Exploiting Domain Knowledge for Object Discovery

Alvaro Collet* Bo Xiong† Corina Gurau‡ Martial Hebert* Siddhartha S. Srinivasa*

*Carnegie Mellon University {acollet, hebert, siddh}@cs.cmu.edu
†Connecticut College bxiong@conncoll.edu
‡Jacobs University cgurau@jacobs-university.de

***Abstract*—In this paper, we consider the problem of *Lifelong Robotic Object Discovery* (LROD) as the long-term goal of discovering novel objects in the environment while the robot operates, for as long as the robot operates. As a first step towards LROD, we automatically process the raw video stream of an entire workday of a robotic agent to discover objects.**

We claim that the key to achieve this goal is to incorporate domain knowledge whenever available, in order to detect and adapt to changes in the environment. We propose a general graph-based formulation for LROD in which generic domain knowledge is encoded as constraints. Our formulation enables new sources of domain knowledge—metadata—to be added dynamically to the system, as they become available or as conditions change. By adding domain knowledge, we discover 2.7× more objects and decrease processing time 190 times. Our optimized implementation, HerbDisc, processes 6 h 20 min of RGBD video of real human environments in 18 min 30 s, and discovers 121 correct novel objects with their 3D models.

I. Introduction

We focus on the problem of *Lifelong Robotic Object Discovery* (LROD): discovering new objects unsupervised in an environment while the robot operates, for as long as the robot operates. Key challenges in LROD are scalability to massive datasets, robustness in dynamic human environments, and adaptability to different conditions over time. As a first step towards LROD, we address these challenges on an entire workday of data, collected as our robot explored a cluttered and populated office building. We show for the first time a system that processes, in under 19 minutes, hundreds of thousands of samples and over 6 h of continous exploration, to discover over a hundred new objects in cluttered human environments.

Our key insight to making LROD feasible is to incorporate *domain knowledge*. More precisely, the data gathered by a service robot is not an unordered collection of anonymous images: range data is also available; we may 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 the images; or we may only be interested in objects of certain sizes or shapes relevant to the robot. In our work, we define the term *metadata* to encode *any* source of domain knowledge. Our definition includes any additional robotic sensing, assumptions, or prior information; in short, any non-visual data that can provide information about the object candidates.

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 may result in solutions of limited applicability. [1] 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 [2], [3]. [4] use 3-frame sequences, motion cues, and assume that images contain a table with known color distribution to discover and accurately segment objects in cluttered scenes. [5] assumes 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 [4] and [5] assume knowledge about the image ordering and sequencing. [6] use datasets consisting of multiple sequences of images collected repeteadly in the same locations, in order to compute per-sequence environment

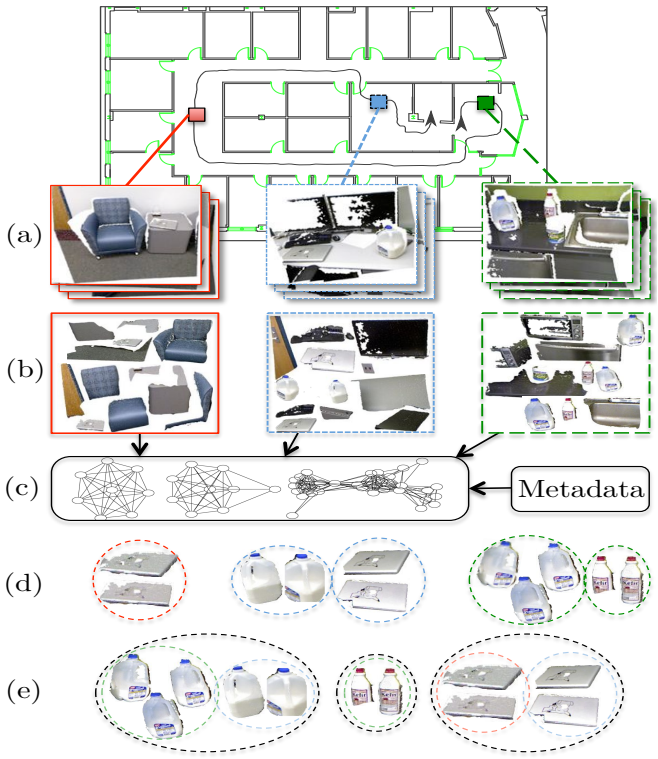


Fig. 1: Object Discovery with Metadata. (Top) Robotic agent navigates through office environment storing an RGBD video stream and localization information. (a) Spatial/temporal constraints separate the video stream in subsets red, green and blue. (b) Images in the sequence are segmented to generate object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

maps and perform scene differencing to discover movable objects. The implicit assumptions include a rough knowledge of the robot location, recording time, and that the robotic agent visits the same locations multiple times.

Consider the example in Fig. 1, in which a robotic agent navigates through an office environment recording an RGBD video stream (Fig. 1(a)). Unsupervised Object Discovery techniques (e.g., [7]) create a pool of object candidates (e.g., the RGBD regions in Fig. 1(b)), which are represented as nodes in a pairwise graph (Fig. 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—recurring patterns—(Fig. 1(d-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.

In contrast, we can exploit metadata to make the problem feasible. In Fig. 1(a), we can split the datastream according to the robot's location and data acquisition timestamps (red-blue-green subsets). The object candidates for each subset (Fig. 1(b)) are compared only within the same subset. The pairwise graphs in Fig. 1(c) can encode the visual similarity between candidates and other cues such as staticity constraints or object priors. In Fig. 1(d), we can 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. Therefore, we perform a global clustering step (Fig. 1(e)) to join the multiple object instances as single object models.

The main contribution of this work is a general framework for object discovery that leverages *any* form of metadata. In our formulation, we do not distinguish between visual similarity and 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 a probability p that quantifies the answer's confidence. 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 $p = s(h_i, h_j)$.

With constraints, we can seamlessly combine multiple similarities and metadata. We define a set of logic operations over constraints to form complex constraint expressions that encode all our knowledge relevant to discovering objects. We formulate the LROD problem as a partitioning of graphs built over constraints, which we term Constrained Similarity Graphs (CSGs). CSGs, coupled with service robotics constraints, are by construction much sparser than regular visual similarity graphs, and produce many connected components. 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. We exploit this advantage in our optimized implementation, HerbDisc, to process over 6 h of RGBD video in under 19 min and discover 121 objects.

By formalizing domain knowledge and visual similarity as constraints, we can change the entire system behavior online, and exploit metadata adaptively, as conditions change. In addition, our constraints-based formulation is general, it covers both generic Unsupervised Object Discovery algorithms (e.g., [8], [7]) and special-purpose algorithms (e.g., [5]).

II. Problem Formulation

A. Inputs and Outputs

The visual input to HerbDisc is a set $\boldsymbol{I}$ of N images with associated range data, which we refer to as data samples I_n. A candidate generator processes the data samples in $\boldsymbol{I}$ to compute a set of data fragments $\boldsymbol{h}$, which we consider the object candidates (e.g., Fig. 1(b)). Each object candidate $h_i = \{h_i^{\boldsymbol{rgb}}, h_i^{\boldsymbol{P}}, h_i^{\boldsymbol{\Phi}}\}$ is defined by a set of image pixels $h_i^{\boldsymbol{rgb}}$, a set of 3D points $h_i^{\boldsymbol{P}}$, and a set of metadata attributes $h_i^{\boldsymbol{\Phi}}$.

The output of this framework is a set of 3D object models $\boldsymbol{M}$. Each object $M_k = \{M_k^{\boldsymbol{rgb}}, M_k^{\boldsymbol{P}}, M_k^{\boldsymbol{h}}\}$ is defined by a set of 3D points $M_k^{\boldsymbol{P}}$ with associated color $M_k^{\boldsymbol{rgb}}$ and the set of object candidates $M_k^{\boldsymbol{h}} = \{h_{1,k}, \ldots, h_{i,k}, \ldots\}$ used to create object M_k.

B. Constraints

Constraints encode generic information about an object candidate h_i or a relationship between candidates h_i, h_j. There are two types of constraints: node constraints Θ^{n} (which encode information about a single candidate) and edge constraints Θ^{e} (which encode information about the relationship between a pair of candidates). Table I shows a list of the constraints we use in this work. We model each constraint Θ as a Bernoulli distribution with probability of success p. Node constraints Θ^{n} modify a single object candidate h_i, such that

$$\Theta^{\text{n}} : h_i \mapsto \{0, 1\} \qquad P(\Theta^{\text{n}}(h_i) = 1) = p. \tag{1}$$

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

$$\Theta^{\text{e}} : h_i, h_j \mapsto \{0, 1\} \qquad P(\Theta^{\text{e}}(h_i, h_j) = 1) = p. \tag{2}$$

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

III. Framework overview

We describe the general flowchart of our framework in this section. In the following sections, we focus on the novel elements of this paper: defining constraints (Section IV), generating CSGs (Section IV-C), and the implementation of constraints and CSGs in HerbDisc (Section VI). We provide a list of the constraints implemented in HerbDisc in Table I.

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 [9] (Section VI-A).

2. CSG Generation. We create a graph of relationships between object candidates using constraints Θ. We denote the CSG built by Θ as $G^{\Theta} = (E^{\Theta}, V^{\Theta})$ (Section IV-C).

If Θ encodes a visual similarity, then G^{Θ} is equivalent to regular pairwise similarity graphs in Unsupervised Object Discovery (e.g., [7]). Applying the constraints in Table I to create G^{Θ} produces multiple connected components G_g^{Θ}.

Constraint	Type	Information	Source	Description	Section
Θ_{motion}	node	Relative camera motion	A	Acquire data samples only if there is motion (no repeated frames).	VI-B
Θ_{seq}	edge	"data comes in sequences"	S	Split data stream in sequences based on camera motion and maximum length.	VI-B
Θ_{support}	node	"objects have surfaces of support"	E	Reject candidates not supported by horizontal or vertical planes (tables or walls).	VI-A
Θ_{static}	edge	"scene is static for a few seconds"	E	Measure 3D overlap between candidates.	VI-C
Θ_{size}	node	Object size	E	Compare candidate's size with object prior.	VI-D
Θ_{shape}	node	Object shape	E	Compare candidate's shape with object prior.	VI-D
Θ_{app}	edge	Visual Similarity	V	Compare visual similarity between candidates using color histograms.	VI-E
Θ_{3D}	edge	Shape Similarity	V	Compare shape similarity between candidates using FPFH features.	VI-E

TABLE I: Constraints used in HerbDisc. For each 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: E (metadata about the environment), A (metadata about the robotic agent), S (metadata about the sensors), or V (visual information).

3. CSG Clustering. We compute groups of candidates for each $G^{\Theta}_g \in G^{\Theta}$ with the graph partitioning algorithm of [10]. [10] is a greedy community discovery method based on Betweenness Centrality, which is very efficient for sparse graphs and works well for our problem. Each cluster C_i contains a set of candidates $\boldsymbol{h}_i$, which are registered together and merged to compute partial 3D models m_i using [11].

4. Object CSG Generation. We compute a CSG $G^m = (E^m, V^m)$ over partial object models $m_i \in \boldsymbol{m}$. The number of nodes in this graph is orders of magnitude smaller than G^{Θ}. Only a subset of the constraints from Table I are available (we use Θ_{size}, Θ_{shape}, Θ_{app}, and Θ_{3D}), as many constraints require local information not relevant for the partial objects.

5. Object Clustering. We compute clusters of partial 3D models using the graph partitioning of [10] on G^m. Each cluster C_k contains partial object models m_i.

6. 3D model generation. We generate full object models M_k from clusters of partial object models C_k. We globally register the partial models with Global Alignment [11] to produce full 3D models.

IV. Information as Constraints

A. Defining Constraints

Consider the pair of scenes illustrated in Fig. 2, in which we encode as constraints the assumptions $\Theta^{\text{n}}_{\text{planar}}$ = "objects are planar", $\Theta^{\text{e}}_{\text{static}}$ = "scene is static", and $\Theta^{\text{n}}_{\text{tables}}$ = "objects lie on tables". Encoding $\Theta^{\text{n}}_{\text{planar}}$ requires answering the question "is candidate h_i planar?". If we can measure whether an object is planar or not (e.g., by computing 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 Fig. 2(2,2). Similarly, to encode Θ_{tables} we must answer the question "is candidate h_i on a table?" for which we need to 1) detect a table, and 2) determine if candidate h_i is on it. If we can measure these two factors, then the assumption can be encoded as a node constraint, with the result shown in Fig. 2(2,3). Finally, the assumption "the scene is static" implies to answer affirmatively that "do candidates h_i at time t and h_j at time $t+1$ occupy the same location in space?" If we can register the two scenes and h_i and h_j occupy the same 3D location, then $\Theta^{\text{e}}_{\text{static}}$ would be satisfied with p proportional to the overlap between h_i and h_j. The result of $\Theta^{\text{e}}_{\text{static}}$ is shown in Fig. 2(1,3).

Some sources of metadata may also operate over both nodes and edges. For example, object tracking can be encoded as a union of an edge constraint Θ^{e} = "are candidates h_i at time t and h_j at time $t+1$ the same object?," and a node constraint Θ^{n} = "is candidate h_i being tracked?" To incorporate such constraints, we redefine Θ as a pair $\Theta \equiv (\Theta^{\text{n}}, \Theta^{\text{e}})$. Constraints that operate only on nodes or edges should 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.

Pairwise similarity functions also induce constraints Θ. In particular, a normalized similarity function $s(h_i, h_j) \in [0, 1]$ induces an edge constraint Θ^{e} with $P_{\Theta^{\text{e}}}(h_i, h_j) = s(h_i, h_j)$. In HerbDisc, we do not distinguish between visual similarity and metadata: they are all encoded as constraints Θ_i. This unification is very useful to combine multiple constraints (Section IV-B) and build CSGs (Section IV-C).

B. 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^{n}_i, Θ^{n}_j be independent node constraints induced by metadata, and $P_{\Theta}(h)$ the probability of candidate h satisfying Θ^{n}. Then, the negation operator $\neg\Theta^{\text{n}}_i$ is computed as

$$P_{\neg\Theta^{\text{n}}_i}(h) = 1 - P_{\Theta^{\text{n}}_i}(h), \tag{3}$$

which represents the probability of h *not* satisfying Θ^{n}. The conjunction operator $\Theta^{\text{n}}_i \wedge \Theta^{\text{n}}_j$ is then computed as

$$P_{\Theta^{\text{n}}_i \wedge \Theta^{\text{n}}_j}(h) = P_{\Theta^{\text{n}}_i}(h) P_{\Theta^{\text{n}}_j}(h). \tag{4}$$

Finally, the disjunction operator $\Theta^{\text{n}}_i \vee \Theta^{\text{n}}_j$ is computed as

$$P_{\Theta^{\text{n}}_i \vee \Theta^{\text{n}}_j}(h) = 1 - P_{\neg\Theta^{\text{n}}_i \wedge \neg\Theta^{\text{n}}_j}(h). \tag{5}$$

We analogously define the conjunction $\wedge$, disjunction $\vee$ and negation $\neg$ operators for edge constraints, by substituting $P_{\Theta^{\text{n}}}(\cdot)$ for $P_{\Theta^{\text{e}}}(\cdot, \cdot)$ in Eq. (3), Eq. (4) and Eq. (5).

Logic operations over constraint pairs $\Theta = (\Theta^{\text{n}}, \Theta^{\text{e}})$ operate on Θ^{n} and Θ^{e} independently, so we define negation $\neg\Theta_i = (\neg\Theta^{\text{n}}_i, \neg\Theta^{\text{e}}_i)$, conjunction $\Theta_i \wedge \Theta_j = (\Theta^{\text{n}}_i \wedge \Theta^{\text{n}}_j, \Theta^{\text{e}}_i \wedge \Theta^{\text{e}}_j)$, and disjunction $\Theta_i \vee \Theta_j = (\Theta^{\text{n}}_i \vee \Theta^{\text{n}}_j, \Theta^{\text{e}}_i \vee \Theta^{\text{e}}_j)$.

Any logic operation can be derived from the negation, conjunction, and disjunction operators. We can now define arbitrarily complex constraint expressions based on logic operations over primitive constraints. In Fig. 2(4,1), to search for objects assuming that "the scene is static" AND that

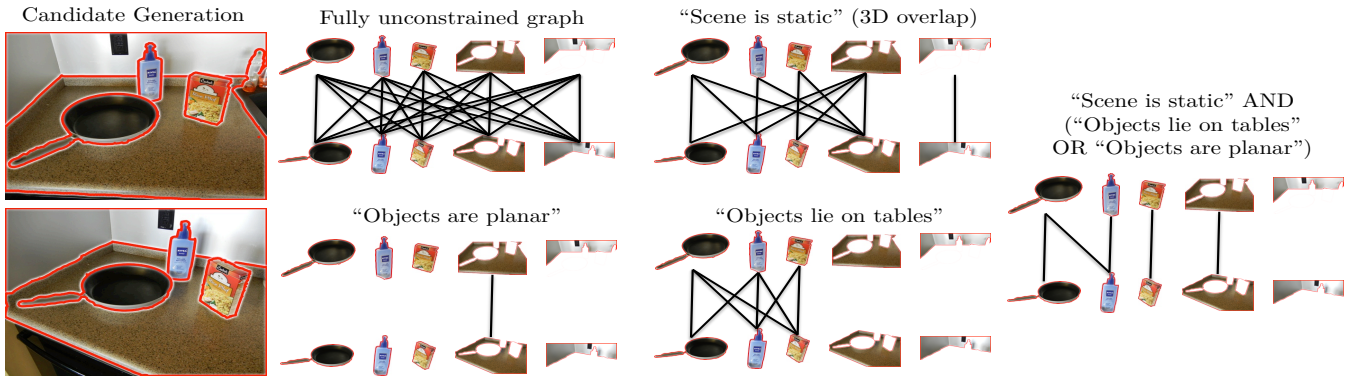


Fig. 2: Metadata induces constraints on pairwise graphs. The fully unconstrained graph is seldom computed in practice, as techniques such as inverted indexes are used to preselect potential matches [12]. Our formulation generalizes such techniques, constraining a graph based on any source of metadata (columns 2-3). Most importantly, our formulation facilitates the creation of complex constraint expressions combining multiple constraints (column 4).

"objects that lie on tables" OR "objects are planar", we simply define the constraint $\Theta = \Theta_{\text{static}} \wedge (\Theta_{\text{tables}} \vee \Theta_{\text{planar}})$.

A generic Θ can be composed of multiple Θ_i using the logic operators defined above,

$$\Theta = \Theta_1 \circ \Theta_2 \circ \ldots \circ \Theta_i \circ \ldots, \tag{6}$$

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

C. 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 Θ if and only if: 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) G^Θ has node weights $w(h_i) = P_{\Theta^{\text{n}}}(h_i)$, and edge weights $w(h_i, h_j) = P_{\Theta^{\text{e}}}(h_i, h_j)$. We generate G^Θ for Θ following Algorithm 1.

Algorithm 1 Building a Constrained Similarity Graph

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

In Algorithm 1, $p_{\min}$ denotes the threshold probability for nodes and edges (in normal conditions, $p_{\min} = 0.5$). The CSG construction and the entire framework are independent of the particular choice of Θ. Θ can be any arbitrarily complex constraint expression, ranging from visual similarity only to multiple sources of metadata, as we implement in HerbDisc.

The CSG construction 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, even complete graphs, which are $\mathcal{O}(n^2)$. Evaluating a constraint for a node or edge can also be expensive (e.g., visual similarities).

In practice, we can speed up the CSG construction by using conjunctive constraint expressions (as in Eq. (4)), and positioning the most restrictive constraints first. 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} as in Algorithm 2.

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

The complexity of Algorithm 2 for a given Θ and G^{Θ_0} is $\mathcal{O}(|E^{\Theta_0}|)$. The size of G^{Θ_0} determines the complexity of building G^Θ. Therefore, an appropriate choice of Θ_0 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. The motion and sequencing constraints $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ (see Section VI-B for details) 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 that m is fixed and $m \ll n$ in realistic situations (in the NSH Dataset, $m = 50$ and $n = 521234$). Then, the CSG construction given G^{Θ_0} has a complexity of $\mathcal{O}(n)$ for the remaining constraints $\Theta_k \in \boldsymbol{\Theta}$. Visual similarities are the most expensive constraints, so it is crucial to perform this optimization to only compute $\mathcal{O}(n)$ of them. See Table II for a quantitative evaluation of the reduced complexity of this method.

The constraints Θ and the generic CSG construction of Algorithm 1 are designed for both soft constraints (i.e. Θ such that $P_\Theta \in [0, 1]$) and hard constraints (i.e. Θ such that



Fig. 3: The Kitchen Dataset (top row) and the NSH Dataset (bottom three rows). Each row show images with ground truth annotations for some of the environments we visited. Some scenes were so challenging (e.g., row 2, col 3-5) that the annotators could not separate the objects in the scene.

$P_\Theta \in 0, 1)$. In HerbDisc, we use Algorithm 2 with a mix of soft and hard constraints.

V. Datasets

We present two datasets of real human environments in which we evaluate HerbDisc: the Kitchen Dataset, and the NSH Dataset (see Fig. 3). We recorded both datasets by driving our robot HERB around the environment and capturing data with a Kinect RGBD camera at 640×480 resolution and an effective framerate of approximately 22 fps (due to throughput limitations).

We manually annotated both datasets to obtain ground truth, with the following labeling procedure. Our goal is to obtain the list of objects that HERB could potentially grasp. Since it is infeasible to annotate every single data sample—there are over half a million—we process each data stream with a motion filter to eliminate redundant samples (described in Section VI-B). We select 10 images for each scene (e.g., office, lab, kitchen), and label all objects in these images with the LabelMe tool [13]. As an estimate of the objects that HERB can grasp, we label any object that:

- is at least 10×5 cm (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).

Fig. 3 shows annotated examples from the Kitchen (top row) and NSH datasets (bottom 3 rows).

The Kitchen Dataset captures a 12-minute recording of HERB in a kitchen environment, with relatively clean scenarios and 40 ground truth objects.

The NSH Dataset is a stream of 6 hours and 20 minutes of HERB exploring the NSH building of Carnegie Mellon University, comprising 521234 data samples. We divided the recording in four fragments (NSH-1 to NSH-4) lasting between 1 h and 1 h 50 min each, one per building floor. 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. We labeled a total of 423 unique ground truth objects.

Time (min)	58.0	102.7	186.9	262.2	319.9	380.6
$\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$	686K	1.2M	2.5M	3.3M	4.0M	4.9M
Θ_{motion}	29.1M	83.9M	263M	517M	803M	1.2B
Raw data	2.9B	10.4B	30.4B	59.9B	89.2B	126.0B

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

VI. Implementation of HerbDisc

In this section, we describe the novel components of HerbDisc, focusing on how to formulate similarities, assumptions, and other metadata from Table I as constraints.

A. Constrained Candidate Generation

Candidate generators that rely on metadata are common in the robotics literature. For example, algorithms that track objects [5], that assume tabletop scenes [2], or that perform scene differencing [6] usually compute better candidates than generic objectness segmentation algorithms. 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 [9] with the assumption that objects have surfaces of support in floors, tables and walls. Θ_{support} is defined as $\Theta^{\text{n}}_{\text{support}}(h_i) = 1$ with $p = 1$ if supported(h_i, I_j). The supported$(\cdot, \cdot)$ function searches for large planes in the data sample I_j that generated candidate h_i, and accepts h_i if it lies within some distance (50 cm) above the planes found.

B. Motion and Sequencing

In LROD, we receive a neverending data stream of information from the robot sensing. 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.

We define Θ_{motion} to sample the input data stream at a dynamic framerate depending on HERB's motion, and we define Θ_{seq} to split the subsampled data stream into small subsets that we term *sequences*. We enforce a maximum sequence length m to limit the order $|V^{\Theta_{\text{seq}}}|$ of any connected component in the CSG.

In practice, Θ_{motion} samples the data stream when there is enough motion γ_{min} between data samples. Given the transformation $T_{k,k-1}$ between consecutive samples I_k and I_{k-1}, and with $M : T \mapsto \mathbb{R}$ a magnitude of the motion T, we model the motion constraint Θ_{motion} for $h_i \in I_k$ as $\Theta^{\text{n}}_{\text{motion}}(h_i) = 1$ with $p = 1$ if $M(T_{k,k-1}) > \gamma_{\text{min}}$.

The sequencing constraint Θ_{seq} is defined as $\Theta^{\text{e}}_{\text{seq}}(h_i, h_j) = 1$ with $p = 1$ if $\text{seq}(h_i) = \text{seq}(h_j)$. For data sample I_k, the sequence identifier $\text{seq}(I_k)$ is incremented if there is too much motion ($M(T_{k,k-1}) > \gamma_{\text{max}}$) between sample I_k and I_{k-1}, or if we reach the maximum sequence length m.

We use $M(T) = \|T\|_F$ as an estimate of the relative motion T. γ_{min} and γ_{max} are calibrated so that we capture m data samples in 20 seconds moving in a straight line at HERB's slowest and fastest speed.

Table II shows the impact of $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$. On the NSH Dataset, the number of edges remaining in the CSG

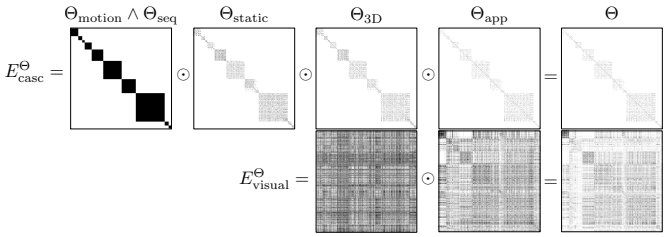


Fig. 4: CSG graphs for the edge constraints in HerbDisc, displayed as adjacency matrices (where a black dot indicates an edge between candidates), in the Kitchen Dataset. The overall graph E^{Θ} (rightmost column) is the Hadamard product of all adjacency matrices in a row. (top) CSGs using conjunctive constraints, as implemented in HerbDisc. (bottom) CSGs for visual similarity Θ_{app} and Θ_{3D} (in this case, E^{Θ} is a regular pairwise similarity graph). The CSG using metadata (top) is much more discriminative than the CSG for visual similarity only.

is decreased from 126 billion to 4.9 million (6 orders of magnitude) when using $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$.

To implement this motion filter, we modify the Kinect Fusion [14] 6DoF tracker available in PCL [15].

C. Spatial Overlap

We described Θ_{static} in Section IV-A. We model Θ_{static} as 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, so that $\Theta^{\text{e}}_{\text{static}} = 1$ with $p = s^{\text{overlap}}_{i,j}$ if $s^{\text{overlap}}_{i,j} > s^{\text{overlap}}_{\text{min}}$.

This constraint is designed to operate in unison with the sequencing constraint Θ_{seq}. 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. We measure $s^{\text{overlap}}_{i,j}$ between 3D point clouds $h^{\boldsymbol{P}}_i, h^{\boldsymbol{P}}_j$ by comparing their voxel grids (using the relative transformations from Kinect Fusion for registration).

D. Size/shape priors

We define a prior based on the sizes and shapes of known objects that HERB can grasp. The soft constraint $\Theta_{\text{prior}} = \Theta_{\text{size}} \wedge \Theta_{\text{shape}}$ is composed of size and shape components.

The function $s^{\text{size}}_i = s^{\text{size}}(h_i, h_{\text{prior}})$ estimates the likelihood that the longest dimension of h_i is sampled from a Gaussian distribution centered at the size given by h_{prior}. 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 [16] (linearity, planarity and scatterness). We then define Θ_{size} and Θ_{shape} analogously to previous sections, using the similarities s^{size}_i and s^{shape}_i respectively.

E. Visual and 3D shape similarity

For appearance features, we compute the color histogram of each candidate in LAB color space, as in [17], 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 [18], 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 $s(\cdot,\cdot) \in [0,1]$. We formulate these similarities as the soft constraints Θ_{app} and Θ_{3D} analogously to previous sections.

VII. Experiments

In this section, we evaluate the impact of metadata in object discovery. We first compare the performance of HerbDisc with and without any metadata. We then evaluate the scalability of metadata-augmented object discovery to very large datasets, such as the NSH Dataset.

A. 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 [9] with Θ_{support}.

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

$$\begin{aligned}\Theta_{\text{local}} = &\Theta_{\text{motion}} \wedge \Theta_{\text{seq}} \quad \wedge \Theta_{\text{support}} \wedge \Theta_{\text{static}} \wedge \\ &\Theta_{\text{size}} \quad \wedge \Theta_{\text{shape}} \wedge \Theta_{\text{app}} \quad \wedge \Theta_{\text{3D}}.\end{aligned} \tag{7}$$

In the Object CSG Clustering, we cluster the CSG built with

$$\Theta_{\text{global}} = \Theta_{\text{size}} \wedge \Theta_{\text{shape}} \wedge \Theta_{\text{app}} \wedge \Theta_{\text{3D}}. \tag{8}$$

In Θ_{local}, we compute the histograms in Θ_{app} with 6 bins per channel, and compute the FPFH features of Θ_{3D} for the centers of a 1 cm voxel grid. In Θ_{global}, we use 10 bins in Θ_{app}, and a 3 mm voxel grid for Θ_{3D}. In our experience, of Θ_{local} has significantly more impact in the overall performance than Θ_{global}. 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. We discretize each parameter in 5 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, 16GB of RAM, a nVidia GTX 580 GPU, and runninng 64-bit Ubuntu Linux 10.04.

B. Evaluation Procedure

We define three purity metrics to evaluate object discovery. **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 [19], we measure the group purity of model M as the largest percentage of *pure* object candidates in $M^{\boldsymbol{h}}_k = \{h_{1,k}, \ldots, h_{i,k}\}$ that belong to the same object. **3D purity:** We define an object's 3D point cloud $M^{\boldsymbol{P}}_k$ as *pure* if the 3D points in $M^{\boldsymbol{P}}_k$ cover over 80% of the area visible in the data samples for that particular object.

We distinguish between three categories of objects: *correct*, *valid*, and *invalid* (see Table III for the evaluation criteria).

We define Precision and Recall as in [7]. In [7], Precision is the ratio of *correct*+*valid* object models to the total number of discovered models, and Recall is the ratio of unique *correct* objects to the total number of ground truth objects.

We use the cluster size to estimate the quality of an object, and use it as the variable to threshold to compute the P-R curves. To summarize the P-R curves in a single number, we use the average F_1-measure.

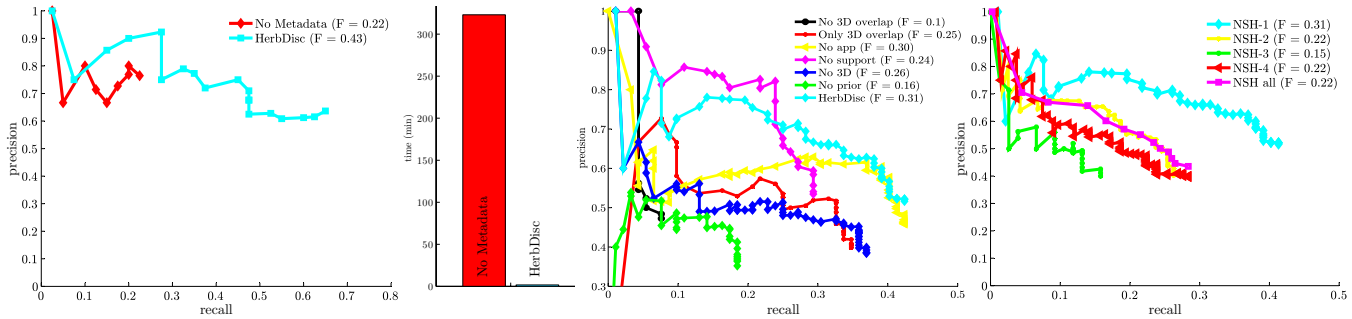


Fig. 5: (a) PR curve comparing Visual similarity vs. HerbDisc for the Kitchen Dataset. (b) Computation time of Visual Similarity vs. HerbDisc for the Kitchen Dataset. HerbDisc is 190× faster. (c) PR curve of the ablative analysis of HerbDisc for the NSH-1 Dataset, disabling one constraint at a time. (d) Results for global NSH Dataset and per floor (NSH-1 to NSH-4).

C. HerbDisc vs. Visual Similarity

Fig. 5(a) 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 to 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 to 24% maximum recall at 77% precision. For the same recall of 24%, HerbDisc achieves 90% precision (13% higher than Θ_{visual} alone). The additional constraints provided by the metadata (and especially Θ_{seq}) allow HerbDisc to process the Kitchen Dataset 190 times faster than if using visual similarity alone, as shown in Fig. 5(b). The main reason for this speedup is the limited number of pairwise visual similarities to evaluate in the CSG compared to the regular pairwise similarity graph from Θ_{visual}. Namely, HerbDisc evaluates 16271 pairwise visual similarities, compared to 1.6 millions in Θ_{visual}.

To illustrate the impact of different constraints on the CSG, we show in Fig. 4 the graphs (displayed as adjacency matrices) generated by each edge constraint for the Kitchen Dataset. Fig. 4(top) displays the CSG after each constraint as evaluated in HerbDisc, cascading the multiple conjunctive constraints for efficiency. The metadata-based constraints Θ_{seq}, Θ_{static} are significantly more discriminative than the visual features Θ_{3D} and Θ_{app}. Fig. 4(bottom) shows the result of using visual similarity constraints with no metadata. Using visual similarity alone, the product of all adjacency matrices is significantly denser (i.e., noisier), which accounts for the increased computation time shown in Fig. 5(b).

D. HerbDisc in the NSH Dataset

In this section, we evaluate the performance of HerbDisc on the NSH Dataset. We could not evaluate the visual similarity baseline Θ_{visual} on this dataset, because the testing machine had barely made any progress after a week of processing. HerbDisc processes the NSH Dataset in 18 min 30 s. This running time does not include data acquisition time (and motion filtering and candidate generation, which we execute in parallel with the data acquisition). Fig. 5(c) shows the PR curves for an ablative analysis of HerbDisc on NSH-1, in which we disable one constraint at a time and evaluate the resulting performance. Metadata constraints such as Θ_{static} show the largest impact on HerbDisc's performance, over visual similarity ones.

Evaluating HerbDisc in the entire NSH Dataset Fig. 5(d), we discover a total of 464 object models, where 121 unique objects are *correct* (28.6% recall) and 85 are *valid* (44.4% precision). We see a clear difference in performance as we move from regular office environments (NSH-1) to laboratories and machine shops. 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., Fig. 3(2, 3-5)), which include multiple metallic objects, specular reflections, and extreme clutter. We show examples of *correct*, *valid* and *invalid* objects in Fig. 6.

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 V, such as chairs, trashcans, or wall-mounted paper holders (see Fig. 6). We believe that the only way to disambiguate between valid and invalid objects is to interact with them during the discovery process, which is a future direction for us. Our framework can be used to leverage interaction information if available, as well as any other metadata, when formulated as constraints.

	Cand. purity	Group purity	3D purity	In ground truth
correct	80%	100%	80%	✓
valid	80%	100%	80%	✗
invalid		(not correct and not valid)		

TABLE III: Criteria for *correct*, *valid*, and *invalid* objects. We rely on an "oracle" evaluation [19], a human annotator who answers the question "Is M_k an object?" when faced with object model M_k. Examples of *valid* objects include those too big or too small to be grasped, immovable objects (e.g., attached to a wall), or parts of complex objects (e.g., a bicycle's seat).

VIII. Conclusions

We have introduced Lifelong Robotic Object Discovery, the problem of discovering new objects in the environment during an entire robot's lifetime. As a first step towards LROD, we have processed the raw video stream of an entire workday of a robotic agent.

Our key insight to making LROD feasible is to incorporate metadata. We have described a graph-based framework to integrate any source of metadata and similarity functions as constraints, and to combine multiple constraints using logic expressions. With constraints, we provide a common formulation to encode any source of information, both visual data and metadata, as metadata-augmented graphs—Constrained Similarity Graphs (CSGs)—for object discovery.

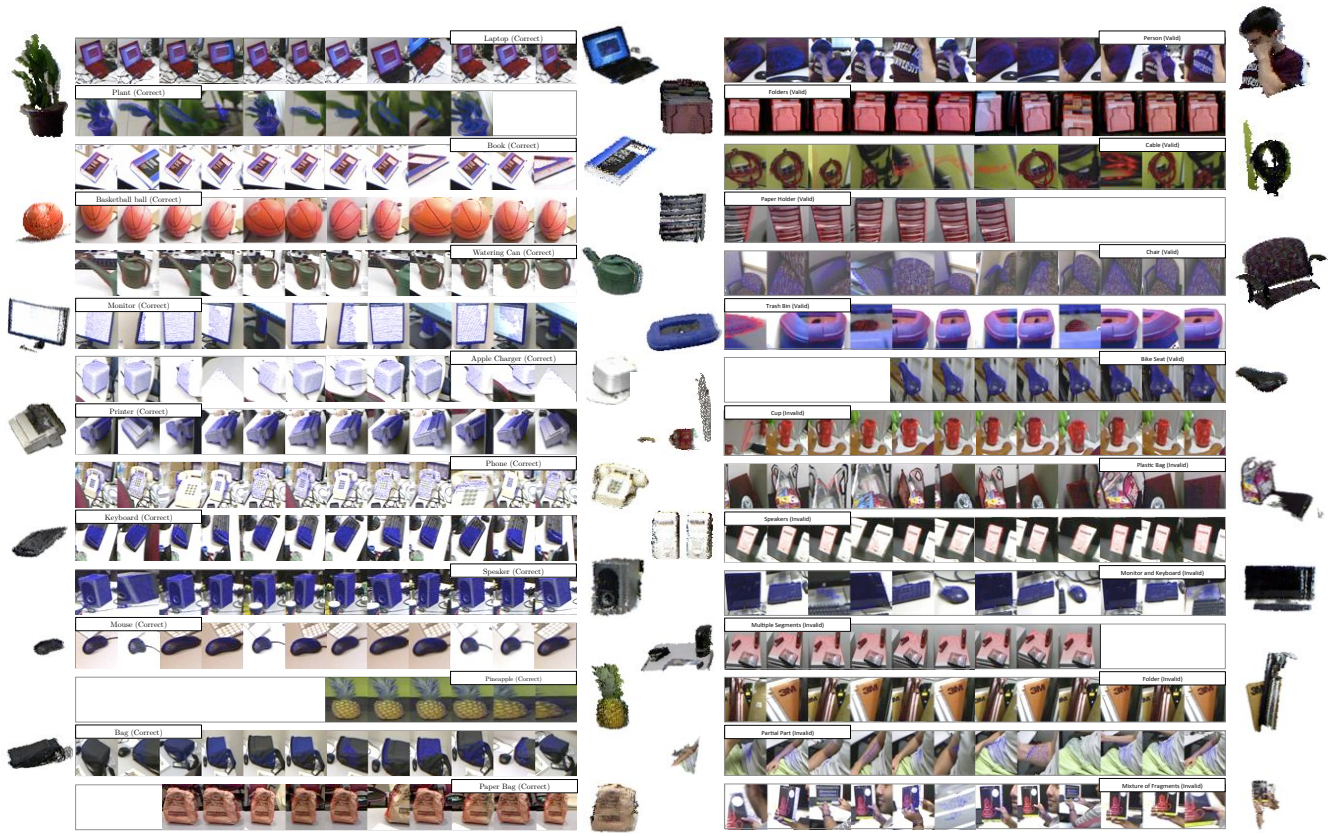


Fig. 6: Examples of *Correct* (left), *Valid* (right-top) and *Invalid* (right-bottom) objects. For each object, we display its object label (text box); its 3D model (left/right); and 10 randomly selected images from the object candidates h_i (center), with the 3D point clouds overlaid in red or blue over each image.

We have introduced HerbDisc, an optimized implementation of our framework which leverages metadata to efficiently discover objects in large datasets. To evaluate the performance of HerbDisc, we have gathered a dataset of 6 h 20 min of raw RGBD video of office and lab environments containing 423 ground truth objects. HerbDisc processed this dataset in under 19 minutes and discovered over a hundred 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). More importantly, we showed that our framework can opportunistically leverage different sources of information adaptively, as conditions change, which is a necessary feature to make LROD feasible in the long term.

See the supplementary material for a demo of HerbDisc discovering objects in a kitchen environment, which we can process in only a few seconds.

REFERENCES

[1] Z.-c. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, "General 3D Modelling of Novel Objects from a Single View," in *IROS*, 2010, pp. 3700–3705.

[2] M. Bjorkman and D. Kragic, "Active 3D scene segmentation and detection of unknown objects," in *ICRA*. IEEE, 2010, pp. 3114–3120.

[3] G. Kootstra and D. Kragic, "Fast and Bottom-Up Object Detection, Segmentation, and Evaluation using Gestalt Principles," in *ICRA*, 2011.

[4] A. K. Mishra and Y. Aloimonos, "Visual Segmentation of Simple Objects for Robots," in *Robotics: Science and Systems*, 2011.

[5] T. Morwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze, "BLORT - The Blocks World Robotic Vision Toolbox," in *ICRA Workshops*, 2010.

[6] E. Herbst, P. Henry, X. Ren, and D. Fox, "Toward Object Discovery and Modeling via 3-D Scene Comparison," in *ICRA*. IEEE, 2011.

[7] H. Kang, M. Hebert, and T. Kanade, "Discovering Object Instances from Scenes of Daily Living," in *ICCV*, 2011.

[8] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman, "Using Multiple Segmentations to Discover Objects and their Extent in Image Collections," in *CVPR*. IEEE, 2006.

[9] A. Collet, S. S. Srinivasa, and M. Hebert, "Structure Discovery in Multi-modal Data : a Region-based Approach," in *ICRA*, 2011.

[10] U. Brandes, "A Faster Algorithm for Betweenness Centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[11] D. Borrmann, J. Elseberg, K. Lingermann, A. Nuchter, and J. Hertzberg, "The Efficient Extension of Globally Consistent Scan Matching to 6 DOF," in *3DPVT*, 2008.

[12] J. Philbin, J. Sivic, and A. Zisserman, "Geometric Latent Dirichlet Allocation on a Matching Graph for Large-scale Image Datasets," *IJCV*, vol. 95, no. 2, pp. 138–153, Jun. 2010.

[13] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "LabelMe: A Database and Web-Based Tool for Image Annotation," *IJCV*, vol. 77, no. 1-3, pp. 157–173, Oct. 2008.

[14] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, S. Hodges, P. Kohli, J. Shotton, A. Davison, A. Fitzgibbon, and D. Freeman, "KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera," in *UIST*, 2011.

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

[16] J.-F. Lalonde, N. Vandapel, D. F. Huber, and M. Hebert, "Natural terrain classification using three-dimensional ladar data for ground robot mobility," *Journal of Field Robotics*, vol. 23, no. 10, pp. 839–861, Oct. 2006.

[17] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert, "Recovering Occlusion Boundaries from a Single Image," in *ICCV*, Oct. 2007.

[18] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," *ICRA*, pp. 3212–3217, May 2009.

[19] T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine, "Unsupervised Object Discovery: A Comparison," *IJCV*, vol. 88, no. 2, pp. 284–302, Jul. 2009.