Learning Object-Based State Estimators for Household Robots

Yilun Du[1] Tomas Lozano-Perez[1] Leslie Pack Kaelbling[1]

***Abstract*— A robot operating in a household makes observations of multiple objects as it moves around over the course of days or weeks. The objects may be moved by inhabitants, but not completely at random. The robot may be called upon later to retrieve objects and will need a long-term object-based memory in order to know how to find them. Existing work in *semantic* SLAM does not attempt to capture the dynamics of object movement. In this paper, we combine some aspects of classic techniques for data-association filtering with modern attention-based neural networks to construct object-based memory systems that operate on high-dimensional observations and hypotheses. We perform end-to-end learning on labeled observation trajectories to learn both the transition and observation models. We demonstrate the system's effectiveness in maintaining memory of dynamically changing objects in both simulated environment and real images, and demonstrate improvements over classical structured approaches as well as unstructured neural approaches. Additional information available at project website [1]: https://yilundu.github.io/obm/.**

I. Introduction

Consider a robot operating in a household, making observations of multiple objects as it moves around over the course of days or weeks. The objects may be moved by the inhabitants, even when the robot is not observing them, and we expect the robot to be able to find any of the objects when requested. We will call this type of problem *entity monitoring*. It occurs in many applications, but we are particularly motivated by robotics applications where the observations are very high dimensional, such as images or point clouds.

One version of this problem is addressed by *semantic* SLAM [2] methods, which focus on constructing a metric map of the occupied space, keeping the robot localized, and detecting static objects and marking their locations in the map. The SLAM aspects of these systems are excellent and we do not intend to replace or improve on them. However, these methods generally do not model objects that might move (and in fact, actively work to avoid adding them to the map) and also do not fuse information about an object that might be acquired from multiple observations that are considerably spaced in time. Another version of this problem, *object tracking*, removes the SLAM considerations, and addresses situations in which observations are closely spaced in time and objects only briefly go out of view.

This paper focuses instead on the problem of maintaining a memory of objects over a long time horizon, with multiple temporally and spatially distant observations of a single object and with objects that may be moved according to characteristic patterns over time. Solving such problems

[1]Computer Science and Artificial Intelligence Laboratory, MIT, USA, yilundu, tlp, lpk@mit.edu.

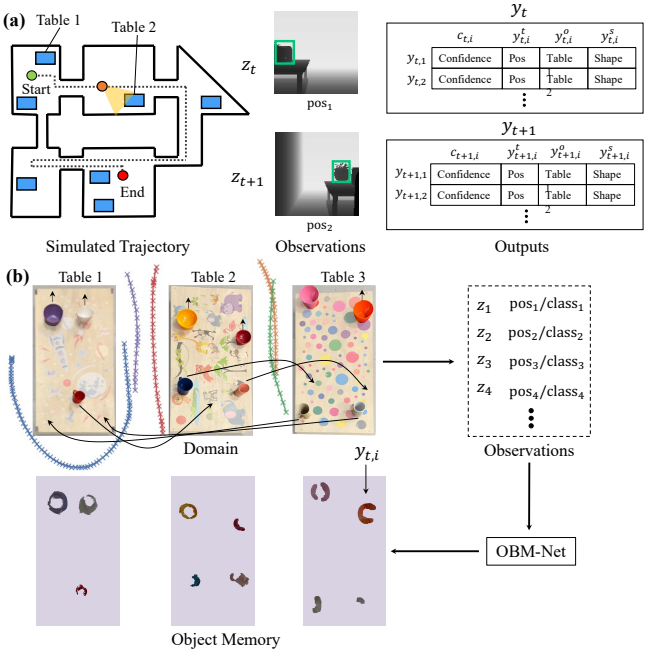


Fig. 1: **(a) Simulated Domains.** OBM-Net takes a set of input observations generated from a trajectory exploring a simulated house. At each timestep, OBM-Net receives an observation $\boldsymbol{z}_t$, consisting of a single segmented depth map input of an object and its corresponding 2D offset on a table. At each timestep, OBM-Net outputs a set of predicted objects $\boldsymbol{y}_t$ in the environment, where each object prediction $\boldsymbol{y}_{ti}$ consists of a predicted confidence of the object and its associated position, shape, and table which it is on (objects move with dynamics across tables). **(b) Real Images.** OBM-Net takes a set of input observations made by a robot driving around three different tables at 5 separate points in time, where individual objects on tables migrate over time (indicated with arrows). Input observations are processed into a set of partial pointcloud detections, with each input observation $\boldsymbol{z}_t$ to OBM-Net corresponding to the 2D position and class identity of a detected partial pointcloud. At each timestep, OBM-Net outputs a set of objects $\boldsymbol{y}_t$, which when reconstructed correspond to predicted objects and their locations on each table.

requires online *data association*, determining which individual objects generated each observation, and *state estimation*, aggregating the observations of each individual object to obtain a representation that is lower variance and more complete than any individual observation. This problem can be addressed by an online recursive *filtering* algorithm that receives a stream of object detections as input and generates, after each input observation, a set of hypotheses corresponding to the actual objects observed by the agent.

A classical solution to the entity monitoring problem, developed for the tracking case but extensible to other dynamic settings, is a *data association filter* (DAF) (the tutorial

of [3] provides a good introduction). A Bayes-optimal solution to this problem can be formulated, but it requires representing a number of possible hypotheses that grows exponentially with the number of observations. A much more practical, though less robust, approach is a maximum likelihood DAF (ML-DAF), which commits, on each step, to a maximum likelihood data association: the algorithm maintains a set of object hypotheses, one for each object (generally starting with the empty set) and for each observation it decides to either: (a) associate the observation with an existing object hypothesis and perform a Bayesian update on that hypothesis with the new data, (b) start a new object hypothesis based on this observation, or (c) discard the observation as noise. As the number of entities in the domain and the time between observations of the same entity increase, the problem becomes more difficult and the system can begin to play the role of the long-term object-based memory (OBM) for an autonomous agent.

The engineering approach to constructing such an OBM requires many design choices, including the specification of a latent state-space for object hypotheses, a model relating observations to object states, another model specifying the evolution of object states over time, and thresholds or other decision rules for choosing, for a new observation, whether to associate it with an existing hypothesis, use it to start a new hypothesis, or discard it. In any particular application, the engineer must tune all of these models and parameters to build an OBM that performs well. This is a time-consuming process that must be repeated for each new application.

In this paper, we develop a method for training neural networks to perform as OBMs for dynamic entity monitoring. *In particular, we train a system to construct a memory of the objects in the environment, without explicit models of the robot's sensors, the types of objects to be encountered, or the patterns in which they might move in the environment.* Although it is possible to train an unstructured recurrent neural network (RNN) to solve this problem, we find that building in some aspects of the structure of the OBM allows faster learning with less data and enables the system to address problems with a longer horizon. We describe a neural-network architecture that uses self-attention as a mechanism for data association, and demonstrate its effectiveness in different robotics domains as illustrated in Figure 1. We first illustrate its application on a simulated robotic domain, enabling the estimation of position, type and shape of objects across time. We further validate our approach on real indoor images gathered in a domain with moving objects using a Jackal mobile robot.

II. Related Work

Semantic mapping Our work is related to a large literature in robotics on *semantic* SLAM [4–9] (Kostavelis et al. [2] provide a good survey), which seeks to obtain a semantic labeling of places and objects in an environment. Typically, *semantic* SLAM operates on static environments, with recent work on dynamic environments [7, 9–11] focusing on obtaining a semantic labeling of only the static places and objects in an environment. In contrast, in this paper, we explore a new setting, in which we are interested in capturing instead, a semantic labeling of the dynamically moving objects in a scene. Such a task is important for household robotics, where objects will be displaced by the occupants of the house.

Learning Object Dynamics To operate a robot autonomously in a household environment, an underlying model of dynamics of objects in the house must be learned. Hawes et al. [12] constructs a architecture for such mobile robot autonomy, where a spatial-temporal clustering method [13] is used to extract dynamic objects from the surrounding environments. Tipaldi et al. [14] represents the dynamics of objects by utilizing dynamic occupancy grids while Kucner et al. [15] learning conditional probabilities of neighboring cells dynamic transitions in occupancy grid. Krajnik et al. [16] further explore model object dynamics utilizing periodic functions. In contrast, we propose to utilize a neural network in the form of DAF to construct a memory of objects in an environment, consisting of both associating objects across time, as well as constructing and learning their associated dynamics.

Data Association Our work is inspired by past work on data-association filtering. The most classic filter, for the case of a single entity, is the Kalman filter [17]. In the presence of data-association uncertainty the Kalman filter can be extended by considering assignments of observations to multiple existing hypotheses in a DAF or ML-DAF. These approaches, all of which require hand-tuned transition and observation models, are described by [3]. We show in supplementary results [1] that our learned approach can learn the underlying transition and observation models and performs comparably to ML-DAF with ground truth system dynamic and observation models on robotic domains.

A special case of the entity monitoring problem where observations are closely spaced in time has been extensively explored in the visual object tracking setting [18–20]. In these problems, there is typically a fixed visual field populated with many smoothly moving objects. This enables some specialized techniques that take advantage of the fact that the observations of each object are typically smoothly varying in space-time, and incorporate additional visual appearance cues. A related problem is *object re-identification* [21–23], which is focused on matching images of specific people and cars, without taking any other information into account.

In contrast, in our setting, there is no fixed spatial field for observations, they may combine a variety of modalities, and they may be temporally widely spaced, as would be the case when a robot moves through the rooms of a house, encountering and re-encountering different objects as it does so.

Algorithmic priors for neural networks Our approach is a instantiation of a method which integrates algorithmic structure with end-to-end neural network training. This approach has been applied to sequential decision making by [24], particle filters by [25], and Kalman filters by [26], as well as to a complex multi-module robot control system by [27]. The results generally are much more robust than completely hand-built models and much more sample-efficient

than completely unstructured deep-learning. We view our work as an instance of this general approach.

III. Problem formulation

We formalize the process of learning an object-based memory system (OBM). Formally, when the OBM is executed online, it receives a stream of input observations $z_1, \ldots z_T$ where $z_t \in \mathbb{R}^{d_z}$, and after each input z_t, it will output two vectors representing a set of predicted properties of hypothesized objects (*e.g.* position, type, and shape) $y_t = [y_{tk}]_{k\in(1..K)}$ and an associated confidence score for each hypothesis, $c_t = [c_{tk}]_{k\in(1..K)}$, where $y_{tk} \in \mathbb{R}^{d_y}$, $c_{tk} \in (0, 1)$. To ensure that confidences are bounded, we constrain $\sum_k c_{tk} = 1$. We limit the maximum number of hypothesis "slots" in advance to K. Dependent on the application, the z and y values may be in the same space with the same representation, but this is not necessary.

We have training data representing N different entity-monitoring problem instances,

$$\mathcal{D} = \{(z_t^{(i)}, m_t^{(i)})_{t\in(1..L_i)}\}_{i\in(1..N)} \quad ,$$

where each training example is an input/output sequence of length L_i, each element of which consists of a pair of input z and $m = \{m_j\}_{j\in(1..J_t^{(i)})}$, which is a set of nominal object hypotheses representing the true *current state* of objects that have actually been observed so far in the sequence. It will always be true that $m_t^{(i)} \subseteq m_{t+1}^{(i)}$ and $J_t^{(i)} \leq K$ because the set of objects seen so far is cumulative.

Our objective is to train a recurrent computational model to perform as an OBM effectively in problems that are drawn from the same distribution over latent domains as those in the training set. To do so, we formulate a model (described in section IV) with parameters θ, which transduces the input sequence $z_1, \ldots, z_L$ into an output sequence $(y_1, c_1), \ldots, (y_L, c_L)$, and train it to minimize the following loss function:

$$\mathcal{L}(\theta; \mathcal{D}) = \sum_{i=1}^{N}\sum_{t=1}^{L_i} \mathcal{L}_{\text{obj}}(y_t^{(i)}, m_t^{(i)}) + \mathcal{L}_{\text{slot}}(y_t^{(i)}, c_t^{(i)}, m_t^{(i)}) + \mathcal{L}_{\text{sparse}}(c_t^{(i)}) \quad .$$

The $\mathcal{L}_{\text{obj}}$ term is a *chamfer loss* [28], which looks for the predicted y_k that is closest to each actual m_j and sums their distances, making sure the model has found a good, high-confidence representation for each true object, with $\epsilon \ll 1$:

$$\mathcal{L}_{\text{obj}}(y, c, m) = \sum_j \min_k \frac{1}{c_k + \epsilon}\|y_k - m_j\| \quad .$$

The $\mathcal{L}_{\text{slot}}$ term is similar, but makes sure that each object the model has found is a true object, where we multiply by c_k to not penalize for predicted objects in which we have low confidence:

$$\mathcal{L}_{\text{slot}}(y, c, m) = \sum_k \min_j c_k\|y_k - m_j\| \quad .$$

Finally, the sparsity loss discourages the model from using multiple outputs to represent the same true object, by encouraging sparsity in object hypothesis confidences:

$$\mathcal{L}_{\text{sparse}}(c) = -\log\|c\| \quad .$$

Details in [1] illustrate how such a loss encourages sparsity among attention weights.

IV. OBM-Nets

Inspired by the the basic form of classic DAF algorithms and the ability of modern neural-network techniques to learn complex models, we have designed the OBM-Net architecture for learning OBMs and a customized procedure for training it from data, motivated by several design considerations. First, because object hypotheses must be available after each individual input and because observations will generally be too large and the problem too difficult to solve from scratch each time, the network will have the structure of a recursive filter, with new memory values computed on each observation and then fed back for the next. Second, because the loss function is *set based*, that is, it doesn't matter what order the object hypotheses are delivered in, our memory structure should also be permutation invariant and independent of the number of objects, and so the memory processing is in the style of an attention mechanism. Finally, in applications where the observations z may be in a representation not well suited for hypothesis representation and aggregation, the memory operates on a latent representation that is related to observations and output hypotheses via encoder and decoder modules.

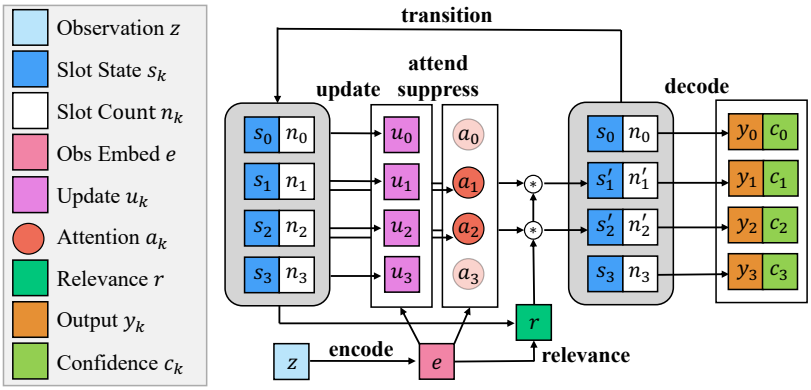
Figure 2 shows the architecture of the OBM-Net model. The memory of the system is stored in s, which consists of K elements, the K hypotheses in DAF, each in $\mathbb{R}^{d_s}$; the length-K vector n of positive values encodes how many observations have been assigned to each slot during the execution so far. New observations are combined with the memory state, and the state is updated to reflect the passage of time by a neural network constructed from seven modules with trainable weights.

When an observation z arrives, it is immediately **encode**d into a vector e in $\mathbb{R}^{d_s}$, which is fed into subsequent modules. First, **attention** weights w are computed for each hypothesis slot, using the encoded input and the existing content of that slot, representing the degree to which the current input "matches" the current value of each hypothesis in memory, mirroring the hypothesis matching procedure in DAFs. Since an observation typically matches only a limited number of hypotheses in DAFs, we force the network to commit to a sparse assignment of observations to object hypotheses while retaining the ability to effectively train with gradient descent, the **suppress** module sets all but the top M values in w to 0 and renormalizes, to obtain the vector a of M values that sum to 1:

$$w_k = \frac{\exp(\textbf{attend}(s_k, n_k, e))}{\sum_{j=0}^{n}\exp(\textbf{attend}(s_j, n_k, e))} \quad ; \quad a = \textbf{suppress}(w) \quad .$$

The a vectors are integrated to obtain n, which is normalized to obtain the output confidence c.

Mirroring hypothesis updates in DAFs, the **update** module also operates on the encoded input and the contents of each hypothesis slot, producing a hypothetical update of the hypothesis in that slot under the assumption that the current



Algorithm 1 OBM-Net execution.

Input: input observations $z_1, \ldots, z_T$, count $n \in \mathbb{R}^K$, state $s \in \mathbb{R}^{K \times D}$
for timestep $t = 1$ to T **do**
  $e \leftarrow$ **encode**(z_t)
  $r \leftarrow$ **relevance**(e, s, n)
  for slot $k = 1$ to K **do**
    $a_k \leftarrow$ **suppress**(**attend**(s_k, n_k, e))
    $u_k \leftarrow$ **update**(s_k, n_k, e)
    $s'_k \leftarrow (1 - ra_k)s_k + ra_k u_k$
    $n_k \leftarrow n_k + a_k$
    $y_k \leftarrow$ **decode**(s'_k)
    $c_k \leftarrow n_k / (\sum_i n_i)$
    $s_k \leftarrow$ **transition**(s'_k)
  end for
end for

Fig. 2: Architecture and pseudocode of OBM-Net. Observations are fed sequentially to OBM-Net, and encoded with respect to each hypothesis. A subset of the hypotheses are updated at each time-step, with corresponding slot counts incremented according to attention weight. Slots are then decoded, with the confidence of an output proportional to underlying slot count.

z is an observation of the object represented by that slot; so for all slots k,

$$u_k = \textbf{update}(s_k, n_k, e) \quad .$$

To enable the rejection of outlier observations, a scalar **relevance** value, $r \in (0, 1)$, is computed from s and e; this value modulates the degree to which slot values are updated, and gives the machine the ability to ignore or downweight an input. It is computed as

$$r = \textbf{relevance}(e, s, n) = \mathrm{NN}_2(\operatorname*{avg}_{k=1}^{K} \mathrm{NN}_1(e, s_k, n_k)) \quad ,$$

where NN_1 is a fully connected network with the same input and output dimensions and NN_2 is a fully connected network with a single sigmoid output unit. The attention output a and relevance r are now used to decide how to combine all possible slot-updates u with the old slot values s_t using the following fixed formula for each slot k:

$$s'_{t\,k} = (1 - ra_k)s_{t\,k} + ra_k u_k \quad .$$

Because most of the a_k values have been set to 0, this results in a sparse update which will ideally concentrate on a single slot to which this observation is being "assigned".

To obtain outputs, slot values s'_t are then **decoded** into the outputs, y, using a fully connected network:

$$y_k = \textbf{decode}(s'_{t\,k}) \quad .$$

Finally, to simulate transition updates in DAFs and to handle the setting in which object state evolves over time, we add a **transition** module, which computes the state s_{t+1} from the new slot values s'_t using an additional neural network:

$$s_{t+1\,k} = \textbf{transition}(s'_t)_k \quad .$$

These values are then fed back, recurrently, as inputs to the overall system.

Given a data set $\mathcal{D}$, we train the OBM-Net model end-to-end to minimize loss function $\mathcal{L}$, with a slight modification. We find that including the $\mathcal{L}_{\text{sparse}}$ term from the beginning of training results in poor learning, but adopting a training scheme in which the $\mathcal{L}_{\text{sparse}}$ is first omitted then reintroduced over training epochs, results in reliable training that is efficient in both time and data.

V. Empirical Results

We evaluate OBM-Net on several different *entity monitoring* tasks. Additional information in [1] provides results in a large number of simple, illustrative tasks, illustrating the ability of OBM-Net to learn to perform as a data-association filter in a variety of situations, some with observations in a very different space from than the desired output hypotheses. In this paper, we focus on dynamic entities in robotics domains. First, we evaluate the performance of OBM-Net on the complex simulated household robot domain shown in Figure 1(a), and then validate the ability of OBM-Net to capture an object with underlying dynamics and complex properties, as well as its utility for downstream robotics object-fetching tasks. Next, we evaluate the performance of OBM-Net on real images captured by a mobile robot in Figure 1(b).

A. Simulated Household Robot Domains

We first validate that OBM-Net can learn to solve the entity monitoring task in simulated robotic settings.

Setup. We model a robot moving within a house, as pictured in Figure 1(a), in the PyBullet simulation environment. In this house, each problem will involve following a trajectory consisting of a sequence of 50 locations. These locations are distributed across 5-6 separate rooms, with later time steps potentially revisiting earlier locations. At each location, the robot looks around and if there is a table within view (which happens about 50% of the time), it will get an observation of one of the objects on the table or an empty observation otherwise. Each new problem has 8 tables whose locations are drawn from a larger set of potential table locations and on each table there will be two objects drawn from a small set of classes, e.g. lamp, cushion, etc. Each object class has a characteristic stochastic movement pattern, with one object class sequentially teleporting between tables (details in web site). The goal is for the robot to be able to construct hypotheses for each distinct object it has seen and to be able to predict for each object the table it is currently on and its location relative to the table. More precisely, the input sequence of observations z corresponds to a segmented depth map of a single object visible given the camera pose at a

	Configuration A						Configuration B						Configuration C					
	Table Accuracy			Position Error			Table Accuracy			Position Error			Table Accuracy			Position Error		
Observations	10	25	50	10	25	50	10	25	50	10	25	50	10	25	50	10	25	50
Non Learned																		
Clustering	0.761	0.695	0.485	0.053	0.070	0.103	0.761	0.695	0.488	0.053	0.070	0.103	0.761	0.695	0.488	0.053	0.069	0.103
STRANDS	0.900	0.733	0.610	0.033	0.057	0.085	0.940	0.841	0.737	0.023	0.048	0.087	0.973	0.832	0.774	0.031	0.055	0.086
DS-SLAM [10]	0.953	0.769	0.641	0.020	0.045	0.080	0.924	0.858	0.775	0.021	0.047	0.083	0.971	0.898	0.776	0.032	0.053	0.083
DAF	0.959	0.807	0.670	0.022	0.043	0.081	0.937	0.871	0.787	0.021	0.047	0.084	0.974	0.914	0.803	0.030	0.053	0.083
Learned																		
Set Transformer	0.883	0.619	0.476	0.034	0.066	0.089	0.919	0.771	0.542	0.024	0.052	0.093	0.885	0.745	0.649	0.037	0.056	0.089
LSTM	0.839	0.661	0.406	0.058	0.093	0.126	0.875	0.716	0.514	0.053	0.094	0.123	0.892	0.717	0.519	0.052	0.091	0.130
OBM-Net	**0.984**	**0.926**	**0.809**	**0.019**	**0.041**	**0.078**	**0.989**	**0.924**	**0.795**	**0.021**	**0.046**	**0.082**	**0.988**	**0.932**	**0.873**	**0.027**	**0.052**	**0.080**

TABLE I: **Quantitative Analysis of OBM-Net on Simulated Household Domain.** Quantitative comparison of OBM-Net with baselines across 3 studied household domain configurations across 10, 25, 50 observations.

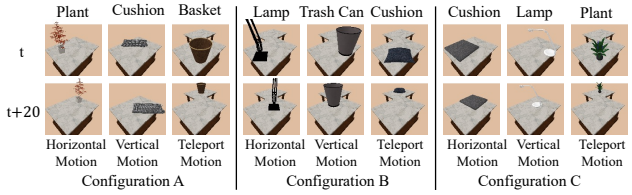


Fig. 3: **Simulated Object Dynamics.** Illustration of object dynamics in simulation across each class of objects in each of the 3 configurations. Images illustrate 20 timesteps of motion.

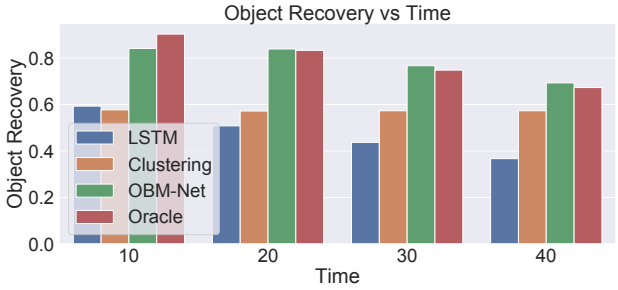


Fig. 4: **Object Recovery over Time.** Percentage of objects correctly recovered as a function of timesteps since seeing the object last. OBM-Net performs similarly to an oracle with ground truth dynamics.

particular location in the trajectory (or an empty observation in the case no object is visible), as well as which table it is resting on and its positional offset relative to the table. The desired output y values are, for each distinct object seen so far, the predicted table y^t it is on currently as well as its associated offset relative to the predicted table, y^o.

We train on a total of 10000 randomly sampled trajectories in the same floor plan, but with new randomly drawn object instances and tables for each trajectory. We test using 1000 trajectories, with test object meshes drawn from a set *disjoint from* the set of object meshes used during training (but sharing the same semantic classes). To test the flexibility of the approach, we consider three different configurations of object classes on tables (each object class contains roughly 20 meshes, with 50% of meshes used for training and the other 50% used for testing).

- Configuration A: Plants (which move horizontally across a table), cushions (which move vertically across a table), and baskets (which move diagonally in a table, but also teleport sequentially to different tables).
- Configuration B: Table Lamps (which move horizontally), Trash Cans (which move vertically) and cushions (which move diagonally across a table, but also sequentially teleport between tables).
- Configuration C: Cushions (which move horizontally), Table Lamps (which move vertically) and plants (which move diagonally across a table, but also sequentially teleport between tables).

We illustrate the underlying dynamics of objects in each configuration in Figure 3.

Crucially, none of the category shape information or dynamics is built in: the observation and transition models as well as the desired output representation are learned entirely from data.

Metrics. To test the efficacy of our approach, we measure to what extent each hypothesis slot m_i can recover both the table that the associated object is on, as well as the object's position relative to the table. We match a hypothesis slot k with each object label y_i by computing $\arg\min_k \|y_i^o - m_k^o\| + \text{Loss}_{\text{CE}}(y_i^t - m_k^t)$. For each match, we report the accuracy of m_k^t matching y_i^t, and as well the mean absolute error between y_i^o and m_k^o. When the table prediction for y_i is incorrect, we set mean absolute error to be equal to half the table size (0.15), as reported table offsets are meaningless in that case. In this setting, both OBM-Net and associated baselines use 10 hypothesis slots.

Baselines. We first compare OBM-Net to online learned baselines of LSTM [29] and Set Transformer [30] where all learned network architectures are structured to use ~ 50000 parameters. We further compare to task-specific baselines for this problem (detail at our website). First, given a localized input-segmented depth map, we extract object offsets by averaging all points in the point cloud associated with each segment. To associate objects dynamically across time, we use batch K-means clustering on the inferred object candidate offsets and associated table identities to obtain a set of objects. We further compare OBM-Net with the more complex spatial-temporal clustering method used in the STRANDS project [12] to infer objects in a real robotic setup from our underlying segmented depth maps, as well as a hand-crafted DAF system using ground truth dynamics. Finally, we compare our approach with *semantic* SLAM approach, DS-SLAM [10] for segmenting dynamically changing objects. For all learned models, we convert the segmented depth maps

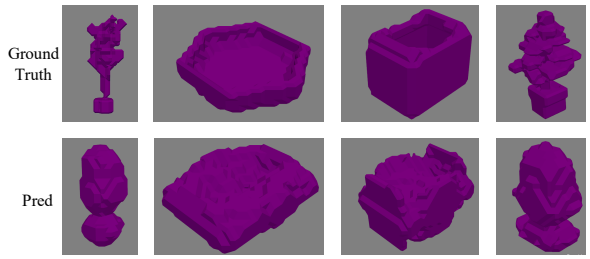


Fig. 5: **3D Shape Reconstruction.** Illustration of predicted 3D shape reconstructions using OBM-Net on unseen shapes at test time, compared to corresponding ground truth shape. OBM-Net is able to capture the coarse detail of individual shapes.

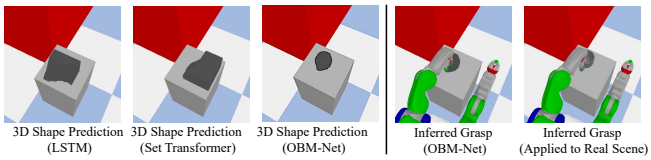


Fig. 6: **(Left) 3D Reconstructions.** Illustration of 3D reconstructions of hypothesis from each model. OBM-Net obtains accurate 3D reconstructions. **(Right) Estimated Grasps.** We utilize the predicted 3D mesh from OBM-Net to infer a grasp which successfully enables the grasp of a real object in the ground truth scene.

into downsampled 3D pointclouds.

Results. Table I shows that OBM-Net outperforms the baselines in both estimating the supporting tables and regressing the relative position of the objects across different numbers of observations. Figure 4 shows the prediction error of all methods as a function of the number of steps since the robot last saw an object; observe that OBM-Net is substantially better at long-term memory than the LSTM and Set Transformer, and still outperforms the clustering and STRANDS baselines even with long inter-observation gaps. As an upper bound, we compare with an oracle model, which knows ground truth object identity and dynamics (ignoring object collision). We find that OBM-Net performs similarly to the oracle model (performance across all models drops due to stochasticity), and in some cases does better, perhaps by modeling object collisions (not modeled in the oracle model).

Shape reconstruction. By adding a shape occupancy prediction head [31] to OBM-Net, we can also regress the underlying 3D shapes of our objects. We predict each shape at $32 \times 32 \times 32$ resolution, decoding each occupancy at each voxel coordinate using a MLP head conditioned on a hypothesis state. Quantitatively, we find that our approach gets 95.33% accuracy compared to 72.74% accuracy obtained by a LSTM and 73.67% obtained by a Set Transformer when predicting voxels for each test mesh in the test set. We provide visualization of predicted shapes from OBM-Net in Figure 5.

Downstream Tasks. Finally, we verify that object hypotheses from OBM-Net can usefully support a task in which a robot has to retrieve an object it has previously observed. First, we consider the task of finding a previously-encountered object. We train LSTM, Set Transformer, and OBM-Net to predict underlying object class y^c for each object hypothesis, as well as shape estimate and location. Given a desired object class (for example, either a plant, cushion, or bucket in configuration A) we wish to find, the robot examines

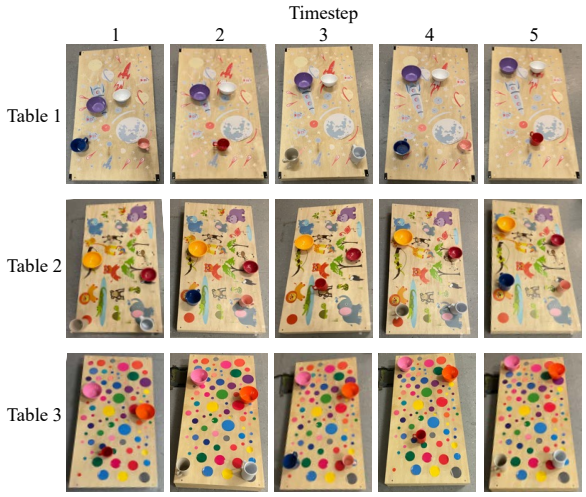


Fig. 7: **Real Object Dynamics.** Illustration of object dynamics across tables across each of 5 different timesteps. Bowls displace vertically on a table while mugs teleport between adjacent tables.

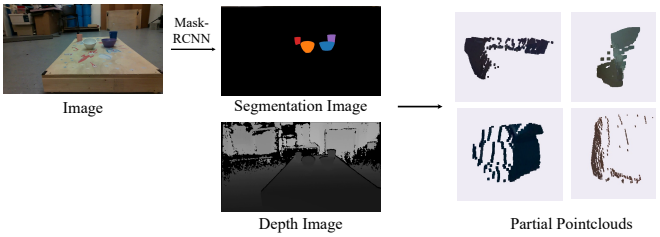


Fig. 8: **Data Processing.** An input image is fed through a Mask-RCNN model to obtain a semantic segmentation of bowls and mugs. The resultant segmentation is projected with the observed depth image to obtain partial pointclouds of objects.

each prediction (y_i, c_i) and navigates in the simulated world to look for an object of the specified class, based on predictions of y_i^t and y_i^o. It first goes to the most confident location of an object of that class, then if it does not locate the object there, it goes to the next most confident, and so on. We measure the number of predictions that need to be queried to find the object, as well as the percentage of trials in which the robot succeeded within 10 attempts. On this task, we find that a LSTM obtains an overall planning success of 68.75% with an average number of 5.38 hypotheses investigated before finding an object. In contrast, the Set Transformer obtains a planning success of 81.25% with on average 4.88 attempts. We find that OBM-Net performs best and is able to find the object of the desired class 100% of the time, with an average of 2.03 hypotheses examined before finding the object.

Next, we qualitatively analyze the 3D reconstructions of each object hypothesis with respect to its ability to support long-horizon manipulation planning. Although final execution of the grasp of an object can often be done based on a partial point-cloud from a single view, it is useful to be able to make predictions about the 3D reconstruction. These can be used for shape-completion of the current view as well as for planning an overall approach to manipulating an object before the object is in direct view of the robot. To test the functional utility of the 3D reconstructions, we compute grasps on the underlying shape by looking for parallel planar

Fig. 9: **Real 3D Shape Reconstructions (side).** Illustration of a side view of reconstructed objects from OBM-Net compared to baselines. OBM-Net is able to more accurately reconstruct the shape of objects, aggregating past observations at both the current table and other tables.

surfaces large enough to accommodate the gripper. We then try to execute that grasp on the target 3D object we wish to grasp in the (simulated) real world. As illustrated in Figure 6, we find that the 3D reconstruction of object hypotheses from OBM-Net is accurate enough to enable grasping of a real 3D shape. In contrast, predictions from LSTM and Set Transformer baselines are significantly poorer and do not enable downstream manipulation.

B. Real Robot Domain

We next validate that OBM-Net can solve the dynamic entity monitoring task on real RGB-D video captured by a LiDAR Camera L515 mounted on a mobile robot. Please see the attached supplemental video for captured footage of our approach as it is running, as well as reconstructed objects obtained by OBM-Net.

Setup. We capture images from the robot moving between three separate tables across five separate points in time. Each observation point consists of a trajectory, during which local observation information is fused. At the end of each trajectory, called a time-step in the following, a set of observations are presented to the OBM-Net.

We illustrate the trajectories and placement of tables at each time-step in Figure 1(b) (blue = time step 1, red = time step 2, green = time step 3, yellow = time step 4, purple = time step 5), with robot looking at table 1 in time steps 1 and 5, table 2 on time steps 2 and 4, and table 3 on time step 3. At each time step, bowls are displaced vertically on their table, while mugs are moved between adjacent tables (illustrated in Figure 7). In this setting, the input sequence of observations z to OBM-Net correspond to the 2D offset of a detected object with respect to the center of the table it lies on, the table it lies on and its associated type. The desired output y values are, for each object seen so far, the predicted table y^t it is on currently as well as its associated offset relative to the predicted table, y^o. For computational efficiency, we use existing local geometric alignment algorithms to reconstruct object shapes from predicted offsets y^o, as opposed to directly regressing shape from OBM-Net as done in Section V-A. We made this choice to enable efficient execution of our system on a robot, but it is straightforward to directly train OBM-Net to reconstruct shapes as done in Section V-A.

	Table 1		Table 2		Table 3	
	Position Error	Object Accuracy	Position Error	Object Accuracy	Position Error	Object Accuracy
Clustering	0.198	67%	0.181	50%	0.119	50%
DS-SLAM [10]	0.028	100%	0.096	50%	0.119	50%
OBM-Net	**0.025**	**100%**	**0.035**	**100%**	**0.035**	**100%**

TABLE II: **Quantitative Analysis of OBM-Net on Real Data.** Quantitative comparison of OBM-Net with baselines on regressing the final positions and presence of individual objects at each table at the final timestep 5. OBM-Net outperforms baselines.

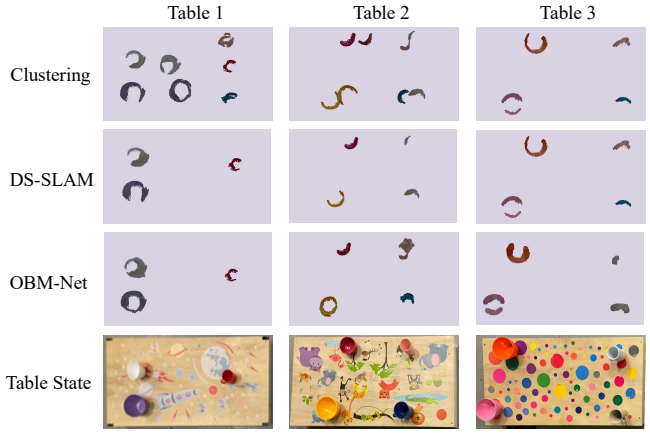


Fig. 10: **Real 3D Shape Reconstructions (top).** Illustration of top view of reconstructed object shapes from OBM-Net compared to each baselines. Compared to baselines, OBM-Net is able to simulate the horizontal movement of bowls, as well as the dynamic movement of mugs. Reconstructions of individual objects, such as mugs in table 2 is aided by observations at other tables.

To extract object detections at each time step, we first compute camera poses for each captured image in the sequence using ORBSLAM2 [32]. Next, we extract partial point clouds of observed mugs and bowls on the table from each image by combining Mask-RCNN [33] segmentations of each RGB image with its corresponding depth image (illustrated in Figure 8). Finally, we fuse partial point clouds across images in a time step using the camera poses from ORBSLAM2 to obtain a set of object detections at that time step. We then compute the 2D offset of a detected object by computing the centroid of the partial point cloud. We train OBM-Net on 1000 simulated trajectories of objects following the same movement dynamics and measure the ability of OBM-Net to generalize to real data. We quantitatively measure object accuracy, the percentage of objects correctly inferred at each table at time step 5, as well as position error, the distance of the nearest inferred object's position from the closest actual object's position on a table.

Baselines. We compare OBM-Net to two of the best-performing baseline approaches in Section V-A to obtain a semantic segmentation of the dynamically changing mugs and bowls on real images. First, we consider the clustering baseline from Section V-A, where we directly aggregate underlying semantic detections across time and apply clustering to the resultant outputs, using the ground truth number of objects as the number of centers. Next, we adopt the semantic SLAM approach of [10] to deal with dynamic objects, where semantic segmentations of objects are updated based on new

observations if they meet a log-odds probability threshold. Neither of the baselines has a model of dynamics of the objects.

Results. We illustrate the reconstructions of predicted objects on each table at time step 5 from OBM-Net and each of the baselines, using a side view in Figure 9 and from a vertical view in Figure 10. In Figure 9 reconstructions of individual shapes from OBM-Net can be seen to be more complete compared to those of the baselines. OBM-Net is also able to aggregate information about a particular object seen from past observations of the object at the same and other tables. In Figure 10, reconstructions of individual shapes can be seen to satisfy the underlying dynamics of objects, with horizontal movement of bowls on each table, as well as the movement of mugs across tables. In particular, in both Figure 9 and Figure 10, while OBM-Net is able to reconstruct the correct mug object at table 2 and 3, both DS-SLAM and clustering baselines reconstruct the incorrect mug objects. Furthermore, on table 2, OBM-Net can reconstruct pink and blue mugs more accurately by aggregating views of the mugs seen at other tables. We further quantitatively evaluate our approach compared to baselines in Table II and find that OBM-Net substantially outperforms the baselines.

VI. Discussion

This work has demonstrated an approach to constructing a dynamic object-based memory for household robots in changing environments. By incorporating algorithmic bias inspired by a classical solution to the problem of filtering to estimate the state of multiple objects simultaneously, coupled with modern machine-learning techniques, OBM-Net learns from experience *how to perform as a data-association filter* in novel environments with complex observation and transition models that would be too difficult to hand-specify. Importantly, our system may be applied with *no prior knowledge* about the types of observations or desired output hypotheses or the frequency of observations. We have demonstrated its effectiveness in a variety of problems including simulated and real robot object memory problems.

References

[1] Yilun Du, Tomas Lozano-Perez, and Leslie Pack Kaelbling. URL: https://yilundu.github.io/obm/.

[2] Ioannis Kostavelis and Antonios Gasteratos. "Semantic mapping for mobile robotics tasks: A survey". In: *Robotics and Autonomous Systems* 66 (2015), pp. 86–103.

[3] Yaakov Bar-Shalom, Fred Daum, and Jim Huang. "The Probabilistic Data Association Filter". In: *IEEE Control Systems Magazine* (2009).

[4] Xieyuanli Chen et al. "Suma++: Efficient lidar-based semantic slam". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 4530–4537.

[5] Liang Zhang et al. "Semantic SLAM based on object detection and improved octomap". In: *IEEE Access* 6 (2018), pp. 75545–75559.

[6] Javier Civera et al. "Towards semantic SLAM using a monocular camera". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 1277–1284.

[7] Linyan Cui and Chaowei Ma. "SOF-SLAM: A semantic visual SLAM for dynamic environments". In: *IEEE access* 7 (2019), pp. 166528–166539.

[8] Kevin Doherty, Dehann Fourie, and John Leonard. "Multimodal semantic SLAM with probabilistic data association". In: *2019 international conference on robotics and automation (ICRA)*. IEEE. 2019, pp. 2419–2425.

[9] Zemin Wang et al. "A computationally efficient semantic slam solution for dynamic scenes". In: *Remote Sensing* 11.11 (2019), p. 1363.

[10] Chao Yu et al. "DS-SLAM: A semantic visual SLAM towards dynamic environments". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 1168–1174.

[11] Linyan Cui and Chaowei Ma. "SDF-SLAM: Semantic depth filter SLAM for dynamic environments". In: *IEEE Access* 8 (2020), pp. 95301–95311.

[12] Nick Hawes et al. "The strands project: Long-term autonomy in everyday environments". In: *IEEE Robotics & Automation Magazine* 24.3 (2017), pp. 146–156.

[13] Rares Ambrus et al. "Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2015, pp. 5678–5685.

[14] Gian Diego Tipaldi, Daniel Meyer-Delius, and Wolfram Burgard. "Lifelong localization in changing environments". In: *The International Journal of Robotics Research* 32.14 (2013), pp. 1662–1678.

[15] Tomasz Kucner et al. "Conditional transition maps: Learning motion patterns in dynamic environments". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 1196–1201.

[16] Tomáš Krajník et al. "Fremen: Frequency map enhancement for long-term mobile robot autonomy in changing environments". In: *IEEE Transactions on Robotics* 33.4 (2017), pp. 964–977.

[17] Greg Welch and Gary Bishop. "An introduction to the kalman filter". In: (2006).

[18] Wenhan Luo et al. "Multiple object tracking: A literature review". In: *arXiv preprint arXiv:1409.7618* (2014).

[19] Yu Xiang, Alexandre Alahi, and Silvio Savarese. "Learning to track: Online multi-object tracking by decision making". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4705–4713.

[20] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, pp. 3464–3468.

[21] Rohit Girdhar and Deva Ramanan. "CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning". In: *arXiv preprint arXiv:1910.04744* (2019).

[22] Song Bai et al. "Re-ranking via metric fusion for object retrieval and person re-identification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 740–749.

[23] Vaibhav Bansal, Gian Luca Foresti, and Niki Martinel. "Where Did I See It? Object Instance Re-Identification With Attention". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 298–306.

[24] Aviv Tamar et al. "Value Iteration Networks". In: *ArXiv* abs/1602.02867 (2016).

[25] Rico Jonschkowski, Divyam Rastogi, and Oliver Brock. "Differentiable Particle Filters: End-to-End Learning with Algorithmic Priors". In: *ArXiv* abs/1805.11122 (2018).

[26] Rahul G. Krishnan, Uri Shalit, and David A Sontag. "Deep Kalman Filters". In: *ArXiv* abs/1511.05121 (2015).

[27] Peter Karkus et al. "Differentiable Algorithm Networks for Composable Robot Learning". In: *ArXiv* abs/1905.11602 (2019).

[28] Harry G. Barrow et al. "Parametric Correspondence and Chamfer Matching: Two New Techniques for Image Matching". In: *IJCAI*. 1977.

[29] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural Comput.* 9.8 (1997), pp. 1735–1780.

[30] Juho Lee et al. "Set transformer: A framework for attention-based permutation-invariant neural networks". In: *arXiv preprint arXiv:1810.00825* (2018).

[31] Lars Mescheder et al. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4460–4470.

[32] Raul Mur-Artal and Juan D Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras". In: *IEEE transactions on robotics* 33.5 (2017), pp. 1255–1262.

[33] Kaiming He et al. "Mask r-cnn". In: *ICCV*. 2017.