Model-Instance Object Mapping

Joydeep Biswas(✉) and Manuela Veloso

School of Computer Science, Carnegie Mellon University, Pittsburgh, USA
{joydeepb,veloso}@ri.cmu.edu

Abstract. Robot localization and mapping algorithms commonly represent the world as a static map. In reality, human environments consist of many movable objects like doors, chairs and tables. Recognizing that such environment often have a large number of instances of a small number of types of objects, we propose an alternative approach, *Model-Instance Object Mapping* that reasons about the models of objects distinctly from their different instances. Observations classified as short-term features by Episodic non-Markov Localization are clustered to detect object instances. For each object instance, an occupancy grid is constructed, and compared to every other object instance to build a directed similarity graph. Common object models are discovered as strongly connected components of the graph, and their models as well as distribution of instances saved as the final Model-Instance Object Map. By keeping track of the poses of observed instances of object models, Model-Instance Object Maps learn the most probable locations for commonly observed object models. We present results of Model-Instance Object Mapping over the course of a month in our indoor office environment, and highlight the common object models thus learnt in an unsupervised manner.

1 Introduction

Robots deployed in human environments frequently encounter movable objects like tables and chairs. These movable objects pose a challenge for the long-term deployment of robots since they obstruct features of the map, and are difficult to map persistently. Areas where the majority of the observations of the robot are of unmapped objects are especially challenging, making it harder for a robot to accurately estimate its location globally.

There have been a number of long-term Simultaneous Localization and Mapping (SLAM) solutions proposed to tackle this problem by estimating the latest state of a changing world. While such approaches may be successful at tracking the changes of frequently observed environments over time, it may be infeasible for a robot to observe all the changes in all parts of a large deployment environment. In such an approach, if a robot were to infrequently visit a place with

This research was partially supported by the National Science Foundation award number NSF IIS-1012733, and by DARPA award number FA8750-12-2-0291. The views in this document are those of the authors only.

R.A.C. Bianchi et al. (Eds.): RoboCup 2014, LNAI 8992, pp. 525–536, 2015.
DOI: 10.1007/978-3-319-18615-3_43

many movable objects, it might fail to register the latest map to the older map due to large differences between them. Another approach is to maintain maps of the different observed states of the environment as local sub-maps. However, the set of possible states of an environment grows exponentially with the number of movable objects due to the exponential number of combinations of different poses of the movable objects.

In this paper, we propose an alternative approach to modelling a changing environment by separating the models of the movable objects from their distribution of poses in the environment. This proposed approach is borne of the realization that movable objects in human environments are often different instances of the same type of object. Furthermore, even though movable objects do move around, they still tend to be situated in roughly the same locations. For example, a work chair will most likely be observed in front of its accompanying work desk, even if the exact location of the chair might change over time. Our proposed approach, *Model-Instance Object Mapping*, models the objects independently of their instances in the environment. Each Model thus represents a unique *type* of object, while Instances represent the different *copies* of it that are observed. Observed short-term features, as classified by Episodic non-Markov Localization [1], are first clustered into separate point clouds for each object instance. An occupancy grid is built for each object instance, and are then compared in pairs to form a directed object similarity graph. Common object models are then detected by looking for strongly connected components in the graph.

2 Related Work

In a changing human environment, one common approach is to extend SLAM to maintain an up-to-date map of the environment when newer observations contradict the older map. Dynamic Pose Graph SLAM [2] is one such approach that extends pose graph SLAM. Dynamic Maps [3] extends SLAM and maintains estimates of the map over several timescales using recency weighted samples of the map at several timescales simultaneously.

An alternative to relying on an up-to-date map is to locally model the variations of a changing environment. The approach of "Temporary Maps" [4] models the effect of temporary objects by performing local SLAM, using the latest global map estimate as an initial estimate for the local map. Saarinen et al. [5] model a dynamic environment as an occupancy grid with associated independent Markov chains for every cell on the grid. Patch Maps [6] represents the different observed states of parts of the map and selects the one most similar to the robot's observation for localization.

There have been a few approaches to detecting movable objects in the environment. Recognizing that many objects in indoor human environments are of similar shapes, the approach of hierarchical object maps [7] assumes certain classes of shapes of objects that are matched to observed unmapped objects. The Robot Object Mapping Algorithm [8] detects moveable objects by detecting differences in the maps built by SLAM at different times. Detection and Tracking of

Moving Objects [9] is an approach that seeks to detect and track moving objects while performing SLAM. Relational Object Maps [10] reasons about spatial relationships between objects in a map. Bootstrap learning for object discovery [11] is similar to the "model" part of our work in that it builds models of unmapped objects, but does not reason about instances. Generalized Approach to Tracking Movable Objects (GATMO) [12] is an approach to tracking the movements of movable objects that is similar to our work in that it models movable objects. Our work however, further tracks every observed instance of the movable objects, thus allowing it to reason about the most likely poses of objects.

In contrast to the related work, our proposed approach, Model-Instance Object Mapping, is novel in its decoupling of the models of objects from the different instances of each model that are observed, and keeps track of every observed instance of the objects, to further reason about the most likely poses of objects.

3 Episodic non-Markov Localization

In a human environment, observations correspond to either immovable objects like walls, movable objects like chairs and tables, and moving objects like humans. Episodic non-Markov Localization (EnML) [1] classifies these different types of observations as Long-Term Features (LTFs), Short-Term Features (STFs) and Dynamic Features (DFs), respectively. LTFs correspond to features from a long-term map, while STFs are related to unmapped movable objects observed at different time steps. The correlations between STF observations across timesteps results in a non-Markovian algorithm. EnML represents the correlations between observations from different timesteps, the static map, and unmapped objects using a Varying Graphical Network (VGN). Figure 1 shows an example instance of a VGN.

EnML further limits the history of observations by partitioning the past observations into "episodes", where the belief of the robot's poses over the latest episode is independent of prior observations given the first pose of the robot in the episode. Episode boundaries commonly occur when the robot leaves one area for another, for example by going through a doorway, or turning a corner.

EnML thus estimates the robot's pose, as well as the pose of the unmapped STFs over each episode. EnML does not maintain a persistent history or database of the STFs that have been observed in the past. However, in a real human environment, the set of movable objects observable by a robot in a given area is finite, and the rest of this paper is devoted to building models of these STFs, and estimating their pose distributions in the world. EnML provides (aside from the poses of the robot) a set S of all the points corresponding to STFs observed in the world during the robot's deployment, and the corresponding poses P of the robot from where the points were observed. Figure 2 shows an example set S of STFs as detected by EnML.

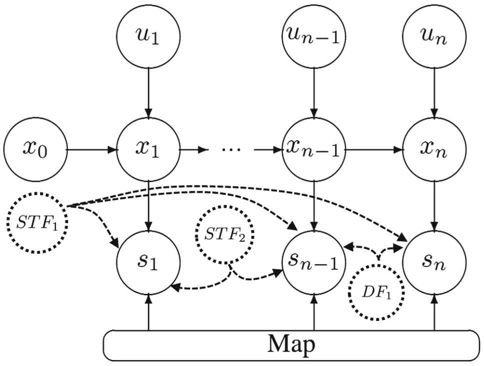

Fig. 1. An example instance of a Varying Graphical Network (VGN) for non-Markov localization. The non-varying nodes and edges are denoted with solid lines, and the varying nodes and edges with dashed lines. Due to the presence of short term features (STFs) and dynamic features (DFs), the structure is no longer periodic in nature. The exact structure of the graph will depend on the STFs and DFs present.

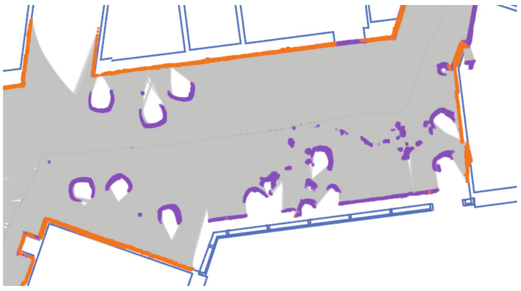

Fig. 2. Step 1 of Model-Instance Object Mapping: Observations S classified as Short-Term Features (purple) by Episodic non-Markov Localization are extracted (Colour figure online).

4 Finding Object Instances

The first step to building models of the objects is to cluster the observed points. The goal of this step is, given S (the set of points in global coordinates corresponding to STFs) and P (the set of poses of the robot from which points in S were observed), to form clusters $C = \{c_i\}_{i=1:n}$ where each cluster c_i is a non-overlapping subset of S, and every point in c_i is within a distance of ϵ of at least one other point in c_i. Here, ϵ is a configurable parameter that determines how close two objects can be, in order to be considered to be part of the same object. Note that each cluster c_i may contain points observed from different poses of the robot, as long as they are spatially separated by at most ϵ from other points in the cluster. We use the point cloud clustering Algorithm from [13] for this step, with a distance threshold of $\epsilon = 3$ cm. Figure 3 shows the clusters that were extracted from the example set S shown in Fig. 3.

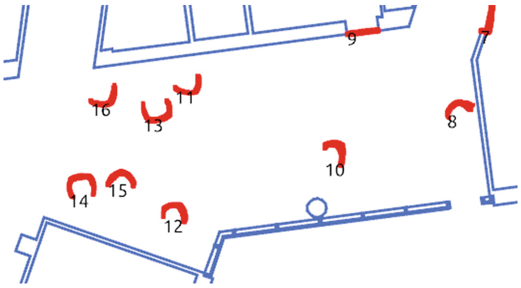

Fig. 3. Step 2 of Model-Instance Object Mapping: Observations S are clustered into set $C = \{c_i\}_{i=1:n}$. Each distinct cluster is denoted by its cluster index. Note that points in S that were sparsely distributed and not within distance ϵ of other points have been discarded.

For each cluster c_i, we build an occupancy grid [14] map m_i from the observed points to model the shape of the object. Algorithm 1 lists the algorithm to build an occupancy grid map m given cluster c, the associated set of poses x, and the width w and height h of the cluster. The grid map m and occupancy counts n are first initialized to zero matrices. For every pixel location l, for every observation that was made at that location, the occupancy value $m(l)$ and observation count $n(l)$ are both incremented. For every pixel location l that is observed to be vacant by observing a point beyond it, the observation count is incremented without incrementing the occupancy value. Finally, the occupancy value at every pixel location is normalized by the observation count for that pixel. Figure 4 shows three example occupancy grid maps constructed from the clusters in Fig. 3.

Algorithm 1. Build Object Model

```
 1: procedure BuildObjectModel($c, x, w, h$)
 2:     $m \leftarrow w \times h$ zero matrix
 3:     $n \leftarrow w \times h$ zero matrix
 4:     for each pixel $l$ in $m$ do
 5:         for each point $p$ in $c$, pose $o$ in $x$ do
 6:             if $p$ is observed at pixel $l$ then
 7:                 $m(l) \leftarrow m(l) + 1$
 8:                 $n(l) \leftarrow n(l) + 1$
 9:             else if $l$ is between $p$ and $o$ then
10:                 $n(l) \leftarrow n(l) + 1$
11:             end if
12:         end for
13:     end for
14:     for each pixel $l$ in $m$ do
15:         $m(l) \leftarrow m(l)/n(l)$
16:     end for
17:     return $m$
18: end procedure
```

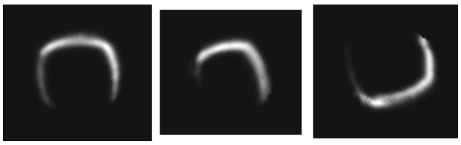

Fig. 4. Step 3 of Model-Instance Object Mapping: Each object instance is used to build an occupancy grid model, and three such example models are shown here.

5 Building Object Models

In human environments, and in particular in office environments, movable objects frequently occur as multiple instances of the same model. For example, in a common study area, there may be many chairs, but there will likely be either a single, or a small number of *types* of chairs. Hence, given the occupancy grid maps m_i for each object instance observed, there may be multiple instances of the same model of object, and the subject of the next section is on how to find these common object models.

As a robot encounters different instances of the same object model, it is likely to observe them at different locations and rotations. Furthermore, due to the presence of other objects, they may be partially occluded. Therefore, before comparing the models of different instances, we first align pairs of objects. For a given relative transform $T = \{\delta_x, \delta_y, \delta_\theta\}$ between the centroids of two object instances m_i and m_j, we define an alignment objective function F_{Align},

$$F_{\text{Align}}(m_i, m_j, T) = \sum_{l \in m_i} m_i(l) m_j(T \times l) \tag{1}$$

computed over all pixel locations l in m_i. Given this alignment object function, the optimal alignment T^*_{ij} between object instances m_i and m_j is defined as

$$T^*_{ij} = \arg\max_T F_{\text{Align}}(m_i, m_j, T). \tag{2}$$

This optimal alignment is found by exhaustive search over all possible relative transforms within a search window $\delta_x|| < \Delta_t, ||\delta_y|| < \Delta_t, ||\delta_\theta|| < \Delta_\theta$.

Given an optimal alignment T^*_{ij} between object instances m_i and m_j, we define a similarity metric F_{Similar} to compare the object instances:

$$F_{\text{Similar}}(m_i, m_j, T^*_{ij}) = \frac{\sum_{l \in m_i} I_\epsilon(m_i(l)) I_\epsilon(m_j(T^*_{ij} \times l))}{\sum_{l \in m_i} I_\epsilon(m_i(l))}. \tag{3}$$

Here, $I_\epsilon(\cdot)$ is the thresholded indicator function that returns 1 when the value passed to it is greater than the threshold ϵ. Note that F_{Similar} is not a symmetric metric : $F_{\text{Similar}}(m_i, m_j, T^*_{ij}) \neq F_{\text{Similar}}(m_j, m_i, T^*_{ji})$. This is because the similarity metric is computed over all pixel locations l in the first object instance m_i, and the two object instances need not necessarily be of the same size. Furthermore, the similarity metric returns values in the range $0 \leq F_{\text{Similar}} \leq 1$.

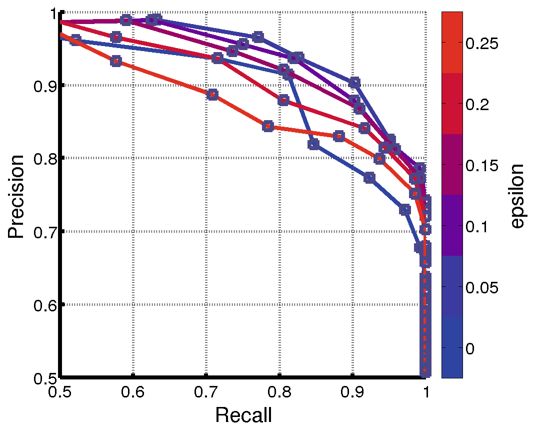

Fig. 5. Precision-recall curves for object similarity classification between pairs drawn from 25 objects, compared to hand-labeled classification.

Given two object instances m_i and m_j, and a similarity threshold Δ_{Similar}, m_i is classified as being similar to m_j if

$$F_{\text{Similar}}(m_i, m_j, T^*_{ij}) \geq \Delta_{\text{Similar}}. \tag{4}$$

Similarity comparisons between pairs of objects thus depend on two parameters, ϵ and Δ_{Similar}. ϵ serves as an outlier rejector, ignoring low occupancy values in the object instance models, while Δ_{Similar} serves as a confidence threshold.

To determine the values of ϵ and Δ_{Similar} to be used, we hand-labelled similarity comparisons between 25 object instances. Next, for several values of ϵ, we varied Δ_{Similar} to evaluate the precision and recall of the similarity classification. Figure 5 shows the precision-recall curves for the different values of ϵ. The values $\epsilon = 0.05$ and $\Delta_{\text{Similar}} = 0.7$ yield a precision as well as recall of 90 %, and we use these values for the subsequent sections.

Figure 6 shows sample results of alignment and similarity comparison between four objects. Each object is classified as being similar to itself as expected, while different instances of the same model (instances 1 and 2, and instances 3 and 4) are correctly aligned and classified as being similar. Objects 1 and 2 were doors, and objects 3 and 4 were two instances of the same type of chair observed by the robot. Note that object instances 1 and 2, as shown by their occupancy grids, were observed in different poses, but still aligned correctly with respect to each other.

The similarity metric F_{Similar} is used to build a directed similarity graph $G_{\text{Similar}} = \langle V, E \rangle$ where vertices $v_i \in V$ denote the object instances the corresponding object instances m_i, and directed edges $e_k \in E$ indicate similarities between object instances. The presence of an edge e_k: $v_i \rightarrow v_j$ denotes that object instance m_i, corresponding to vertex v_i, is similar to object v_j, as evaluated by Eq. 4. Given this directed similarity graph G, we find sets of object

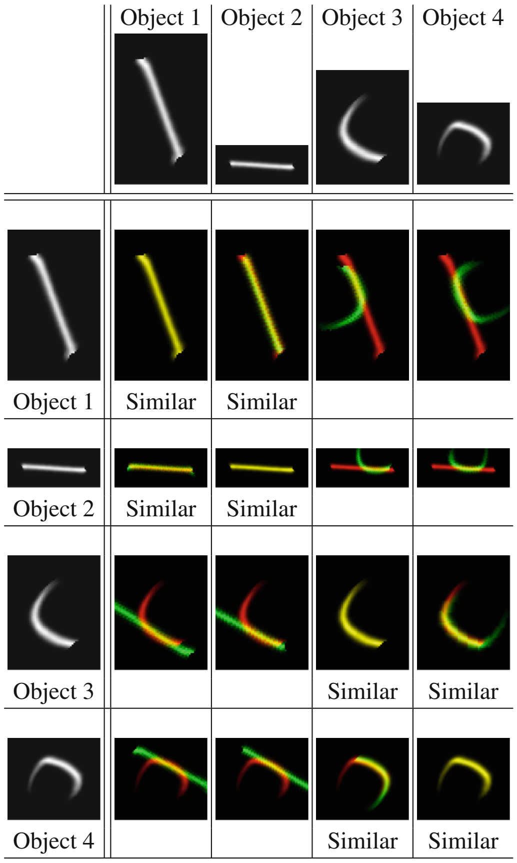

Fig. 6. Alignment and similarity comparison grid between four object instances. The top row and left-most column show the occupancy grids of the object instances. In each cell, the instance corresponding to the row is drawn in red, and the optimal alignment $T*$ of the instance corresponding to the column is drawn in green. Similarity matches are labelled in each cell (Colour figure online).

instances corresponding to the same object model by computing the strongly connected components of G. Let $S = \{s_i\}_{i=1:N_S}$ be the set of N_s strongly connected components extracted from graph G. Each strongly connected component s_i thus corresponds to a common model that is shared by the object instances represented by the vertices in s_i. Figure 7 shows an example similarity graph constructed by comparing the observed object instances from Fig. 3.

6 Updating the Model-Instance Map Across Deployments

So far we have focussed on how common models, detected as strongly connected components s_i, are extracted from the observations of STFs during a single

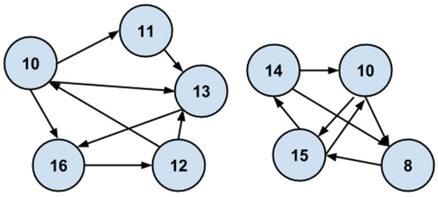

Fig. 7. Step 4 of Model-Instance Object Mapping: Every object instance is compared to every other object instance, and a similarity graph is constructed by connecting nodes (representing object instances) by directed edges to indicate similarity between them. Note that in this example, there are two strongly connected components, and hence the Model-Instance Object Mapping algorithm will find two common object models.

deployment log of the robot. However, as a lifelong learning algorithm, Model-Instance Object Mapping needs to reason about the persistence of object models and their instances across *all* deployments of the robot. In this section we present how models s_i from the latest deployment of the robot are merged with older persistent models p_j from all past deployments of the robot.

Persistent models p_j, just like models s_i, include a common occupancy grid model of the object, and a list of poses of the observed instances of the object. For the first deployment of the robot, the list of persistent objects is empty, and thus all models s_i are saved as unique persistent objects. For every subsequent deployment, every model s_i is aligned (Eq. 2) and compared to every persistent model p_j using the similarity metric F_{Similar} (Eq. 3). For each object s_i, every persistent object p_j that satisfies the similarity test of Eq. 4 bidirectionally is declared to be of the same model. All such sets of objects that are declared to be of the same model are then merged. Note that this merging step also attempts to merge persistent objects that were previously distinct, but which are determined to be bidirectionally similar to a newly observed object. This procedure for updating and merging persistent objects preserves the property that all instances of that object form a strongly connected component. Figure 8 shows the pose instances of an example persistent object, as extracted from the similarity graph of Fig. 7, which was constructed by comparing the object instances of Fig. 3.

7 Results

We have been deploying our robot, CoBot in an actual office environment since 2011, and have gathered extensive logs of the robot's deployments [15]. We processed a subset of the logs thus gathered between the January and April of 2014, in one of the floors in our building, through the Object-Instance Mapping algorithm. There were 133 such deployments in total, over the course of which the robot traversed a total distance of more than 55 km. We present here the resultant Model-Instance map thus learnt.

There were a total of 2526 objects observed, out of which there were 221 unique object models discovered. Out of the 221 unique object models, 50 had

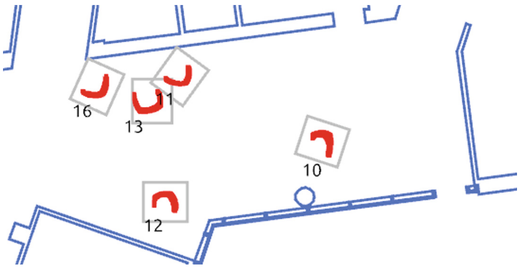

Fig. 8. Step 5 of Model-Instance Object Mapping: Poses for every instance of one example persistent objects are visualized here by duplicating the model of the persistent object at the poses of the instances on the map.

Table 1. Frequency counts of the unique object models discovered. The first row lists the range of the number of instances, and the second row lists the number of objects that were discovered within that range of instances.

Num Instances	1	2-5	6-10	11-20	21-40	41-80	81-160	161-1440
Num Objects	171	25	11	2	5	4	1	2

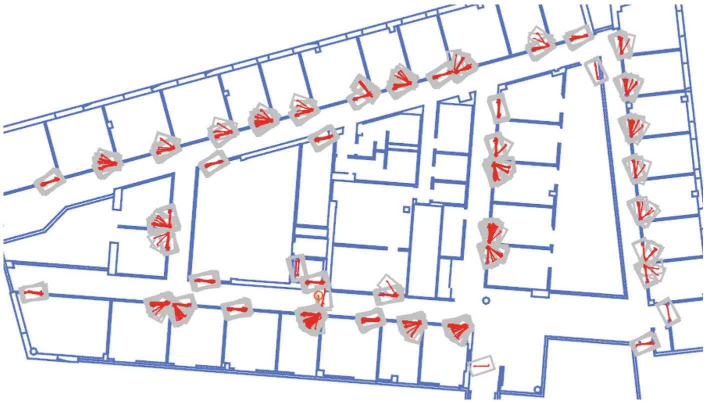

Fig. 9. The most commonly observed object, doors, as discovered by Model-Instance Object Mapping. Our algorithm correctly detected all the doors in the environment without any supervision.

2 or more observed instances. Table 1 lists the distribution of the frequency of the unique object models. The most commonly observed object model is that of doors, and the algorithm detected 1440 instances of them. Figure 9 shows the poses of the doors that were detected by Model-Instance Object Mapping without any supervision. The algorithm correctly detected all the doors in the environment, and also catalogued all the observed poses of the doors observed. From the figure, it is easy to discern the opening edges of the doors. Figure 10 shows some of the most commonly observed object models and their instances.

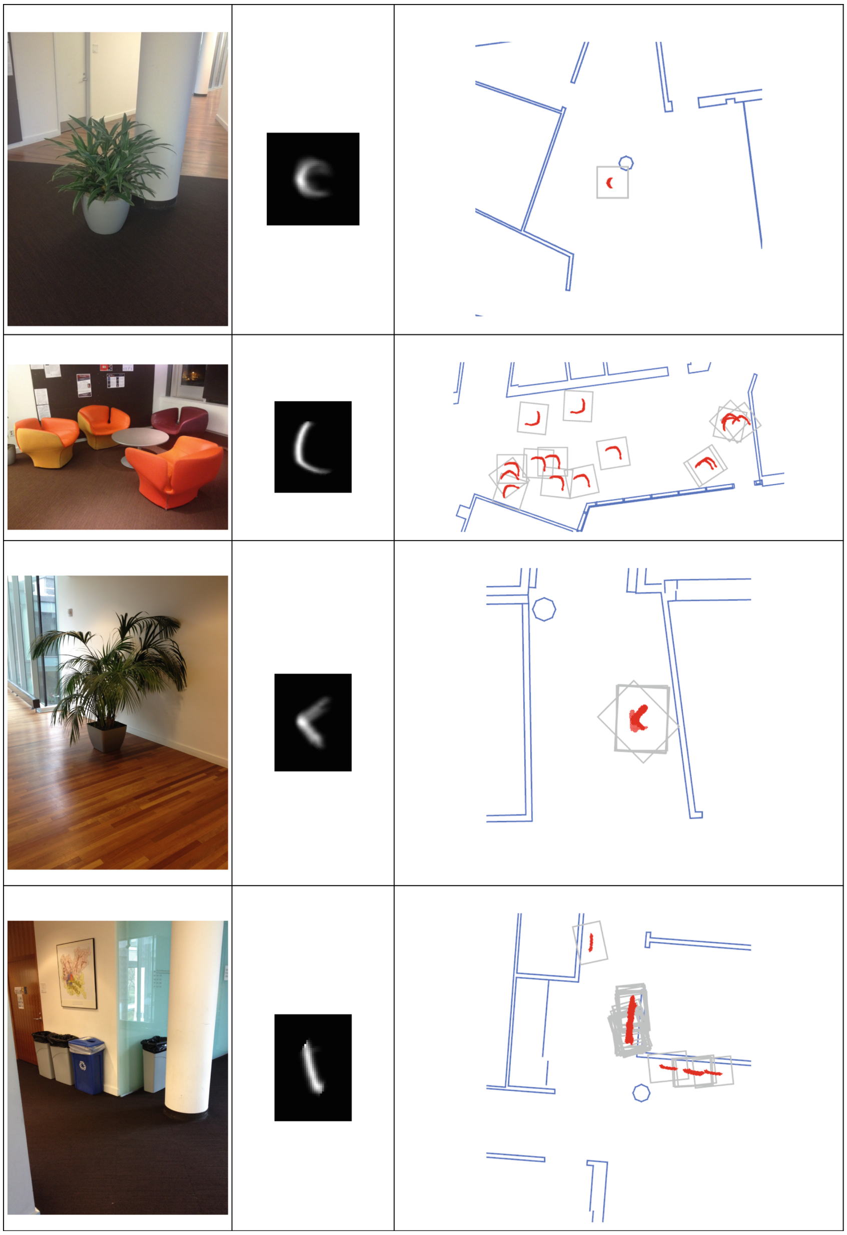

Fig. 10. Some examples of objects discovered by Model-Instance Object Mapping: (from top to bottom) a round planter, a commonly found chair, a square planter, and trashcans in the kitchen area. The first column shows a photograph of the object, the second column shows the occupancy grid model built, and the third column shows the instances that were observed for each object model, across all the deployments. Note that instances of these objects were detected at multiple locations, but only one location is shown for each object here.

8 Conclusion

In this paper we introduced the Model-Instance Object Mapping approach to separately estimate the model as well as instances of common recurring objects in human environments. This approach enables the robot to find common objects and track the poses of their instances in an unsupervised manner in order to represent the changing nature of the environment.

References

1. Biswas, J., Veloso, M.: Episodic Non-Markov localization: reasoning about short-term and long-term features. In: ICRA (2014)
2. Walcott-Bryant, A., Kaess, M., Johannsson, H., Leonard, J.: Dynamic pose graph slam: long-term mapping in low dynamic environments. In: IROS, pp. 1871–1878 (2012)
3. Biber, P., Duckett, T.: Dynamic maps for long-term operation of mobile service robots. In: RSS 2005, pp. 17–24 (2005)
4. Meyer-Delius, D., Hess, J., Grisetti, G., Burgard, W.: Temporary maps for robust localization in semi-static environments. In: IROS 2012, pp. 5750–5755 (2012)
5. Saarinen, J., Andreasson, H., Lilienthal, A.: Independent markov chain occupancy grid maps for representation of dynamic environment. In: IROS 2012, pp. 3489–3495 (2012)
6. Stachniss, C., Burgard, W.: Mobile robot mapping and localization in non-static environments. In: AAAI 2005, pp. 1324–1329 (2005)
7. Anguelov, D., Biswas, R., Koller, D., Limketkai, B., Thrun, S.: Learning hierarchical object maps of non-stationary environments with mobile robots. In: UAI 2002, pp. 10–17 (2002)
8. Biswas, R., Limketkai, B., Sanner, S., Thrun, S.: Towards object mapping in non-stationary environments with mobile robots. In: 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 1, pp. 1014–1019. IEEE (2002)
9. Wang, C.-C., Thorpe, C., Thrun, S., Hebert, M., Durrant-Whyte, H.: Simultaneous localization, mapping and moving object tracking. The Int. J. Rob. Res. **26**(9), 889–916 (2007)
10. Limketkai, B., Liao, L., Fox, D.: Relational object maps for mobile robots. In: IJCAI, pp. 1471–1476 (2005)
11. Modayil, J., Kuipers, B.: Bootstrap learning for object discovery. In: Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), vol. 1, pp. 742–747. IEEE (2004)
12. Gallagher, G., Srinivasa, S.S., Bagnell, J.A., Ferguson, D.: Gatmo: a generalized approach to tracking movable objects. In: IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 2043–2048. IEEE (2009)
13. Rusu, R.B.: Semantic 3d object maps for everyday manipulation in human living environments. KI-Künstliche Intelligenz **24**(4), 345–348 (2010)
14. Elfes, A.: Using occupancy grids for mobile robot perception and navigation. Computer **22**(6), 46–57 (1989)
15. Biswas, J., Veloso, M.M.: Localization and navigation of the cobots over long-term deployments. The Int. J. Rob. Res. **32**(14), 1679–1694 (2013)