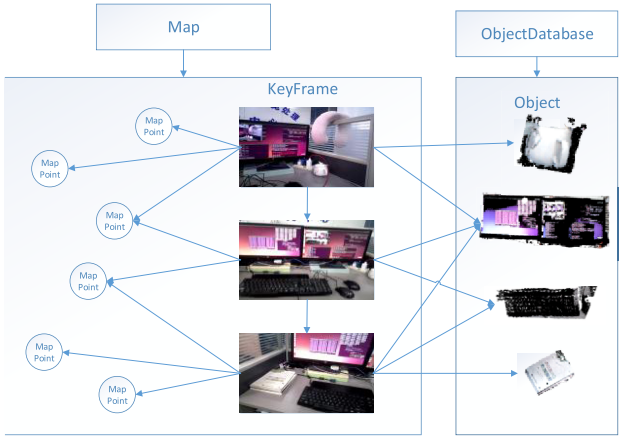

FIGURE 3. An illustration of the relationship between keyframes, map points and objects.

Next, object regularization based on CRF is used to correct the probabilities of each object computed by YOLO. In this process, constraints between objects are computed according to the statistics of MS-COCO Dataset, and it is then used to optimize the object probabilities computed by YOLO detection.

When accurate labels of each object are captured, filter process is used to provide more stable features and remove the unstable features which are always located on the moving objects. At the same time, the temporary objects are created, which contain point clouds produced by projection.

When temporary object is generated, we use data association module to decide either to create a new object or associate it with existing object in the map according to the matching score. In data association, in order to find correspondence between existed objects and temporary objects, we first build relationship between keyframes and objects. Then, Kd-Tree structure is used to accelerate the computation of matching score. When the existing objects can be combined with the temporary objects, the former can be updated with the new detection by a recursive Bayesian process.

Finally, Map Generation uses point clouds stored in objects to generate map based on Octomap, which is accelerated by multi-threads realization and Fast Line Rasterization algorithm.

D. RELATIONSHIP BETWEEN KEYFRAMES AND OBJECTS

In order to integrate the concept of semantic into the framework of ORB_SLAM2, we construct relationship between keyframes and objects by referring to the implementation method between keyframes and map points, which has existed in ORB-SLAM2. In ORB_SLAM2, each keyframe stores map points that it has observed in the frame image, at the same time, each map point records the keyframes which have observed the map point sequentially. As a result, we can build relationship between keyframes and perform some optimization, such as analyzing whether a keyframe is redundant or deciding whether a map point has high quality. Therefore, we build the relationship between keyframe and each object as followings.

In our realization, each object O_i contains:

1) Word coordinates of each point cloud that are located on the object.
2) A fixed number of class labels and the corresponding confidence score which is calculated through a recursive Bayesian update.
3) Keyframes which can observe this object.
4) Kd-tree structure generated through the object's point clouds, which is used for fast search.
5) The class label which this object belongs to.
6) The number of observations.

Each keyframe K_i should store:

1) The corresponding RGB image which is used to detect objects.
2) The corresponding depth image which is used to generate point clouds.
3) Objects that have been observed in this keyframe.

From above description, we can build relationship between keyframe and objects as keyframes and map points. Therefore, keyframe can find its relative objects, and vice versa. At the same time, we create an object database, in which all the detected objects are stored.

IV. SEMANTIC MAPPING

In this section, we introduce each module of the semantic SLAM, which include Improved SLAM, Object Detection, Object Regularization, Temporary Objects Generation, Data Association, Object Model Update, and Map Generation

A. IMPROVED SLAM

The proposed semantic SLAM is constructed based on ORB-SLAM2. In ORB_SLAM2, **Tracking thread** localizes the camera with every frame through four steps. First, ORB features are extracted from RGB images. Second, ORB features are used to perform feature matching with the reference frame, preliminarily calculate the camera pose and return the number of matched map points. Third, the camera pose is optimized again with the matched local map points which are searched through the relative keyframes. Finally, tracking thread decides whether a new keyframe is inserted based on some principles. In order to combine ORB_SLAM2 and Deep Learning based Object Detection seamlessly, tracking thread module is modified in the following three ways.

In order to reduce the effect of dynamic objects, the second step in **tracking thread** is changed to track by keyframes instead of reference frames. If SLAM tracks by reference frame, the camera pose calculation can easily be effected by large moving objects. This is because, when a large moving object passes by, original SLAM will track features on the moving object, which affects the tracking accuracy. However, if SLAM tracks feature by keyframes, it can still calculate the correct camera pose before the new keyframe insertion. The essential reason is that old keyframe doesn't contain features of the moving object. We choose the Levenberg-Marquardt method, from G2O [45] which contains several optimization algorithms to optimize the pose of the current frame. This method needs a good initial estimated value for optimization. Therefore we use the constant velocity motion model to predict the position of the current frames as a G2O initial value before optimization.

The third step in tracking process is modified to compare the number of matched inliers with the result of the second step to judge whether the tracked current frame is lost. In ORB-SLAM2, matched inliers are compared with a constant value in the third step. It is easy to lose tracking when the camera moves fast, because the ORB feature points of current frame may only match with the map points observed by the last frame. In this case, the number of the observed map points will be less than the constant value. Therefore, the third step should compare the number of matched inliers with the number of matched inliers computed by the last frame. The second step function computes the number of matched map points between the last keyframe and the current frame, therefore we should compare the number of matched inliers with the result of the second step.

Tracking thread is changed to create more keyframes. In ORB-SLAM2, tracking module mainly tracks by keyframes, fewer and fewer features can be seen when camera moves, which leads to lose tracking easily. Therefore, more keyframes are needed to create more map points and it is useful for correct tracking.

B. OBJECT DETECTION

In semantic SLAM, individual objects are important entities which not only can supply semantic information for the map, but also can enhance the localization accuracy, therefore a fast and robust object recognition method is needed. In deep-learning area, DeepLab and FCN [2] can provide pixel level semantic segmentation, however, RCNN, Fast-RCNN and Faster-RCNN can supply the object level's bounding box detection. Furthermore, they may generate too many object proposals which cause detection of same region multiple times. Thus, many of them are too slow, and they cannot satisfy the real-time requirement when they are integrated with SLAM. For example, Faster-RCNN just can process 5 images per second which is very slow for SLAM. In the algorithm of YOLO(You Only Look Once), it divides each

image into N×N grids, and each grid is only detected once. As a result, YOLO can process 45 images per second, therefore we choose YOLO as the object detection method to generate a number of object proposals in the form of bounding boxes for every keyframe in our proposed semantic SLAM system.

YOLO is a state-of-the-art, real-time object detection system. We use the network trained on the COCO dataset instead of PASCAL VOC, because COCO dataset contains 80 types of objects, while PASCAL VOC dataset only has 20 kinds of objects. Although Yolo can process images at 40-90 FPS on a Titan X with normal YOLO weights, GT730 only has 1GB memory which is not enough for normal YOLO version. After some experiments, we find that normal YOLO still takes about 0.04s per image which cannot fulfill real-time requirement. However, Tiny YOLO weights can fulfill all these requirements. It only uses about 700MB memory and takes about 0.02s per image which is twice as much fast as the normal YOLO version. As a result, we use Tiny YOLO weights instead of normal YOLO weights. In the implementation, YOLO detects the new keyframe in Local Mapping module after the new keyframe is added and wrong map points are culled sequentially.

C. OBJECT REGULARIZATION

Although we can get the semantic label for each object through YOLO algorithm, semantic context among objects has not been explicitly incorporated into the object categorization models. Some researchers have found that context information is good for semantic segmentation, however, object context requires access to the referential meaning of the object [25]. In other words, when performing the task of object categorization, objects' category label must be assigned with respect to other objects in the scene, assuming there is more than one object present. To illustrate this further, we can have a look at the following example.

In the scene of a tennis match, four types of objects are detected and categorized respectively, labeled as ''Tennis court'', ''Person'', ''Tennis Racket'', and ''Lemon''. Using a categorization system without a semantic context module, these generated labels would be the final classification results. However, in context, one of these labels is not satisfactory. Namely, the object labeled ''Lemon'', with an appearance very similar to a ''Tennis Ball'' is probably mis-labeled, due to the ambiguity in visual appearance. By enforcing semantic contextual constraints, the label of the yellow blob changes to ''Tennis Ball'', as this label fits more precisely in the context with other labels.

In the task of image segmentation, context information is used to optimize the final result with CRF (Conditional Random Filed) algorithm. The history of CRF for image segmentation tasks has started before [26]–[30]. The main problem that CRF can solve effectively is how to model the class scores calculated by some classifiers and local information of images simultaneously. Then this problem can be treated as a problem of maximizing a posteriori. We can define unary potentials to model the probabilities that each pixel or patches belongs to each category, and pairwise potentials to model relation between two pixels or patches. The frequently used CRF models contain unary potentials, pairwise potentials and some weighted parameters. The pairwise potentials are modeled on 4 or 8 neighbors, like [26], [29], [42]–[46], therefore this structure is limited in modeling long distances on the image. Aware of the above problem, a lot of expanded CRF models have been developed in recent years [29]–[31]. Toyoda and Hasegawa [32] proposed a fully connected CRF to integrate local information and global information jointly which sets up pairwise potentials on all pairs of pixels or patches in semantic image labeling task. By modeling long range interactions, dense CRF provides a more detailed labeling compared to its sparse version. The dense CRF with Gaussian kernel potentials has emerged as a popular framework for semantic image segmentation tasks [33].

Inspired by the above idea, we construct a probabilistic object-based dense CRF. Compared with CRF based on pixel level, our proposed model can reduce the computational complexity significantly. The corresponding Gibbs energy function of our object-based model is given by:

$$P(x) = \frac{1}{Z} exp(-E(x)) \tag{1}$$

$$E(x) = \sum_{i} \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j) \tag{2}$$

where x is the label assignment for objects given its features, and i, j range from 1 to k (k is the number of objects in the map). Z is the normalized factor. In object-based CRF model, the objective is to minimize $E(x)$ to obtain the final label assignment. The unary potential ψ_u models each vertex of CRF and the pairwise potentials ψ_u models connections between vertices. The general forms of unary and pairwise potentials can be represented as

$$\psi_u(x_i) = -\log P(x_i) \tag{3}$$

$$\psi_u(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{K} \omega_m exp\left(-a_m \left(f_{i,j}\right)^2\right) \tag{4}$$

where $P(x_i)$ is the label probabilistic distribution at i^{th} objects detected by YOLO algorithm, ω is the linear combination weights. μ is called the label compatibility function which describes the possibility of simultaneous occurrence of two different classes in adjacent locations. The Potts Model is the simplest label compatibility function and it is used in our system:

$$\mu(\mathrm{x_i}, \mathrm{x_j}) = \left\{ \begin{array}{l} 0, \text{ if } \mathrm{x_i} = \mathrm{x_j} \\ 1, \text{ otherwise} \end{array} \right\} \tag{5}$$

$f_{i,j}$ is the constraint of i^{th} object and j^{th} object, in our technique, $f_{i,j}$ is calculated as following:

$$f_{i,j} = \frac{1}{p_{i,j}} \tag{6}$$

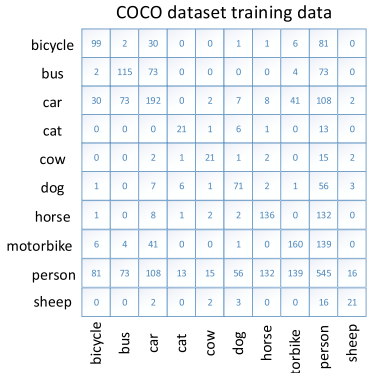

FIGURE 4. **The counts of different labels.**

where $p_{i,j}$ is the probability that i^{th} and j^{th} objects are appeared at the same place, which is acquired through statistics of the COCO dataset. Each image has different kinds of objects which are shown at the same time. Therefore, a label co-occurrence counts matrix is calculated according to those images. An entry i, j in this matrix represents that the times of an object with label i appears in a training image with an object of label j. The diagonal entries indicate the frequency of the object in the training set. The part of the label co-occurrence counts matrix i.e. the confusion matrix is shown in Fig. 4.

Then, $p_{i,j}$ can be calculated through

$$p_{i,j} = \frac{n_{i,j}}{n_j} \tag{7}$$

where n_j is the frequency of the object j, $n_{i,j}$ is the frequency of object i that appears at the same time with an object j. After we get the unary and pairwise potentials, we can use mean fields method to optimize the CRF model. Based on this method, we not only utilize the YOLO object detection, but also integrate the object context information to refine the final object confidence score [47]–[55].

D. TEMPORARY OBJECTS GENERATION

After getting the accurate semantic label, we can filter features and map points according to the semantic label type, thus the effects of dynamic objects can be reduced to the maximum extent. We call this process as feature filter. First, the object is classified into two types, one is static objects, such as sofa, bed and so on, and another is dynamic objects, such as person and car. Then, while computing labels and bounding boxes from above algorithm, we exclude ORB features, map points and DBoW features that belong to the dynamic objects, and retain features on static objects.

Fig.5 demonstrates this process. An original image is shown in Fig.5(a); Fig.5(b) shows ORB features extracted from the original image; Fig.5(c) shows the semantic messages extracted from the original image; Fig.5(d) shows the result of the features filter. As we can see, features that belong to the dynamic object are removed.

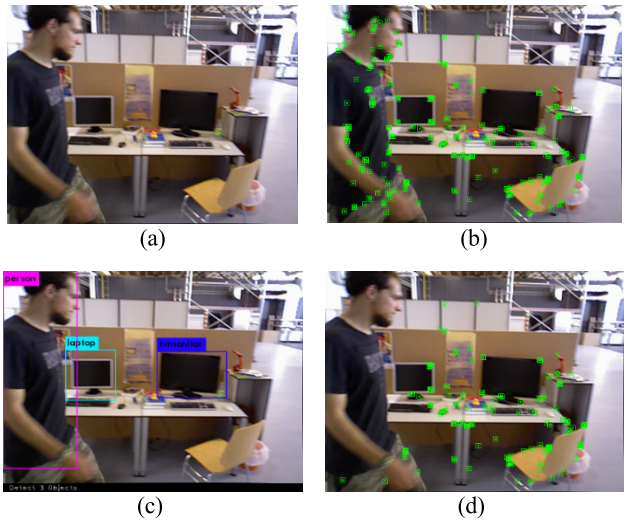

FIGURE 5. **Features filter. Features belonging to moving objects are removed.**

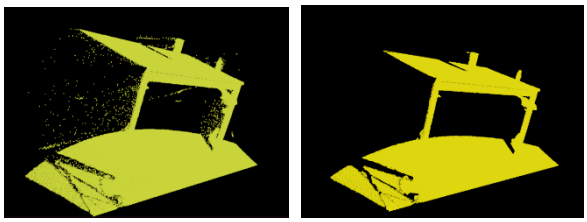

FIGURE 6. **Noises removing.**

After the feature filter process, we generate some temporary objects which contain object size, object type, object confidence scores, and the corresponding point clouds. However, point clouds generated by the RGB-D camera contain some noises. For example, as Fig.6 shows, there are some outliers around the desk. In order to remove these noises, we apply statistical calculation to point clouds. If the points deviate from the average, they may be noises, and can therefore be removed.

In our realization, in order to save memory, point clouds are down-sampled with 5 mm resolution. When getting the robust temporary objects and point clouds, we use data association to decide whether those temporary objects are new objects or already exist in the map.

E. DATA ASSOCIATION

Data association is very important for robust SLAM. In our semantic SLAM system, data association is used to judge the detected objects. In the proposed method, there are two steps for data association.

First, we need to find the candidate objects for each temporary object. Through the relationship between keyframes and map points, we can easily find keyframes which are relative to the current keyframe. These keyframes not only are close to the current keyframe, but also are more likely to contain same objects because they have enough shared map

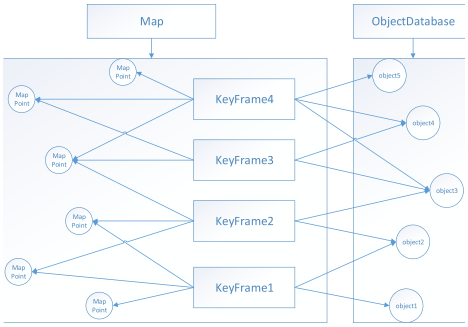

FIGURE 7. Relationship between keyframes and mappoints.

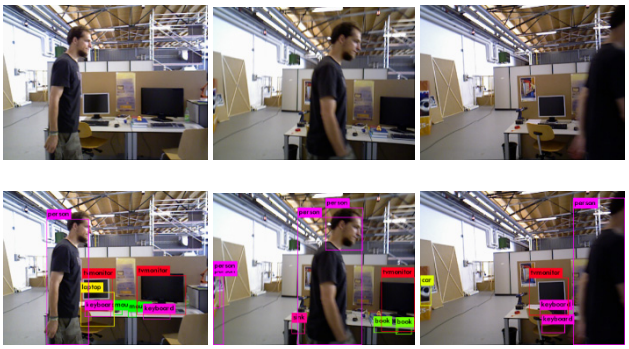

FIGURE 8. The scene with dynamic objects. The first line shows original images. The second line shows semantic messages extracted from the above original images. The person is the dynamic object in this scene.

points. With the relationship between keyframes and objects, the objects seen by these relative keyframes are considered as the candidate objects for every temporary object.

For example, as Fig.7 shows, KeyFrame1, KeyFrame2 and KeyFrame3 have been inserted into the SLAM system. When KeyFrame4 is inserted, the system detects object3, object4 and object5, which are treated as three temporary objects. In order to find the candidate objects for such three temporary objects, we take the following two steps. First, we search the relative keyframes for KeyFrame4. As we see, KeyFrame4 and KeyFrame3 have two common map points, and KeyFrame4 and KeyFrame2 have one common map points, while KeyFrame4 and KeyFrame1 have no common map points. Therefore, KeyFrame3 and KeyFrame4 are the relative keyframes. Second, we can infer that Object2, Object3 and Object4 can be observed by KeyFrame3 and KeyFrame4, therefore Object2, Object3 and Object4 are regarded as candidate objects for temporary objects observed by KeyFrame4.

With the relationship between keyframes and objects, we can avoid some undesirable situation caused by moving objects. For instance, as Fig. 8 shows, a man passes by a desk in the office, and three frames are selected as keyframes. The first keyframe shows that a man starts to block a white TV monitor, and we detect the TV monitor from the first keyframe. Then, the second keyframe represents that a man blocks the TV monitor, and we cannot detect TV monitor in this keyframe. However, the third keyframe shows that the man has passed in front of the TV monitor; it appears again. In the third keyframe, it generates a temporary object whose label is TV monitor. According to the relationship between keyframes and objects, we can easily find that the TV monitor on the first keyframe is a candidate object for the temporary object observed in the third keyframe. The first keyframe and the third keyframe have a lot of same map points, which reveals that the first keyframe is one of the relative keyframes for the third keyframe. If we use the object seen by the last keyframe (it is the second keyframe in this condition) as candidate objects, the temporary object (TV monitor in the third keyframe) will be regarded as a new object by data association because the last keyframe does not contain the white TV monitor.

Second, among the candidate objects, we need to select which one is most similar to the temporary object. In our realization, we perform a nearest neighbor search between 3D points in candidate and temporary objects, and calculate the Euclidean Distance between the matched point pairs. In this stage, k-d tree is used to accelerate the matching process. According to the matched point pairs, scores between candidate and temporary objects can be calculated. A candidate object with the highest score which is also higher than the threshold, is selected as the associated object. If all the objects do not fulfill real-time requirements, the temporary object is considered as a new object which can be inserted into the SLAM system. The score between candidate and temporary objects is calculated by:

$$\mathrm{S} = \frac{M}{N} \tag{8}$$

M means the number of the matched points which have distance of 2cm or less, while N means the number of the points from the temporary object.

F. OBJECT MODEL UPDATE

When we find the correspondence between candidate and temporary objects, the point clouds and confidence scores associated with them should be fused together. In YOLO object detection, RGB image is inputted into the CNN architecture, given the data I_k of the kth image, the output of YOLO can be interpreted in a simplified manner as a per-object independent probability distribution over the class labels, such as $P(O_u) = l_i|I_k$, with u denoting the detected objects and l_i denoting the i^{th} class label. This enables us to update all the objects in the visible set $V_k \in M$ with the corresponding probability distribution by means of a recursive Bayesian update:

$$P\left(l_i|I_{1,\ldots,k}\right) = \frac{1}{Z} P\left(l_i|I_{1,\ldots,k-1}\right) P\left(O_u = l_i|I_k\right) \tag{9}$$

Equation (9) is applied to all the label probabilities for each object, and finally it is normalized with constant Z to yield a proper distribution.

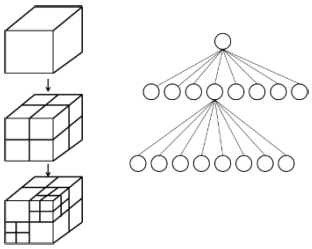

FIGURE 9. The structure of an octree.

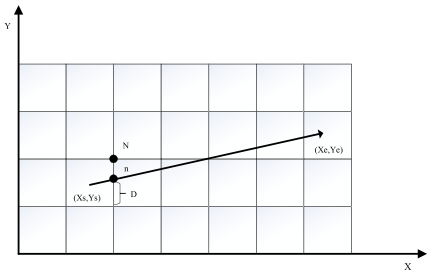

FIGURE 10. Ray casting algorithm in 2D plane.

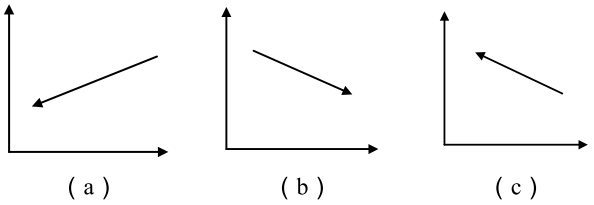

FIGURE 11. Three different situation about ray casting.

At the same time, storing the point clouds with every object allows us to re-build object model when the SLAM system updates its trajectory estimation after a loop closure.

G. MAP GENERATION

In our system, 3D point clouds are stored in every keyframe, and the segmented 3D point clouds are also stored in the corresponding object. Therefore, the map based on point clouds can be generated by projecting the stored 3D points according to the associated poses. However, the map based on point clouds is useless for advanced mission such as path planning or grasp point selection, because point clouds do not use any structures to store each point, which is bad for searching, and each point has no volume information, which makes collision detection and 2D maps generation easily fail. Besides, point clouds cannot distinguish the unknown area, the empty area, and cannot eliminate noise. Octomap uses Octree to store point clouds, it can distinguish the unknown and empty areas and eliminate noise when a new point is inserted. Furthermore, each voxel has a volume.

Octomap is a probabilistic 3D mapping framework based on octrees. As Fig.9 shows, a space is divided into eight small cubes of voxels, and each small space can even be divided into eight smaller spaces. In the realization, root node of an octree represents the whole space, and its eight children denote eight small spaces. The leaf node of an octree represents the smallest resolution voxel in the space.

During realization, the moving object and error in the range measurements can cause a lot of noises in maps. In order to decrease such influence, octomap uses a probabilistic model to solve this problem. Each leaf node in an Octomap stores its probability to be occupied or free. When a new 3D point is inserted, its corresponding leaf node updates its probability in the following way:

$$L\left(n|z_{1:t}\right) = L\left(n|z_{1:t-1}\right) + L\left(n|z_t\right) \tag{10}$$

with

$$L\left(n\right) = log\left[\frac{P\left(n\right)}{1-P\left(n\right)}\right] \tag{11}$$

where n represents a leaf node n, z_t represents the measurements z_t. Therefore, $P\left(n|z_t\right)$ denotes the probability of voxel n to be occupied given the measurements z_t.

Octomap can easily generate the 3D or 2D map based on voxels or pixels, which is useful for path planning and grasp.

At the same time, each voxel in Octomap can store not only its probability to be occupied or free, but also a fixed number of class labels and confidence scores. In the original Octomap, there are two steps to generate map.

First step is to compute the position of the empty voxels from the camera to occupied voxels. Unfortunately, this takes too much time to compute the empty voxels. In order to accelerate this step, we optimize this process. As shown in Fig.10, we consider 2D plane as an example.

In Fig.10, there is a straight line from (X_s, Y_s) to (X_e, Y_e), the slope of this line is computed by:

$$K = \frac{Y_e - Y_s}{X_e - X_s} \tag{12}$$

The grid (X, Y) is denoted by:

$$X = \frac{X_s}{V}, Y = \frac{Y_s}{V} \tag{13}$$

where V is the size of grid.

The offset in this grid is denoted by:

$$X_0 = X_s\%V, Y_0 = Y_s\%V \tag{14}$$

Finally, we can compute D by:

$$D = K \times (V - X_0) + Y_0 \tag{15}$$

If D is smaller than V, it means that n is below N. Therefore, the next empty voxel is $(X+1, Y)$. If D is bigger than V, the next empty voxels are $(X, Y+1)$ and $(X+1, Y+1)$. Then, we can compute the empty voxels from the start point to the end point quickly.

Besides, we have three other situations as displayed in Fig.11.

In the situation (a) in Fig.11, we can swap the start point and the end point and compute empty voxels as above. However, in the condition (b), we convert $L1$ to $L2$ to compute empty voxels as Fig.12 shows.

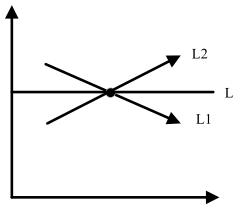

FIGURE 12. Convert L2 to L1 according to L.

We first compute the axis of symmetry L:

$$Y = \frac{Y_s + Y_e}{2} \tag{16}$$

Then, we create a new line L2 by reserving the origin line L1 according to the Line L. Based on that, we can compute empty voxels as above and each voxel should be reserved according to the Line L.

In the situation (c) in Fig.11, we can convert it into (b) by swapping the start and end points.

In the second step, all new voxels are inserted into an Octomap, which locates the corresponding leaf node for every new voxel through the structure of Octomap. Then, the probability of corresponding leaf node is updated with the new voxel.

However, both of these two steps use a single thread to process. Although the first step can use OpenMP to create multi threads to compute the empty voxels, we find that it runs slower than the one with a single thread because too many threads are created. In order to accelerate mapping with multi-threads, three strategies are proposed.

First, we use a thread Pool to take the place of OpenMP. The second step is modified to use eight threads to insert voxels into an Octomap. And the third is that the whole architecture can be accelerated by using a producer-consumer model.

V. EVALUATION

In this section, we give detailed evaluation of the proposed semantic SLAM system in three different ways. First, we use TUM dataset to evaluate the accuracy of tracking. Second, PASCAL dataset is used to evaluate the accuracy of detection. Third, we compare the efficiency between improved and original Octomap.

A. TRACKING

TUM dataset is an excellent dataset to evaluate the accuracy of camera localization as it provides accurate ground truth for the sequences. It contains seven kinds of sequences recorded by a RGB-D camera at 30fps and a resolution of 640 x 480. We only use Handheld SLAM sequence, Robot SLAM sequence, Structure vs. Texture sequence and Dynamic Objects sequence among the seven sequences because these four sequences represent most of the scenes of every day. Furthermore, these sequences contain different kinds of objects, which can ensure more semantic information than other kinds of sequences. Handheld SLAM sequence is recorded by hands, and therefore it has complex and unstable trajectories, while Robot SLAM sequence is recorded by real robots, hence it has stable and simple trajectories. However, Structure vs. Texture sequence mainly focuses on textures and structures of the environment. But all of them are experimented in the scene without dynamic objects. Furthermore, Dynamic Objects sequence is recorded by hands, however when the recordings have some dynamic objects, the trajectory is found to be unstable in such scenes. For comparison, we use different RGB-D SLAM in the benchmark. Each sequence is processed 5 times, and we use RMSE to judge its localization Accuracy, while RMSE is computed by:

TABLE 1. Frame localization error comparison in the TUM RGB-D benchmark.

datasets	Absolute Frame Trajectory RMSE ($m \times 10^{-2}$)	
	Improved SLAM	ORB-SLAM2
Fre1_desk (static)	2.1019	**1.9618**
Fre1_floor (static)	**2.8640**	3.0056
Fre1_xyz (static)	1.2917	**1.0843**
Fre3_walking_half (dynamic)	**10.4595**	17.8241
Fre3_walking _half_val(dynamic)	**11.8822**	15.7652
Fre3_walking_xyz(dynamic)	**20.9494**	27.8164
Fre3_waling_xyz_valid(dynamic)	**7.0189**	30.0308

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left(X_{obs,i} - X_{model,i}\right)^2}{n}} \tag{17}$$

Where n means the number of observation, i denotes the i^{th} observation. $X_{obs,i}$ is the groundtruth of the i^{th} observation, while $X_{model,i}$ is the computation result of the i^{th} observation.

B. LOCALIZATION ACCURACY WITH IMPROVED SLAM

First, we verify the localization accuracy with Improved SLAM. For comparison we have executed ORB-SLAM2 with a RGB-D camera in the benchmark. Table 1 shows the median RMSE error of the benchmark. It can be seen that the RMSE error of Improved SLAM is little worse than the error of ORB-SLAM2 in static environment, which means the accuracy of our SLAM is close to ORB-SLAM2. The reason of worse RMSE error is that when tracking reference frames, the number of common features that the last reference frame and the current keyframe have is larger than that the last keyframe and the current keyframe have. Because of the number of common features, SLAM tracking by reference frames can provide more accurate localization in the static environment.

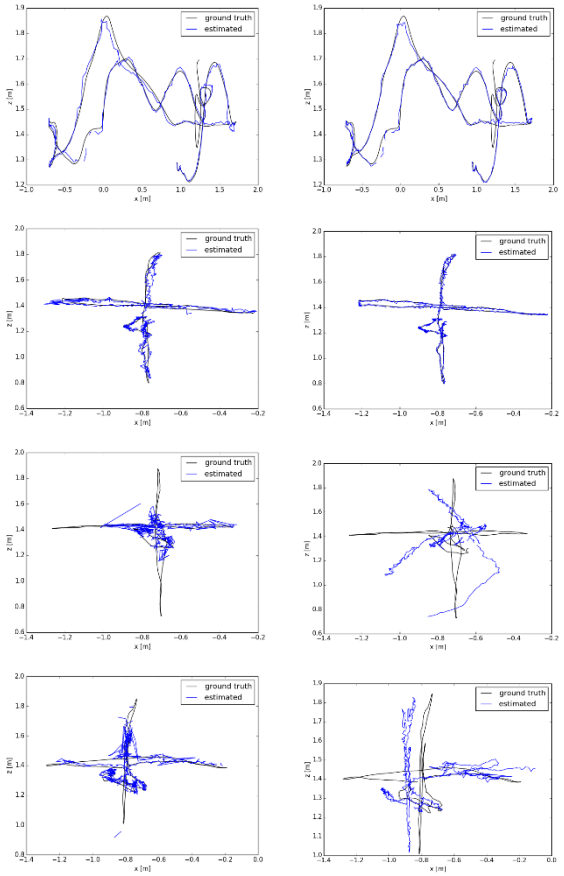

FIGURE 13. Sequences Comparison in TUM RGB-D Dataset. First column shows the trajectories generated by our SLAM. Second column shows the trajectories generated by ORB-SLAM2. First row uses Fre1_desk and second row uses Fre3_sitting_xyz. Both of them are the static environment. Third row uses Fre3_walking_xyz and fourth row uses Fre3_waling_xyz_valid, they are dynamic environment.

However, in the scene with dynamic objects, our SLAM is better than ORB-SLAM2. This is because our SLAM tracks by keyframes instead of reference frames, when a big moving object passes through the camera, the trajectory is easy to follow the moving object if SLAM tracks by reference frames. Since a big moving object contains a lot of features, and SLAM are based on the assumption that the number of features in moving objects is much smaller than the number of features in static objects, so in this case, original ORB-SLAM2 considers the big moving object to be static. If SLAM tracks by keyframes, it can reduce such influence significantly. Before the new keyframe which contains a moving object is added into Local Mapping thread, SLAM only tracks by the last keyframe which only contains static objects.

C. LOCALIZATION ACCURACY WITH SEMANTICS

In this section, we verify the improvements of our semantic SLAM. For comparison we have also executed RGB-D SLAM[6] in the benchmark. Fig.13 shows the comparisons of trajectories calculated by semantic SLAM, with the trajectories calculated by ORB-SLAM2 and RGBD-SLAM. As we observe, our trajectories are similar to the trajectories calculated by ORB-SLAM2 in the scenes without dynamic objects. Table 2 shows the median RMSE error of the benchmark. From Table 2, we can see that the RMSE error of our semantic SLAM is little worse than the error of ORB-SLAM2 and Improved SLAM, which means the accuracy of our SLAM is on par with ORB-SLAM2. The reason causing worse RMSE error is that Local Mapping thread takes more time to detect objects, which means that

TABLE 2. Frame localization error comparison in The TUM RGB-D benchmark.

Dataset	Absolute Frame Trajectory RMSE ($m \times 10^{-2}$)		
	Our SLAM	ORB-SLAM2	RGBD-SLAMv2
Fre1_desk (static)	2.0537	**1.9618**	4.6471
Fre1_floor (static)	3.1381	**3.0056**	9.6217
Fre1_xyz (static)	1.3363	**1.0843**	1.1953
Fre2_360_kidnap (static)	8.1014	**7.9486**	77.7447
Fre2_desk (static)	1.8023	**1.3807**	6.3590
Fre2_desk_with_person(static)	2.4953	**1.3823**	4.1659
Fre3_long_office(static)	1.9562	**1.3252**	2.5010
Fre3_long_office_val(static)	2.2992	**1.5368**	3.0429
Fre3_nostruct_text_far(static)	6.8997	**2.6620**	3.8133
Fre3_nostruct_text_far_val(static)	4.7455	**2.1409**	4.2396
Fre3_nostruct_text_near_loop(static)	1.5803	**1.1160**	1.8341
Fre3_nostruct_text_near_loop_val(static)	2.1017	**1.4544**	1.6238
Fre3_sitting_half(static)	3.2395	**1.4793**	4.4355
Fre3_sitting_half_val(static)	1.9996	**1.6557**	2.2166
Fre3_sitting_xyz(static)	2.7856	**1.5187**	1.0985
Fre3_sitting_xyz_val(static)	2.1190	**1.6577**	1.4112
Fre3_struct_text_far(static)	1.2438	**1.0091**	1.4831
Fre3_struct_text_far_val	1.3975	**1.0580**	2.2037
Fre3_struct_text_nea(static)r	1.2718	**1.0682**	2.8285
Fre3_struct_text_near_val(static)	1.1940	**0.9316**	2.9512
Fre3_walking_half(dynamic)	**6.3607**	17.8241	61.7521
Fre3_walking_half_val(dynamic)	**5.3591**	15.7652	26.2341
Fre3_walking_xyz(dynamic)	**3.3594**	27.8164	88.0750
Fre3_waling_xyz_valid(dynamic)	**3.8746**	30.0308	145.3703

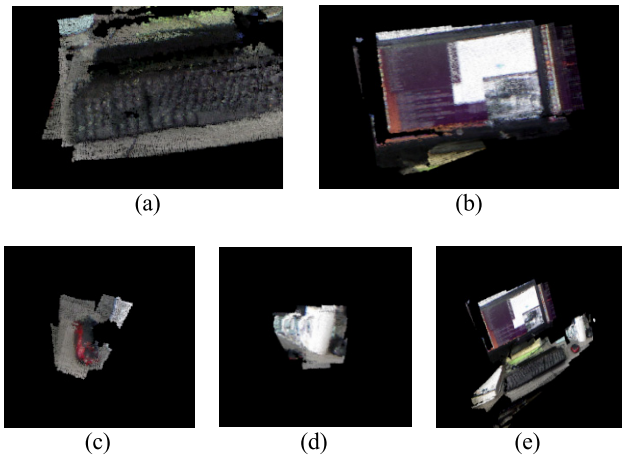

FIGURE 14. After getting semantic messages, we insert these semantic messages into Octomap. The detected objects are converted to Octomap and the size of voxel is 0.01m. (a) is a keyboard. (b) is a monitor. (c) is a mouse. (d) is a cup. (e) shows all the objects.

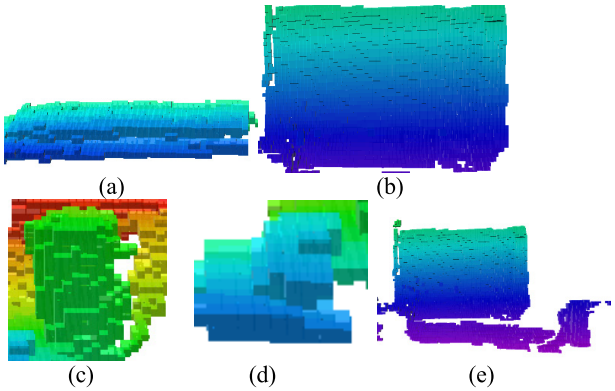

FIGURE 15. After getting semantic messages, we insert these semantic messages into Octomap. The detected objects are converted to Octomap and the size of voxel is 0.01m. (a) is a keyboard. (b) is a monitor. (c) is a mouse. (d) is a cup. (e) shows all the objects.

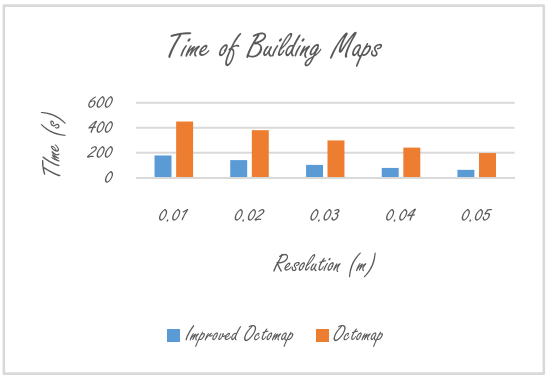

FIGURE 16. Time of building maps.

it takes more time to process keyframes. As a result, some keyframes generated by Tracking thread may be dropped.

In the scene with dynamic objects, our SLAM is much better than ORB-SLAM2. Fig.13 shows the trajectory estimated by our SLAM, obviously it is much closer to the ground truth in the dynamic scenes. The RMSE error of our SLAM is much smaller than ORB-SLAM2. There are two major reasons: first, our semantic SLAM retains features on static objects and removes features which may belong to dynamic objects for each keyframe. Second, because each frame is tracked by keyframe and it can provide more stable features, therefore the localization can be more accurate in the scene with dynamic objects.

D. DETECTION

In order to demonstrate the capabilities of our object-oriented semantic mapping system, we record some RGB-D sequences in indoor environments, and generate a full map of each environment, then compare the number of objects recorded in map for each class. The result is shown in Fig.14. As demonstrated, our system is able to identify the majority of objects in the scenes, but misses few objects. For example, there are two monitors in the desk scene. But it only detects one monitor (image c in Fig.14) because they are too close to each other. In some scenes, it identifies a monitor and a keyboard as a laptop because they are closely located in the scene. Besides, the result given by YOLO is in the form of bounding box, therefore each object contains some part of its background.

E. MAPPING

The detected objects are converted to Octomap and the result is shown in Fig.15.

In the localization experiments, TUM dataset is used to evaluate the accuracy. The TUM dataset contains the RGB and depth images. In the proposed semantic SLAM, we combine RGB and Depth images to generate point clouds, and get a position for each point clouds. Therefore, point clouds and their positions can be used to create a map through Octomap.

In order to compare the improved Octomap and its original Octomap, we use a sequence chosen from TUM dataset to build map with multi-threads in five different resolutions. Both of them use continuous 30 frames from a sequence to build map 5 times. Fig.16 shows the time of building map using improved Octomap and original Octomap. It can be seen that the improved Octomap is much faster than original Octomap.

As illustrated above, an important step in Octomap creation is computing empty voxels. As Fig.17 shows, the speed of empty voxel computation of improved Octomap is about twice less than that of original Octomap. The major reason is that we use a more effective algorithm to compute empty voxels and use threads pool to reduce thread creation, while the original Octomap uses OpenMP to create many threads to compute empty voxels.

Another important step in Octomap creation is inserting new voxels. After adding multi-threads to improve speed, the speed of new Octomap is about four times faster than the

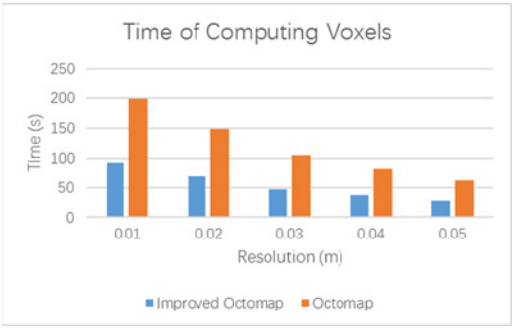

FIGURE 17. Time of computing empty voxels.

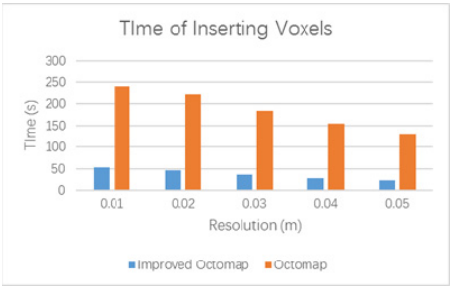

FIGURE 18. Time of inserting voxels.

original one, which is shown in Fig.18. The major reason is that original Octoamp only has one thread to update voxels, while improved Octomap has eight threads to update voxels which belong to the correspond child node.

VI. CONCLUSIONS AND FUTURE DIRECTION

In this paper, we propose a Semantic SLAM system which builds the semantic maps with object-level entities, and it is integrated into ORB-SLAM. In the proposed system, we modify ORB-SLAM2 in three ways to integrate the object detection: using YOLO to detect and recognize objects to generate semantic information, building the relationship between keyframes and objects to generate semantic maps and filtering feature points to improve the accuracy of SLAM with semantic information. Besides, we use a Fast Line Rasterization Algorithm and multi-threads to improve the efficiency of Octomap and accelerate the speed of mapping. Finally, we evaluate our Semantic SLAM from three different aspects. We demonstrate that our Semantic SLAM is more accurate than ORB-SLAM2 and RGB-D SLAM in dynamic environments and is faster in building maps with the improved Octomap. Moreover, we demonstrate the efficiency of object detection through quantitative evaluation on a real-world dataset recorded over an office.

In future work, we can improve our semantic SLAM system in the following ways.

1) We use YOLO to extract semantic messages in the form of Bounding Box from images. However, objects' models generated by Bounding Box contain too much background. Therefore, we need a Deep-Learning Method which can generate more accurate profiles.
2) We use semantic messages to filter features extracted from images. In order to make the utmost of these messages, we can consider semantic messages as landmark and improve the accuracy of localization by the graph-based optimization.
3) In Data Association, we perform a nearest neighbor search to compute the similarity between candidate objects and temporary objects, but this method is easy to be affected by the accuracy of camera localization. Therefore, we can extract FPFH or other features from temporary and candidate objects, and compute the number of match features between temporary and candidate objects, in order to reduce the effect of camera localization for data association.

REFERENCES

[1] K. Tateno, F. Tombari, and N. Navab, "When 2.5D is not enough: Simultaneous reconstruction, segmentation and recognition on dense SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 2295–2302.
[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3431–3440.
[3] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.
[4] W. Hess, D. Kohler, D. Andor, and H. Rapp, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2016, pp. 1271–1278.
[5] S. Izadi *et al.*, "KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. UIST*, 2011, pp. 559–568.
[6] F. Endres, J. Hess, J. Sturm, D. Cremers, W. Burgard, and N. Engelhard, "An evaluation of the RGB-D SLAM system," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2012, pp. 1691–1696.
[7] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE ACM Int. Symp. Mixed Augmented Reality*, Nov. 2007, pp. 1–10.
[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015.
[9] J. Engel, V. Koltun, and D. Cremers. (Oct. 2016). "Direct sparse odometry." [Online]. Available: https://arxiv.org/abs/1607.02565
[10] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.
[11] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, May/Jun. 2014, pp. 15–22.
[12] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2008, pp. 1–8.
[13] R. Girshick, J. Donahue, J. Malik, and T. Darrell, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 580–587.
[14] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1440–1448.
[15] J. Redmon and A. Farhadi. (Dec. 2016). "YOLO9000: Better, faster, stronger." [Online]. Available: https://arxiv.org/abs/1612.08242
[16] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2015, pp. 21–37.
[17] D. Gálvez-López, M. Salas, J. M. M. Montiel, and J. D. Tardós, "Real-time monocular object SLAM," *Robot. Auton. Syst.*, vol. 75, pp. 435–449, Jan. 2016.
[18] S. Pillai and J. Leonard. (Jun. 2015). "Monocular SLAM supported object recognition." [Online]. Available: https://arxiv.org/abs/1506.01732
[19] R. C. Anati, "Semantic localization and mapping in robot vision," Ph.D. dissertation, Dept. Comput. Inf. Sci., Univ. Pennsylvania, Philadelphia, PA, USA, 2016.

[20] J. Mccormac, A. Handa, S. Leutenegger, and A. Davison, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May/Jun. 2016, pp. 4628–4635.
[21] B. Douillard *et al.*, "Hybrid elevation maps: 3D surface models for segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2010, pp. 1532–1538.
[22] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2006, pp. 2276–2282.
[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robot.*, vol. 34, no. 3, pp. 189–206, 2013.
[24] L. Niu and Z. Shao, "A fast line rasterization algorithm based on pattern decomposition," *J. Comput.-Aided Des. Comput. Graph.*, vol. 22, no. 8, pp. 1286–1292, 2010.
[25] I. Biederman, R. J. Mezzanotte, and J. C. Rabinowitz, "Scene perception: Detecting and judging objects undergoing relational violations," *Cognitive Psychol.*, vol. 14, no. 2, pp. 143–177, 1982.
[26] X. He, R. S. Zemel, and M. A. Carreira-Perpinan, "Multiscale conditional random fields for image labeling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Washington, DC, USA, vol. 2, Jun./Jul. 2004, pp. II-695–II-702.
[27] S. Kumar and M. Hebert, "A hierarchical field framework for unified context-based classification," in *Proc. IEEE Conf. Comput. Vis. (ICCV)*, vol. 2, Oct. 2005, pp. 1284–1291.
[28] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *Proc. IEEE Conf. Comput. Vis. (ICCV)*, Rio De Janeiro, Brazil, Oct. 2007, pp. 1–8.
[29] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *Int. J. Comput. Vis.*, vol. 81, no. 1, pp. 2–23, Dec. 2007.
[30] P. Kohli, L. Ladický, and P. H. S. Torr, "Robust higher order potentials for enforcing label consistency," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Anchorage, AK, USA, May 2008, pp. 1–8.
[31] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *Proc. IEEE Conf. Comput. Vis. (ICCV)*, Sep./Oct. 2009, pp. 670–677.
[32] T. Toyoda and O. Hasegawa, "Random field model for integration of local information and global information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 8, pp. 1483–1489, Aug. 2008.
[33] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected CRFs with Gaussian edge potentials," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2011, pp. 109–117.
[34] N. D. Reddy, P. Singhal, and K. M. Krishna, "Semantic motion segmentation using dense CRF formulation," in *Proc. ICVGIP*, 2014, Art. no. 56.
[35] S. Ren, K. He, J. Sun, and R. Girshick, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.* Cambridge, MA, USA: MIT Press, 2015, pp. 91–99.
[36] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining Knowl. Discovery*, vol. 2, no. 2, pp. 121–167, 1998.
[37] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Apr. 2013.
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[39] R. Kümmerle, G. Grisetti, K. Konolige, W. Burgard, and H. Strasdat, "G^2o: A general framework for graph optimization," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2011, pp. 3607–3613.
[40] J. Stückler, B. Waldvogel, H. Schulz, and S. Behnke, "Dense real-time mapping of object-class semantics from RGB-D video," *J. Real-Time Image Process.*, vol. 10, no. 4, pp. 599–609, 2015.
[41] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2013, pp. 1–8.
[42] S. Liu, X. Cheng, Y. Zhou, Q. Li, and W. Fu, "Numeric characteristics of generalized M-set with its asymptote," *Appl. Math. Comput.*, vol. 243, pp. 767–774, Sep. 2014.
[43] Z.-H. Li, G. Liu, Z.-Y. Ji, and R. Zimmermann, "Towards cost-effective cloud downloading with tencent big data," *J. Comput. Sci. Technol.*, vol. 30, no. 6, pp. 1163–1174, Nov. 2015.
[44] X. Fan, H. Song, X. Fan, W. Wei, and J. Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden Markov for cloud computing," *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 78–89, Jan./Feb. 2016, doi: 10.1109/TSC.2016.2528246.
[45] W. Wei, H. Song, W. Li, P. Shen, and A. Vasilakos, "Gradient-driven parking navigation using a continuous information potential field based on wireless sensor network," *Inf. Sci.*, vol. 408, pp. 100–114, Oct. 2017, doi: 10.1016/j.ins.2017.04.042.
[46] W. Wei *et al.*, "GI/Geom/1 queue based on communication model for mesh networks," *Int. J. Commun. Syst.*, vol. 27, no. 11, pp. 3013–3029, Nov. 2014.
[47] W. Wei, Z. Sun, H. Song, H. Wang, X. Chen, and X. Fan, "Energy balance-based steerable arguments coverage method in WSNs," *IEEE Access*, vol. 6, pp. 33766–33773, 2017, doi: 10.1109/ACCESS.2017.2682845.
[48] W. Wei, H. Song, H. Wang, and X. Fan, "Research and simulation of queue management algorithms in ad hoc networks under DDoS attack," *IEEE Access*, vol. 5, pp. 27810–27817, 2017, doi: 10.1109/ACCESS.2017.2681684.
[49] W. Wei, X. Fan, H. Song, and H. Wang, "Video tamper detection based on multi-scale mutual information," *Multimedia Tools Appl.*, pp. 1–18, Sep. 2017, doi: 10.1007/s11042-017-5083-1.
[50] W. Wei and Y. Qi, "Information potential fields navigation in wireless ad-hoc sensor networks," *Sensors*, vol. 11, no. 5, pp. 4794–4807, 2011.
[51] W. Wei, X.-L. Yang, B. Zhou, J. Feng, and P.-Y. Shen, "Combined energy minimization for image reconstruction from few views," *Math. Problems Eng.*, Oct. 2012, Art. no. 154630.
[52] W. Wei, X. L. Yang, P. Y. Shen, and B. Zhou, "Holes detection in anisotropic sensornets: Topological methods," *Int. J. Distrib. Sensor Netw.*, vol. 8, no. 10, p. 135054, Oct. 2012.
[53] W. Wei, Y. Qiang, and J. Zhang, "A bijection between lattice-valued filters and lattice-valued congruences in residuated lattices," *Math. Problems Eng.*, Jul. 2013, Art. no. 908623.
[54] W. Wei, H. M. Srivastava, Y. Zhang, L. Wang, P. Shen, and J. Zhang, "A local fractional integral inequality on fractal space analogous to Anderson's inequality," *Abstract Appl. Anal.*, Jun. 2014, Art. no. 797561.
[55] Q. Ke, J. Zhang, H. Song, and Y. Wan, "Big data analytics enabled by feature extraction based on partial independence," *Neurocomputing*, vol. 288, pp. 3–10, May 2017, doi: 10.1016/j.neucom.2017.07.072.

LIANG ZHANG received the Ph.D. degree in instrument science and technology from Zhejiang University, China, in 2009. In 2009, he joined the School of Software, Xidian University, where he is currently an Associate Professor and the Director of the Embedded Technology and Vision Processing Research Center. He has published over 40 academic papers in peer-reviewed international journals and conferences. His research interests lie in the areas of multi-core embedded systems, computer vision, deep learning, simultaneous localization and mapping, human–robot interaction, and image processing.

LEQI WEI received the master's degree from the School of Computer Science and Technology from Xidian University, Xi'an, China, in 2018. His research interests lie in the areas of computer vision, SLAM, and image processing.

PEIYI SHEN received the Ph.D. degree from Xidian University in 1999 and the Ph.D. degree from MTRC, Department of Computer Science, University of Bath. He was a Research Officer with MTRC, Department of Computer Science, University of Bath, under the supervision of Prof. P. Willis, and a Research Fellow with CVSSP, University of Surrey, under the supervision of Prof. A. Hilton. He was with Agilent Technologies, USA, U.K., Malaysia, and Singapore, from 2000 to 2003. He was also a Post-Doctoral Research Fellow with the School of Computing, National University of Singapore, in 2000. He is currently a Professor with the National School of Software, Xidian University. His research interests are in computer vision, volume visualization, and its applications.

WEI WEI (SM'17) received the M.S. and Ph.D. degrees from Xi'an Jiaotong University in 2011 and 2005, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, Xi'an University of Technology, Xi'an, China. He ran many funded research projects as a principal investigator and a technical member. He has published around 100 research papers in international conferences and journals. His research interest is in the area of wireless networks, wireless sensor networks application, image processing, mobile computing, distributed computing, and pervasive computing, Internet of Things, sensor data clouds, and so on. He is a Senior Member of CCF. He is an Editorial Board Member of FGCS, AHSWN, IEICE, KSII, and so on. He is also a TPC member of many conferences and a regular Reviewer of IEEE TPDS, TIP, TMC, TWC, and many other Elsevier journals.

GUANGMING ZHU received the Ph.D. degree in instrument science and technology from Zhejiang University, China, in 2015. He is currently a Post-Doctoral Researcher with the School of Software, Xidian University. His major research fields are information fusion, human action/gesture recognition, scene recognition, and deep learning.

JUAN SONG received the B.S. degree from the School of Communication Engineering, Hohai University, Nanjing, China, in 2006, and the Ph.D. degree in communication and information system from Xidian University, Xi'an, China, in 2012. She is currently an Associated Professor with the National School of Software, Xidian University. She has published over 20 academic papers in peer-reviewed international journals and conferences. Her research interests include image processing and pattern recognition.