2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Daejeon Convention Center
October 9-14, 2016, Daejeon, Korea

Object Detection and Tracking in RGB-D SLAM via Hierarchical Feature Grouping

Esra Ataer-Cansizoglu and Yuichi Taguchi

***Abstract*— We present an object detection and tracking framework integrated into a simultaneous localization and mapping (SLAM) system using an RGB-D camera. We propose a compact representation of objects by grouping features hierarchically. Similar to a keyframe being a collection of features, an object is represented as a set of segments, where a segment is a subset of features in a frame. Just like keyframes, segments are registered with each other in a map, which we call an object map. We use the same SLAM procedure in both offline object scanning and online object detection modes. In the offline scanning mode, we scan an object using an RGB-D camera to generate an object map. In the online detection mode, a set of object maps for different objects is given, and the objects are detected via appearance-based matching between the segments in the current frame and in the object maps. In the case of a match, the object is localized with respect to the map being reconstructed by the SLAM system by a RANSAC registration. In the subsequent frames, the tracking is done by predicting the poses of the objects. We also incorporate constraints obtained from the objects into bundle adjustment to improve the object pose estimation accuracy as well as the SLAM reconstruction accuracy. We demonstrate our technique in an object picking scenario using a robot arm. Experimental results show that the system is able to detect and pick up objects successfully from different viewpoints and distances.**

I. Introduction

3D geometric and semantic representation is vital for autonomous systems that localize themselves and interact with the surrounding environment. It is hence important to detect and localize high level entities in a 3D map. In this paper, we present an object detection and localization framework integrated into a feature-based RGB-D SLAM system.

Our key contribution is representing objects based on hierarchical feature grouping, which is illustrated in Figure 1. Just like a keyframe being a collection of features, a subset of features in a frame defines a segment. Keyframe-based SLAM systems reconstruct a map containing keyframes registered with each other, which we call a SLAM map. Similarly, we group a set of segments registered with each other to generate a map corresponding to an object, which we call an object map. More specifically, in the first level of hierarchy features are grouped into segments. At the second level, segments are grouped into object maps. Since an instance of an object in a frame might contain multiple segments, the object map can contain multiple segments from a single frame as well. The object map provides a compact representation of the object observed under different viewpoint and illumination conditions.

Mitsubishi Electric Research Labs (MERL), Cambridge, MA 02139, USA
{cansizoglu,taguchi}@merl.com

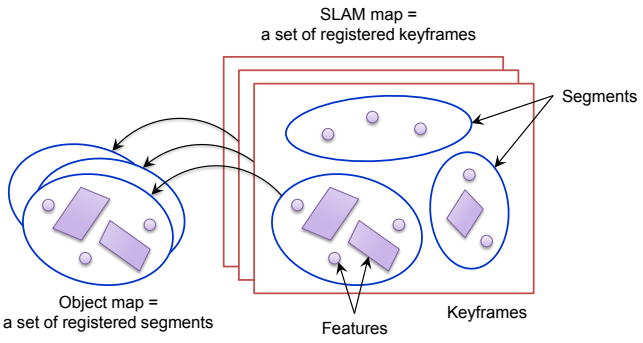

Fig. 1. The key idea of hierarchical feature grouping. A SLAM map typically stores a set of registered keyframes, each containing a set of features. We introduce another hierarchy based on segments to represent an object. A segment contains a subset of features in a keyframe, and an object map includes a set of registered segments. The object map is used for the object detection and pose estimation. In our system the segments are generated by using a depth-based segmentation algorithm and a sliding window approach.

Our system exploits the same SLAM procedure to handle the following two modes: the offline object scanning mode and the online object detection mode. Both of the modes are essential to achieve an object detection and tracking algorithm that can incorporate a given object instantly into the system. The goal of the offline object scanning mode is to generate an object map by collecting the appearance and geometry information of the object. We perform this process with user interaction: our system shows candidate segments that might correspond to the object to the user, and then the user selects the segments corresponding to the object in each keyframe that is registered with the SLAM system. In the online object detection mode, the system takes a set of object maps corresponding to different objects as the input, and then localizes these object maps with respect to the SLAM map that is generated in the online SLAM session.

Our system first detects and localizes objects in an input frame. We generate several segments in each frame using a depth-based segmentation algorithm and a simple sliding window approach. Each segment is matched to the segments in the object maps using VLAD [1] descriptors computed using the feature descriptors assigned to the segment. In the case of a match, a RANSAC registration is performed to localize the segment in the current frame with the object map. Segments with successful RANSAC registration initiate objects, which are stored in the SLAM map as object landmark candidates. The pose of each object landmark candidate is then refined by a prediction-based registration,

and if it is successful, the candidate becomes an object landmark. The list of object landmarks are then merged by looking at the refined poses, i.e., if two object landmarks correspond to the same object map and have similar poses, then they are merged. In the subsequent frames, we use the same prediction-based registration and merging processes to track the object landmarks. Consequently, an object landmark in the SLAM map serves as the representation of the object in the real world. Note that this procedure applies to both offline object scanning and online object detection modes: in the offline mode the object map is incrementally reconstructed using the segments specified in the previous keyframes, while in the online mode the object map is fixed.

We demonstrate the use of our method in an object picking scenario with a robot arm. We report analysis about the accuracy of object tracking with respect to the distance to the object. The results show that our system is capable of detecting and localizing objects from different viewpoints and distances.

A. Related Work

Object detection and pose estimation using 3D or RGB-D data have been widely studied in the robotics and computer vision communities, in particular recently thanks to the emergence of low-cost 3D sensors such as Kinect. Similar to 2D feature descriptors [2], [3] used for 2D image-based object detection, 3D feature descriptors that represent the local geometry have been defined for keypoints in 3D point clouds [4], [5], [6]. Simpler 3D features such as pair features [7] have been also used in voting-based frameworks [8], [9]. Those 3D feature-based approaches work well for objects with rich structure variations, but are not suitable for detecting objects with simple 3D shapes, such as boxes.

To handle simple as well as complex 3D shapes, RGB-D data have been exploited. Hinterstoisser et al. [10] defined multimodal templates for the detection of objects, while Drost and Ilic [11] defined multimodal pair features for the detection and pose estimation. Those approaches compute object poses in a single frame. On the other hand, our approach estimates them in a SLAM system using multiple frames and bundle adjustment, leading to better pose estimation accuracy.

Several systems have been proposed that incorporate object detection and pose estimation into a SLAM framework, similar to ours. Salas-Moreno et al. [12] detected objects from depth maps using [8] and incorporated them as landmarks in the map for bundle adjustment. Their method uses the 3D data only and thus requires rich surface variations for objects as described above. Fioraio et al. [13] proposed a semantic bundle adjustment approach for performing SLAM and object detection simultaneously. Based on a 3D model of the object, they generate a validation graph that contains the object-to-frame and frame-to-frame correspondences among 2D and 3D point features. Their method lacks a suitable framework for object representation, resulting in many outliers after correspondence search. Hence, the detection performance depends on bundle adjustment, which might become slower as the map grows.

II. Object Detection and Tracking via Hierarchical Feature Grouping

We build our object detection and tracking framework on the point-plane SLAM system [14], [15]. The original point-plane SLAM system [14] localizes each frame with respect to a growing map using both 3D points and 3D planes as primitives. Its extended version [15] further uses 2D points as primitives and computes 2D-to-2D, 2D-to-3D, and 3D-to-3D correspondences to exploit information in regions where the depth is not available (e.g., too close or far from the sensor). In this paper, our segments include 3D points and 3D planes (but not 2D points) as features similar to [14], while the SLAM procedure exploits all the 2D points, 3D points, and 3D planes as features to handle the case where the camera is close to the object and the depth information is not available. We use SIFT [2] detector and descriptor for 2D and 3D point measurements.

Figure 2 shows an overview of our system. Our system performs initial object detection in an input frame by the following four major steps: (i) segment generation, (ii) appearance similarity search and RANSAC registration between matching segments, (iii) object pose refinement via prediction-based object registration, and (iv) object merging. Once the object is detected and initiated as the object landmark in the SLAM map, we track the object using a prediction-based registration. Detected object landmarks are also used as constraints in our bundle adjustment procedure, which aims to globally optimize landmarks, keyframe poses, and object poses. Each of the steps is detailed in the following subsections.

A. Object Detection and Localization in the Frame

The goal of this step is to generate hypotheses about the objects existing in the frame and their poses. Hence, this step outputs a set of objects and poses, which will be refined in the subsequent stages.

For the initial detection of objects, first we need to know which subset of features refers to a particular object. Thus, we first generate a group of feature subsets, where each subset is considered as a segment. Each segment is then verified by appearance similarity search and RANSAC registration in the following steps. In other words, a segment is first matched against the set of segments from the object maps in terms of appearance. Second, matching segments are geometrically verified by RANSAC registration.

1) Segment Generation: We generate several segments using both a depth-based segmentation and a simple sliding window approach.

Depth-Based Segmentation: Our SLAM system extracts planes as features, which provide good segmentation in typical indoor scenes where planes are dominant structures. Thus we consider each of the extracted planes to generate a segment. Next, for the remaining non-planar regions of the frame, we apply a depth-based segmentation algorithm. This

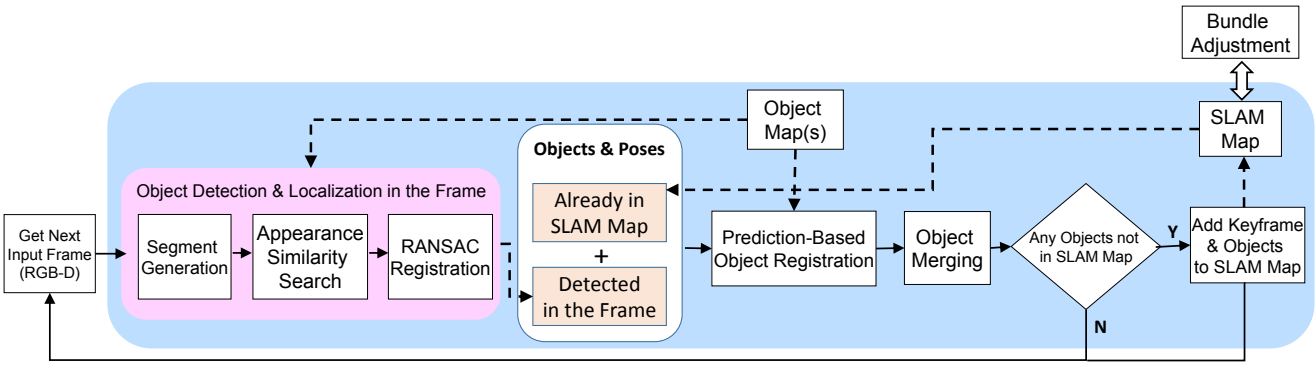

Fig. 2. System overview. For initial object detection, our system generates several segments in the current frame. For each segment, it then uses VLAD-based appearance similarity search followed by RANSAC registration to register the segment with respect to an object map. If the registration is successful, then the object poses are refined by using a prediction-based registration, which are used to merge segments corresponding to an identical object. The detected objects are finally added to the SLAM map as object landmarks. Once the object landmarks are initiated in the SLAM map, for the subsequent frames our system tracks their poses by using the prediction-based registration. The object landmarks are also used as constraints in the bundle adjustment procedure, which runs asynchronously using a separate thread. Dashed arrows represent the flow between data and processes, while solid arrows indicate the flow between processes.

method generates clusters by performing region growing on the remaining 3D point cloud. Based on the planes and the clusters of non-planar regions, we initiate segments using the set of features (3D points and 3D planes) that fall inside each region. We only use regions whose sizes are within a predefined range to omit too large or too small regions and speed up the object detection process.

Sliding Window Approach: Objects touching the table or another object might have been missed in the depth-based segmentation. Hence, we also exploit a sliding window approach in order to generate additional subsets of features. We consider the set of 3D point features that fall inside a $W_w \times W_h$ patch by sliding the patch over the image. The segments with too few features are discarded to avoid unnecessary computation.

In offline object scanning mode, we only use depth-based segmentation, since the aim is to track a single object in a simple scene. In online object detection mode, we utilize both depth-based segmentation and sliding window methods in order to come up with as many hypotheses as we can from a cluttered scene.

2) Appearance Similarity Search and RANSAC Registration: We aim to ensure that the segments are verified in terms of both appearance and geometry. In order to find the associated object map for each segment in the current frame, we first find k closest segments in the set of object maps by using VLAD descriptor matching [1]. Matches with a score above a threshold are returned, since there might be segments that do not belong to any object of interest. The segment in the frame is registered with the matching segment in the object map by using a RANSAC procedure. Namely, we perform all-to-all descriptor matching between the point features of the two segments followed by a RANSAC-based registration that also considers all possible plane correspondences. The segment that generates the largest number of inliers is recalled as the corresponding object. If RANSAC fails for all of the k matching segments of the object maps, the segment of the frame is discarded.

This step produces object landmark candidates. We consider them as *candidates*, because they are only registered with a single segment in the object map, not with the object map as a whole. An object might also correspond to multiple segments in a frame, resulting in repetitions in this list of object landmark candidates. Thus, we proceed with pose refinement and merging processes.

B. Object Pose Refinement by Prediction-Based Registration

An object map contains a set of segments that are acquired from different viewpoints. In this step, we register the entire object map with respect to the current frame by using the estimated pose of the object as a predicted pose.

We project all point and plane landmarks of the object map to the current frame based on the predicted pose of the object landmark candidate. Matches between point measurements of the current frame and point landmarks of the object map are computed. We ignore unnecessary matches based on two rules: (i) a point measurement is matched with a point landmark if the projected landmark falls within an r pixel neighborhood, and (ii) a point measurement is matched with a point landmark if the landmark was observed from a similar viewing angle when the object map was reconstructed. The first rule is to avoid unnecessary point pairs that are too far on the object and the second rule is to avoid performing matches for point landmarks that fall behind the object from the current viewing angle of the frame. Similarly, a plane landmark is considered in RANSAC if it is visible from the viewing angle of the frame. Note that the features of the object map are matched not only with the features included in the segment, but with all the features in the frame.

We use a RANSAC-based registration after finding the correspondences. If it succeeds, the candidate becomes an object landmark in the SLAM map with its refined pose and measurement-landmark associations of the point and plane features.

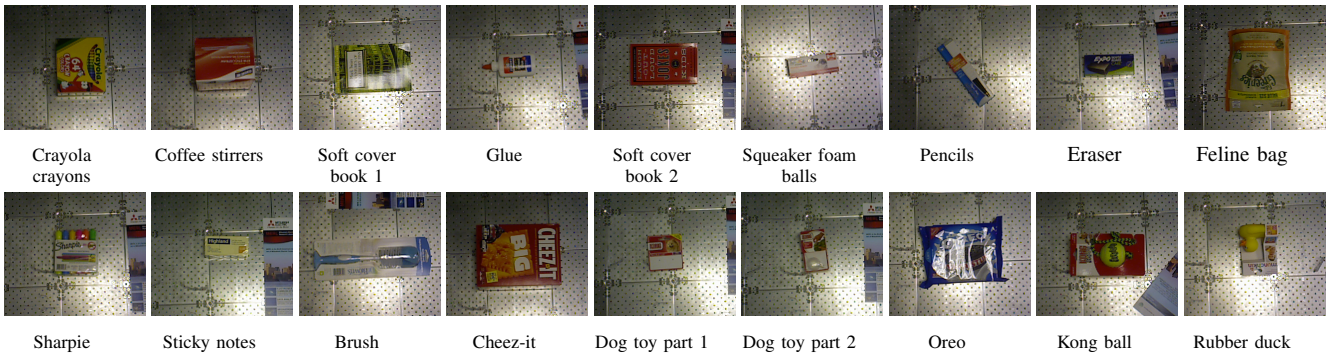

Fig. 3. List of objects used in our experiments.

C. Object Merging

The generated list of object landmarks might have repetitions, since we do not consider disjoint feature sets in segment generation and an object might refer to multiple segments in a frame. As a result, we merge the object landmarks that have similar poses, belonging to an identical object.

D. SLAM Map Update

Typically in the SLAM system, a frame is added to the SLAM map as a keyframe if its pose is different from the poses of any existing keyframes in the SLAM map. In addition to this condition, our system also adds a frame as a keyframe if it includes new object landmarks. This ensures the initialization of newly seen object landmarks and maintenance of the associations between frame measurements and object map landmarks.

E. Object Tracking

Once an object is detected and the object landmark is added to the SLAM map, it is tracked by the prediction-based registration. We use the pose of the object landmark and the pose of the current frame estimated by the SLAM system to predict the pose of the object map in the current frame. Then, we register the object map using the prediction-based registration method explained in Section II-B. If the tracking is successful and the current frame is selected as a new keyframe, then we update the associations of the object landmark in the SLAM map.

F. Bundle Adjustment

Let the triplet (k,l,m) denote an association between feature landmark $\mathbf{p}_l$ and feature measurement $\mathbf{p}_m^k$ of keyframe k with pose $\mathbf{T}_k$. Let I contain the triplets representing all such associations generated by the SLAM system in the current SLAM map. On the other hand, let the tuple (k,l,m,o) denote an object association such that object landmark o with pose $\tilde{\mathbf{T}}_o$ contains an association between the feature landmark $\mathbf{p}_l^o$ of the object map and feature measurement $\mathbf{p}_m^k$ in keyframe k. Let I_o contain the tuples representing such associations between the SLAM map and the object map.

The error that comes from the registration of the keyframes in the SLAM map is

$$E_{kf}(\mathbf{p}_1,\cdots,\mathbf{p}_L;\mathbf{T}_1,\cdots,\mathbf{T}_K)=\sum_{(k,l,m)\in I} d(\mathbf{p}_l,\mathbf{T}_k(\mathbf{p}_m^k)), \quad (1)$$

where $d(\cdot,\cdot)$ denotes the distance between a feature landmark and a feature measurement and $\mathbf{T}(\mathbf{f})$ denotes application of transformation $\mathbf{T}$ to the feature $\mathbf{f}$. The distance is defined for each of the different association types (3D-to-3D point associations, 2D-to-3D point associations, and 3D-to-3D plane associations) as described in [15].

The error that comes from object localization is

$$E_{obj}(\mathbf{T}_1,\cdots,\mathbf{T}_K;\tilde{\mathbf{T}}_1,\cdots,\tilde{\mathbf{T}}_O)=\sum_{(k,l,m,o)\in I_o} d(\mathbf{p}_l^o,\tilde{\mathbf{T}}_o\mathbf{T}_k(\mathbf{p}_m^k)). \quad (2)$$

Note that the object maps are fixed, i.e., $\mathbf{p}_l^o$ are fixed in each object map. This enables us to introduce more constraints to the bundle adjustment and compute better object poses.

The bundle adjustment procedure aims to minimize the total error with respect to the landmark parameters, keyframe poses, and object poses:

$$\begin{aligned}\underset{\forall\mathbf{T}_k,\tilde{\mathbf{T}}_o,\mathbf{p}_l}{\arg\min}\quad & E_{kf}(\mathbf{p}_1,\ldots,\mathbf{p}_L;\mathbf{T}_1,\ldots,\mathbf{T}_K)+\\ & E_{obj}(\mathbf{T}_1,\ldots,\mathbf{T}_K;\tilde{\mathbf{T}}_1,\ldots,\tilde{\mathbf{T}}_O). \quad (3)\end{aligned}$$

We used Ceres Solver [16] to perform the bundle adjustment.

III. Experiments

We used our method for object detection and localization during SLAM in indoor scenes. We also used our method to pick up objects with a robot arm. We report qualitative results for the SLAM in indoor scenes and detailed quantitative analysis for the robot system. Our SLAM system runs at 1.5 frames per second on a Windows 7 machine with 3.40 GHz Intel Core i7-2600 processor.

A. Objects Used in Experiments

We evaluated our method using 18 objects from Amazon Picking Challenge (APC) 2015 [17], [18]. We omitted seven APC objects that have few keypoints or that are not suitable to be picked with our gripper. Figure 3 shows the objects used in our experiments. Each image is a cropped version of the initial frame captured during the object scanning

Fig. 4. Example frames from the offline object scanning mode for `coffee stirrers` (top) and `brush` (bottom) objects.

phase with the robot arm. As can be seen articulated objects such as `squeaker foam balls` or `dog toy` are scanned partially using only their non-deformable parts. Note also that `dog toy` has green and yellow versions that are considered as different objects in APC. However, since their non-deformable parts are identical we used the same object model for detecting and localizing them. For each object or object part, we created two object maps for front and back sides.

B. Object Detection and Localization during SLAM

To generate object maps, we used a tablet-based platform consisting of an Asus Xtion sensor and a Surface tablet to scan the objects. Figure 4 shows some example frames used to generate the object maps for two example objects. Using a hand-held sensor provided flexibility to vary the viewpoint as much as we want in object map construction. Figure 5 depicts object detection and pose estimation results in an office sequence using the obtained object maps. Please see supplementary video for more results on different sequences.

C. Object Picking with Robot Arm

We applied the detection and tracking method to an object picking scenario using a robot arm. As shown in

Fig. 5. Object detection results in an office sequence: input frames (top), object localization results overlayed on depth map (bottom).

Figure 7 (left), we mounted an Asus Xtion sensor on a robot arm, which were calibrated with each other via hand-eye calibration.

The object maps were generated semi-automatically with the robot arm. We placed each object on the table and sampled 25 camera poses located on two circles of radius $r_1 = 100$ mm and $r_2 = 200$ mm where the sensor looked at the center of the table for each location. For the initial frame, a user clicked on the segments corresponding to the object. For the subsequent frames, the robot arm traveled on the sampled poses and the system grew the map by tracking the initially selected object.

1) Object Pose Estimation Accuracy: We first analyzed the accuracy of the pose estimation with respect to the distance to the object. Considering the picking scenario where the robot arm starts from a far location and approaches to the object, we started from 13 different poses in various positions and viewing angles. For each starting point, we approached one step (i.e., 1cm) to the object by centering it in the RGB frame. The pose of the object was updated at every step by adding the frame to the SLAM map. In each step, we recorded the pose of the object landmark and the distance between the object and the sensor. The distance was computed by considering the mean of all point landmark coordinates associated with the object. After the experiments, the reference object pose was computed as the mean of all estimations after removing outliers with medoid-shift clustering. Namely, we clustered all the poses and used the largest cluster as the data points to be used in reference pose computation. The experiments were carried out on four different objects with different structures.

Figure 6 shows the scatter plots for rotational and translational errors. Red line is the median of the error values in a 1 cm bin in terms of distance. Green dashed line indicates the distance where no 3D measurements were visible on the object. Our method is able to localize the object by tracking even when there are no 3D point and plane measurements extracted from the object, since we additionally use 2D point measurements. The results also show that the tech-

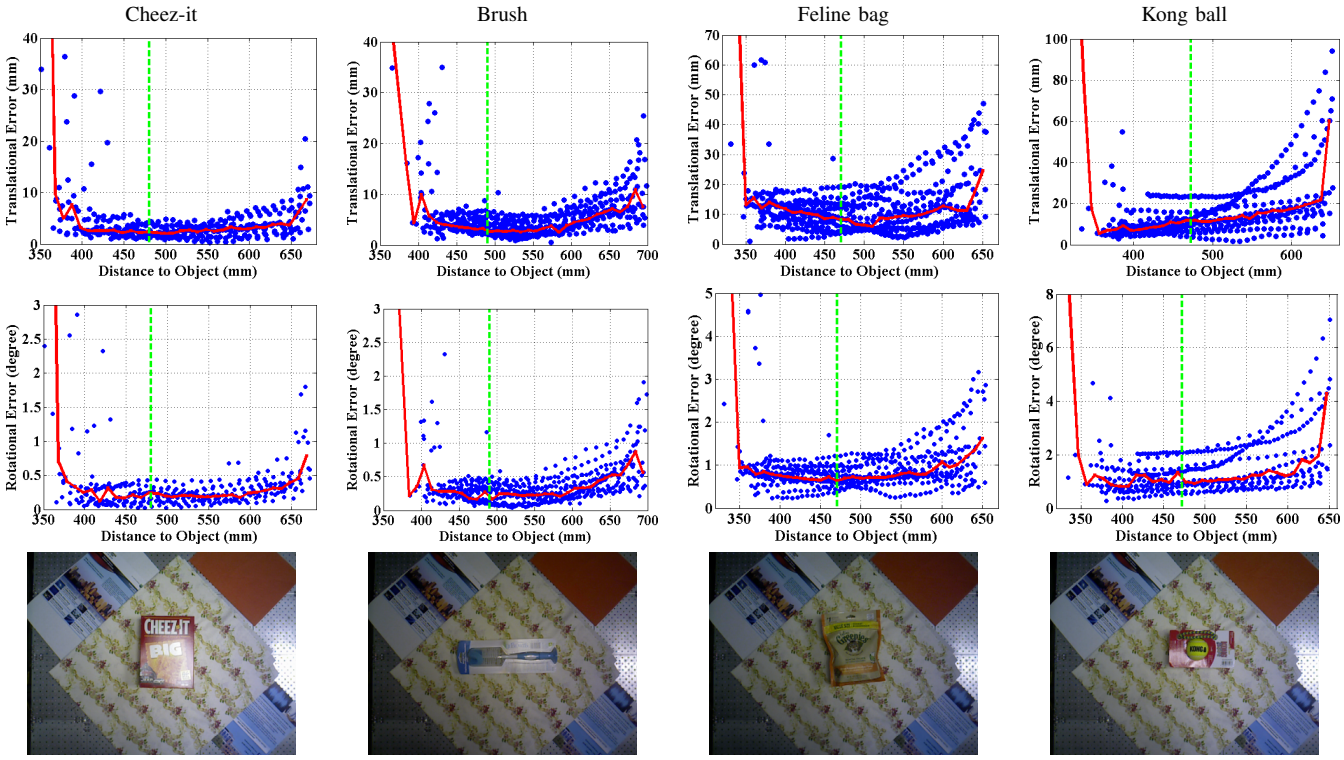

Fig. 6. Accuracy of the localization with respect to the distance to the object. Each blue point indicates an estimation. Red line is the median of the error values in a 1 cm bin in term of distance. Green dashed line indicates the distance where no 3D measurements were visible on the object.

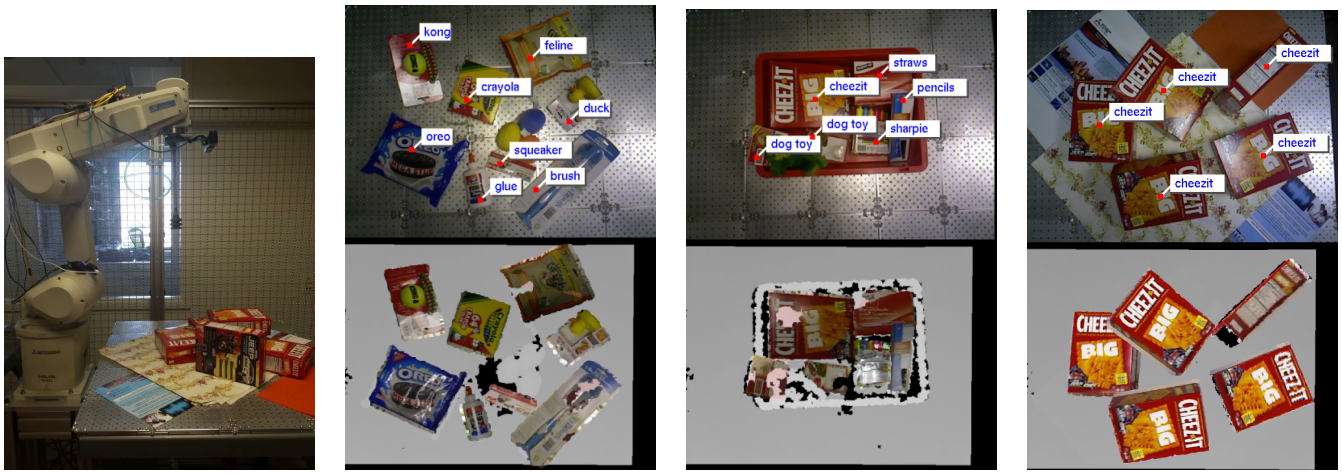

Fig. 7. Examples of object detection and localization results (right) using our robot platform (left): top panel shows the RGB frame, where red dots indicate gripping position and bottom panel shows localization results overlayed on the depth maps.

nique provides different accuracy ranges for different object structures. For example, planar objects such as `cheez-it` yield smaller errors compared to `feline bag` and `kong ball`, which have curvy surface structures.

2) Object Pickup Performance: For object picking experiments, we used a vacuum gripper that is specialized for uneven surfaces and has a suction power to carry a maximum weight of 0.8 pounds. For each object map, we annotated a region where the object can be gripped. Once the object is detected and localized, we transfer each point in the input gripping area based on the pose of the object, and fit local planes to a $w \times w$ patch, where $w = 25$ pixels in our experiments. If the local plane fitting succeeds, the robot arm goes to that position in the direction of local plane normal to pick up the object. With the use of depth information, we also avoid using occluded areas in the object as pickup locations. If multiple objects are detected and localized, we first pick up the object that has the maximum number of feature associations with its object map. Note that we avoid regions close to object boundary during the gripping region annotation.

For each experiment, we put K randomly selected objects

TABLE I

PICKUP SUCCESS RATIO PER NUMBER OF OBJECTS USED IN EXPERIMENTS.

# Objects (K)	Success Ratio	# Failed Attempts	# Misdetections
3	96.7%	2	1
5	92.0%	8	0
10	92.5%	15	2
15	89.0%	33	2
Total	**91.2%**	**58**	**5**

TABLE II

PICKUP SUCCESS RATIO FOR EACH OBJECT.

Objects	# Attempts	# Success	Success Ratio
Crayola crayons	44	44	100.0 %
Coffee stirrers	27	27	100.0 %
Book 1	45	44	97.8 %
Glue	43	42	97.7 %
Book 2	36	35	97.2 %
Squeaker foam balls	35	34	97.1 %
Pencils	32	31	96.9 %
Eraser	35	33	94.3 %
Feline bag	46	43	93.5 %
Sharpie	28	26	92.9 %
Sticky notes	40	36	90.0 %
Brush	28	25	89.3 %
Cheez-it	35	31	88.6 %
Dog toy (green)	43	38	88.4 %
Dog toy (yellow)	50	44	88.0 %
Oreo	30	25	83.3 %
Kong ball	39	31	79.5 %
Rubber duck	24	13	54.2 %

on the table. We carried out 20 experiments for each value of $K = \{3,5,10,15\}$. In each attempt, the robot arm moves to random positions until an object is detected in the scene. If no object is detected after the robot arm traveled 25 random positions, we call the attempt a pickup failure due to misdetection. We obtained 91.2% pickup success ratio over all the experiments. Table I reports the performance with respect to the number of objects in each experiment. Among all 660 pickup attempts, only 5 of them failed due to misdetection. Note that we never misclassified an object. As can be seen, the success ratio decreased as the number of objects on the table increased. Table II shows pickup success ratio per object.

Please refer to the supplementary video for the qualitative results. The detection results from three different runs can be seen in Figure 7. Our system is able to detect multiple objects on a table or in a cluttered environment such as in a box. Moreover, detecting multiple instances of the same object is also possible with the proposed object representation scheme. The gripping positions are displayed with red dots at the upper panel. The lower panel shows the localization results, where we overlay the segment from the object map that is the most similar to the pose of the object in the frame.

3) Discussion: Pickup failures mostly occurred due to inaccurate computation of the gripping position and direction. All misdetections occured when there was a single object on the table and the system was not able to detect any objects after a certain number of robot arm positions. The system failed because of misdetection in only 5 of the 58 failed attempts, which shows the competence of our object detection framework. Missed objects were the back side of objects that has very few keypoints, such as `eraser` and `sticky notes`. For the remaining failure cases, there were two major reasons: (i) the target object was touching another object or the table which resulted in a local plane with an inaccurate gripping location, and (ii) the target object was heavy and/or partially occluded by other objects such that during pickup the object slid from the gripper. Both factors can be avoided by working on system integration. The first type of failures can be avoided by choosing the gripping location as the best local plane centered at the interior points rather than the local planes centered at the boundary points. Moreover, strategies to further improve the accuracy of object map reconstruction and object localization can help more precise gripping position computation. For avoiding second type of failure cases, we can use different grippers based on the weight of the target object.

Gripping location computation affected the performance per object and performance per number of objects. For example, the `rubber duck` object has a package part that is small, hence the gripping location must have been very accurate for a successful pickup. On the other hand, planar objects in boxes were easier to pickup as they are mostly at the top rows of Table II. Our strategy of partially scanning non-deformable parts of the articulated objects worked well. For example, for `squeaker foam ball` and `dog toy` objects focusing on the region with the plush toys might have made the pickup impossible. However, for those objects our system achieved a performance of at least 88%. There were more occlusions when we used more objects in the experiments. Hence, the pickup success ratio decreased as the number of objects increased as reported in Table I.

IV. CONCLUSION AND FUTURE WORK

We presented an object detection and tracking framework that jointly runs with an RGB-D SLAM system. We introduced the concept of hierarchical feature grouping by using segments and represented an object as an object map consisting of a set of registered segments. Both the offline scanning and online detection modes were described in a single framework by exploiting the same SLAM procedure, enabling instant incorporation of a given object into the system. We applied our framework to object picking using a robot arm and showed successful object detection, pose estimation, and grasping results.

The proposed representation is compact. Namely, there is an analogy between keyframe-SLAM map and segment-object map pairs respectively. Both of them use the same features, in our case planes, 3D points, and 2D points that are extracted from input RGB-D data. Moreover, relocalization and tracking techniques are applicable to both concepts as explained in this paper. Thus, one can easily use different

sets of features or different relocalization/tracking techniques instead of ours.

A limitation of our method is on segment generation. The segments are generated based on a simple sliding window approach or depth-based segmentation. The first one generates too many segments that would contain redundancy, while the latter is too conservative as it can miss objects touching other surfaces. This can be improved with a better segmentation method which also takes into account texture information or other important features. It is important to note that we only make use of segments for initial detection of the objects. Thus, object detection module can be easily changed with any kind of object detection method.

In this study we did not focus on how incorporation of object constraints affects the registration accuracy in SLAM, as the scope of this paper was on object detection and tracking rather than improving SLAM accuracy via objects. However, the experiments showed that joint optimization of object poses and the SLAM map improves the object localization performance, since the localization becomes more accurate as we capture more frames. In the future, we plan to perform detailed analysis on joint bundle adjustment and its effect on object and keyframe localization. In addition, this analysis can consider different weights for different objects in optimization since object localization accuracy varies for objects with different structures.

Lastly, an important future extension of our work is on improving the registration accuracy in an object map. Offline pinpoint postprocessing [15] can be applied to the object map after scanning. This would improve the detection and tracking performance in online detection mode as the system relies on the generated object map.

ACKNOWLEDGMENTS

We would like to thank John Barnwell for his invaluable support in building the robot platform. We also thank Jay Thornton, Srikumar Ramalingam, Chen Feng, Alan Sullivan, and Yukiyasu Domae for fruitful discussion.

REFERENCES

[1] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sept. 2012.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int'l J. Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, June 2008.

[4] F. Stein and G. Medioni, "Structural indexing: Efficient 3-D object recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 125–145, Feb. 1992.

[5] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3D scenes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 5, pp. 433–449, May 1999.

[6] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (FPFH) for 3D registration," in *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, May 2009, pp. 3212–3217.

[7] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: A statistical 3D-shape representation for rapid classification," in *Proc. Int'l Conf. 3-D Digital Imaging and Modeling (3DIM)*, Oct. 2003, pp. 474–481.

[8] B. Drost, M. Ulrich, N. Navab, and S. Ilic, "Model globally, match locally: Efficient and robust 3D object recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 998–1005.

[9] C. Choi, Y. Taguchi, O. Tuzel, M.-Y. Liu, and S. Ramalingam, "Voting-based pose estimation for robotic assembly using a 3D sensor," in *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, May 2012, pp. 1724–1731.

[10] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, Nov. 2011, pp. 858–865.

[11] B. Drost and S. Ilic, "3D object detection and localization using multimodal point pair features," in *Proc. Int'l Conf. 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, Oct. 2012, pp. 9–16.

[12] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. J. Kelly, and A. J. Davison, "SLAM++: Simultaneous localisation and mapping at the level of objects," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[13] N. Fioraio and L. D. Stefano, "Joint detection, tracking and mapping by semantic bundle adjustment," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1538–1545.

[14] Y. Taguchi, Y.-D. Jian, S. Ramalingam, and C. Feng, "Point-plane SLAM for hand-held 3D sensors," in *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, May 2013, pp. 5182–5189.

[15] E. Ataer-Cansizoglu, Y. Taguchi, and S. Ramalingam, "Pinpoint SLAM: A hybrid of 2D and 3D simultaneous localization and mapping for RGB-D sensors," in *Proc. IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2016.

[16] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[17] "Amazon picking challenge," http://amazonpickingchallenge.org/.

[18] N. Correll, K. E. Bekris, D. Berenson, O. Brock, A. Causo, K. Hauser, K. Okada, A. Rodriguez, J. M. Romano, and P. R. Wurman, "Lessons from the amazon picking challenge," *ArXiv e-prints*, Jan. 2016.