2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Unsupervised Discovery of Object Classes with a Mobile Robot

Julian Mason and Bhaskara Marthi and Ronald Parr

Abstract— **Object detection and recognition are fundamental capabilities for a mobile robot. Objects are a powerful representation for a variety of tasks including mobile manipulation and inventory tracking. As a result, object-based world representations have seen a great deal of research interest in the last several years. However, these systems usually assume that object recognition is well-solved: they require that accurate recognition be available for every object they might encounter. Despite steady advances, object recognition remains a difficult, open problem. Existing object recognition algorithms rely on high-resolution three-dimensional object models or on extensive hand-labeled training data. The sheer variety of objects that occur in natural environments makes manually training a recognizer for every possible object infeasible. In this work, we present a robotic system for unsupervised object and class discovery, in which objects are first discovered, and then grouped into classes in an unsupervised fashion. At each step, we approach the problem as one of robotics, not disembodied computer vision. On a very large robotic dataset, we discover object classes with 98.7% precision while achieving 71.8% recall. The scale and quality of these results demonstrate the merit of our approach, and prove the practicality of long-term large-scale object discovery. To our knowledge, no other authors have investigated robotic object discovery at this scale, making direct quantitative comparison impossible. We make our implementation and ground-truth labelings available, and evaluate our technique on a very large dataset. As a result, this work is a baseline against which future work can be compared.**

I. INTRODUCTION

The fundamental problems of map building and robot localization have been a subject of study for many years, going back to the work of Moravec and Elfes [1]. Today, two-dimensional occupancy grid mapping (and localization in these maps) is effectively solved for robots with appropriate sensors. As a result, attention has shifted to maps that include objects, not just free and occupied space.

Given working object recognition, object mapping is easy: as a robot navigates, it recognizes the objects that it observes and notes their positions in the map. However, the set of objects that can appear in general environments is essentially unbounded: while manually training a recognizer for each and every object is possible in principle, it is prohibitively expensive in practice.

We propose a different perspective. Rather than recognize from a set of known objects, the robot should *discover* the objects in its environment and learn to recognize them in an unsupervised fashion. This approach has the advantage of requiring no human annotation of individual objects, allowing it to scale to large, general environments. It also makes good use of the mobile nature of the robot: rather than need to manually collect many views of each object (as is often done in object recognition; see, e.g., Rusu et al. [2] and Rublee et al. [3]), multiple views are collected "accidentally" as the robot (and potentially the object) move over time. Multiple views could also be guaranteed using an active search strategy. Another possible solution to the infeasibility of hand-labeling is to use a pre-existing database of objects models like those provided by RoboEarth [4]. However, this requires that the object be recognized from among hundreds of classes, many of which never appear in the robot's environment. This greatly complicates the recognition problem. Because our system learns the objects from the robot's environment, it is *specific*: it need only recognize those objects that appear.

In earlier work [5], we described a system for discovering objects using unsupervised segmentation and for performing change detection over those objects. However, the definition of "object" used was quite weak: objects were defined solely by their position, and no effort was made to learn object classes. In this paper, we extend that work to include a concept of object class, and demonstrate the ability to cluster objects through time and across space. The work described here runs on a Willow Garage PR2 robot with a Microsoft Kinect (although it only requires a localized base and RGB-D camera). The system runs unsupervised and demonstrates high performance on a standard robotic dataset which spans a large environment over a long period of time. The output of this system is a list of discovered objects, their positions over time, and class labels for each object. As these labels are of the form "class 1" or "class 2" not "coffee cup" or "textbook", this is not a complete semantic mapping solution. However, the human cost of labeling each class is far smaller than the human cost of manually segmenting objects from each image frame, or putting each object on a turntable.

As with our previous work in this area, the complete implementation of our system and the data and labels used to validate it are already publicly available; please see `http://ros.org/wiki/megaworldmodel`. Further details of this work are available in Mason [6].

II. PRIOR WORK

Semantic mapping seeks to move the study of robotic mapping beyond two- or three-dimensional occupancy and towards higher-level map constructs like objects, rooms, and available actions. A common first step is to assign semantic *labels* to perceptual data. Nüchter et al. [7] present an early

Julian Mason is with Google Research. `jmason@google.com`
Ronald Parr is with Duke University. `parr@cs.duke.edu`
Bhaskara Marthi is with Vicarious Systems. `bhaskara@gmail.com`
This work supported by NSF CAREER award IIS-0546709. Any opinions, findings, conclusions, or recommendations are those of the authors only.

example, in which three-dimensional points are labeled as floor, ceiling, or object points. Rusu et al. [8] focus on segmentation of objects at close range in tabletop settings. These methods attempt to build useful systems without directly recognizing objects, but this limits their capabilities. Recognizing and labeling environments is powerful. Various techniques have been used for recognition, including the fiducial markers of Galindo et al. [9], techniques based on SIFT features (e.g. Pronobis et al. [10]) and three-dimensional features (e.g. Blodow et al. [11]). In all cases, however, the recognizer was assumed to have been trained previously with the objects of interest. Our focus in this work is on closing the loop so that acquiring data and training recognizers in an unsupervised manner is part of the ongoing operation of the overall semantic mapping system.

Our approach is an example of the larger problem of *object discovery*, which seeks to segment the world into "object" and "non-object" components, and then do data association between the objects. A common approach to object discovery is to rely on object motion as a cue: an object is simply a set of points that is observed to move as a group. Kang et al. [12] use this idea on RGB images, while Herbst et al. [13] and Mason et al. [14] do so with RGB-D data. Motion is a strong cue, but also an onerous requirement. Each of the papers cited above is evaluated on a dataset where object movement is guaranteed; it is not clear if this is a reasonable assumption in general.

Finally, Kang et al. [12] approach the object discovery problem in a general dataset of images from "daily living": high-resolution, close-range images of a variety of objects in an indoor setting. They combine a hierarchical oversegmentation with visual features and color information to perform unsupervised clustering. In a followup, Kang et al. [15] address the sparsity of the object views in the "daily living" dataset, extending their earlier work to leverage a database of product images scraped from the internet.

By contrast, we collect views using a robot. Our work and Kang's are complementary, as object views collected in the local environment could be augmented with product images, which would likely improve performance. However, many of the objects our robot encounters are truly unique to the environment, and are unlikely to appear in any product database. Furthermore, our approach allows *specificity*: our association algorithms need only distinguish among those objects that actually appear in the the robot's environment, not those from the larger set, easing the association problem.

III. Background

In the literature, "object", "instance", and "class" are heavily overloaded terms. In this work, We use *instance* to mean a specific object in a specific location, as in Figures 1a and 1b. Because instances include location, Figures 1a and 1d are different instances, despite being the same physical object. We define a *class* to be every object of a particular type, independent of position. Under our definition, every object in Figure 1 (as well as Figure 2a) are in the same class: "houseplant."

For a robot to recognize objects accurately, they must be observed more than once. In keeping with the common case in robotic mapping, our system runs as a background process. In this case, these observations will be collected "accidentally"; that is, by the robot while it navigates the environment. To collect sufficiently many views, the robot must be operating on a long-term basis.[1]

To this end, we evaluate our system on the RGB-D dataset (the "Willow Garage" dataset) introduced in our earlier work [5], which is the largest dataset of repeated observations of an environment of which we are aware. The dataset consists of 67 runs of a PR2 robot traversing a large office environment over the course of six weeks and includes observations of a wide variety of objects in many locations. These observations include a wide variety of difficult cases including localization errors, sensor errors, and variations in lighting.

We also evaluate our work on the "large" dataset from Mason et al. [14]. (For clarity, and because it includes object movement, we refer to it as the "Mobile Objects" dataset.) This dataset is smaller (in duration, as well as the number and variety of objects). However, it presents a qualitatively different situation in which a robot observes a series of objects at close range. This approximates a robot executing an active object-search strategy.

IV. System

We begin with a localized RGB-D frame. Supporting surfaces are detected and added to the global plane state. Segments (Section V) are then extracted from the depth image. Instance-level data association (Section VI-A) is then performed, connecting the new segments to existing segments in the same location. Finally, class-level data association (Section VI-B) is performed, connecting the new segments to segments in the same class, but different instances.

At each step, the data are stored in a MongoDB "NoSQL" database, allowing the system to run both online and in batch mode. The database also provides persistence, a prerequisite for long-term operation.

V. Segmentation

Traditional object-recognition algorithms take as input an image and return the location of zero (or more) specific, labeled objects in that image. As we are without recognition, we are limited to unsupervised segmentation. Given an image (an RGB-D pair in our case), we return zero or more *segments*: lists of pixels corresponding to (unlabeled) objects or pieces of objects. Our algorithm is described below. The results are detailed in Section VII.

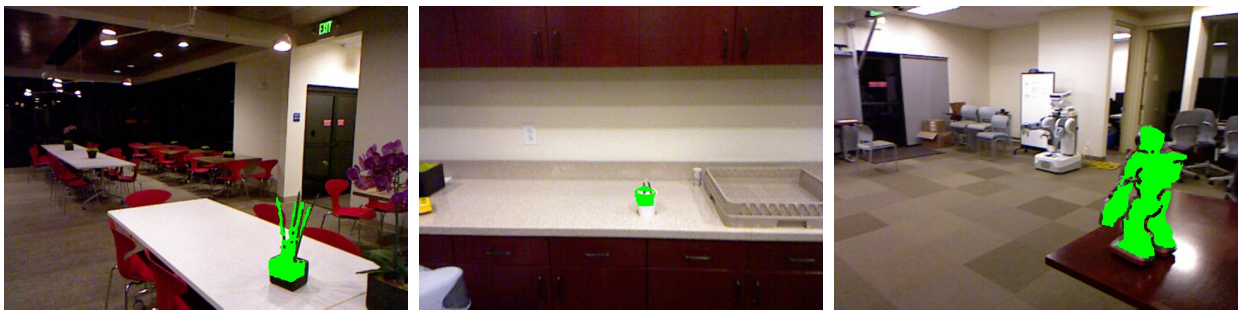
Earlier work in building maps that include objects (e.g. Rusu et al. [8], Trevor et al. [16], Mason and Marthi [5]) performs object segmentation using the *supporting planes assumption*, which we also adopt here. The assumption is that the world contains large, flat, horizontal surfaces, and that objects are those things which rest atop them. While

[1]Note that an active search algorithm could guarantee many views of each object; nothing in our approach precludes this.

(a) (b) (c)

(d) (e) (f)

Fig. 1: Examples of instances and classes (see Section III). First column: two segments corresponding to the same instance. Second column: two segments corresponding to the same physical object as the first column, but a different instance (because the object has moved). Third column: two other segments belonging to the class "houseplant."



(a) A houseplant. (b) A coffee cup. (c) A bipedal robot. (A different instance of this object can be seen in Figure 4.)

(d) An undersegmentation. (e) An RGB-D synchronization error. (f) A non-object.

Fig. 2: Examples of successful ((a), (b), (c)) and failed ((d), (e), (f)) segmentations. In each image, the segment is overlaid in green over the RGB frame. Note that the segment is generated solely from the depth image (hence the possibility of misalignment seen in (e)). See Section V for details. *This figure is best viewed in color.*

this disqualifies certain types of objects (for example, objects resting on the floor, which is not treated as a supporting plane as it trivially supports everything), it contains many objects of interest, particularly those which could be manipulated by a robot. Our earlier work [5] includes some quantitative evaluation of this assumption.

Our implementation operates in the depth image. We maintain a global horizontal-plane state, which tracks every supporting surface in the environment by storing its convex hull. Given a new depth image, planes (of all orientations) are found using the algorithm presented by Trevor [16]. These planes are used to mask out parts of the depth image. Sharp depth discontinuities (depth edges) are also added to this mask. Next, the newly-found horizontal planes are added to the global plane state. Finally, any points from the depth image that are above any supporting plane are used to seed a connected-component analysis in the mask. The resulting components are our segments. Note that this analysis does not always produce a single segment per object, nor do the segments necessarily cover the entire object (for example, consider Figure 4, in which the object is only partially discovered). For evaluation, we allow partial segmentations (so-called "good segments"), and disallow undersegmentations and non-object segments ("bad segments").

Because this technique works in the depth image, it encounters two specific problems: RGB-D misalignment (Figure 2e) and clutter. Without RGB data, cluttered environments like that in Figure 2d run the risk of producing one segment, not several. These errors were surprisingly rare in the Willow Garage dataset (36 segments; 2.4% of the total). This indicates that (at least in our environment), such object configurations are rare. Note that given sufficiently dense data (as in Karpathy et al. [17]) objects can be extracted from clutter directly. However, our passively-collected data do not permit such analysis.

VI. ASSOCIATION

Once the segments are generated (see Section V), we must perform data association. The goal of association is to group the segments (each of which corresponds to part of an object) into meaningful clusters. As the goal of this work is class discovery, we are interested in clusters that correspond to classes (as defined in Section III). As objects in the same instance are necessarily in the same class, we leverage both the properties of instances (in particular, spatial location) and classes (in particular, appearance and 3D shape) to perform association. To quantify the benefits of our class-association step, we measure performance on both instance and class clusters.

We approach association as a graph connectivity problem. In this view, each segment is a node in a graph, and two segments are connected by an undirected edge if they satisfy a set of criteria detailed below. The connected components of the resulting graph are our clusters. These "hard" assignments make our approach brittle to false-positive edges: a single false positive can (in principle) lead to the misassociation of very many segments. In practice, we avoid this problem; see Section VII for details. Note that both instance- and class-level association are performed on the same graph: our technique is not hierarchical, and the connectivity criteria apply to pairs of segments, not pairs of instances. Of course, other clustering techniques are possible; we chose graph connectivity for its simplicity.

A. Instance Association

The need for instance-level association can arise in two ways: viewpoint changes and partial segments. As segments are extracted on a per-frame basis, two different views of the same object are necessarily different segments. Secondly, our segmentation can oversegment or generate partial segments.

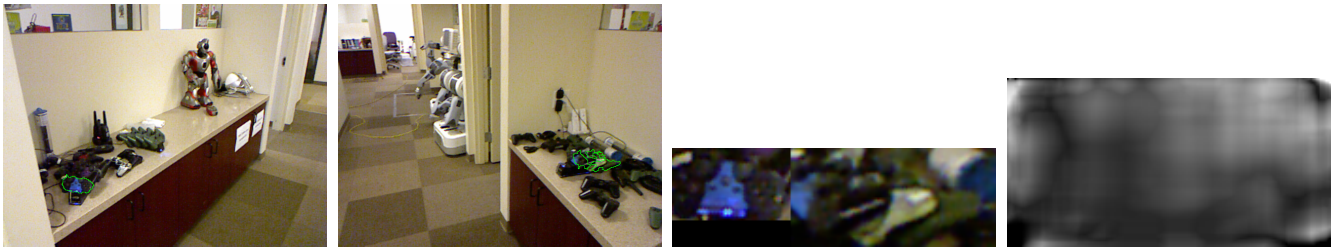
As our robot is localized, we can position the point clouds from our RGB-D sensor in a shared coordinate frame. We use this information to determine if two segments overlap, and therefore if they are part of the same instance. However, localization is only accurate to a few centimeters, so techniques that rely solely on localization (e.g. Mason and Marthi [5]) suffer from both false-positive associations (when two different segments are "smudged together" by localization mistakes) and false-negative associations (when two segments are pulled apart). At the instance level, we improve on pure localization by correcting for localization errors and by considering three-dimensional object overlap.

Consider two segments, s and t, and the RGB-D frames that generated them, f_s and f_t. To determine if s and t are part of the same instance, we need to check them for spatial overlap. However, doing so accurately requires correcting for localization error. In full generality, correction would be done using a SLAM algorithm, but SLAM adds considerable implementation and computational complexity. We do something simpler: pairwise alignment.

We begin by projecting the points corresponding to s and t into two dimensions by discarding the z (vertical) axis. We then compute the convex hulls of these projected points, and check to see if the hulls overlap (our earlier work [5] stops here). If the hulls overlap at all, we proceed to the next step: otherwise, we do not add an edge between s and t. This filtering step is strictly an optimization: the full alignment is computationally expensive.

Given some convex-hull overlap, we proceed to align the full point clouds of f_s and f_t using the Iterated Closest Point (ICP) algorithm [18] (as implemented in PCL [19]). We initialize ICP using the transformation estimate provided by localization. As ICP is run between full frames (each of which may contain several segments) the results are cached (in the database; see Section IV).

Next, we compute the three-dimensional overlap between s and t. Computing the overlap between the three-dimensional convex hulls is sensitive to noise: consider the case of minor undersegmentation where a single point on the background has been included. The resulting hull would be forced to include the volume between the object and background, which could be large relative to the size of the object. We compute an approximation to volumetric overlap: voxel grid overlap. Consider a dense grid of voxels, each



(a) (b) (c) (d)

Fig. 3: Histograms for appearance-based association, as discussed in Section VI-B. In these figures, we color the outline of the segment (rather than every pixel) to leave the colors visible. Figure (c) shows the the "zoomed" versions of the regions, and the resulting heatmap is shown in (d). The appearance cost assigned to this pair is equal to the minimum value in the map.

(a) (b)

Fig. 4: An example requiring instance association. These segments both belong to a single instance of a bipedal robot. Another instance of the same object can be seen in Figure 2c.

$1 \times 1 \times 1$cm, covering the entire environment. For each point p in s, we compute which voxel contains p, and add that voxel to a set. We do the same for t, and our three-dimensional overlap is the number of voxels in the intersection of these two sets. We do not require that the set of voxels be connected: in our point-on-the-background example, we would introduce only one extraneous voxel. The voxelization operation is extremely fast: to convert a point (whose coordinates are in meters) into voxel coordinates, all that is needed is to multiply by a constant (in this case, 100: converting from meters to centimeters) and convert from a floating-point representation to an integer. Because the segments have small spatial dimensions, we can store just those voxels that occur, rather than allocating the dense grid described above, making the entire operation fast and memory-efficient. Let S and T denote the voxel sets for segments s and t. We declare that s and t overlap if $|S \cap T| \geq 0.1|S|$ and $|S \cap T| \geq 0.1|T|$. We also declare an overlap if $S \subseteq T$ or $T \subseteq S$. If s and t overlap, the edge (s, t) is added to our graph. Our improved analysis substantially outperforms the simple two-dimensional technique; see Section VII and Figure 6 for details.

B. Class-level Association

Associating segments into classes ("class discovery") is the more fundamental problem and is the primary goal of this work. Unlike instance association, class discovery cannot rely on location information: segments belonging to the same class can occur in any location. For example, every segment in Figure 1, as well as the segment in Figure 2a, is a member of the class "houseplant," despite appearing in a variety of locations. To discover object classes, we must therefore consider other information. Our algorithm relies on two basic assumptions: segments that belong to the same class should have similar appearance and similar shape.

As any two segments in the same instance are necessarily in the same class, we begin by running instance-level association. Class-level association then (potentially) adds more edges to the graph, and our classes are the connected components of the result.

A common approach for measuring appearance (e.g. Kang et al. [12]) is to measure the distance between color histograms. Because the histogram discards the geometry of the object, histogram distance has the advantage of being robust to alignment errors. However, it is not robust to certain kinds of partial segments. Consider the two segments shown in Figure 3. These are two instances of the same class (a rack of video-game controllers). Because Figure 3a is a partial segment, its color histogram is primarily blacks and blues, while Figure 3b has a more uniform histogram, including greens and whites. To permit matches in such cases, we perform a search over possible alignments of the two segments, computing a histogram distance at each alignment.

We do this by taking the rectangular bounding box of both segments. The smaller rectangle is then swept across the larger rectangle, and the histogram distance between the overlaps is computed. This process creates a heatmap, as seen in Figure 3d. The final appearance distance between two segments is the smallest value in the heatmap.[2]

The discussion above omits an important fact: by working directly in pixel space, the histogram analysis is sensitive to scale. Consider the bipedal robot seen in Figure 2c and Figure 4. These are two instances of the same object, but taken from different distances. As a result, a pixel in

[2]Although our implementation computed this value by brute force, our sliding window search is an excellent candidate for the Efficient Subwindow Search algorithm of Lampert et al. [20].

(a) (b)

Fig. 5: An example of the difficulty posed by using only appearance in matching segments. Both segments seen here are flat white, but should not be matched.

one segment corresponds to a different amount of physical space than a pixel in the other segment. To correct for this, we would like to "zoom in" the more-distant segment until we are observing it from the same distance as we are observing the closer segment. Because we have depth information, the distance to each segment is known. Let z_s denote the average distance to the points in segment s, z_t denote the same for segment t, and let $z_s < z_t$. Under weak perspective projection, "zooming in" s is equivalent to simply enlarging s by the factor z_s/z_t. We perform this correction before performing the overlap analysis described above. As our camera has finite resolution, we are necessarily interpolating between pixels in the more-distance segment. To avoid matching image-scaling artifacts, we skip entirely those segment pairs where $z_s - z_t$ is greater than one meter.

Our implementation uses RGB histograms with four buckets per channel, compared using the total-variation distance.

Next, we compute the height, width, depth, and total (voxelized) volume of each segment, and require that they each differ by no more than a fixed threshold.

Finally, we note that appearance alone can fail. Consider the two segments in Figure 5. Both are basically uniform white, but one is a mug, while the other is a bowl. Furthermore, their approximate dimensions are the same. Therefore, we introduce one more cue: a general shape descriptor. We use the Viewpoint Feature Histogram [2], which computes a single 308-element descriptor for each segment, and compute a distance between segments using the χ^2 distance.

We introduce edges to our graph for those pairs of segments whose histogram cost is below a threshold H, whose ratio of height, width, and depth are each above a threshold V, and whose shape-cost difference is below a threshold F. As before, we then perform a connected-component analysis on the resulting graph, and deem each connected component to be a class. Section VII contains results and Section VII-A details our thresholds and their values.

VII. RESULTS

We evaluate our system on two datasets (Willow Garage and Mobile Objects). To evaluate segmentation quality, we assigned each of the segments to one of three categories: non-objects, undersegmentations, and correct segments. Non-object segments do not correspond to any object (as in Figure 2f). Undersegmentations (Figure 2d) correspond to more than one object, while everything else is a correct segment. Note that "correct" includes oversegmentations. We rely on our instance- and class-level clustering steps to join up oversegmented objects correctly.

In the Willow Garage dataset, segmentation produced 1519 segments. Of these, 36 (2.4%) are undersegmentations, and 183 (12%) are non-objects, giving a total of 219 (14.4%) "bad" segments. In the Mobile Objects dataset, it produced 103 segments. Of these, 2 (1.9%) were undersegmentations, and 8 (7.7%) were non-objects, giving a total of 10 (9.7%) "bad" segments.

To analyze the instance- and class-level clustering, we manually assigned each "good" segment to an instance and each instance to a class ("bad" segments have no meaningful instance or class label). In the Willow Garage dataset, this produced 179 instances of 86 distinct object classes (see Figure 7). In the Mobile Objects dataset, this produced 15 instances drawn from 10 classes.

Consider two segments s and t. In the ground-truth labeling, s and t can be *disconnected* (not in the same class, and therefore not in the same instance), *intra-instance connected* (in the same instance, and therefore the same class) and *inter-instance connected* (in the same class, but not in the same instance). However, our association algorithm can make only two choices: s and t may or may not be in the same connected component.

We compute three values: classwise precision, intra-instance recall, and inter-instance recall. For classwise precision, we consider every pair (s, t) of segments that our algorithm has associated. The pair is a *true positive* if s and t are in the same ground-truth class, and a *false positive* otherwise. Classwise precision is defined as true positives divided by total positives, and tells us what percentage of our positive associations are correct. To compute intra-instance recall, consider every (s, t) pair in the same ground-truth instance. Intra-instance recall is the fraction of these pairs such that s and t are associated by our algorithm. Intra-instance recall tells us how completely our algorithm recovers instances from segments. Inter-instance recall is similar to intra-instance recall, but considers the (s, t) pairs such that s and t are in the same ground-truth class, but different ground-truth instances. Note that all s and t in the same ground-truth class must have either intra- or inter-instance connections (but not both). Inter-instance recall tells us how completely we discover the class structure of the data.

We compute these values over all (s, t) pairs, but do not double-count: if (s, t) is considered, (t, s) and (s, s) are not. We describe our choice of parameters below, and present the results in Figure 6.

A. Parameters

As noted above, our association algorithm has three parameters: the appearance threshold H, the spatial threshold V, and the shape-cost threshold F. As with any unsupervised technique, values for these parameters must be chosen. Here,

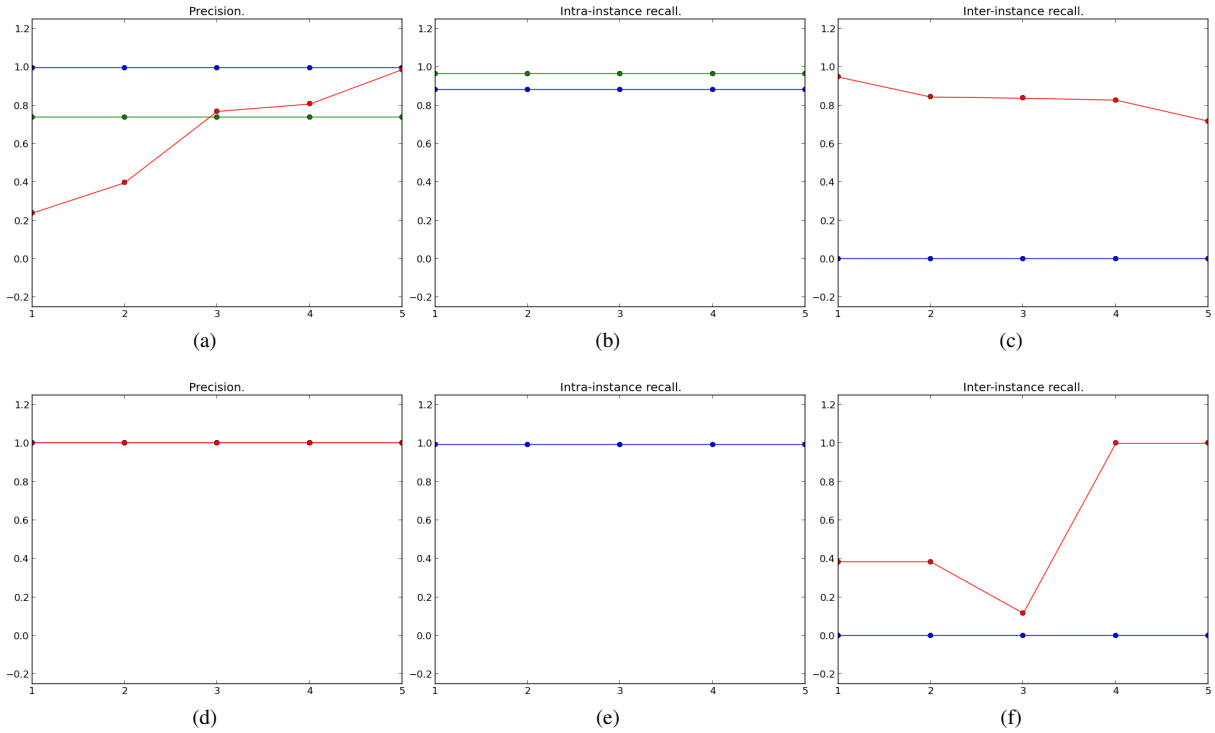


Fig. 6: Results of our algorithm (Section VII) on the Willow Garage (top row) and Mobile Objects (bottom row) datasets. In each figure the the blue points correspond to running only instance-level association (Section VI-A), while the red points include class-level association (Section VI-B), and the green points include only the two-dimensional instance-association discussed in Section VI. The horizontal axis specifies the training set used in our parameter-tuning approach: 1 corresponds to training on the first fifth of the data, 2 to training on the first two-fifths of the data, and so on. Each point is computed by evaluating over the entire dataset. On Willow Garage, our three-dimensional instance-level association (blue) finds very few false positive associations (a), but finds effectively zero inter-instance associations (c). As the three-dimensional technique is a refinement of the two-dimensional techinque, it achieves higher precision (a) at the cost of reduced recall (b). On that dataset, our complete algorithm (red) maintains 98.6% precision while discovering 71.8% of the inter-instance relationships (rightmost point in (c)) Because we use a single graph for both instance- and class-level association, adding edges generated by class-level association removes the distinction between an instance and a class. Therefore, no red line appears in (b). We hypothesize that the dip seen in (f) is due to simple bad luck: many choices of parameters achieve 100% precision and recall on that training set, and the arbitrarily-chosen winner generalized poorly.

we present a technique for parameter selection that leverages the temporal nature of robotic exploration.

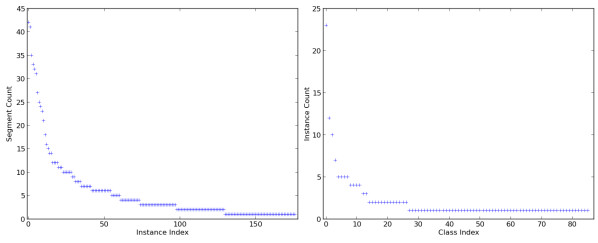
Nothing about our object discovery algorithm requires that the parameters be chosen in this way. We describe our approach to make our choice of parameters transparent, and to support deploying our algorithm in a novel environment.

We chose the parameters automatically using a sweep: for each training set (detailed below), we varied H from 0.01 to 0.55 by steps of 0.01, V from 0.5 to 1.0 by steps of 0.01, and F from 1 to 300 by steps of 10. (The ranges and step sizes were determined manually.) At each step, we recorded the parameter setting that generated the highest inter-class recall while also maintaining a precision (on the training set) of 0.98 or greater. This high precision threshold is suggested by our use of hard association decisions: a false-positive association between two segments can lead to two large clusters being incorrectly associated.

Traditional cross-validation is a poor fit for our problem, as a held-out set that is a small fraction of the total data will contain few (s, t) pairs that should be connected. As a result, few potential connections will be considered, let alone found. Instead, consider what might happen should we deploy our system in a novel environment. The system would begin with default values for the parameters. Should performance prove poor, a set of segments would be hand-labeled, and used to train new parameters. Should these new parameters not prove good enough, further data would be labeled. (Because of our emphasis on high precision, the inferred labels would provide a high-quality starting point for labeling further data.)

We simulate this process by sorting our segments by the time of their observation, and then partitioning them into five groups. We perform our parameter sweep only on group one, then on groups one and two, and so on. Rather than evaluate on a held-out set, we evaluate on the entire dataset at each step. The results of this process are detailed in Figure 6.

The parameters found by training on the entire Willow



(a) (b)

Fig. 7: Ground truth instance and class counts for the Willow Garage dataset. Figure (a) plots the number of segments that belong to each (hand-labeled) instance. Figure (b) plots the number of instances (not segments) that belong to each ground-truth class. Two instances are left off of Figure (a) for reasons of scale; they contain 213 segments and 94 segments.

Garage dataset (rightmost data point in Figure 6a) were $H = 0.21$, $V = 0.84$, and $F = 121$. These parameters achieved a precision of 98.7% and inter-instance recall of 71.8%.

For the Mobile Objects dataset (rightmost data point in Figure 6f), the values are $H = 0.35$, $V = 0.71$, and $F = 291$, which achieve both 100% precision and 100% inter-instance recall.

VIII. CONCLUSIONS AND FUTURE WORK

We have presented a novel method for leveraging the capabilities of a mobile to robot to discover objects efficiently and accurately in classes in large, general settings. By combining the 3D capabilities of a modern robot with classic RGB image analysis, we segment objects from the world, and then group them into instances and classes. Importantly, the system can operate unsupervised: no manual segmentation or labeled training sets are required, and the system has only three parameters, which can be easily tuned to the environment given minimal user feedback.

Our experiments demonstrate our ability to discover 71.8% of the inter-segment connections while maintaining a precision of 98.6%. That we can achieve this impressive performance by combining standard algorithms demonstrates the value of using a robot, not just a camera, for object and class discovery and proves the feasibility of unsupervised object and class discovery in general settings. Beyond proving the concept, our algorithm also provides a powerful way of discovering what object classes occur in an environment.

These results suggest several directions for future work, including incorporating active search strategies, instance-to-instance (rather than segment-to-segment) association algorithms, and databases of "known" objects to improve performance. Another possibility is to leverage the scalability of our system to learn object "behavior": where object classes tend to appear. For example, are coffee cups more common in the kitchen or the dining room? Such a model could be used to plan efficient search strategies.

Finally, the most obvious direction for future work is to bring a wider range of computer vision and clustering techniques to bear on this problem. We do not intend for this work to be the last word on this topic. Instead, we hope that the most important contribution of this work will be to demonstrate the feasibility of object discovery as an embodied, long-term task, and to invite broader participation in this endeavor through the baseline and benchmarks we have provided.

REFERENCES

[1] H. Moravec and A. E. Elfes, "High Resolution Maps from Wide Angle Sonar," in *IEEE International Conference on Robotics and Automation*, March 1985.

[2] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D Recognition and Pose Using the Viewpoint Feature Histogram," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," *International Conference on Computer Vision*, Nov. 2011.

[4] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft, "RoboEarth," *Robotics Automation Magazine*, vol. 18, no. 2, 2011.

[5] J. Mason and B. Marthi, "An Object-Based Semantic World Model for Long-Term Change Detection and Semantic Querying," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[6] J. Mason, "Object Discovery with a Mobile Robot," Ph.D. dissertation, Duke University, June 2013.

[7] A. Nüchter, O. Wulf, K. Lingemann, J. Hertzberg, B. Wagner, and H. Surmann, "3D Mapping with Semantic Knowledge," in *RoboCup 2005: Robot Soccer World Cup IX*, 2006.

[8] 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," *Intelligent Robots and Systems*, Oct. 2009.

[9] C. Galindo, J.-A. Fernández-Madrigal, J. Gonzalez, and A. Saffioti, "Robot Task Planning using Semantic Maps," *Robotics and Autonomous Systems*, 2008.

[10] A. Pronobis and P. Jensfelt, "Large-scale Semantic Mapping and Reasoning with Heterogeneous Modalities," in *ICRA*, 2012.

[11] N. Blodow, D. Jain, Z. Marton, and M. Beetz, "Perception and Probabilistic Anchoring for Dynamic World State Logging," in *IEEE-RAS International Conference on Humanoid Robots*, Dec. 2010.

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

[13] 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.

[14] J. Mason, B. Marthi, and R. Parr, "Object Disappearance for Object Discovery," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 2836–2843.

[15] H. Kang, M. Hebert, A. A. Efros, and T. Kanade, "Connecting Missing Links: Object Discovery from Sparse Observations Using 5 Million Product Images," in *European Conference on Computer Vision*, 2012.

[16] A. Trevor, "PCL::Segmentation — planes, clusters, and more," in PCL tutorial at IROS 2012. As of submission unpublished material, but available at pointclouds.org.

[17] A. Karpathy, S. Miller, and L. Fei-Fei, "Object Discovery in 3D scenes via Shape Analysis," in *International Conference on Robotics and Automation (ICRA)*, 2013.

[18] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, 1992.

[19] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *ICRA*, 2011.

[20] C. H. Lampert, M. B. Blaschko, and T. Hofmann, "Efficient Subwindow Search: A Branch and Bound Framework for Object Localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, 2009.