Article

The International Journal of Robotics Research
2015, Vol. 34(1) 3–25

DOI: 10.1177/0278364914546030
ijr.sagepub.com
SAGE

HerbDisc: Towards lifelong robotic object discovery

Alvaro Collet[1], Bo Xiong[2], Corina Gurau[3], Martial Hebert[1] and Siddhartha S. Srinivasa[1]

Abstract

Our long-term goal is to develop a general solution to the lifelong robotic object discovery (LROD) problem: to discover new objects in the environment while the robot operates, for as long as the robot operates. In this paper, we consider the first step towards LROD: we automatically process the raw data stream of an entire workday of a robotic agent to discover objects. Our key contribution to achieve this goal is to incorporate domain knowledge (robotic metadata) in the discovery process, in addition to visual data. We propose a general graph-based formulation for LROD in which generic domain knowledge is encoded as constraints. To make long-term object discovery feasible, we encode into our formulation the natural constraints and non-visual sensory information in service robotics. A key advantage of our generic formulation is that we can add, modify, or remove sources of domain knowledge dynamically, as they become available or as conditions change. In our experiments, we show that by adding domain knowledge we discover 2.7× more objects and decrease processing time 190 times. With our optimized implementation, HerbDisc, we show for the first time a system that processes a video stream of 6 h 20 min of continuous exploration in cluttered human environments (and over half a million images) in 18 min 34 s, to discover 206 new objects with their 3D models.

Keywords

Object discovery, sensor fusion, lifelong learning, object segmentation, objectness

1. Introduction

An important goal in the field of service robotics is to achieve lifelong autonomy, interacting with non-expert users and operating in regular households. In this scenario, a robot should learn about the objects in the household to manipulate them; and learn constantly, as things change often. Large databases of precomputed objects are useful for common objects, but discovering new objects in the environment is critical for true lifelong autonomy in service robotics. Our long-term goal is to develop a general solution to the problem of discovering new objects in the environment while the robot operates, for as long as the robot operates. We term this problem as the *lifelong robotic object discovery* (LROD) problem. A specialization of the Unsupervised Object Discovery problem (e.g. Russell et al., 2006; Kang et al., 2011), LROD focuses on massive datasets of dynamic human environments gathered by a robotic agent.

As a first step towards LROD, we automatically process the raw video stream of an entire workday of a robotic agent. Considering the autonomy and charging times of current service robots (e.g. Srinivasa et al., 2010), a robotic workday amounts to approximately 6–8 hours of raw sensor data (e.g. RGBD video feed) and over half a million data samples (e.g. RGBD images). We show for the first time a system that processes, in under 19 minutes, hundreds of thousands of samples and over 6 h of continuous exploration, to discover hundreds of new objects in cluttered human environments.

The goal of unsupervised object discovery is to jointly segment and learn the appearance of unknown objects in the environment. Unsupervised object discovery is a very challenging problem, partially because it is ill-defined: there is no clear definition of *object*. Most research in this area models objects as recurring patterns in the data, thus attempting to jointly segment and learn the appearance of

[1]The Robotics Institute, Carnegie Mellon University, USA
[2]Connecticut College, USA
[3]Jacobs University, Germany

Corresponding author:
Alvaro Collet, The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA.
Email: acollet@cs.cmu.edu

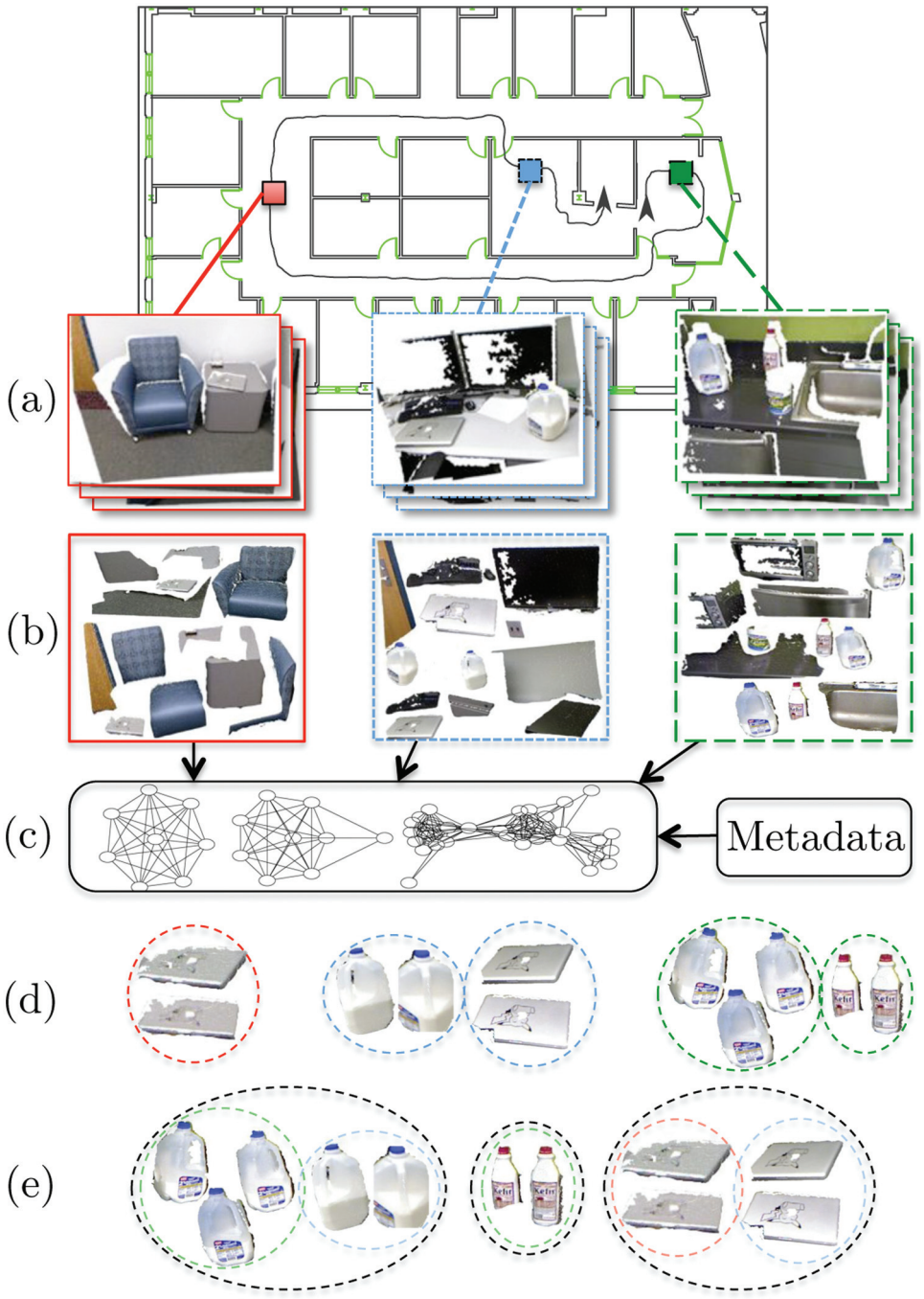

Fig. 1. Example of object discovery with metadata. (Top) Robotic agent navigates through office environment storing an RGBD video stream and localization information. (a) Spatial/temporal constraints separate the video stream in subsets red, green and blue. (b) Images in the sequence are segmented to generate object candidates. (c) Object Discovery with Metadata: the different sequence subsets are processed independently for efficiency, using robot localization, object size/shape constraints and external knowledge to find (d) individual object instances. (e) Global object discovery performed on discovered object instances (d) to obtain a single representation for each object.

objects by searching for such patterns. Generic techniques for unsupervised object discovery do not scale well to the volume and complexity of the LROD input data stream. In massive datasets of millions of samples, with cluttered scenes containing many small and similar-looking objects, recurring visual patterns appear everywhere. In addition, the computational cost of finding recurring patterns skyrockets as the amount of data to process grows.

Consider the example in Figure 1, in which a robotic agent navigates through an office environment recording an RGBD video stream (Figure 1(a)). Unsupervised object discovery techniques (e.g. Kang et al., 2011) create a pool of

object candidates (e.g. the RGBD regions in Figure 1(b)), which are represented as nodes in a pairwise graph (Figure 1(c)). The graph edges are computed by comparing the visual similarity between every pair of object candidates. Then, clustering techniques are used to group similar object candidates, i.e. recurring patterns, (Figure 1(d) and (e)). Building the pairwise graph requires $\mathcal{O}(n^2)$ similarity comparisons; as the length of the video stream grows, this cost becomes prohibitively expensive. Most of the computation time is spent comparing candidates with very low likelihood of being grouped together (e.g. the candidates in the corridor in Figure 1(b)(left) and the kitchen in Figure 1(b)(right)). If we analyze the input data stream based on the visual information alone, we are forced to evaluate every pair of object candidates. However, we know intuitively that objects in the kitchen and objects in the corridor have little in common. We also know that two data samples acquired in the same location within a few seconds of each other are more likely to contain the same objects than data samples acquired in different years. We can use this external information, this *metadata*, to drastically reduce the computation time and improve the quality of the discovered objects.

Our key insight to making LROD feasible is to incorporate *robotic metadata*. The data gathered by a service robot is not just an unordered collection of anonymous images. First, range data is also available. In addition, we may also know where and/or when the images are captured, as well as their ordering; we may know where interesting objects usually appear for a particular environment, such as in tables or cabinets; we may have additional sensing (e.g. robot localization, odometry) for some or all of the images; or we may only be interested in objects of certain sizes or shapes relevant to the robot. In our work, we define *robotic metadata* as *any* source of additional information about the visual/range data. Our definition of robotic metadata includes any additional robotic sensing, assumptions, or prior information about the objects, environment, the robot's sensors, or task; in short, any non-visual data that can provide information about the object candidates. Consider the example in Figure 1, now using metadata. A robotic agent navigates through an office environment recording an RGBD video stream, using the robot's location and data acquisition timestamps to separate the data stream (the red–blue–green subsets in Figure 1(a)). The object candidates for each subset (Figure 1(b)) are compared only within the same subset. The pairwise graphs in Figure 1(c) encode the visual similarity between candidates, as well as other cues such as if candidates overlap in space, or object priors based on the robot's grasping capabilities. In the clustering step (Figure 1(d)), we group object candidates with similar visual information and metadata. The metadata-augmented similarity graphs encode local information to discover individual object instances, and we may discover multiple instances of the same objects in different data subsets. We perform a global clustering step (Figure 1(e)) to join the multiple object instances as single object models.

The main theoretical contribution of this work is a general framework for object discovery that leverages *any* form of metadata, and in particular the natural constraints that arise in service robotics scenarios. Multiple works in the robotics literature use specific types of metadata (see Section 3 for references), often by imposing restrictions on the environment, data acquisition, or agent motion, to improve performance at the cost of limited applicability when the assumptions are violated. Specific solutions could be implemented to use particular sources of metadata, but the solutions would lack adaptability, degrading with any environment changes during the lifetime of the robotic agent. For LROD, we need instead a general architecture to opportunistically leverage and adapt to the available metadata, and incorporate new metadata as it becomes available.

In our formulation, we do not distinguish between visual similarity and robotic metadata. We encode all similarities and metadata as an intermediate representation that we term a *constraint*. The definition of a constraint is very simple: a *measurable* yes/no question about an object candidate or a relationship between candidates, with some p, a probability of success, about the answer. For example, an appearance similarity function $s(\cdot,\cdot)$ is encoded as the constraint "are candidates h_i and h_j similar in appearance?". The answer would be yes/no, with probability $p = s(h_i,h_j)$. Metadata can be similarly encoded as constraints, as we describe in detail in Sections 5 and 7.

With this intermediate representation of constraints, we can seamlessly combine multiple similarities and other metadata sources. We define a set of logic operations over constraints to form complex constraint expressions that encode all of our knowledge relevant to discover objects. We formulate the general LROD problem as a distributed partitioning of graphs built over constraints, which we term constrained similarity graphs (CSGs). Our distributed graph partitioning formulation is shown in Figure 1(d) and (e), and the CSGs are illustrated in Figure 1(c).

These CSGs, when coupled with service robotics constraints, are by construction much sparser than regular visual similarity graphs, and produce many connected components. With service robotics constraints, this graph sparsity effectively reduces the number of visual similarities to compute (the most expensive operations) from $\mathcal{O}(n^2)$ (with respect to the number of images n) to $\mathcal{O}(n)$, as well as greatly improving the performance of the graph partitioning algorithm. In addition, our constraints-based formulation is general, in the sense that it covers both generic unsupervised object discovery algorithms (e.g. Russell et al., 2006; Kang et al., 2011) and purpose-specific algorithms (e.g. Morwald et al., 2010).

Our main applied contribution is HerbDisc, an optimized implementation of this framework in the robotic system HERB (Srinivasa et al., 2010). Our framework seamlessly integrates visual and 3D shape similarity with spatial and temporal constraints, size/shape object priors, and motion information in a flexible and extensible way. We drove our service robot to over 200 offices from 4

floors of a university building, recording 6 h 20 min of continuous RGBD video of real human environments, totaling over half a million images. HerbDisc processed this dataset in 18 min 34 s using a single quad-core machine and discovered 206 novel objects (44.5% precision, 28.6% recall), showcasing both the efficiency of this framework and the robustness of its results.

Preliminary versions of this work have been published at Collet (2012); Collet et al. (2013).

2. Problem formulation

Consider the example of robotic object discovery shown in Figure 2. We identify five major components. The *World* Ω represents the underlying physical phenomenon (i.e. the environment) where we discover objects. A *physical agent* $\mathcal{A}$ (e.g. a robot) gathers data through observation or interaction with the world Ω. The physical agent uses *sensors* $\mathcal{S}$ (e.g. a camera) to gather data samples I (e.g. images). A *candidate generator* $\mathcal{H}$ produces object candidates h from data samples. Finally, the *discoverer* $\mathcal{D}$ groups recurring object candidates into objects.

In this paper, we describe a general architecture for an object discoverer $\mathcal{D}$ that uses metadata from the world Ω, the physical agent $\mathcal{A}$ and the sensors $\mathcal{S}$, alongside visual information from the object candidates h, to discover objects robustly and efficiently.

2.1. Inputs and outputs

The visual input to HerbDisc is a set $\boldsymbol{I}$ of N images with associated range data:

$$\mathbf{I} = \{I_1, \ldots, I_n, \ldots, I_N\}, \quad I_n = \{I_n^{\boldsymbol{rgb}}, I_n^{\boldsymbol{P}}\} \tag{1}$$

where $I_n^{\boldsymbol{rgb}}$ is the set of color RGB values in image n, and $I_n^{\boldsymbol{P}}$ is the set of 3D points available from the viewpoint of image n.

A candidate generator $\mathcal{H}$ generates a set of data fragments $\boldsymbol{h}$ from image and range data in $\boldsymbol{I}$, which we consider the object candidates. Each object candidate

$$h_i = \{h_i^{\boldsymbol{rgb}}, h_i^{\boldsymbol{P}}, h_i^{\Phi}\} \tag{2}$$

is characterized by a set of color pixels $h_i^{\boldsymbol{rgb}}$, a set of 3D points $h_i^{\boldsymbol{P}}$, and a set of metadata attributes h_i^{Φ}.

The output of this framework is a set of metric 3D models of objects $\boldsymbol{M}$. Each object model

$$M_k = \{M_k^{\boldsymbol{rgb}}, M_k^{\boldsymbol{P}}, M_k^{\boldsymbol{h}}\} \tag{3}$$

is characterized by the set of object candidates $M_k^{\boldsymbol{h}} = \{h_{1,k}, \ldots, h_{i,k}, \ldots\}$ used to create object M_k, and by the set of colored 3D points $M_k^{\boldsymbol{rgb}}$, $M_k^{\boldsymbol{P}}$ that comprise its 3D model.

2.2. Constraints

Constraints encode generic information about an object candidate h_i or a relationship between candidates h_i, h_j. In our formulation, we define these constraints as *node constraints* Θ^{n} and *edge constraints* Θ^{e}, respectively. We model each constraint Θ as a Bernoulli distribution with probability of success p (and, conversely, a probability of failure $q = 1 - p$). Node constraints Θ^{n} encode information about a single object candidate h_i,

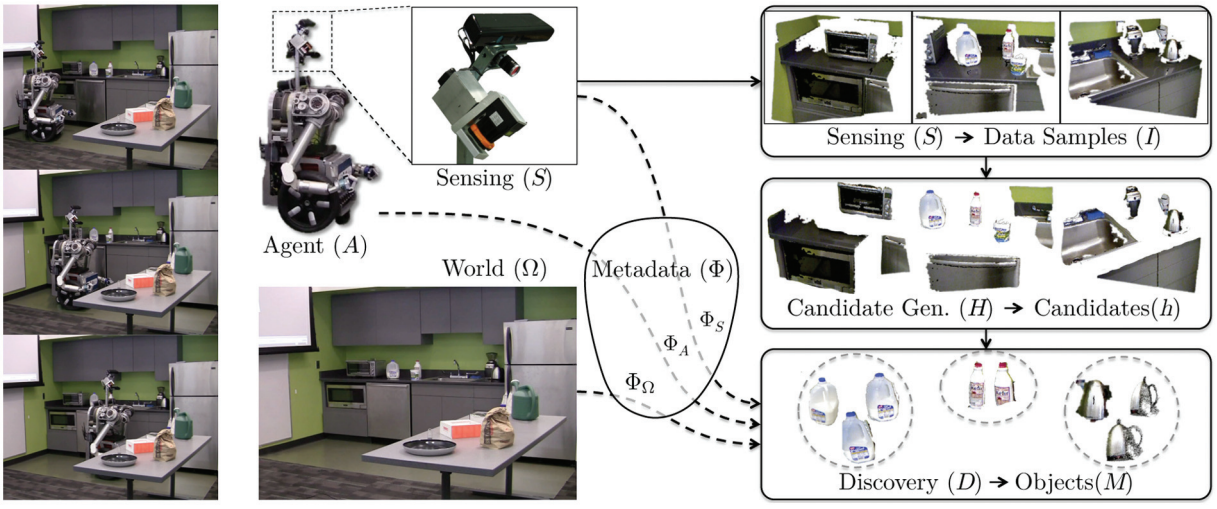

Fig. 2. Main components in robotic object discovery. (Left) The robot HERB moves through a kitchen searching for novel objects. (Center) The three physical components of robotics object discovery are: the world Ω, the robotic agent $\mathcal{A}$, and the sensors $\mathcal{S}$. (Right) The sensors capture data samples x to be processed by a candidate generator $\mathcal{H}$ to produce object candidates. The discoverer $\mathcal{D}$ groups recurring object candidates into objects, using candidate data and metadata sources Φ_Ω (e.g. assumption "objects lie on tables"), $\Phi_\mathcal{A}$ (e.g. robot localization data), Φ_S (e.g. image ordering and timestamps).

Table 1. Constraints used in HerbDisc. For each constraint Θ_i, we provide: the type of information encoded in Θ_i; whether Θ_i is applied on a single object candidate (node) or a relation between a pair of candidates (edge); the information source(s) encoded in Θ_i; uses any metadata or not; a short description of the meaning of Θ_i; and the section in which Θ_i is described in detail. The possible sources of information are: Φ_Ω (metadata about the environment), $\Phi_\mathcal{A}$ (metadata about the robot), $\Phi_\mathcal{S}$ (metadata about the sensors), or V (visual information).

Constraint	Type	Information	Source	Description	Section
Θ_{motion}	Node	Relative camera motion	$\Phi_\mathcal{A}$	Acquire data samples only if there is motion (no repeated frames)	Section 7.2
Θ_{seq}	Edge	"Data comes in sequences"	$\Phi_\mathcal{S}$	Split data stream in short sequences based on camera motion and maximum sequence length	Section 7.2
Θ_{support}	Node	"Objects have surfaces of support"	Φ_Ω	Reject candidates not supported by horizontal or vertical planes (tables or walls)	Section 7.1
Θ_{static}	Edge	"Scene is static for a few seconds"	Φ_Ω	Measure 3D overlap between candidates	Section 7.3
Θ_{size}	Node	Object size	Φ_Ω	Compare candidate's size with object prior	Section 7.4
Θ_{shape}	Node	Object shape	Φ_Ω	Compare candidate's shape with object prior	Section 7.4
Θ_{app}	Edge	Visual similarity	V	Compare visual similarity between candidates using color histograms	Section 7.5
Θ_{3D}	Edge	Shape similarity	V	Compare shape similarity between candidates using FPFH features	Section 7.5

$$\Theta^{\text{n}} : h_i \mapsto \{0, 1\} \tag{4}$$

$$P(\Theta^{\text{n}}(h_i) = 1 | h_i) = p \tag{5}$$

Analogously, edge constraints Θ^{e} encode information about the relationship between a pair of object candidates h_i, h_j, such that

$$\Theta^{\text{e}} : h_i, h_j \mapsto \{0, 1\} \tag{6}$$

$$P(\Theta^{\text{e}}(h_i, h_j) = 1 | h_i, h_j) = p \tag{7}$$

We provide a list with all of the constraints used in this paper in Table 1.

In the interest of brevity, we use the shorthand notation $P_{\Theta^{\text{n}}}(h) \equiv P(\Theta^{\text{n}}(h) = 1|h)$ and $P_{\Theta^{\text{e}}}(h_i, h_j) \equiv P(\Theta^{\text{e}}(h_i, h_j) = 1|h_i, h_j)$ in the remainder of this paper.

3. Related work

The aim of unsupervised object discovery (Weber et al., 2000; Russell et al., 2006) is to jointly segment and learn the appearance of unknown objects in the environment. Unsupervised object discovery is very challenging, in part because the definition of object is subjective, as it depends on the observer. Furthermore, different works use different input sources (e.g. unorganized collections of images (Weber et al., 2000; Kang et al., 2011), image sequences (Morwald et al., 2010), images with disparity (Somanath et al., 2009), laser data (Ruhnke et al., 2009)) to produce different data outputs (e.g. clusters of images (Weber et al., 2000), clusters of bounding boxes (Lee and Grauman, 2011), clusters of image segments (Russell et al., 2006; Kang et al., 2011), 3D models (Somanath et al., 2009)), and using different assumptions (e.g. one object per image (Weber et al., 2000), only tabletop scenes (Kootstra and Kragic, 2011), multiple views of the same scene (Herbst et al., 2011)) depending on the application. Comparing the performance between methods is very challenging due to this disparity in inputs, definition of objects, assumptions, and outputs.

Methods in unsupervised object discovery that assume an unorganized collection of images as input are very common in computer vision research (e.g. Weber et al., 2000; Sivic et al., 2005; Russell et al., 2006; Philbin et al., 2010; Kang et al., 2011; Lee and Grauman, 2011; Kang et al., 2012, and the general survey of Tuytelaars et al. (2009)). Using an unorganized collection of images as input implies, in terms of Figure 2, that we assume no knowledge about the world, the physical agent, or the sensing. Some of these methods, such as Weber et al. (2000); Tuytelaars et al. (2009), focus only on grouping entire images in categories (i.e. assuming that each image mostly contains a single, large object), which is equivalent to not using a candidate generator $\mathcal{H}$.

The key difference between unsupervised object discovery and LROD is the amount and variety of information sources. Most methods in unsupervised object discovery assume that no information is available about the world Ω, the physical agent $\mathcal{A}$, or the sensors $\mathcal{S}$. As datasets grow larger, visual information becomes less discriminative and recurring visual patterns appear everywhere. In addition, algorithms often require pairwise operations over all pairs of candidates, which makes them computationally expensive. In LROD, metadata (i.e. non-visual information from Ω, $\mathcal{A}$, and $\mathcal{S}$) is not only available, but necessary; we need

a general architecture to leverage both visual information and metadata to discover objects and adapt as conditions change.

Prior work in robotics has widely used metadata to limit computational costs and improve robustness in perception. The metadata is mostly incorporated by imposing restrictions on the environment, data acquisition, or agent motion, which often result in single-purpose solutions of limited applicability. Common assumptions include partial knowledge of the world Ω, usually about the scene configuration or the appearance or shape of objects. Marton et al. (2010) assumes that interesting objects lie on tables to segment novel objects in 3D point clouds. A horizontal plane detector is used to pre-segment the scene and enforce the tabletop assumption. This same assumption is shared by other works in the robotics literature, such as Bjorkman and Kragic (2010) and Kootstra and Kragic (2011). Mishra and Aloimonos (2011) use three-frame sequences, motion cues, and assume that images contain a table with known color distribution to discover and accurately segment objects in cluttered scenes. Morwald et al. (2010) assume that relevant objects may be modeled by simple shapes (such as boxes or cylinders) and that images come in sequences to perform automated modeling of household objects, enforcing temporal consistency with tracking. Both Mishra and Aloimonos (2011) and Morwald et al. (2010) assume some knowledge on constraints about the sensors $\mathcal{S}$ (image ordering and sequencing). Herbst et al. (2011) use datasets consisting of multiple sequences of images collected in the same locations, in order to compute per-sequence environment maps and perform scene differencing to discover movable objects. The implicit assumptions include the knowledge of the robot location, recording time, and that the robotic agent $\mathcal{A}$ visits the same locations multiple times. Rusu et al. (2008) assume strong prior shape and location knowledge to segment cabinets, drawers and shelves in kitchen environments, which are in turn used as cues for the most likely locations of objects. Other works assume an active robotic agent $\mathcal{A}$ that interacts with Ω, $\mathcal{S}$ and $\mathcal{H}$ to modify the environment and improve the object discovery process; for example, Fitzpatrick (2003) track movable objects through random interactions with the environment. All of these works use metadata and assumptions to improve performance and efficiency for their particular setups, at the cost of underperforming (and, often, not working at all) in alternative types of scenes. Our general architecture addresses these shortcomings with a common formulation for metadata, thus allowing us to opportunistically take advantage of different sources of information as conditions change.

In our framework, we combine multiple sources of information (visual similarity and metadata) in CSGs, and cluster the CSGs to obtain groups of object candidates. In the clustering literature, this area is known as multi-similarity (or multi-source) clustering. While multi-similarity clustering applied to unsupervised object discovery is a novelty of this work, other fields (e.g. bioinformatics) commonly use multi-similarity clustering to combine multiple heterogeneous data sources. Zeng et al. (2010) combine gene expression data, text, and clustering constraints induced by the text data, to identify closely related genes. Zeng et al. (2010) use a variant of EM in which parameter estimation and cluster reassignment are performed over a single data source picked at random at each iteration. Troyanskaya et al. (2003) introduce a Bayesian framework to cluster protein–protein interaction patterns based on multiple sources of protein relations. The Bayesian network combines multiple clusterings (one for each data source) using human expert knowledge to estimate the prior probabilities of the interaction patterns.

Other fields such as machine learning and data mining have also shown interest in multi-similarity clustering. Bouvrie (2004) considers the problem of multi-similarity clustering with partially missing data, where not all data sources are available for all points. Bouvrie (2004) optimizes an information-theoretic objective function over pairs of co-occurrence matrices, which requires $\binom{n}{2}$ clustering steps (for n data sources). Tang et al. (2009) propose link matrix factorization, in which multiple graphs for different data sources are approximated by a graph-specific factor and a factor common to all graphs, where the common factor is the consensus partition. Strehl and Ghosh (2002) combine multiple clusterings as a combinatorial optimization problem over the shared mutual information between clusterings. This method performs clusterings for individual data sources first, and a clustering over the co-occurrences of data labels, which the authors term a *cluster ensemble*. Hore et al. (2006) modify the cluster ensembles of Strehl and Ghosh (2002) to use clustering centroids instead of clustering labels. This change enables the combination of disjoint datasets into the same cluster ensemble, with centroids acting as representatives for the data in their clusters.

All previously mentioned methods for multi-similarity clustering except Hore et al. (2006) suffer from poor scalability, as they all require computing and storing multiple clusterings of the full dataset for each individual data source. In object discovery, some data sources (in particular, visual similarity) are very expensive to compute; therefore, clustering each individual data source can be very costly. Some cases, such as Tang et al. (2009), also require multiple full adjacency matrices in memory, which is infeasible for large datasets. In our work, we take the route of Hore et al. (2006) of computing consensus clusters over disjoint datasets. The key differences between Hore et al. (2006) and our work arise from our clustering method being tailored for object discovery. First, we compute disjoint subsets of data samples dynamically from metadata, and not random splits. Second, we use partial 3D object models as intermediate representations, and not centroids. The partial 3D models encode more information than centroids or individual candidates h_i, so our clustering method is asymmetric: the similarity functions that create the disjoint subsets (visual features and metadata) are different

than the similarity functions in the consensus clustering (more complex visual and 3D features).

4. Framework overview

This section contains a brief summary of the discovery framework and its components, alongside a description of how each component is implemented in HerbDisc. In the following sections, we focus on the novel elements of this paper: defining constraints (Section 5), generating CSGs (Section 5.3), and the implementation of constraints and CSGs in HerbDisc (Section 7). We provide a list of the constraints implemented in HerbDisc in Table 1.

1. **Candidate generation**. We compute object candidates h_i from each data sample $I_n \in \boldsymbol{I}$. We use the *objectness*-based segmentation algorithm of Collet et al. (2011) (Section 7.1).
2. **CSG generation**. We create a graph of relationships between object candidates using constraints Θ. We define the CSG built by constraint Θ as $G^{\Theta} = (E^{\Theta}, V^{\Theta})$ (Section 5.3).

   If the constraint Θ encodes a visual similarity, then the CSG G^{Θ} is equivalent to regular pairwise similarity graphs in unsupervised object discovery (e.g. Kang et al., 2011). Applying the constraints in Table 1 to create G^{Θ} produces multiple connected components G^{Θ}_g.
3. **CSG clustering**. We compute groups of candidates for each $G^{\Theta}_g \in G^{\Theta}$ with the graph partitioning algorithm of Brandes (2001). This algorithm is a greedy community discovery method based on the betweenness centrality metric, which is very efficient for sparse graphs and works well for our problem.

   Each cluster C_i contains a set of object candidates $\boldsymbol{h}_i$, which are registered together and merged to compute partial 3D models m_i. The set of all partial models discovered is denoted as $\boldsymbol{m}$.

   Each object $m_i = \{m_i^{\boldsymbol{rgb}}, m_i^{\boldsymbol{P}}, m_i^{\boldsymbol{h}}\}$ is defined by a set of 3D points $m_i^{\boldsymbol{P}}$ with associated color $m_i^{\boldsymbol{rgb}}$ and the set of object candidates $m_i^{\boldsymbol{h}}$ used to create object m_i.
4. **Object CSG graph generation**. We compute a CSG graph $G^m = (E^m, V^m)$ over partial object models $m_i \in \boldsymbol{m}$. The number of nodes in this graph is orders of magnitude smaller than G^{Θ}, so we can afford to compute more complex constraints if needed. Only a subset of the constraints from Table 1 are available for partial object models m_i. In particular, we use Θ_{size}, Θ_{shape}, Θ_{app}, and Θ_{3D}, as the others require local information that is not relevant for the partial objects.
5. **Object clustering**. We compute clusters of partial 3D models using the graph partitioning algorithm of Brandes (2001) on the graph G^m, analogously to step 3. Each cluster C_i contains a set of partial object models m_i.
6. **3D model generation**. We generate full object models M_i from clusters of partial object models C_i. Each cluster of partial object models C_i is globally registered to produce full 3D models. We use the global alignment algorithm of Borrmann et al. (2008) for the global registration of partial 3D models.

5. Information as constraints

In the introduction, we defined a constraint Θ as a *measurable* yes/no question about a node or edge, with probability of success p about the answer. In Section 2, we modeled each constraint Θ as a Bernoulli distribution. In this section, we describe how to encode information as constraints; we define logic operations of constraints that allow us to create complex constraint expressions; and how to compute CSGs from constraints.

5.1. Defining constraints

Constraints encode generic information about the world. Consider the assumptions $\Theta^{\text{n}}_{\text{planar}} = ''\text{objects are planar}''$, $\Theta^{\text{e}}_{\text{static}} = ''\text{scene is static}''$, and $\Theta^{\text{n}}_{\text{tables}} = ''\text{objects lie on tables}''$, illustrated in the example scenes in Figure 3. To encode these assumptions as constraints, we need to express them as a *measurable* yes/no question about a node or edge. For example, Θ_{planar} requires answering the question "is candidate h_i planar?". If we can measure whether an object is planar or not (e.g. by evaluating the reconstruction error of a planar approximation of h_i's 3D points), then we can encode the assumption as a constraint, with the result shown in Figure 3 (row 2, col 2). Similarly, we must answer the question "is candidate h_i on a table?" to encode Θ_{tables}, for which we need to (1) detect a table, and (2) determine whether candidate h_i is on it. The assumption can be encoded as a node constraint if we can measure those two factors, with the result shown in Figure 3 (row 2, column 3). To encode the assumption "the scene is static", we must answer the question that "Do candidates h_i at time t and h_j at time $t + 1$ occupy the same location in space?". The constraint Θ_{static} would be satisfied if we can register the two scenes and h_i and h_j occupy the same 3D location, with p proportional to the overlap between h_i and h_j. Figure 3(row 1, column 3) shows the result of applying Θ_{static} to our example scenes.

Basic constraints, as in the example above, operate over a node or an edge. However, more complex constraints may operate over both nodes and edges. To support this class of constraints, we redefine in our formulation the constraint Θ as a pair $\Theta \equiv (\Theta^{\text{n}}, \Theta^{\text{e}})$. Constraints that operate only on nodes or edges are considered to implement a default operator for nodes ($\Theta^{\text{n}} = 1$) or edges ($\Theta^{\text{e}} = 1$) which satisfies the constraint with $p = 1$ for any input. An example of constraint that operates over nodes and edges is object tracking. Object tracking can be encoded as a union of an edge constraint Θ^{e} = "are candidates h_i at time t and

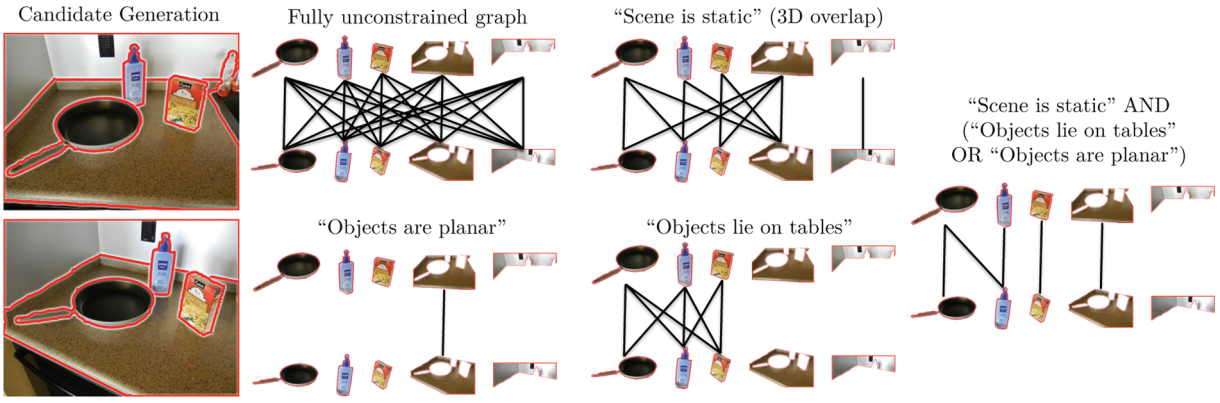

Fig. 3. Metadata induces constraints on pairwise similarity graphs. We illustrate this effect on a pair of manually segmented images for simplicity. The fully unconstrained graph is seldom computed in practice, as techniques such as inverted indexes are used to preselect potential matches (Philbin et al., 2010). Our formulation generalizes such techniques, constraining a graph based on any source of metadata (columns 2–3). Most importantly, our formulation facilitates the creation of complex rules from the combination of multiple sources of metadata (column 4).

h_j at time $t + 1$ the same object?", and a node constraint Θ^{n} = "is candidate h_i being tracked?".

A key advantage of this formulation is that we can encode any source of information as a constraint, including the pairwise similarity functions typical in object discovery and many other computer vision problems. In particular, a normalized similarity function $s(h_i, h_j) \in [0,1]$ induces an edge constraint Θ^{e} with $P_{\Theta^{\mathrm{e}}}(h_i, h_j) = s(h_i, h_j)$. In HerbDisc, we do not distinguish between visual similarity and other metadata: they are all encoded as constraints Θ_i. In the following sections, we see how to combine multiple constraints (Section 5.2) and build CSGs (Section 5.3), both of which are only possible thanks to this unification.

5.2. The logic of constraints

A key consequence of our generic constraint formulation is that we can seamlessly combine multiple sources of metadata using logic statements. In order to take full advantage of Boolean algebra, we define the logic operations of conjunction $\wedge$, disjunction $\vee$ and negation $\neg$ over node and edge constraints induced by metadata. Let Θ_i^{n}, Θ_j^{n} be independent node constraints induced by metadata, and $P_{\Theta}(h)$ the probability of candidate h satisfying constraint Θ^{n}. Then, the negation operator $\neg\Theta_i^{\mathrm{n}}$ is computed as

$$P_{\neg\Theta_i^{\mathrm{n}}}(h) = 1 - P_{\Theta_i^{\mathrm{n}}}(h) \tag{8}$$

which represents the probability of h *not* satisfying constraint Θ^{n}. The conjunction operator $\Theta_i^{\mathrm{n}} \wedge \Theta_j^{\mathrm{n}}$ is then computed as

$$P_{\Theta_i^{\mathrm{n}} \wedge \Theta_j^{\mathrm{n}}}(h) = P_{\Theta_i^{\mathrm{n}}}(h) P_{\Theta_j^{\mathrm{n}}}(h) \tag{9}$$

Finally, the disjunction operator $\Theta_i^{\mathrm{n}} \vee \Theta_j^{\mathrm{n}}$ is computed as

$$P_{\Theta_i^{\mathrm{n}} \vee \Theta_j^{\mathrm{n}}}(h) = 1 - P_{\neg\Theta_i^{\mathrm{n}} \wedge \neg\Theta_j^{\mathrm{n}}}(h) \tag{10}$$

We analogously define the conjunction $\wedge$, disjunction $\vee$ and negation $\neg$ operators for edge constraints, by substituting $P_{\Theta^{\mathrm{n}}}(\cdot)$ for $P_{\Theta^{\mathrm{e}}}(\cdot, \cdot)$ in Equations (8), (9) and (10).

Logic operations over constraint pairs $\Theta = (\Theta^{\mathrm{n}}, \Theta^{\mathrm{e}})$ operate on Θ^{n} and Θ^{e} independently, so that

$$\neg\Theta_i = (\neg\Theta_i^{\mathrm{n}}, \neg\Theta_i^{\mathrm{e}}) \tag{11}$$

$$\Theta_i \vee \Theta_j = (\Theta_i^{\mathrm{n}} \vee \Theta_j^{\mathrm{n}}, \Theta_i^{\mathrm{e}} \vee \Theta_j^{\mathrm{e}}) \tag{12}$$

$$\Theta_i \wedge \Theta_j = (\Theta_i^{\mathrm{n}} \wedge \Theta_j^{\mathrm{n}}, \Theta_i^{\mathrm{e}} \wedge \Theta_j^{\mathrm{e}}) \tag{13}$$

Any logic operation can be derived from the conjunction, disjunction and negation operators. A generic constraint Θ can be composed of multiple constraints Θ_i using the logic operators defined above,

$$\Theta = \Theta_1 \circ \Theta_2 \circ \ldots \circ \Theta_i \circ \ldots \tag{14}$$

where the composition operator $\circ$ denotes any logic operation using Boolean algebra.

We can now define arbitrarily complex constraint expressions based on logic operations over primitive constraints. In Figure 3(row 1, column 4), we illustrate this behavior with the three hard constraints: Θ_{static}, Θ_{tables}, Θ_{planar}. To search for objects assuming that "the scene is static" AND that "objects that lie on tables" OR "objects are planar", we simply define the constraint

$$\Theta = \Theta_{\mathrm{static}} \wedge (\Theta_{\mathrm{tables}} \vee \Theta_{\mathrm{planar}}) \tag{15}$$

5.3. Constrained similarity graphs

CSGs are undirected graphs which encode information from constraints into nodes, edges, node weights and edge weights. Let $G^{\Theta} = (E^{\Theta}, V^{\Theta})$ be an undirected pairwise graph. G^{Θ} is a CSG of constraint Θ if and only if:

Algorithm 1 Building a constrained similarity graph.

$V^{\Theta} = \oslash$
$E^{\Theta} = \oslash$
for h_i in $\boldsymbol{h}$ **do** ▷Add nodes that satisfy Θ
  if $P_{\Theta^n}(h_i) > p_{\min}$ **then**
    $V^{\Theta} \leftarrow V^{\Theta} \bigcup \{h_i\}$
    $w(h_i) \leftarrow P_{\Theta^n}(h_i)$
for h_i in $\boldsymbol{V}^{\Theta}$ **do** ▷Add edges that satisfy Θ
  for h_j in $\boldsymbol{h}$ with $j > i$ **do**
    if $P_{\Theta^e}(h_i, h_j) > p_{\min}$ **then**
      $E^{\Theta} \leftarrow E^{\Theta} \bigcup \{e_{i,j}\}$
      $w(e_{i,j}) \leftarrow P_{\Theta^e}(h_i, h_j)$

1. every node $h_i \in V^{\Theta}$ satisfies Θ^n;
2. every edge $e_{i,j} \in E^{\Theta}$ satisfies Θ^e; and
3. has node weights $w(h_i) = P_{\Theta^n}(h_i)$, and edge weights $w(h_i, h_j) = P_{\Theta^e}(h_i, h_j)$.

We generate the CSG G^{Θ} for constraint Θ following Algorithm 1. The CSG construction in Algorithm 1 and the entire framework are independent of the particular choice of Θ. Θ can be any arbitrarily complex constraint expression, ranging from the multiple sources of metadata and visual similarity we implement in HerbDisc, to the visual similarity-only that transforms the CSG into a regular pairwise similarity graph. In Algorithm 1, $p_{\min}$ denotes the threshold probability for nodes and edges (typically, $p_{\min} = 0.5$).

Building a generic CSG has necessarily a worst-case complexity of $\mathcal{O}(n^2)$, where $n = |\boldsymbol{h}|$, since the CSG must be able to build any graph including pairwise similarity graphs, or even complete graphs, which are $\mathcal{O}(n^2)$. In addition, evaluating a constraint expression for a node or edge can be expensive, especially if computing complex visual similarities. In practice, we can use conjunctive constraint expressions (as in Equation (9)) to simplify the construction of a CSG, by positioning the most restrictive constraints first. Evaluating a conjunctive constraint expression is much faster than evaluating generic constraint expressions, as we only need to evaluate a constraint in the constraint expression if all previous constraints are successful.

Consider a constraint Θ_0 that generates the CSG $G^{\Theta_0} = (E^{\Theta_0}\ V^{\Theta_0})$. We can compute the CSG G^{Θ} from G^{Θ_0} using conjunctive constraints as in Algorithm 2.

The advantage of conjunctive constraints is clear: the complexity of Algorithm 2 for a given Θ and G^{Θ_0} is $\mathcal{O}(|E^{\Theta_0}|)$, where the graph G^{Θ_0} determines the complexity of building G^{Θ}. Therefore, an appropriate choice of Θ_0 to build a sparse CSG very quickly can greatly improve the performance of the overall algorithm. Some of the natural constraints in service robotics are excellent for this purpose, such as spatiotemporal constraints. In HerbDisc, we define motion and sequencing constraints $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ in Table 1 (see Section 7.2 for details) to split the data stream into subsets of samples with limited motion and at most m samples per subset. Using $\Theta_0 = \Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ yields a CSG G^{Θ_0} with $|E^{\Theta_0}| = \mathcal{O}(nm) \approx \mathcal{O}(n)$ edges, considering

Algorithm 2 Building a CSG with conjunctive constraints.

$V^{\Theta} = V^{\Theta_0}$
$E^{\Theta} = E^{\Theta_0}$
for h_i in V^{Θ} **do**
  for Θ_k in Θ **do** ▷Erase nodes that do not satisfy Θ_k
    if $P_{\Theta_k^n}(h_i) < p_{\min}$ **then**
      $V^{\Theta} \leftarrow V^{\Theta} - \{h_i\}$
      break
    else
      $w(h_i) \leftarrow w(h_i) P_{\Theta_k^n}(h_i)$
for h_i in $\boldsymbol{V}^{\Theta}$ **do**
  forΘ_k in Θ **do** ▷Erase edges that do not satisfy Θ_k
    for h_j in $\mathcal{N}^{\Theta}(h_i)$ **do**
      if $P_{\Theta_k^e}(h_i, h_j) < p_{\min}$ **then**
        $E^{\Theta} \leftarrow E^{\Theta} - \{e_{i,j}\}$
        break
      else
        $w(e_{i,j}) \leftarrow w(e_{i,j}) P_{\Theta_k^e}(h_i, h_j)$

Table 2. Effect of motion and sequencing in computational cost, for the NSH Dataset. Number of edges to evaluate if using (1) the motion and sequencing constraints, (2) only the motion constraint, and (3) the raw data stream.

Time (min)	$\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$	Θ_{motion}	Raw data
58.0	0.7M	29.1M	2.9B
102.7	1.2M	83.9M	10.4B
186.9	2.5M	263M	30.4B
262.2	3.3M	517M	59.9B
319.9	4.0M	803M	89.2B
380.6	4.9M	1.2B	126.0B

that m is fixed and $n \gg m$ in realistic situations (in the NSH Dataset, $m = 50$ and $n = 521{,}234$). Under these conditions, the CSG construction given G^{Θ_0} has a complexity of $\mathcal{O}(n)$ for the remaining constraints $\Theta_k \in \boldsymbol{\Theta}$. Given that the visual similarities are the most expensive constraints, it is crucial to perform this optimization to only compute $\mathcal{O}(n)$ similarities. See Table 2 for a quantitative evaluation of the reduced complexity of this method.

The CSG constructions of Algorithms 1 and 2, as well as the constraints Θ, are designed for both soft constraints (i.e. Θ such that $P_{\Theta} \in [0, 1]$) and hard constraints (i.e. Θ such that $P_{\Theta} \in \{0, 1\}$). In HerbDisc, we use Algorithm 2 with a mix of soft and hard constraints. The hard constraints are positioned first in the constraint expression to purposefully split the CSG into many small connected components as quickly as possible. We then use soft constraints to better evaluate the nuances of appearance and shape similarity for those candidates with real potential of being part of the same object.

6. Datasets

We evaluate HerbDisc on two datasets of real human environments: the Kitchen Dataset and the NSH Dataset (see Figure 4). We captured both datasets from the sensory data

Fig. 4. The Kitchen Dataset (top row) and the NSH Dataset (bottom three rows). Each row depicts the Kinect 3D point clouds (top) and their corresponding images with ground truth annotations (bottom) for some of the environments we visited. The Kitchen Dataset captures a low-clutter environment with 20 objects of interest. The NSH Dataset captures office and lab environments, ranging from moderate to extreme clutter. Some scenes were so challenging (e.g. row 2, columns 3–5) that the annotators could not separate the objects in the scene.

of our robot, HERB, driving around different working environments of a university building. The captured data is an RGBD video stream from a Kinect camera at 640 × 480 resolution. The framerate was set to 30 fps, but due to throughput limitations the effective framerate is approximately 22 fps. For the remainder of this paper, we refer to each pair of color image and depth image as a *data sample*.

The Kitchen Dataset captures four 3-minute recordings of HERB in a kitchen environment, with relatively clean scenarios and 20 ground truth objects that HERB must discover. We refer to the four individual recordings as Kitchen-1 to Kitchen-4, and their union (a 12-minute recording with 14,282 data samples) as the Kitchen Dataset.

The NSH Dataset is a workday-length recording of HERB exploring the NSH building of Carnegie Mellon University, containing 6 h and 20 min of RGBD video and 521,234 data samples. The dataset is divided in four

fragments lasting between 1 h and 1 h 50 min each, one per building floor. We refer to the four individual recordings as NSH-1 to NSH-4, and the full-length stream as the NSH Dataset. For this dataset, we visited over 200 real offices and laboratories to capture the real conditions in which people work, with scenes ranging from moderate to extreme clutter. This dataset also captures the wide differences in lighting conditions in human environments (from dim light to bright sunlight), which degrade the data acquisition and to which a lifelong agent must be robust. We labeled a total of 423 unique ground truth objects. We can analyze the object statistics of this dataset by aggregating the ground truth objects into common classes. The most popular object classes are coffee mugs (19 labeled), monitors (17 labeled), keyboards (16 labeled), laptops (13 labeled), and books (12 labeled). Among the object classes with one labeled instance, we find objects as diverse as a toy rocket, a pineapple, a bike padlock, a quadrocopter, and various mechanic tools such as a pair of plyers.

We followed the labeling procedure described below to manually annotate both datasets and obtain ground truth. Our goal is to obtain the list of objects that HERB could potentially grasp. Since it is infeasible to annotate every single data sample, we process each data stream with a motion filter to eliminate redundant samples (the same motion filter used in HerbDisc, described in Section 7.2). Then, we select 10 images from each office, lab, kitchen, etc., we visited, and label all objects with the LabelMe tool (Russell et al., 2008). As a rough estimate of the objects that HERB can grasp, we consider valid any object that:

- is at least 10 × 5 cm in its two largest dimensions (e.g. a smartphone);
- is at most 60 cm long in its longest dimension (e.g. a monitor);
- appears unoccluded in at least one data sample; and
- is movable, with free space around it to be grasped (e.g. a stack of books in a bookshelf is not labeled).

Figure 4 shows examples of data samples from the Kitchen (top row) and NSH Dataset (bottom 3 rows), alongside the annotated data.

7. Implementation of HerbDisc

In this section, we describe how to formulate similarities, assumptions, and other metadata from Table 1 as constraints, and how each component is integrated into our optimized implementation, HerbDisc. The advantage of formulating the different components as constraints is the adaptability of the system. We can completely control and modify the behavior of HerbDisc (e.g. to adapt it a particular task) without modifying a single line of code, as HerbDisc only depends on the constraint expression Θ to construct the CSGs. For example, we could measure if the assumptions for specific algorithms hold before using them, and revert to safer algorithms if they do not, modifying only the constraint expression. By modifying Θ when environmental conditions change, we can adapt and opportunistically select the best constraints for each task.

We show experimental results on the impact of each component in the different subsections. See Section 8 for a description of the baseline and the evaluation procedure.

7.1. Constrained candidate generation

The candidates h produced by a candidate generator $\mathcal{H}$ can be refined with constraints to adapt to the particular algorithm assumptions, either by entirely enabling/disabling a candidate generator based on metadata, or by rejecting unnecessary candidates for the particular task. An example of such a constraint would be the requirement that "objects lie on tables".

Candidate generators that rely on metadata are common in the robotics literature. For example, algorithms that track objects (Morwald et al., 2010), that assume tabletop scenes (Bjorkman and Kragic, 2010), or that perform scene differencing (Herbst et al., 2011) usually compute better candidates than generic objectness segmentation algorithms. These "specialized" candidate generators all have one thing in common: they impose restrictions on the environment to simplify the task and improve performance, at the cost of limited applicability in alternative types of scenes. In our framework, we can include multiple candidate generators and use them when their assumptions are met, and revert to more generic candidate generators otherwise.

In HerbDisc, we combine the generic objectness segmentation algorithm of Collet et al. (2011) with the assumption that objects have surfaces of support in floors, tables, and walls. The constraint $\Theta_{\text{support}} = (\Theta^{\text{n}}_{\text{support}}, 1)$ is defined as

$$\Theta^{\text{n}}_{\text{support}}(h_i) = \begin{cases} 1, \text{with } p = 1 & \text{if supported}(h_i, I_j) \\ 0, \text{with } q = 1 & \text{otherwise} \end{cases} \tag{16}$$

where $q = 1 - p$ is the probability of failure of $\Theta^{\text{n}}_{\text{support}}$, the supported$(\cdot)$ function searches for large planes in the data sample I_j that generated candidate h_i, and accepts h_i if it lies within a certain distance above the planes found. In simple scenes, Θ_{support} can be used as a standalone candidate generator, by clustering the point clouds above the detected planes into a few connected components. For the standalone Θ_{support}, we use the implementation of Rusu and Cousins (2011).

In Figure 6, we compare the performance of Rusu and Cousins (2011) and Collet et al. (2011) with Θ_{support} used in HerbDisc. We see in Figure 6 that the standalone Θ_{support} achieves better precision as it accurately segments simple scenes better. However, the performance degrades in complex scenes (see Figure 5 for examples), as the connected components may include large groups of objects. Combining the generic Objectness segmentation with Θ_{support} yields a good tradeoff between generating enough

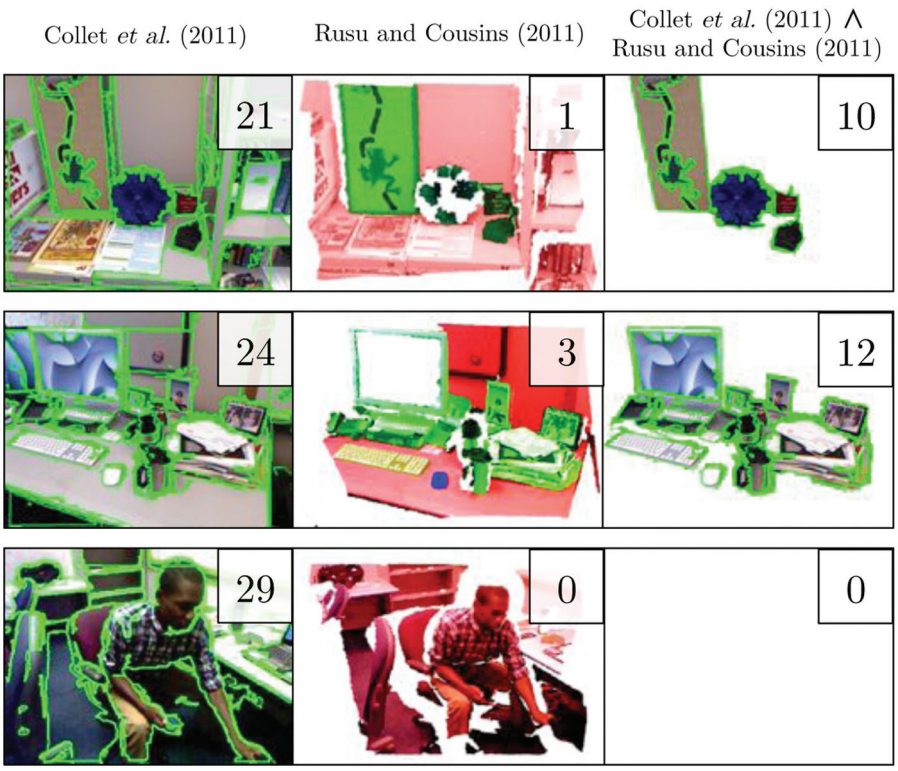

Fig. 5. Examples of constrained candidate generation in the NSH-1 dataset. The number of candidates in each data sample is shown at the top right corner of each image. (Left) RGBD Objectness segmentation algorithm of Collet et al. (2011). (Center) Rejected areas according to Θ_{support} are shown in red; the connected components of accepted 3D points are shown in green/yellow/blue. In cluttered scenes, multiple objects are sometimes grouped together. Scenes with no visible support are rejected (e.g. row 3). (Right) Combining Collet et al. (2011) and Θ_{support} limits the number of candidates but does not result in undersegmentation.

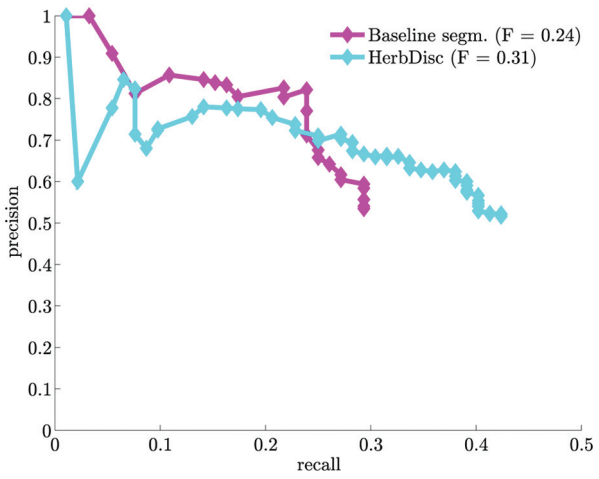

Fig. 6. Impact of Θ_{support} (Rusu and Cousins (2011), Baseline Segm.) versus HerbDisc's Objectness + Θ_{support} in the NSH-1 dataset. Rusu and Cousins (2011) achieves higher precision (80% precision at 20% recall, compared with 78% precision of HerbDisc) at the cost of 14% lower maximum recall.

candidates for complex scenes, and filtering unlikely candidates for efficiency. In Figure 5, we show the generic objectness segmentation of Collet et al. (2011), and compare it with the standalone candidate generator from Rusu and Cousins (2011) and the combination of (Collet et al., 2011) and Rusu and Cousins (2011).

7.2. Motion and sequencing

In the general LROD problem, the incoming data stream should be continuous (the raw RGBD video stream) and neverending. In particular, we assume that the data stream is:

1. an ordered sequence of data samples; and
2. recorded at a frame rate high enough so that there is spatial overlap between data samples.

During the data acquisition, the motion of HERB influences the amount of spatial overlap between data samples. In particular, HERB may (a) not be in motion and acquiring repeated data samples, (b) be in motion and fulfilling assumption 2, or (c) be in motion and violating assumption 2 (i.e. moving too fast). We address these issues with constraints Θ_{motion} and Θ_{seq}.

In particular, we sample the input data stream at a dynamic framerate depending on HERB's motion, and split

the subsampled data stream into small subsets that we term *sequences*. Using Θ_{motion} and Θ_{seq}, we do not process repeated samples, and we do not consider any edges between data samples that violate assumption 2. We enforce a maximum sequence length m to limit the order $|V^{\Theta_{\text{seq}}}|$ of any connected component in the CSG.

Let $T_{k,k-1} \in \mathbb{R}^{4\times4}$ be the transformation between sample I_k and the previous sample in the data stream I_{k-1}, and $M : T \mapsto \mathbb{R}$ the magnitude of the motion T. We model the motion constraint $\Theta_{\text{motion}} = (\Theta^{\text{n}}_{\text{motion}}, 1)$ for $h_i \in I_k$, as

$$\Theta^{\text{n}}_{\text{motion}}(h_i) = \begin{cases} 1, \text{with } p = 1 & \text{if } M(T_{k,k-1}) > \gamma_{\min} \\ 0, \text{with } q = 1 & \text{otherwise} \end{cases} \tag{17}$$

Θ_{motion} only samples the data stream when there is enough motion $\gamma_{\min}$ between data samples.

The sequencing constraint $\Theta_{\text{seq}} = (1, \Theta^{\text{e}}_{\text{seq}})$, where

$$\Theta^{\text{e}}_{\text{seq}}(h_i, h_j) = \begin{cases} 1, \text{with } p = 1 & \text{if } \text{seq}(h_i) = \text{seq}(h_j) \\ 0, \text{with } q = 1 & \text{otherwise} \end{cases} \tag{18}$$

limits the potential edges to candidates $h_i \in I_k$, $h_j \in I_l$ which belong to the same sequence. We compute seq(·) during the data acquisition. For data sample I_k, the sequence identifier

$$\text{seq}(I_k) = \begin{cases} \text{seq}(I_{k-1}) + 1 & \text{if } M(T_{k,k-1}) > \gamma_{\max} \\ \text{seq}(I_{k-1}) & \text{otherwise} \end{cases} \tag{19}$$

is incremented if there is too much motion ($\gamma_{\max}$) between the current sample I_k and I_{k-1} (or if we reach the maximum sequence length m).

Here $M(T) = ||T||_F$ is an estimate of the relative motion T. We calibrate $\gamma_{\min}$ and $\gamma_{\max}$ so that we capture m data samples in 20 seconds moving in a straight line at HERB's slowest and fastest speed. In practice, sequences are finished because $\gamma_{\max}$ is exceeded in approximately 73% of the sequences (often due to sharp turns), and reaching our limit of $m = 50$ in 27% of the cases, mainly in long, straight corridors. HerbDisc is not very sensitive to particular choices of the maximum sequence length; halving or doubling the maximum sequence length ($m = 25$ and $m = 100$, respectively) yields a decrease of less than 3% in maximum recall with respect to our default choice of $m = 50$.

We evaluate the impact of the motion and sequencing constraints Θ_{seq} to computational complexity for the NSH Dataset in Table 2. We calculate the total number of potential edges remaining in the CSG, which is a measure of the computational cost, in the cases of: (1) using $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ to generate connected components; (2) only using Θ_{motion} to downsample the input data stream; and (3) the raw data stream. Our implementation in HerbDisc, which uses $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ as the initial constraint (using Algorithm 2), yields equivalent computational cost after processing 6 h 20 min as Θ_{motion} after approximately 18 min, or as the raw data stream after 2 min 24 s. Figure 7 compares the

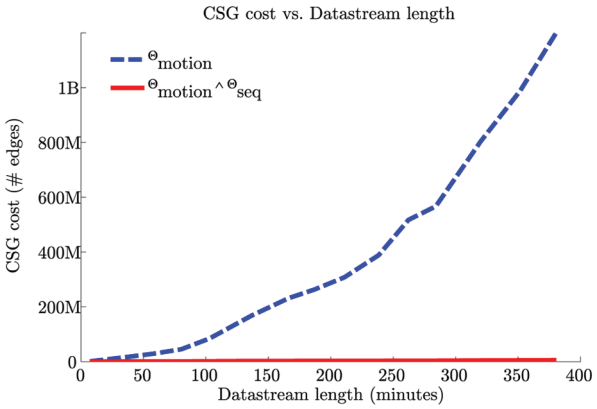

Fig. 7. Comparing the computational cost of HerbDisc when using $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ and Θ_{motion} for the NSH Dataset, with respect to the data stream length. Using $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ results in linear cost in the number of samples, compared with the squared cost of Θ_{motion}.

trend in computational cost with respect to the data stream length. While using Θ_{motion} is two orders of magnitude more efficient than the raw data stream, it still yields a squared cost with respect to the data stream length, compared with the linear cost of $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$.

For the actual implementation, we considered two alternatives: (1) to track the robot motion using the robot's odometry; or (2) to track the camera motion using real-time techniques such as Kinect Fusion (Izadi et al., 2011). We decided to implement the Kinect Fusion approach, because the odometry does not track the camera tilt and shake while HERB drives. These artifacts can be pretty significant depending on the surface (e.g. camera shake on floor tiles, and tilt on thick carpeting). To implement this motion filter, we modify the Kinect Fusion six-degree-of-freedom tracker available in PCL (Rusu and Cousins, 2011). Our implementation of Θ_{motion} and Θ_{seq} in HerbDisc, including the PCL Kinect Fusion tracking, runs in real time (up to 30 fps) during the data acquisition process to compute the initial CSG $G^{\Theta_0} = G^{\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}}$ from Algorithm 2.

7.3. Spatial overlap

Many objects in human environments are only moved occasionally, and remain static across most observations. The constraint Θ_{static} encodes our assumption that objects remain static for at least a few seconds at a time. To encode this assumption in our framework, we consider the question "do candidates h_i and h_j overlap in space within a sequence?". Relationships between candidates that do not overlap in space should not be considered any further, as they most likely belong to different objects.

The constraint $\Theta_{\text{static}} = (1, \Theta^{\text{e}}_{\text{static}})$, where

$$\Theta^{\text{e}}_{\text{static}}(h_i, h_j) = \begin{cases} 1, \text{with } p = s^{\text{overlap}}_{i,j} & \text{if } s^{\text{overlap}}_{i,j} > s^{\text{overlap}}_{\min} \\ 0, \text{with} q = 1 & \text{otherwise} \end{cases} \tag{20}$$

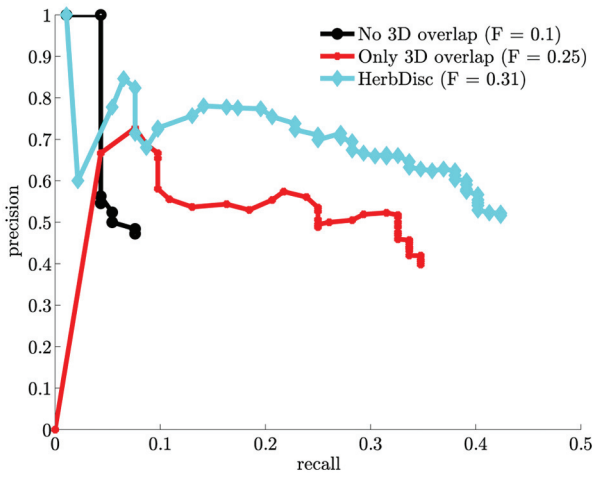

Fig. 8. Impact of Θ_{static} in HerbDisc for the NSH-1 dataset. Not using the 3D overlap similarity of Θ_{static} yields a 27% drop in recall compared with HerbDisc. Comparatively, using the 3D overlap similarity Θ_{static} alone with no visual features in HerbDisc only results in a decrease of 7% recall and 12% precision at maximum recall with respect to HerbDisc.

is a soft constraint that measures the amount of 3D overlap $s^{\text{overlap}}_{i,j} = s^{\text{overlap}}(h_i, h_j)$ between candidates h_i, h_j, and returns true with probability $s^{\text{overlap}}_{i,j}$ if the overlap is above a threshold $s^{\text{overlap}}_{\text{min}}$.

This constraint is designed to operate in unison with the sequencing constraint Θ_{seq}. Here Θ_{seq} splits the data stream into small subsets of samples close in time (sequences), and Θ_{static} ensures that, within the same sequence, we only evaluate groups of candidates in a similar position in space.

To measure the overlap between hypotheses, we use the incremental registration provided by PCL Kinect Fusion to register all data samples within a sequence with respect to some common coordinate frame T^s (the first sample in that sequence). We transform all object candidates $\boldsymbol{h}$ to the common coordinate frame, and measure the 3D overlap $s^{\text{overlap}}_{i,j}$ between 3D point clouds $h^{\boldsymbol{P}}_i, h^{\boldsymbol{P}}_j$ by comparing their voxel grids.

In Figure 8, we compare the impact of using the 3D overlap constraint Θ_{static} in HerbDisc. We see that Θ_{static} is a crucial metadata constraint in HerbDisc, as disabling Θ_{static} yields a maximum recall of 8% at 47% precision in the NSH-1 dataset, a difference of 27% recall at the same precision when enabled. Furthermore, disabling the visual similarity features and using only Θ_{static} as an edge constraint results in a drop of only 7% recall and 12% in precision (at maximum recall). These results reinforce our claim that visual features alone are not descriptive enough for large-scale datasets, and that metadata plays a key role in LROD.

7.4. Size/shape priors

Part of the reason why there is no clear definition of *object* is because its meaning is subjective: it depends on the observer. In service robotics, different robots might have different definitions of objects depending on their capabilities. For HerbDisc, we consider a definition of object based on the manipulation capabilities of HERB. In particular, we define a prior based on the sizes and shapes of known objects that HERB can grasp.

In order to build an object prior for our framework, we define it as a constraint $\Theta_{\text{prior}} = \Theta_{\text{size}} \wedge \Theta_{\text{shape}}$ composed of size and shape components. Let $\Theta_{\text{size}} = (\Theta^{\text{n}}_{\text{size}}, 1)$ be a constraint on an object candidate's size, such that

$$\Theta^{\text{n}}_{\text{size}}(h_i) = \begin{cases} 1, \text{with } p = s^{\text{size}}_i & \text{if } s^{\text{size}}_i > s^{\text{size}}_{\text{min}} \\ 0, \text{with } q = 1 & \text{otherwise} \end{cases} \quad (21)$$

The function $s^{\text{size}}_i = s^{\text{size}}(h_i, h_{\text{prior}})$ estimates the likelihood that the longest dimension of h_i could be sampled from a Gaussian distribution centered at the size given by h_{prior}.

Analogously, $\Theta_{\text{shape}} = (\Theta^{\text{n}}_{\text{shape}}, 1)$ is a constraint on the candidate's shape, such that

$$\Theta^{\text{n}}_{\text{shape}}(h_i) = \begin{cases} 1, \text{with } p = s^{\text{shape}}_i & \text{if } s^{\text{shape}}_i > s^{\text{shape}}_{\text{min}} \\ 0, \text{with } q = 1 & \text{otherwise} \end{cases} \quad (22)$$

The measure $s^{\text{shape}}_i = s^{\text{shape}}(h_i, h_{\text{prior}})$ estimates the similarity between h_i and h_{prior} according to the PCA-based shape features of Lalonde et al. (2006) (linearity, planarity, and scatterness). The effect of this constraint is to essentially require that object candidates have some volume and are not purely planes or lines.

In Figure 9, we evaluate the impact of size and shape priors in HerbDisc for the NSH-1 dataset. The main effect of Θ_{prior} is to limit the amount of candidates to cluster, with Θ_{prior} rejecting 63% of the original pool of candidates. The increased number of candidates when Θ_{prior} is disabled yields a 301% increase in the number of objects discovered, most of which are just repetitions due to cluster fragmentation. The final output without Θ_{prior} yields a decrease of 7% recall and 10% in precision (at maximum recall), compared with HerbDisc.

7.5. Visual and 3D shape similarity

We describe and compare candidates with features based on 3D shape and appearance. Using these features alone to compute a CSG would result in a pairwise similarity graph as in Kang et al. (2011). For appearance features, we compute the color histogram of each candidate in LAB color space, as in Hoiem et al. (2007), and compare a pair of candidates h_i, h_j with the χ^2 distance between normalized color histograms. For 3D shape, we use the FPFH features of Rusu et al. (2009), which compute a histogram of the local geometry around each 3D point. We compare the FPFH features of a pair of candidates h_i, h_j by estimating the average χ^2 distance among the nearest neighbor 3D points between h_i, h_j. Both similarity metrics $s^{\text{app}}(\cdot,\cdot)$ and $s^{\text{3D}}(\cdot,\cdot)$ are normalized so that $s(\cdot,\cdot) \in [0,1]$.

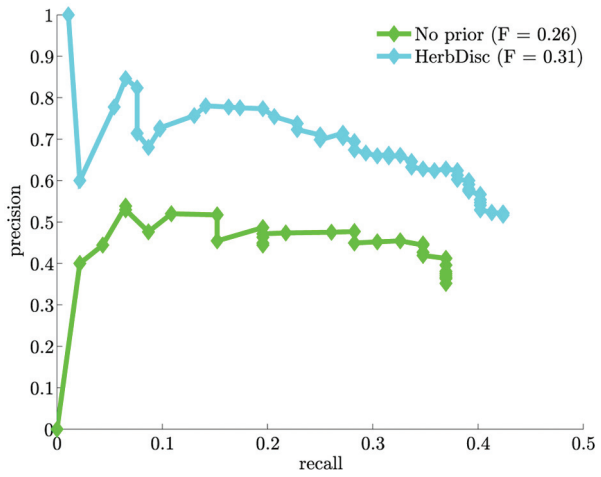

Fig. 9. Impact of Θ_{prior} in HerbDisc for the NSH-1 dataset. Not using Θ_{prior} yields a decrease of 7% recall and 10% in precision (at maximum recall).

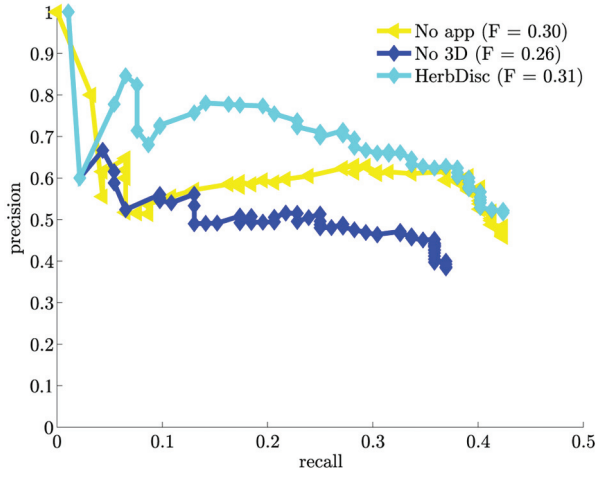

Fig. 10. Impact of Θ_{3D} and Θ_{app} in HerbDisc for the NSH-1 dataset. Disabling Θ_{3D} in HerbDisc decreases 7% recall and 15% precision at maximum recall, as well as 20% lower precision at 20% recall. Disabling Θ_{app} yields a decrease of 1% recall and 3% precision at maximum recall, and 19% lower precision at 20% recall.

In order to use these similarities in our framework, we reformulate them as constraints Θ_{app} and Θ_{3D}. In particular, we define $\Theta_{app} = (1, \Theta^{e}_{app})$ as a soft constraint such that

$$\Theta^{e}_{app}(h_i, h_j) = \begin{cases} 1, \text{with } p = s^{app}_{i,j} & \text{if } s^{app}_{i,j} > s^{app}_{min} \\ 0, \text{with } q = 1 & \text{otherwise} \end{cases} \quad (23)$$

where $s^{app}_{i,j} = s^{app}(h_i, h_j)$. Analogously, we define $\Theta_{3D} = (1, \Theta^{e}_{3D})$ as a soft constraint such that

$$\Theta^{e}_{3D}(h_i, h_j) = \begin{cases} 1, \text{with } p = s^{3D}_{i,j} & \text{if } s^{3D}_{i,j} > s^{3D}_{min} \\ 0, \text{with } q = 1 & \text{otherwise} \end{cases} \quad (24)$$

where $s^{3D}_{i,j} = s_{3D}(h_i, h_j)$.

In Figure 10, we compare the impact of Θ_{app} and Θ_{3D} in HerbDisc. Disabling the 3D shape similarity Θ_{3D} yields a decrease of 7% recall and 15% precision at maximum recall, compared to HerbDisc, as well as a more significant drop in precision at low recalls (e.g. a 28% decrease in precision at 20% recall). The contribution of Θ_{app} is less noticeable: disabling Θ_{app} results in a decrease of 1% in maximum recall at only 3% lower precision, although it is significant at lower recall (e.g. disabling Θ_{app} yields a 19% decrease in precision at 20% recall).

8. Experiments

In this section, we evaluate the impact of using metadata to discover objects. We first compare the performance of HerbDisc with and without any metadata on the Kitchen Dataset in Section 8.3.1. Using metadata, we evaluate the ability of HerbDisc to discover novel objects during a whole workday of operating in challenging human environments. We perform an ablative analysis to assess the impact of each constraint in the constraint expression Θ. Thanks to our framework, performing such an analysis only requires modifying the definition of the constraint expression Θ, but not any change in the source code. This feature is critical for our goal to develop a system that can adapt its behavior as conditions change, using metadata opportunistically.

Our main testbed is the NSH Dataset (Section 6), with 6 h 20 min of HERB driving into over 200 offices and engineering labs, and containing 423 annotated ground truth objects. We use the smaller Kitchen Dataset in Section 8.3.1 to evaluate the visual similarity-only baseline, as it is too computationally expensive to execute in the NSH Dataset.

8.1. Baseline and training

The baseline for all our experiments is the full system HerbDisc, with all constraints enabled. The default candidate generator is the objectness segmentation of Collet et al. (2011) with $\Theta_{support}$. In each experiment, we enable/disable individual components (through the constraint expression) and analyze the resulting behavior.

The constraint expression Θ_{local} we use in the CSG construction step of HerbDisc is

$$\Theta_{local} = \Theta_{motion} \wedge \Theta_{seq} \wedge \Theta_{support} \wedge \Theta_{static} \wedge \Theta_{size} \wedge \Theta_{shape} \wedge \Theta_{app} \wedge \Theta_{3D} \quad (25)$$

In the object CSG clustering, we cluster the CSG built with

$$\Theta_{global} = \Theta_{size} \wedge \Theta_{shape} \wedge \Theta_{app} \wedge \Theta_{3D} \quad (26)$$

We design the constraints Θ_{app} and Θ_{3D} to be more exhaustive for Θ_{global} than Θ_{local}. In Θ_{local}, we compute the histograms in Θ_{app} with six bins per channel, and compute the FPFH features of Θ_{3D} only for the centers of a

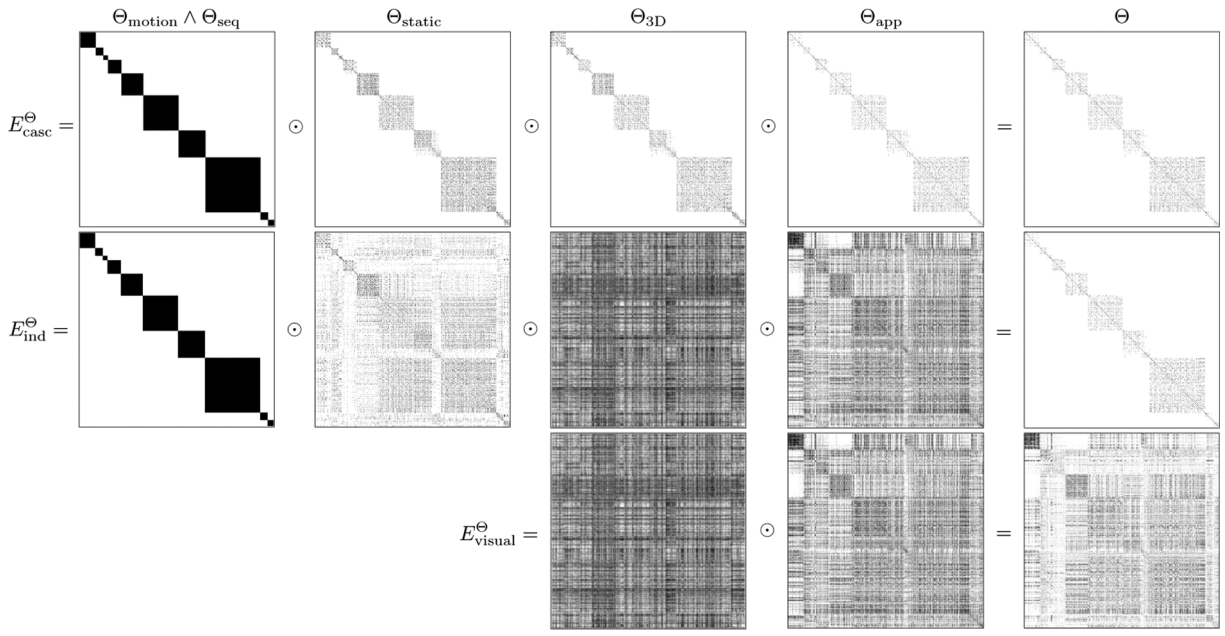

Fig. 11. CSG graphs for the edge constraints in HerbDisc, displayed as adjacency matrices (where a black dot indicates an edge between candidates), in the Kitchen Dataset. The overall graph E^{Θ} (rightmost column) is the product of each adjacency matrix. (Top) Cascaded CSGs using conjunctive constraints, as implemented in HerbDisc. (Center) CSGs computed for each constraint independently. The overall CSG E^{Θ} is the same for the cascaded and independent CSGs. (Bottom) CSGs for the visual similarity constraints Θ_{app} and Θ_{3D}. The overall CSG E^{Θ} for this case is a regular pairwise similarity graph. The CSGs using metadata (top/center cols) are much more discriminative than the CSG for visual similarity only.

1 cm voxel grid. In Θ_{global}, the partial objects contain significantly more information than individual hypotheses. We use 10 bins for the histograms in Θ_{app} and compute FPFH features for Θ_{3D} for the centers of a 3 mm voxel grid. In our experience, the choice of Θ_{local} has significantly more impact in the overall performance than Θ_{global} for object CSG clustering. We therefore focus our experiments on the local step and modify *only* Θ_{local}, while we keep Θ_{global} constant throughout the experiments.

We use the first 20% of the NSH-1 dataset (not included in the evaluation) to train the parameters and thresholds in HerbDisc, by maximizing the average F_1-measure (defined in Section 8.2). To do so, we discretize each parameter in five settings in the range [0,1] and choose the best-performer configuration according to a grid search. We do not modify any parameter in any experiment after the initial training phase. All experiments were performed on a computer with an Intel Core i7-920 CPU, 16 GB of RAM, a nVidia GTX 580 GPU, and runninng 64-bit Ubuntu Linux 10.04.

8.2. Evaluation procedure

In this section, we describe the metrics to evaluate the ability of HerbDisc to discover objects during HERB's workday. For a given object model M_k, we define the metrics of *candidate purity, group purity*, and *3D purity*, as follows.

Candidate purity. We describe an object candidate h_i as *pure* if over 80% of the area in $h_{i,k}$ overlaps with a ground truth object.

Group purity. Following Tuytelaars et al. (2009), we measure the group purity of model M as the largest percentage of *pure* object candidates in $M_k^{\boldsymbol{h}} = \{h_{1,k}, \ldots, h_{i,k}\}$ that belong to the same object.

3D purity. We require that the 3D models reconstruct the partial viewpoints seen by HERB. Therefore, we define an object's 3D point cloud $M_k^{\boldsymbol{P}}$ as *pure* if the 3D points in $M_k^{\boldsymbol{P}}$ cover over 80% of the area visible in the data samples for that particular object.

Given the open and unsupervised nature of LROD, we often discover objects that do not appear in the ground truth set, despite being real objects. Following Kang et al. (2011), we distinguish between three categories of objects: *correct, valid*, and *invalid*.

We define an object model M_k as ***correct*** if (1) it is an object annotated in the ground truth, (2) its 3D point cloud is pure, and (3) every object candidate $h_{i,k}$ associated to M_k is pure, i.e. if the set $M_k^{\boldsymbol{h}}$ is 100% pure. Other works in the literature commonly define correct objects as clusters with some minimum percentage of purity (e.g. 80% in Kang et al. (2011)), but we believe that object models need to be 100% correct to be of any use for robotics. Figure 15 shows multiple examples of *correct* objects.

We define an object model M_k as ***valid*** if (1) its 3D point cloud is pure, (2) the set of candidates $M_k^{\boldsymbol{h}}$ is 100%

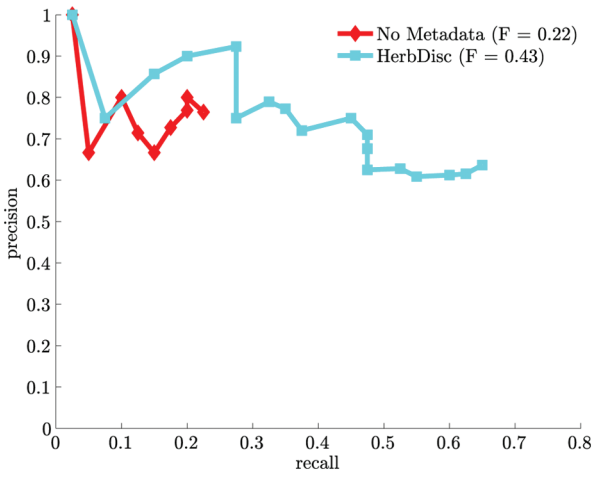

Fig. 12. Impact of using metadata in HerbDisc for the Kitchen Dataset, compared with visual and 3D similarity alone. HerbDisc achieves with a maximum recall of 65% at 62% precision, compared with 24% maximum recall at 77% precision. For the same recall of 24%, HerbDisc achieves 90% precision (13% higher than the visual similarity Θ_{visual} alone).

Table 3. Running times of HerbDisc vs. Θ_{visual} in the Kitchen Dataset. Using no metadata (Θ_{visual}) is 190 × slower than using metadata in this dataset, mainly due to the extra cost of constructing the graph. The Θ_{visual} needs to evaluate 1.6 million pairwise visual similarities from 1806 object candidates, compared with the 16,271 pairwise visual similarities to evaluate when using metadata in HerbDisc.

Component	HerbDisc	Θ_{visual}
CSG construction	35.9 s	18,981.8 s
CSG clustering	61.3 s	394.0 s
Object CSG clustering	4.4 s	2.8 s
Total processing time	101.6 s	19,378.6 s

pure (as with *correct* objects), but (3) it has not been labeled as a ground truth object. We rely on an "oracle" evaluation, as in Tuytelaars et al. (2009). The "oracle" evaluation is a human annotator who answers the questions "Is M_k an object?" and "Does M_k have a name?" when faced with the set of candidates for object model M_k. The category *valid* mainly contains objects too big or too small to be grasped (e.g. chairs), immovable objects (e.g. attached to a wall), or parts of complex objects (which could be objects themselves, such as a bicycle's seat). Figure 16 (top) shows multiple examples of *valid* objects.

We define an object model M_k as ***invalid*** if it is neither *correct* nor *valid*. The category *invalid* mainly includes models M_k of groups of objects erroneously segmented, of a single object but < 100% group purity, or a mix of multiple objects erroneously clustered together. Figure 16 (bottom) shows multiple examples of *invalid* objects.

We define precision and recall as in Tuytelaars et al. (2009) and Kang et al. (2011). In Kang et al. (2011), Precision is the ratio between the number of *correct* + *valid* object models and the total number of discovered models:

$$\text{Precision} = \frac{\#\text{correct} + \#\text{valid}}{\#\text{correct} + \#\text{valid} + \#\text{invalid}} \tag{27}$$

We measure recall as the ratio between the number of unique *correct* objects and the total number of ground truth objects.

$$\text{Recall} = \frac{\#\text{unique correct obj.}}{\#\text{unique ground truth obj.}} \tag{28}$$

We use the cluster size to estimate the quality of an object, and use it as the variable to threshold to compute the precision–recall curves. To summarize the precision–recall curves in a single number, we use the average F_1-measure, which balances Precision and Recall for each sample i in the precision–recall curve:

$$F_1 = \frac{1}{N}\sum_{i}^{N}\frac{2\text{Precision}_i\text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{29}$$

8.3. Results

In this section, we evaluate the impact of metadata to discover objects. We evaluate the use of metadata to using visual similarity alone in Section 8.3.1, and then show that we can leverage metadata to process very large datasets such as the NSH Dataset in Section 8.3.2.

8.3.1 HerbDisc versus visual similarity. Figure 12 shows the performance of using a CSG with visual similarity only ($\Theta_{\text{visual}} = \Theta_{\text{motion}} \wedge \Theta_{\text{3D}} \wedge \Theta_{\text{app}}$), compared with the full HerbDisc, in the Kitchen Dataset. We include the motion filter Θ_{motion} in the evaluation of Θ_{visual} so that both systems have the same initial pool of object candidates.

HerbDisc is the clear winner in the Kitchen Dataset, with a maximum recall of 65% at 62% precision, compared with 24% maximum recall at 77% precision. For the same recall of 24%, HerbDisc achieves 90% precision (13% higher than the visual similarity Θ_{visual} alone). The additional constraints provided by the metadata (and especially Θ_{seq}) allow HerbDisc to process the Kitchen Dataset 190 × faster than if using visual similarity alone, as shown in Table 3. The main reason for this speedup is the limited number of pairwise similarities to evaluate in the CSG (mainly due to Θ_{seq}) compared with the regular pairwise similarity graph from Θ_{visual}. Namely, HerbDisc evaluates 16,271 pairwise visual similarities, compared to 1.6 million in Θ_{visual}.

To illustrate the impact of different constraints on the CSG, we show in Figure 11 the graphs (displayed as adjacency matrices) generated by each edge constraint $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$, Θ_{static}, Θ_{3D}, and Θ_{app}, for the Kitchen Dataset. Figure 11 (top) displays the CSG after each constraint as evaluated in HerbDisc, cascading the multiple

Table 4. Running times of HerbDisc in the NSH Dataset. The motion and sequencing constraint and the candidate generation are executed in parallel with the data acquisition and are not included. The overall running time is 1113.9 seconds (18 min 34 s), to discover 121 correct and 85 valid objects from an RGBD video feed of 6 h 20 min (521,234 samples).

Component	Time (s)
Data acquisition	22,836
Read sequence/candidate data Θ_{seq}	25.9
CSG construction	710.1
CSG clustering	211.9
Object CSG clustering	54.6
Model registration	111.4
Total processing time	1113.9

Table 5. Impact of each component and quantities generated for the NSH Dataset, from 521,234 input images to 464 output models (with 121 *correct* and 85 *valid* objects).

Component	Output	Quantity
$\mathcal{S}$	Input samples I	521,234
Θ_{motion}	Samples I	19,614
Θ_{seq}	Disjoint sequences I^s	732
$\mathcal{H} \wedge \Theta_{\text{support}}$	Object candidates h	58,682
$\Theta_{\text{size}} \wedge \Theta_{\text{shape}}$	CSG nodes V^{Θ}	49,230
$\Theta_{\text{static}} \wedge \Theta_{\text{3D}} \wedge \Theta_{\text{app}}$	CSG edges E^{Θ}	431,121
CSG clustering	Partial objects m_k	2215
Object CSG clustering	Full objects M_k	464

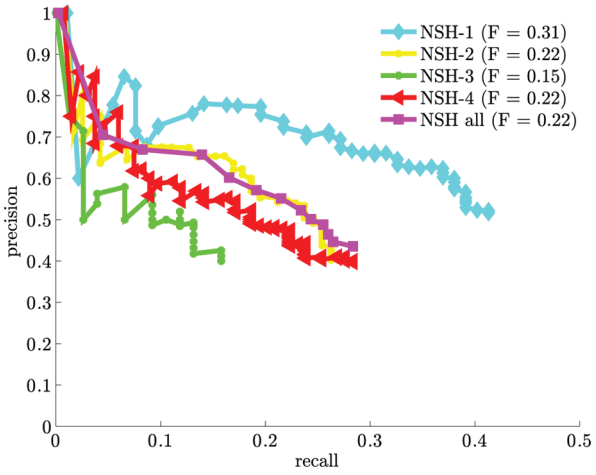

Fig. 13. Floor-by-floor evaluation of HerbDisc on the NSH Dataset. HerbDisc achieves a 28% higher recall in regular office environments (NSH-1) compared with laboratory and machine shop environments (NSH-3). Mixed environments containing both laboratories and offices (NSH-2 and NSH-4) achieve similar recall. HerbDisc achieves a maximum recall of 28.6% in the overall NSH Dataset at 44.4% precision, compared with 43.9% maximum recall in office environments (NSH-1) and 15% in laboratories and machine shops (NSH-3).

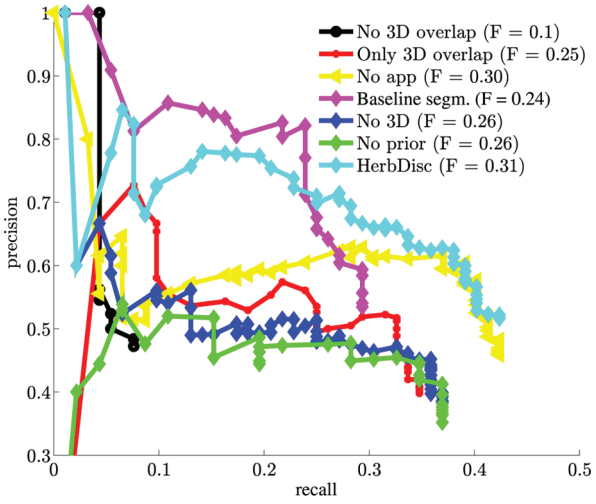

Fig. 14. Summary of precision–recall curves for the ablative analysis. HerbDisc is the best-performing method, combining the results of all constraints for improved object discovery in the NSH-1 dataset.

conjunctive constraints for efficiency. Figure 11 (middle) shows the CSG for each constraint independently. The product of all adjacency matrices (rightmost column) is the same for both approaches, but HerbDisc is more efficient. The metadata-based constraints Θ_{seq}, Θ_{static} are significantly more discriminative than the visual features Θ_{3D} and Θ_{app}. The adjacency matrix for $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ also illustrates the behavior of the dynamic motion filter, generating sequences of different length (i.e., squares of different size) depending on HERB's motion. Figure 11 (bottom) shows the result of using visual similarity constraints with no metadata. In this case, the product of all adjacency matrices (rightmost column) is significantly denser than in HerbDisc, which accounts for the increased computation time shown in Table 3.

8.3.2 HerbDisc in the NSH Dataset. In Section 7, we explored the impact of each individual component of HerbDisc. We provide a summary plot in Figure 14 that combines the precision–recall curves of all components.

The attempt to evaluate Θ_{visual} on the NSH Dataset was unsuccessful, after the testing machine had made barely any progress after a week of processing. HerbDisc processes the NSH Dataset in 18 min 34 s. We show an itemized list of running times for the different steps in Table 4, and the statistics for images, candidates, edges, etc., in Table 5. The overall running time does not include data acquisition time (and motion filtering and candidate generation, which we execute in parallel with the data acquisition). The most expensive step is the CSG construction, which processes 732 connected components in the

CSG, for a total of 49,230 nodes and 4.9 million edges, with 431,121 edges satisfying all constraints, in 11 min 49 s. The CSG clustering step is the second most expensive step, separating 2215 clusters (i.e. partial objects) in 3 min 31 s. The object CSG clustering and model registration are the most expensive per-unit steps. However, they leverage the filtered information from the CSGs to cluster and register 464 objects in 2 min 45 s.

We discover a total of 464 object models in the NSH Dataset, where 121 unique objects are *correct* (28.6% recall) and 85 are *valid* (44.4% precision). In Figure 13, we show the precision–recall curves for the NSH Dataset, as well as a floor-by-floor analysis (NSH-1 to NSH-4). We see a clear difference in performance as we move from regular office environments (NSH-1) to laboratories and machine shops (NSH-3). In office environments, HerbDisc displays a maximum recall of 43.9% at 52% precision, and 78% precision at 20% recall. In contrast, we only achieve a maximum recall of 15% at 41% precision in the laboratories of NSH-3 (e.g. Figure 4 (2, 3–5)), which include multiple shiny metallic objects, specular reflections, and extreme clutter.

We can also modify the configuration of HerbDisc on the fly to achieve different behaviors. For example, if we are more interested in precision than recall, we can use Θ_{support} as a standalone candidate generator and achieve 82% precision at 25% recall (on NSH-1), or reject the lowest-ranked models and achieve 60% precision at 40% recall. We show examples of *correct* objects in Figure 15 and of *valid* and *invalid* objects in Figure 16.

The *correct* objects discovered by HerbDisc are predominantly objects we would expect in an office environment, such as laptops, books, phones, monitors, keyboards, and mice. Other objects, such as basketball balls, watering cans, plants, and food items, showcase the object diversity, and therefore difficulty, from the NSH Dataset. We require objects to be 100% pure to be considered *correct*, which assures a high quality for potential robotics applications. For example, for the maximum recall configuration, we discover 75% of the labeled keyboard instances, 66% of books, 63% of mugs, and 51% of laptops. Shiny, metallic objects are particularly hard to discover, as the Kinect often fails to produce decent point clouds with them. Therefore, objects such as plyers, screwdrivers, adjustable wrenches, and other mechanic tools are all outright missed.

In an open task such as object discovery, it is nearly impossible to obtain comprehensive ground truth. HerbDisc discovers objects that the annotators considered outside the guidelines for ground truth in Section 6, such as chairs, trashcans, or wall-mounted paper holders (see Figure 16). The discovery of such objects can be due to several reasons. First, the object priors specified in HerbDisc may not be specific enough, accepting objects that HERB cannot manipulate (e.g. chairs and people). Other objects are not considered correct due to semantic meaning (e.g. the object is a part of a more complex object, such as a bike seat or a chair's armrest), because the object is immovable (e.g. a wall-mounted paper holder), or because the annotators did not notice or recognize the object (e.g. paper folders, cables). We believe that the only way to disambiguate between these cases is to interact with the objects during the discovery process, which is a future direction. Our framework can be used to leverage interaction information if available, as well as any other source of metadata, when formulated as constraints.

Among the *invalid* objects, we identify three main categories: (1) correct but impure objects; (2) groups of objects; and (3) mixtures of fragments. The first case refers to correctly discovered objects that contain a few (or sometimes only one) misplaced candidates, such as the red cup or the plastic bag in Figure 16. Objects in the second case are usually compound of multiple objects very close to each other or touching each other, such as groups monitor–keyboard–mouse or stapler–stapler–table. The third case comprises unrecognizable groups of object candidates from multiple objects. Invalid objects in cases (1) and (3) are mostly due to clustering errors, which improperly unite candidates from different objects. Objects in case (2) are mostly due to candidate generation/segmentation errors, failing to separate the individual objects in complex scenes. At maximum recall, 69% of the missed objects are part of invalid objects (i.e. at least one image of the object is correctly segmented, but incorrectly associated with an invalid object), and 31% are outright missed (mostly shiny, metallic objects, such as adjustable wrenches and other mechanic tools, whose images produce very poor segmentation results). Among the invalid objects, 64% are groups of objects, 26% are correct but impure objects, and 10% are mixtures of fragments.

9. Conclusions

In this paper, we have proposed a solution to discover objects during an entire workday of a robotic agent, processing over 6 hours of raw video stream in under 19 minutes. This solution is a first step toward solving our long-term goal of LROD. The LROD problem, which we have also introduced in this paper, is the problem of discovering new objects in the environment during an entire robot's lifetime: while the robot operates, for as long as the robot operates.

We claim that the key to make LROD a feasible problem is domain knowledge (robotic metadata). We have introduced a novel formulation to encode generic domain knowledge and visual information as graph constraints in a graph-based framework. In this formulation, we can combine multiple constraints using boolean logic expressions. The resulting metadata-augmented graphs, which we term CSGs, provide a common framework to encode any source of information for object discovery.

To assess the validity of our framework, we have introduced an optimized implementation of object discovery called HerbDisc. In HerbDisc, we efficiently discover objects in large datasets by leveraging the natural

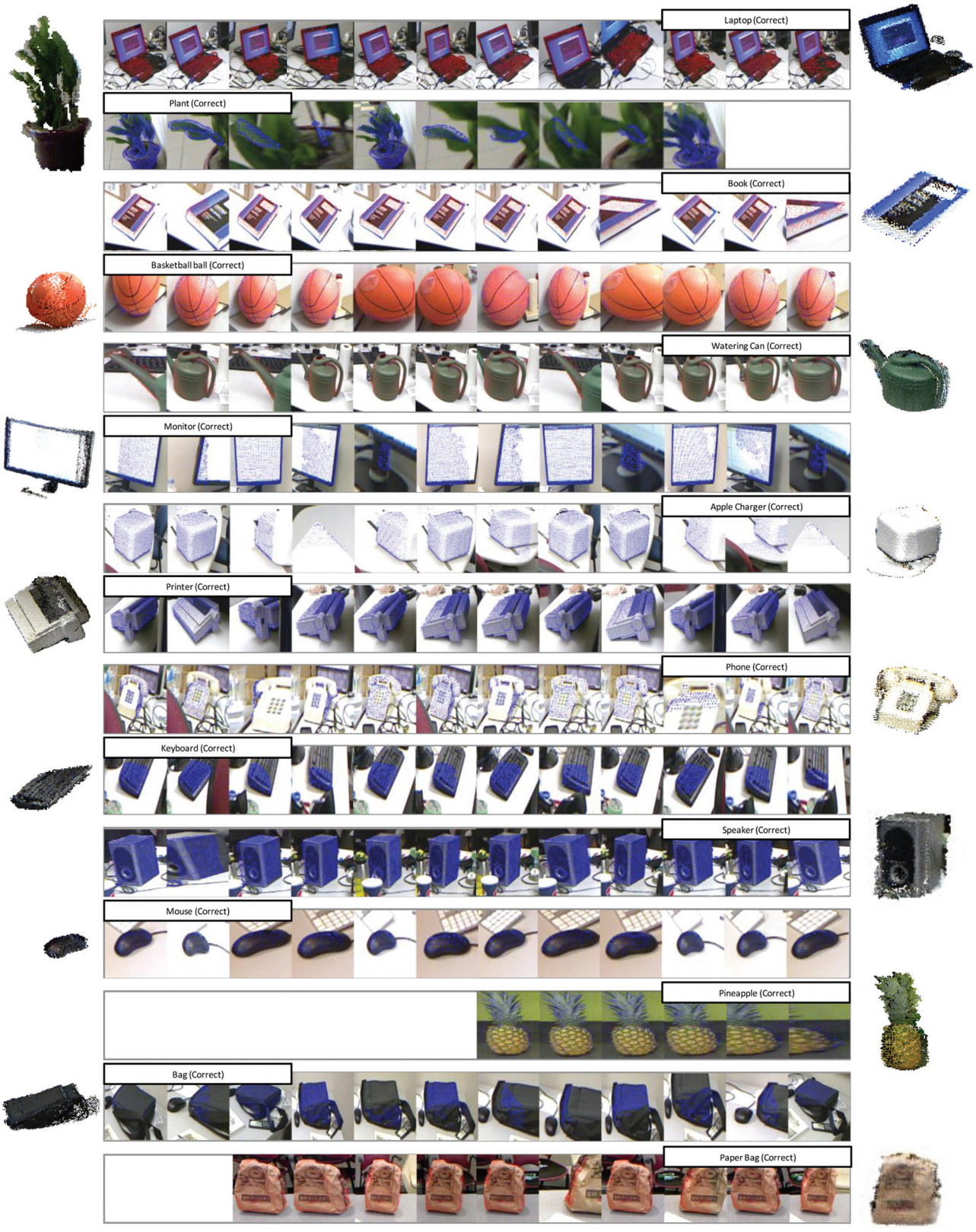

Fig. 15. Examples of *Correct* objects. For each object, we display its object label (text box); its 3D model (left/right); and 10 randomly selected images from the set of object candidates h_i (center), with the 3D point clouds h_i^P overlaid in red or blue over each image.

constraints of service robotics about the environment, the robotic agent, and the sensors, as well as the visual information. We have gathered a dataset of over half a million RGBD images (6 h 20 min of raw RGBD video) of

Fig. 16. Examples of *Valid* and *Invalid* objects. For each object, we display its object label (text box); its 3D model (left/right); and 10 randomly selected images from the set of object candidates h_i (center), with the 3D point clouds $h_i^{\boldsymbol{P}}$ overlaid in red or blue over each image.

office and lab environments, ranging from moderately to extremely cluttered, and containing 423 ground truth objects, in order to evaluate HerbDisc in a dataset of a realistic robotic workday. HerbDisc processed this dataset in

under 19 minutes and discovered 206 novel objects, such as monitors, keyboards, plants, and food items, with a maximum recall of 28.6% at 44.4% precision, and 68% precision at 15% recall (and, for regular office environments, maximum recall of 43.9% at 52% precision, and 78% precision at 20% recall). A key feature to make LROD feasible is system adaptability to changing conditions. In our framework, we showed that we can opportunistically leverage different sources of information adaptively, when conditions change, just by changing the operating constraints in the graph-based formulation.

And yet, despite discovering hundreds of novel objects, HerbDisc failed to discover over half of the total number of objects. We believe that, in order to truly solve the LROD problem, it will be necessary to transform the robot from an observer to an active agent, interacting with objects and leveraging that information to discover and validate discovered objects. With our framework, we can encode the information coming from interaction as more effective graph constraints, to discover objects resulting from that interaction. A future direction for our research is to develop effective interaction strategies to discover novel objects, to disambiguate when uncertain, and to validate the discovered objects by interacting with them.

Another related future direction is to explore online techniques for object discovery. The framework described here is essentially a batch process, so that it can be processed during the robot's downtime. However, online processing could be performed using the sequences provided by the motion filter. Once the motion filter generates a new sequence, of up to 20 seconds, we can cluster and generate partial objects for that sequence. We would perform an object CSG clustering step every few hours, to join the most recent partial objects with the full objects found in previous object CSG clusterings.

Acknowledgements

Special thanks are due to the members of the Personal Robotics Lab at Carnegie Mellon University for insightful comments and discussions.

Funding

This work was partially supported by the National Science Foundation (grant number EEC-0540865).

References

Bjorkman M and Kragic D (2010) Active 3D scene segmentation and detection of unknown objects. In: *IEEE international conference on robotics and automation*, pp. 3114–3120. Piscataway, NJ: IEEE Press.

Borrmann D, Elseberg J, Lingermann K, Nuchter A and Hertzberg J (2008) The efficient extension of globally consistent scan matching to 6 DOF. In: *International symposium on 3D data processing, visualization and transmission*.

Bouvrie JV (2004) *Multi-Source Contingency Clustering*. Master's Thesis, Massachusetts Institute of Technology (MIT), USA.

Brandes U (2001) A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology* 25(2): 163–177.

Collet A (2012) *Lifelong Robotic Object Perception*. Doctoral dissertation, Carnegie Mellon University, USA.

Collet A, Srinivasa SS and Hebert M (2011) Structure discovery in multi-modal data: a region-based approach. In: *IEEE international conference on robotics and automation*, Shanghai. Piscataway, NJ: IEEE Press.

Collet A, Xiong B, Gurau C, Hebert M and Srinivasa SS (2013) Exploiting domain knowledge for object discovery. In: *IEEE international conference on robotics and automation*.

Fitzpatrick P (2003) First Contact: an active vision approach to segmentation. In: *international conference on intelligent robots and systems*.

Herbst E, Ren X, Fox D and Henry P (2011) Toward object discovery and modeling via 3-D Scene comparison. In: *IEEE international conference on robotics and automation*. Piscataway, NJ: IEEE Press.

Hoiem D, Stein AN, Efros AA and Hebert M (2007) Recovering occlusion boundaries from a single image. In: *IEEE international conference on computer vision*, pp. 1–8. Piscataway, NJ: IEEE Press.

Hore P, Hall L and Goldgof D (2006) A cluster ensemble framework for large data sets. *IEEE international conference on systems, man and cybernetics*, pp. 3342–3347.

Izadi S, Kim D, Hilliges O, et al. (2011) KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In: *ACM symposium on user interface software and technology.*

Kang H, Hebert M, Efros AA and Kanade T (2012) Connecting missing links : object discovery from sparse observations using 5 million product images. In: *European conference on computer vision*.

Kang H, Hebert M and Kanade T (2011) Discovering object instances from scenes of daily living. In: *international conference on computer vision*.

Kootstra G and Kragic D (2011) Fast and bottom-up object detection, segmentation, and evaluation using Gestalt principles. In: *IEEE international conference on robotics and automation*, pp. 3423–3428.

Lalonde JF, Vandapel N, Huber DF and Hebert M (2006) Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics* 23(10): 839–861.

Lee YJ and Grauman K (2011) Learning the easy things first: self-paced visual category discovery. In: *IEEE conference on computer vision and pattern recognition*, pp. 1721–1728.

Marton ZC, Pangercic D, Blodow N, Kleinehellefort J and Beetz M (2010) General 3D modelling of novel objects from a single view. In: *IEEE international conference on intelligent robots and systems*, pp. 3700–3705.

Mishra AK and Aloimonos Y (2011) Visual segmentation of "simple" objects for robots. In: *Robotics: science and systems*.

Morwald T, Prankl J, Richtsfeld A, Zillich M and Vincze M (2010) BLORT - The Blocks World Robotic Vision Toolbox. In: *IEEE international conference on robotics and automation workshops*.

Philbin J, Sivic J and Zisserman A (2010) Geometric Latent Dirichlet Allocation on a Matching Graph for Large-scale Image Datasets. *International Journal of Computer Vision* 95(2): 138–153.

Ruhnke M, Steder B, Grisetti G and Burgard W (2009) Unsupervised learning of 3D object models from partial views. *IEEE international conference on robotics and automation*, pp. 801–806

Russell BC, Freeman WT, Efros AA, Sivic J and Zisserman A (2006) Using multiple segmentations to discover objects and their extent in image collections. In: *IEEE conference on computer vision and pattern recognition*. Piscataway, NJ: IEEE Press.

Russell BC, Torralba A, Murphy KP and Freeman WT (2008) LabelMe: a database and Web-based tool for image annotation. *international journal of computer vision* 77(1–3): 157–173.

Rusu RB, Blodow N and Beetz M (2009) Fast Point Feature Histograms (FPFH) for 3D registration. *IEEE international conference on robotics and automation*, pp. 3212–3217.

Rusu RB and Cousins S (2011) 3D is here: Point Cloud Library (PCL). In: *IEEE international conference on robotics and automation*.

Rusu RB, Marton ZC, Blodow N, Dolha M and Beetz M (2008) Towards 3D point cloud based object maps for household environments. *Robotics and Autonomous Systems* 56(11): 927–941.

Sivic J, Russell BC, Efros AA, Zisserman A and Freeman WT (2005) Discovering objects and their location in images. In: *IEEE international conference on computer vision*. Piscataway, NJ: IEEE Press.

Somanath G, Rohith MV, Metaxas D and Kambhamettu C (2009) D - Clutter: building object model library from unsupervised segmentation of cluttered scenes. In: *IEEE conference on computer vision and pattern recognition*. Piscataway, NJ: IEEE Press.

Srinivasa SS, Ferguson D, Helfrich CJ, et al. (2010) HERB: a home exploring robotic butler. *Autonomous Robots* 28(1): 5–20.

Strehl A and Ghosh J (2002) Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research* 3(1): 583–617.

Tang W, Lu Z and Dhillon IS (2009) Clustering with multiple graphs. In: *IEEE international conference on data mining*.

Troyanskaya OG, Dolinski K, Owen AB, Altman RB and Botstein D (2003) A Bayesian framework for combining heterogeneous data sources for gene function prediction (in *Saccharomyces cerevisiae*). *Proceedings of the National Academy of Sciences* 100(14): 8348–8353.

Tuytelaars T, Lampert CH, Blaschko MB and Buntine W (2009) Unsupervised object discovery: a comparison. *International Journal of Computer Vision* 88(2): 284–302.

Weber M, Welling M and Perona P (2000) Towards automatic discovery of object categories. In: *IEEE conference on computer vision and pattern recognition*, vol. 2, pp. 101–108. Piscataway, NJ: IEEE Press.

Zeng E, Yang C, Li T, et al. (2010) Clustering genes using heterogeneous data sources. *International Journal of Knowledge Discovery in Bioinformatics* 1(2): 12–28.