Object Disappearance for Object Discovery

Julian Mason and Bhaskara Marthi and Ronald Parr

***Abstract*— A useful capability for a mobile robot is the ability to recognize the objects in its environment that move and change (as distinct from background objects, which are largely stationary). This ability can improve the accuracy and reliability of localization and mapping, enhance the ability of the robot to interact with its environment, and facilitate applications such as inventory management and theft detection. Rather than viewing this task as a difficult application of object recognition methods from computer vision, this work is in line with a recent trend in the community towards *unsupervised* object discovery and tracking that exploits the fundamentally temporal nature of the data acquired by a robot. Unlike earlier approaches, which relied heavily upon computationally intensive techniques from mapping and computer vision, our approach combines visual features and RGB-D data in a simple and effective way to segment objects from robot sensory data. We then use a Dirichlet process to cluster and recognize objects. The performance of our approach is demonstrated in several test domains.**

I. Introduction

Objects are a fundamental primitive in robotics. The ability to recognize, track, and map objects and their locations is a requirement for a wide variety of useful robotic capabilities. However, object detection and recognition are difficult, open problems. Ideally, one might hope to provide a robot with high-resolution 3D object models. However, such models must be painstakingly built using a turntable or other specialized sensing equipment. Even if this were practical, the underlying assumption that the objects a robot *might* encounter could be thoroughly cataloged in advance is questionable at best. In a general environment, maintaining a database of "all possible objects" is intractable: objects may be introduced or removed at any time, objects may change shape, and new objects have to be individually modeled.

We develop an approach to object perception based on the principle of *object discovery*. The fundamental problem in object discovery is to segment incoming sensor readings into "object" and "non-object" segments. Observing that general environments are fundamentally dynamic, we assume that objects are things that move. By detecting that objects have moved, we allow the environment to provide us with a segmentation directly. Intuitively, if we can detect that something has disappeared, whatever *used to be there* must have been an object. We then send these segments to a tracking system that performs data association through time, and over object appearances at various locations. The data association is done by inference in a probabilistic model based on a Dirichlet process.

Our system, from segmentation to data association, is entirely unsupervised, and runs on a standard mobile robotic platform. We demonstrate it on three datasets of increasing size and complexity, and empirically evaluate the results.

The contributions of this work are threefold: a sparse feature map for efficient detection of object disappearance, an accurate depth-based object segmentation system leveraging the sparse feature map, and a visual-word-based probabilistic model for object appearances in different locations that allows accurate data association across time and space.

All of the code and data used in this work are publicly available [1].

II. Prior Work

The idea that the capacity of an object to move is inherent to its status as an object is (at least) decades old; the classic work of Gibson [2] termed this *detachability*. More recently, Biswas et al. [3], Anguelov et al. [4] and Sanders et al. [5] worked in this area. These early papers provide examples of two common approaches to this problem: *scene differencing* and *live motion*. Scene differencing approaches construct several static maps of an environment, with a period of time in between. Changes can be detected by comparing two or more such maps; these changes can then be grouped into objects. Biswas et al. work with two-dimensional occupancy grids built using sonar, and perform EM to infer object models. Anguelov et al. extend this to include object class templates. Like our work, these approaches assume that the world is static in the short term. Kang et al. [6] infer a "background" from a collection of unordered images. In contrast, live video approaches observe the world changing moment-to-moment, and rely on these direct motion cues. Sanders et al. use several (fixed) cameras, and analyze the time-series of each image pixel to group pixels according to how they change. By operating in pixel space, this work limits itself to a fixed-camera system. Modayil and Kuipers [7] use a 2D occupancy map to classify sensor readings as "static" or "dynamic"; dynamic readings are then clustered and tracked. Southey and Little [8] provide another example of a live-video system, combining stereo vision with optical flow techniques to segment manipulable objects in video, and visual features to group these segments. Ayvaci and Soatto [9] use motion in video to find occlusion cues which are integrated to partition the image into depth layers. Sivic et al. [10] do frame-to-frame tracking in video, and

Julian Mason and Ronald Parr are with the Duke University Department of Computer Science. 308 Research Drive, Durham NC 27708. {mac, parr}@cs.duke.edu

Bhaskara Marthi is with Willow Garage. 68 Willow Road, Menlo Park CA 94025. bhaskara@willowgarage.com

This work supported by Willow Garage and NSF CAREER award IIS-0546709. Any opinions, findings, conclusions, or recommendations are those of the authors only.

aggregate groups of points that move together to segment objects. The tracked frames become exemplars of the object's appearance, allowing object-level queries.

A state-of-the-art technique in scene differencing is that of Herbst et al. [11], [12], which takes the difference between three-dimensional maps built with an RGB-D sensor. By working directly with a 3D representation, their system generates 3D models directly, and shows excellent performance. However, their technique requires the precise alignment of very high quality maps. Although this can be done automatically, it limits their technique to environments where extremely accurate dense map registration is possible; they demonstrate their technique on tabletop-sized examples. The computational cost is also proportional to the size of the environment. In contrast, our sparse map allows us to scale to large environments; we show examples of our method associating segments across several appearances in a large (1600 m^2) environment. A direct quantitative comparison with this work is impossible, as neither an implementation of their technique nor their input data are publicly available.

Object discovery can also be thought of as a *cosegmentation* problem (see, e.g., Rother et al. [13] and Vicente et al. [14]). In cosegmentation, the information shared between groups of images is leveraged to improve segmentation. Importantly, cosegmentation algorithms take as input pictures of objects, where the task is *finding* the object in the picture rather than inferring if an object is present at all.

Sudderth et al. [15] present a Dirichlet process-based generative model for a parts-based objects and backgrounds, and demonstrate impressive performance on scenes of streets and offices. However, their data are normalized so that objects in the same class appear at roughly the same scale in each image, and they require labeled training data.

Finally, Kang et al. [16] present a technique for object discovery in image sets. They use repeated oversegmentations to generate a large pool of segments; these are then clustered according to color, visual features, and shape descriptors. Like this work, they require that objects move to be correctly discovered. Kang's focus on unordered RGB image sets makes the technique general, but the lack of depth data or a localized sensor greatly complicates their algorithm. Again, a direct quantitative comparison is impossible as implementation is not publicly available, and the available data are RGB-only.

III. Background

Our data association algorithm is based on Dirichlet processes (DPs) [17], and we provide a brief overview here. DPs are probability distributions suited for clustering problems in which the number of clusters is not known in advance. A DP has two parameters: a *base measure* G_0, and a *concentration parameter* (or *new cluster rate*) $\alpha > 0$. A sample from the DP is itself a probability distribution over the same space as the base measure.

The probabilistic model used in clustering problems is actually an extension known as the Dirichlet process mixture. Intuitively, each cluster will correspond to a particular probability distribution from which the assigned elements are drawn. For example, if the elements to be clustered are from a finite set, each cluster may have a multinomial distribution over this set. The base measure G_0 is a prior distribution over these cluster distributions. In the example, the base measure may be a Dirichlet prior over the set of multinomial distributions on the given set. Drawing from the Dirichlet distribution then yields a specific countably infinite mixture G of multinomial distributions. Each observed element is then generated by sampling a mixture component from G, then sampling from that component's multinomial distribution. Two elements are in the same cluster if they were generated by the same mixture component.

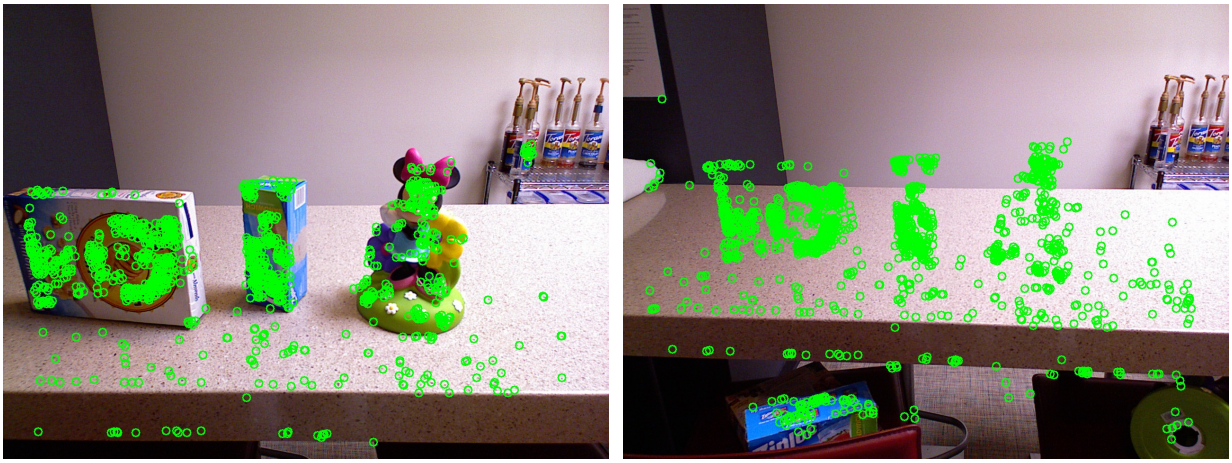
Given a set of observed elements to be clustered, the unobserved quantities are the DP sample G, and the identity of the cluster from which each element was drawn. Although G has infinitely many mixture components, in practice we only care about how the observed items are partitioned into clusters. There exist several inference algorithms [18] based on Markov-chain Monte Carlo (MCMC) that will generate approximately i.i.d. sample clusterings from the posterior distribution conditional on the observations. We may then use these samples to answer any inferential questions about the clustering, for example by picking the most likely one.

IV. Problem

We wish to *discover* and *track* objects in a general environment with a mobile robot. Robot localization is a well-studied problem; in indoor environments with good sensors, it is effectively solved by AMCL [19]. Furthermore, the release of the Microsoft Kinect has made available high-resolution RGB-D data. Taking these together, a modern robot can reasonably produce localized RGB-D data. Our particular robot is a Willow Garage PR2, but our technique requires only a localized RGB-D sensor.

Our driving premise is that objects move (or are moved) while the robot isn't looking. We seek to detect when movement has occurred; if features formerly present in the world have disappeared, something in the space occupied by the features was probably an object. Detecting that features have disappeared is easier than detecting when space is vacated or newly occupied, as these require storing 3D metric information about the entire world just in case it becomes relevant to object discovery. Octree-based techniques like OctoMap [20] can be used for this, but these dense techniques are extremely sensitive to small localization errors, particularly at the resolutions necessary to detect objects. Rather than use a dense technique, we focus on using feature disappearance to cue object discovery, though our approach could be easily extended to use feature appearances as well (see Section X). To facilitate efficient discovery of features which have disappeared, we use a sparse representation: a map of timestamped visual features, posed in 3D space. See Section V for details.

A genuinely sparse feature map cannot label every pixel in a sensor reading as "object" or "non-object". Here, we can leverage the depth data provided by our sensor. Rather



(a) An image, with the features in M_1 drawn on.

(b) A later observation of the same place, with M_4 drawn on.

Fig. 1: An example of detecting visual feature disappearance (Section V). Note that some feature clusters are not on objects; however, they are on planes (or on connected components with too few points), and are therefore filtered out. *Best viewed in color.*

than segmenting in RGB, we can use the points which have disappeared to seed a segmentation analysis in the depth image. The segmenter is detailed in Section VI.

Given these segments, we must now perform a tracking step, to associate segments in both time ("I'm seeing this object again") and in space ("I'm now seeing this object in a different location.") We take the common (see, e.g., Sivic et al. [10] and Kang et al. [16]) approach of using a bag of visual words. Because our segmenter uses full-fledged features, not visual just visual words, we can learn our words from only those features that appear on objects. This opportunity to tune our tracker to the specific objects in our scene is an important benefit of our approach. See Section VII for details.

Finally, we use a DP model over these visual words to perform tracking and data association. Our output is a set of clustered RGB-D segments. These could be the input, for example, to a 3-D reconstruction system. Data association is detailed in Section VIII.

V. Visual Features

To detect that something has disappeared (and is therefore an object), one must know that there was something there in the first place. Rather than build a dense 3D map like that of Herbst et al. [11], [12], we build a sparse map of visual features projected into 3D. Recall that our goal is to detect object disappearances; given a new RGB-D frame pair, the question we wish to answer is "Which feature points *should have* been observed, but were not?" Such points are then treated as candidate on-object points.

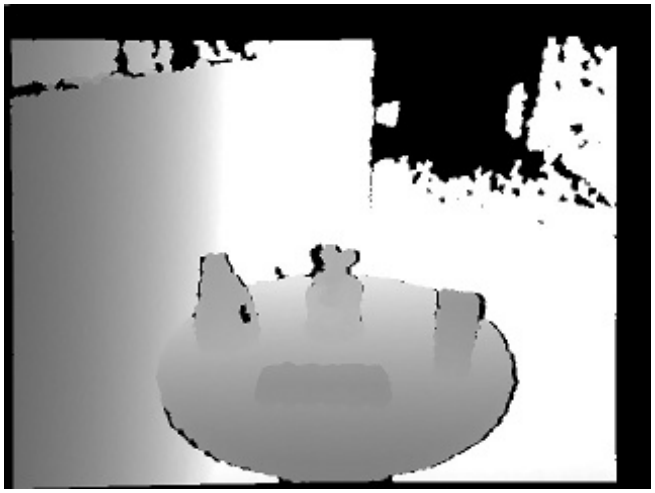
The use of an RGB-D camera makes the geometric part of this question straightforward. Given the current sparse feature map M_1 and the robot's current localization estimate, we can project the points into RGB pixel coordinates. The camera z-coordinate (out of the image plane) can then be checked against the depth image to see if the point is occluded in this frame. This analysis leaves us with a set of points M_2 that should have been observed in this frame.

Next, we must determine which points were actually observed. We compute visual features in the current RGB frame, and project them into 3D. We define a spatial threshold s and a descriptor-distance threshold d. A new feature f is deemed to match a map feature m if f is within both the s and d thresholds. Applying these thresholds to each element of M_2 gives us a set M_3 of features that should have been observed, but were not. These features are added to the map. See Figure 1 for an example.

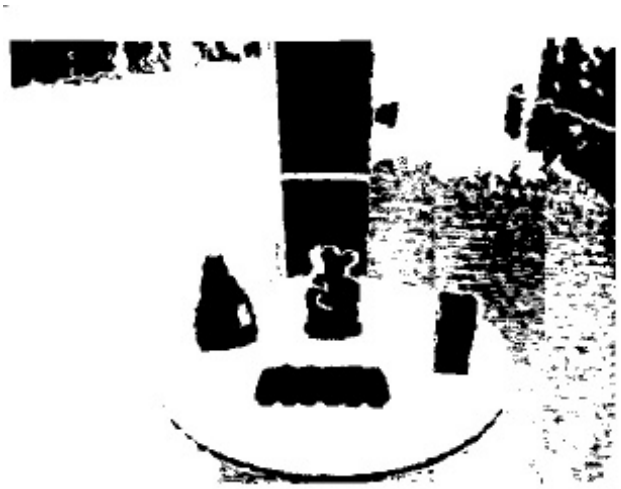
A feature map can be stored and used efficiently if it is not populated too heavily with useless features. This is a concern for most methods of generating visual features since they can produce a large number of features per image, but many of these may suffer from instability due to image noise, camera motion, and (in our case) accidental occlusion due to localization error. Because we use negative feature detections as a cue for positive object detections, we wish to avoid *false negatives*, even at the risk of false positives. We introduce two techniques to prune our features. First, we enforce *temporal stability*. We require that a feature be observed (seen for the first time, or matched) for k frames in a row before it is added to the map. This helps to filter those features that are highly sensitive to image noise. We also enforce temporal consistency on the matching side: to count as a candidate, a feature must fail to match (that is, be in M_3) k times in a row; this helps to account for transient misses due to image noise.

Secondly, we introduce the concept of a *feature cluster*. Because features will be seen from several different viewpoints, the same point in 3D space may generate a variety of different feature descriptors. To account for this, our feature clusters store several descriptors. The cost for a new feature f to match against a cluster c is then the minimum of the

(a) The RGB image. The projected feature clusters (used to seed the flood-filling operation) are drawn on in red.



(b) The depth image, scaled so that 2 m fills the range (per our depth cutoff; see Section IX).



(c) The planes-and-edges mask. White indicates areas that will be masked out of the image. Flood fill occurs within the black areas.

(d) The segments (false color). Seed points are drawn here as well.

Fig. 2: The stages of the segmentation algorithm (Section VI). *Best viewed in color.*

descriptor distance between f and any of the descriptors in c. To update feature clusters, we introduce a second spatial threshold, the *integration threshold* i. If f is within i of c, we add f's descriptor to c's set. Importantly, we do not perform a descriptor-distance check first: the goal of a feature cluster is to capture the variability of descriptor values due to viewpoint; requiring a close descriptor match would defeat this. Applying the temporal stability criteria to the features in M_3 gives us M_4, which we pass as input to the segmenter.

As described, feature clusters will grow without bound if a location steadily changes appearance during repeated observations. This could be addressed by setting a maximum cluster size, and timing out cluster members that are too old, or have not been directly matched recently enough. Cluster growth did not pose a problem in our experiments.

Our feature clusters draw some inspiration from the HOC descriptor [21], which also seeks to handle viewpoint effects in visual features.

In our experiments, we use the ORB descriptor [22]. We set the spatial threshold $s = 5$ cm, the descriptor distance threshold $d = 150$, the temporal stability threshold $k = 5$, and the integration threshold $i = 2$ cm, and use the same parameters for every experiment.

VI. Segmenter

When a set of candidate object points is identified, these points are handed off to the segmenter. The job of the segmenter is to discover the RGB-D data in an earlier frame that corresponds to the missing object in the current frame. This is achieved by going backwards in time to all frames that contained the missing feature. For each such frame, a segment in the depth image is extracted.

The segmenter relies on a simple assumption: objects must be *supported*. In particular, we assume that objects must rest on planes. This assumption throws out certain types of objects (hanging overhead lamps, for example), but includes

nearly everything else. For example, the objects found by Herbst et al. [11], [12] are all on tabletops. Similarly, all but 10 of the 175 images in the ADL dataset from Kang et al. [16] are of objects on large planes. This plane assumption has also been used in the semantic mapping literature, often to ease object recognition in controlled close-range environments. See, e.g., Rusu et al. [23] and Trevor et al. [24].

As input, our segmenter takes an image and the temporally stable set M_4 (Section V, Figure 2a). Working with the corresponding depth image (Figure 2b) planes are extracted using RANSAC, as implemented in PCL [25]. A mask is formed using the pixels found to be on planes. This mask will be used to separate the planes from the objects by subtracting away the planes. Next, pixels of large depth discontinuity (depth edges) are found, and added to the mask. The resulting mask is shown in Figure 2c. Finally, we project the points in M_4 into the masked depth image and flood-fill outward (from black to white in Figure 2c). The resulting connected regions correspond to segments and can be used to extract depth or color data for the objects, as shown in Figure 2d. Very large connected components (larger than 0.7 m on any side in our experiments) are filtered out, as are those that contain fewer than three feature clusters.

The output of this step is a set of segments, represented in pixel coordinates. We rely upon our DP object model to determine whether segments correspond to the same physical object. Performance is discussed in Section IX.

VII. Visual Words

During tracking, rather than working directly with feature descriptors in our object model, we use a *bag of words* approach, in which descriptors are quantized into visual words [10]. This provides faster performance, and greater robustness and generalization across different viewpoints than working with raw features. We use a vocabulary of words of size W (in our experiments $W = 250$). These words are generated after segmentation, meaning that they are tuned to the specific segments in our data.

For each segment, we first recompute ORB features for that segment alone (the previously computed ORB features for the frame covered the entire image, and therefore may not have many features lying on the segment). Next, given the ORB descriptors for all features on all segments in our dataset, we apply K-means clustering in descriptor space to get a set of centroids $d_1, \ldots, d_W$ (also in descriptor space).

Each feature descriptor on each segment is then replaced with the closest centroid. If d_w is the closest centroid to descriptor d, then it is simply represented as the integer w in the inference algorithm, since the only operation that will be performed on the visual words is equality checking. The output of this stage consists of the segments from the previous stage, and the visual words for each segment.

VIII. Data Association

The perception pipeline described thus far produces, for each frame for which a group of features has disappeared, some number of segments, each consisting of a set of visual words and associated position (in the sensor frame). The remaining task is to determine which of these segments correspond to the same object. Even if segments are produced from two consecutive frames, they may not be identical due to noise in the color and depth images. When the robot and objects have moved in the scene, the segments corresponding to the same object will certainly differ due to changes in the size of the object in the image and pose relative to the robot.

Given a history of observed segments, we aim to produce a clustering or, more generally, a probability distribution over clusterings, where clusters correspond to objects. Thus, a clustering of a set of segments consists of a set of hypothesized object IDs and, for each segment, the ID of the object to which it belongs.

Clustering and data association are well-studied problems. A challenge in our setting is that the segments being clustered are complex, with varying dimensionality. Further, it is challenging to define a notion of distance between clusters, due to occlusions (not all visual words are observed) and due to the fact that the absolute coordinates of the feature points will change when the object moves.

We use a model based on DPs to approach the problem. The base measure for the DP is a Dirichlet distribution over the set of visual words $\{1, \ldots, W\}$. Therefore, each mixture component of the DP corresponds to a multinomial distribution drawn from this base measure. An observed segment is generated by choosing a component based on the DP, then sampling independent visual words from the component's multinomial distribution.

The mixture components of this DP model correspond to underlying objects in the world; each has an associated multinomial distribution over visual words, corresponding to the features on the object. The observed elements are individual visual words. Our goal is simply to cluster the observed segments; we assume more detailed modeling of the objects will be done at a later stage. Note that the setting is slightly different from the standard Dirichlet process mixture, in that the visual words appearing on a given segment are assumed to come from the same object (i.e., we assume at this stage that the segmentation is correct).

Our algorithm will output a set of samples, each of which is a clustering that assigns an object ID to each observed segment. To determine a single best clustering given the samples, we could simply take the most likely clustering, but the samples are very high dimensional, and any given sample occurs only a handful of times. Using the most likely clustering therefore ignores much of the information in the samples. We instead separately assign each segment to the object ID it was most often associated with in the samples. This estimator will not work for arbitrary sampling schemes, because it depends on the specific values of the object ID and is not permutation invariant. But it works well for our Gibbs sampler described below, in which it is unlikely for the IDs of all the segments corresponding to an object to change simultaneously to a new value.

We use a version of the collapsed sampler described by

Dataset	S	M	L
Hand segments	270	394	419
Auto segments	270	396	423
True positives	270	392	357
False positives	0	4	66
False negatives	0	2	51
Precision	100%	98.9%	84.4%
Recall	100%	99.4%	87.5%

TABLE I: Segmenter performance (Section IX-A).

Dataset	S	M	L	K-means on L
Unique objects	2	4	7	...
Auto segments	270	396	423	...
Invalid segments	0	4	36	...
Precision	100%	100%	86.2%	68.9%
Recall	100%	100%	72.2%	59.9%

TABLE II: Tracker performance (Section IX-B), and compared to a K-means baseline on the large dataset.

Neal [18], modified to deal with the fact that observations are individual visual words, but clustering is done at the level of segments. In more detail, the algorithm maintains samples of the form $(x_1, \ldots, x_M)$ where M is the number of segments. Each x_m is an object ID, which we represent as a positive integer. Initial samples may be generated in any way; we use an initial sample $(1, 2, \ldots, M)$. At each iteration, the algorithm flips the m^{th} component, where m repeatedly sweeps over $(1, \ldots, M)$. Flipping the m^{th} component is equivalent to assigning segment m to some existing object, or to a new object. The probability of assigning segment m to object c is proportional to:

$$W_{-m,c} \int F(s_m, \phi) dH_{-m,c}(\phi)$$

where $W_{-m,c}$ is the number of visual words on segments other than m currently assigned to c, $H_{-m,c}$ is the posterior distribution over multinomial distribution ϕ based on the prior Dirichlet distribution G_0 and observations of these visual words, and F is the likelihood of the words in segment s_m given ϕ. The probability of assigning m to a new object (not assigned to any other segments) is proportional to:

$$\alpha \int F(s_m, \phi) dG_0(\phi)$$

where α is the parameter to the DP (we use $\alpha = 5.0$ in our experiments). The above quantities are computed for all existing object IDs and the new object ID, then normalized.

We incorporate various optimizations to perform sampling efficiently. The main idea is that the various quantities used in sampling can be updated incrementally with the sample. In particular, we maintain a reverse mapping from cluster ID to segments belonging to it, as well as, for each cluster, the current posterior Dirichlet distribution for that cluster given all the segments currently assigned to it. This allows our sampler to perform several thousand flips per second.

IX. Performance

We validate the performance of our system on three datasets of increasing size and complexity. As noted before, we require only a localized RGB-D camera; in our mobile datasets, this is a Microsoft Kinect mounted atop a Willow Garage PR2, capturing RGB frames at 1280×960 and depth frames at 640×480, both at 5 Hz. The depth images are limited to a range of 2 m, to minimize range errors from the Kinect. Frames during which the robot is moving too quickly are filtered out to minimize visual feature errors due to motion blur. However, they are included in the posted data [1]. The datasets are called:

- *Small:* A fixed-camera example, as a sanity check. The dataset consists of 101 frames of a static, empty scene, followed by 135 frames in which two objects have been added, and then 114 frames in which the objects have been removed. Example images can be seen in Figure 3. The hand-segmentation results in 270 segments, and two unique objects.
- *Medium:* A dataset taken from a mobile robot navigating in an office environment. The robot observes a table (with objects) and then looks away while the objects are removed, and observes the table again. The robot then travels roughly 18 m, and repeats this process with a counter that contains the same objects. The same objects were used in both places to test data association across locations (with commensurate changes in lighting, etc.) There are four unique objects, 394 segments found by hand, and 484 total frames. Example images can be seen in Figure 4.
- *Large:* A dataset that ranges over several rooms of a 40 m $\times$ 40 m office environment, for a total distance of 181.5 m. There are two passes over this environment. In the first pass, the robot observes several objects in each room. In the second pass, all the objects are removed. There are a total of seven unique objects, 419 segments found by hand, and 397 frames. The frame count is lower than the medium dataset because the robot was not allowed to linger as long in any location. Example images can be seen in Figure 5.

All three datasets and their hand segmentations are publicly available [1].

A. Segmenter Performance

To provide a baseline against which to compare our segmenter, we hand-segmented each dataset. For every occurrence of an object in our data, we manually find the bounding rectangle and assign a label according to the object name. Our automatic segmentations are not, in general, rectangular (or even convex). To compute the overlap between a hand segmentation h and an automatic segmentation a, we first find the bounding rectangle r of a. We then declare h and a equal if $\text{Area}(h \cap r) \geq 0.5 \cdot \text{Area}(r)$ and $\text{Area}(h \cap r) \geq 0.5 \cdot \text{Area}(h)$. This 50-50 overlap criterion is common; Kang et al. [16] use it (for non-rectangular segments), for example. We can then compute precision and recall scores for each dataset. Table I shows the results.

B. Data Association

Given the output from the segmentation, we ran the Gibbs sampling algorithm described in Section VIII on each of the

(a) An image from before the objects appear. (b) An image when the objects are visible. (c) An image with the segmentations drawn on.

Fig. 3: Example images from the small dataset (Section IX). The abrupt change in lighting between (a) and (b) is from the Kinect's automatic gain. The scene after the objects are removed is equivalent to (a), so is not shown. Image (c) shows a segmentation. False-color pixels denote the segmentation results; thick boxes denote the bounding rectangle of the segment. Thin boxes denote the hand segmentation.

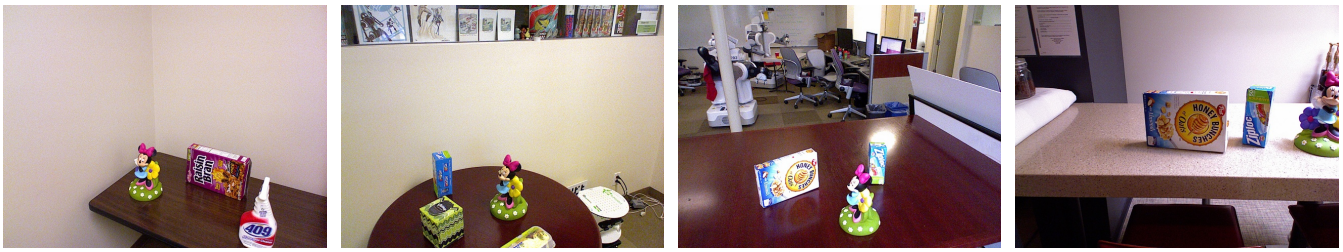
(a) An example observation of the table, with objects.

(b) An example observation of the table, without objects.

(c) An example observation of the counter, with objects.

(d) An example observation of the counter, without objects. (The objects seen here are never seen to move, and so are not detected by our system.)

Fig. 4: Examples from the medium dataset (Section IX). The four unique objects seen to move in this dataset are all visible in (a).



(a) The first of the four places investigated.

(b) The second of the four places investigated.

(c) The third of the four places investigated.

(d) The fourth of the four places investigated.

Fig. 5: Examples from the large dataset (Section IX). All seven unique objects can be seen. Some objects appear in several locations; others, only one. All four places were also seen without objects (not shown).

datasets. The sampler was run for 5000 scans over the set of segments, where each scan involves flipping each of the segment object IDs in turn.

Given the clustering output, we first label as "invalid" those segments which do not intersect an actual object. This is slightly different from the overlap criterion used for our analysis of the segmenter; we want to be able to track partial objects (for example, when an object is partway out of the field of view) even when the segmenter fails. We then define:

- TP (true positives) is the number of pairs of (valid) segments (i, j) such that i and j come from the same object in reality, and are in the same cluster;
- FP (false positives) is the number of pairs such that i and j do not come from the same object but are in the same cluster;
- FN (false negatives) is the number of pairs such that i and j come from the same object but are not in the same cluster.

We then report, for each dataset, the precision $TP/(TP + FP)$ and recall $TP/TP+FN$. For a baseline, we compared our results on the large dataset against K-means clustering. Treating each segment's (normalized) vector of visual word counts as a multinomial distribution, we clustered according to total variation distance. Unlike our algorithms, K-means was provided with the true cluster count (seven). Results are shown in Table II.

Model parameters were tuned by searching over a range of values jointly for α and θ (the parameter for the symmetric Dirichlet base measure); the parameter values used in the experiments were $\alpha = 9.0$ and $\theta = 100.0$. We also performed a sensitivity analysis around the chosen parameter values, allowing each parameter to vary by up to 40% in either direction. Overall performance (summed precision and recall) varied by at most 15% over this range.

We note two extreme cases for comparison, assuming n objects that appear l times in each of k locations. First, an algorithm that simply clustered all segments together would have precision $1/n$ and recall 1, while an algorithm that matched instances of an object within locations but not across would have precision 1 and recall $1/k$. For example, on the large dataset, where each object appears in ≈ 3 locations, baseline 1 would have precision 0.14 and recall 1.0, while baseline 2 would have precision 1.0 and recall 0.33.

X. Conclusions and Future Work

We have presented a simple and effective method for detecting objects in RGB-D data streams of the type that would be collected by a mobile robot. Unlike previous approaches, our method does not rely upon highly accurate 3D maps and does not use computationally intensive image processing techniques. Our method uses the disappearance of visual features as a cue to construct segments, which are then associated across space and time using a DP object model that effectively clusters segments into objects. We demonstrate the performance of this system across several domains with varying numbers of objects.

Currently, we detect only object disappearances. However, detecting appearances is a straightforward extension. Consider the lifespan of disappearing feature cluster: at some time t it is stably visible, and at some later time t' it is stably missing. To detect object appearance, we need only detect those clusters which are stably visible at t' and stably invisible at t; that is, we need to run the identical analysis, but backwards through time, not forwards.

While the bag of words model of objects appears to be sufficient for object discovery, it does not exploit the 3D data contained in the segments. This data could be used for even more accurate object detection by helping to disambiguate similar objects. It could also be used to build 3D models of objects, thus enabling the completely unsupervised construction of an object model database.

References

[1] "Accompanying data, software, and code." http://ros.org/wiki/Papers/IROS2012_Mason_Marthi_Parr.

[2] J. Gibson, *The Ecological Approach to Visual Perception*. Lawrence Erlbaum, 1986.

[3] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun, "Towards Object Mapping in Non-Stationary Environments With Mobile Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, pp. 1014–1019.

[4] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, S. Sanner, and S. Thrun, "Learning Hierarchical Object Maps of Non-Stationary Environments With Mobile Robots," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2002, pp. 10–17.

[5] B. C. S. Sanders, R. C. Nelson, and R. Sukthankar, "A Theory of the Quasi-Static World," in *16th International Conference on Pattern Recognition*, 2007, pp. 1–6.

[6] H. Kang, A. A. Efros, M. Hebert, and T. Kanade, "Image Composition for Object Pop-out," *IEEE Workshop on 3D Representation for Recognition*, 2009.

[7] J. Modayil and B. Kuipers, "Towards Bootstrap Learning for Object Discovery," *AAAI Workshop on Anchoring Symbols to Sensor Data*, 2004.

[8] T. Southey and J. J. Little, "Object Discovery through Motion, Appearance and Shape," *AAAI Workshop on Cognitive Robotics*, 2006.

[9] A. Ayvaci and S. Soatto, "Detachable Object Detection: Segmentation and Depth Ordering from Short-Baseline Video," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[10] J. Sivic, F. Schaffalitzky, and A. Zisserman, "Object Level Grouping for Video Shots," *International Journal of Computer Vision*, pp. 189–210, Jan. 2006.

[11] E. Herbst, P. Henry, X. Ren, and D. Fox, "Toward Object Discovery and Modeling via 3-D Scene Comparison," in *IEEE International Conference on Robotics and Automation*, May 2011, pp. 2623–2629.

[12] E. Herbst, X. Ren, and D. Fox, "RGB-D Object Discovery via Multi-Scene Analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept. 2011, pp. 1–7.

[13] C. Rother, T. Minka, A. Blake, and V. Kolmogorov, "Cosegmentation of Image Pairs by Histogram Matching — Incorporating a Global Constraint into MRFs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 993–1000.

[14] S. Vicente, V. Kolmogorov, and C. Rother, "Cosegmentation Revisited: Models and Optimization," in *European Conference on Computer Vision*, Sept. 2010.

[15] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky, "Describing Visual Scenes using Transformed Objects and Parts," *International Journal of Computer Vision*, 2007.

[16] H. Kang, M. Hebert, and T. Kanade, "Discovering Object Instances from Scenes of Daily Living," in *International Conference on Computer Vision*, Nov. 2011.

[17] T. S. Ferguson, "A Bayesian Analysis of Some Nonparametric Problems," *Annals of Statistics*, 1973.

[18] R. M. Neal, "Markov Chain Sampling Methods for dirichlet Process Mixture Models," *Journal of Computational and Graphical Statistics*, pp. 249–265, 2000.

[19] D. Fox, "Adapting the Sample Size in Particle Filters Through KLD-Sampling," *The International Journal of Robotics Research*, pp. 985–1003, Dec. 2003.

[20] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation*, May 2010.

[21] K. Pirker, M. Rüther, and H. Bischof, "Histogram of Oriented Cameras — a New Descriptor for Visual Slam in Dynamic Environments," in *Proceedings of British Machine Vision Conference*, 2010.

[22] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an Efficient Alternative to SIFT or SURF," *International Conference on Computer Vision*, pp. 1–8, Nov. 2011.

[23] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Close-range Scene Segmentation and Reconstruction of 3d Point Cloud Maps for Mobile Manipulation in Domestic Environments," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009.

[24] A. J. B. Trevor, J. G. R. III, C. Nieto-Granda, and H. I. Christensen, "Tables, Counters, and Shelves: Semantic Mapping of Surfaces in 3D," *IROS Workshop on Semantic Mapping*, pp. 1–6, Sept. 2010.

[25] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 9-13 2011.