Human-Guided Object Mapping for Task Transfer

TESCA FITZGERALD and ASHOK GOEL, Georgia Institute of Technology, USA
ANDREA THOMAZ, University of Texas at Austin, USA

When transferring a learned task to an environment containing new objects, a core problem is identifying the mapping between objects in the old and new environments. This object mapping is dependent on the task being performed and the roles objects play in that task. Prior work assumes (i) the robot has access to multiple new demonstrations of the task or (ii) the primary features for object mapping have been specified. We introduce an approach that is not constrained by either assumption but rather uses structured interaction with a human teacher to infer an object mapping for task transfer. We describe three experiments: an extensive evaluation of assisted object mapping in simulation, an interactive evaluation incorporating demonstration and assistance data from a user study involving 10 participants, and an offline evaluation of the robot's confidence during object mapping. Our results indicate that human-guided object mapping provided a balance between mapping performance and autonomy, resulting in (i) up to 2.25× as many correct object mappings as mapping without human interaction, and (ii) more efficient transfer than requiring the human teacher to re-demonstrate the task in the new environment, correctly inferring the object mapping across 93.3% of the tasks and requiring at most one interactive assist in the typical case.

CCS Concepts: • **Computing methodologies** → **Cognitive robotics**; **Spatial and physical reasoning**; *Reasoning about belief and knowledge*;

Additional Key Words and Phrases: Task transfer, object mapping

ACM Reference format:
Tesca Fitzgerald, Ashok Goel, and Andrea Thomaz. 2018. Human-Guided Object Mapping for Task Transfer. *ACM Trans. Hum.-Robot Interact.* 7, 2, Article 17 (October 2018), 24 pages.
https://doi.org/10.1145/3277905

1 INTRODUCTION

Our work is aimed at enabling robots to flexibly adapt skills for efficient deployment in dynamic human environments. While there are a number of methods to allow a robot to learn a new task through interactive demonstrations (Akgun et al. 2012; Argall et al. 2009; Chernova and Thomaz 2014), transferring the learned tasks to new environments remains a challenge due to the many possible differences between the original environment where the robot learned the task (referred to as the *source* environment) and the new environment (referred to as the *target* environment)

This material is based on work supported by the NSF Graduate Research Fellowship under Grant No. DGE-1148903, ONR grant N000141410120, and the IBM PhD Fellowship.
Authors' addresses: T. Fitzgerald and A. Goel, Georgia Institute of Technology, 801 Atlantic Dr. NW, Atlanta, GA, 30332; emails: {tesca.fitzgerald, goel}@cc.gatech.edu; A. Thomaz, University of Texas at Austin, 2501 Speedway, Austin, TX, 78712; email: athomaz@ece.utexas.edu.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
2018 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
2573-9522/2018/10-ART17
https://doi.org/10.1145/3277905

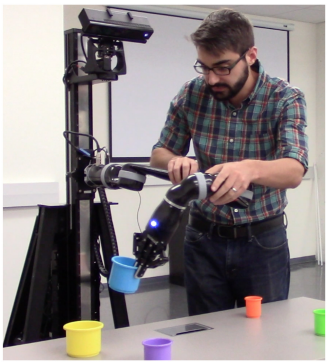


Fig. 1. Interactive Task Demonstration.

(Taylor and Stone 2009). To succeed in target environments containing new objects, the robot must identify correspondences between the new objects and those in the original environment, a problem known as *object mapping*. This is a complex problem, however; given n objects in each environment, there are $n!$ possible mappings to consider, in principle. In this article, we focus on this object mapping problem. We assume that a task model was previously learned (by demonstration or otherwise) and do not address the problem of goal learning or the entire process of task transfer at this time. Rather, by evaluating object mapping independently of any specific task learning algorithm, we demonstrate that it would be compatible with a variety of learning algorithms, provided that they produce a list of object-directed steps for each task.

While object mapping has been addressed in related work (see Section 2), it is typically assumed that the robot can gain additional experience in the target environment or knows which object feature(s) to use when evaluating object correspondences. However, when the robot learns a task from human interaction (such as that shown in Figure 1), such assumptions lead to two challenges. First, gaining additional experience in the target environment can be a time-consuming task in which the human teacher must continue to provide task demonstrations or feedback to the robot as it explores the environment. Simply helping the robot gain the correct mapping would more efficiently utilize the teacher's time.

As an example, an assistive robot deployed in a home setting will encounter many opportunities for object mapping. Routine tasks can be performed with multiple object types (e.g., a task model for *stacking* can be reused to stack cups or books), as well as various instances of a single object class (e.g., a single home may contain multiple bowls that vary in their dimensions, shapes, and other visual features). Rather than re-learn the task for each object variation, learning the correspondences between objects would enable the robot to reuse its previously learned task models. An additional use case for object mapping is object replacement; if a robot knows how to hammer a peg, but does not have a hammer available to it, then the robot should ideally use a suitable replacement (e.g., a similarly-sized rock or paperweight) to repeat the task rather than needing to re-learn the task for the new object.

A second challenge of object mapping is that the robot does not have the same contextual knowledge that the human teacher has about the task and thus does not know which object features are relevant to successfully repeating the task. For example, in transferring a *cup-stacking* task, the robot would need to identify which cups in the source and target environments are similar according to their size feature. In another task in which the robot learns to *sort* the same cups by their color, the robot would need to instead map objects according to their *hue* feature. While the human teacher knows which features are relevant in the context of that task, the robot does not.

1.1 Outline

In this article, we describe a human-guided approach to task-dependent object mapping (*"situated mapping"*) problems. In Section 3, we define the situated mapping problem. Since the human teacher knows the roles of objects in the task, we posit that human teachers can readily provide assistance in this regard. We describe *Mapping by Demonstration* (MbD), our interactive approach to situated mapping problems, in Section 4.

We then present the results of three experiments. In our first experiment, first described in Fitzgerald et al. (2016), we perform an extensive evaluation of MbD in *simulated* domains, applying this approach to variations of tasks containing five, six, or seven objects. We then follow-up with a case study demonstrating the approach's applicability to real-world objects in realistic tasks. In both the simulated and real-world tasks in this experiment, mapping assistance is provided according to the ground-truth object mapping, and thus there is no opportunity for error in the mapping assistance provided to the algorithm.

In our second experiment, we evaluate the MbD algorithm's effectiveness in the *interactive* use case. We conduct a user study recording how a human teacher assists the robot in learning and transferring three ordered, pick-and-place tasks. We conclude from this experiment that structured interaction with the human teacher during task mapping is effective in both (i) reducing the number of interactions needed to repeat the task to less than 50% of those needed to re-learn the task and (ii) increasing mapping accuracy and confidence in comparison to predicting an object mapping without human interaction.

The final experiment we describe is an offline evaluation of our method for confidence-based interaction, addressing the errors that arise from mapping assistance in the interactive use case (and that are not present in the first, simulated experiment). In this approach, we aim to minimize interaction errors by evaluating the amount of assistance needed to infer the object mapping. From this experiment, we conclude that a robot can use a confidence threshold to moderate the number of assistance requests it makes, *balancing* autonomy and interaction to infer an object mapping.

1.2 Contributions

In this article, we describe the MbD algorithm first presented in our earlier work (Fitzgerald et al. 2016), and expand on it by making the following contributions:

(1) The first adaptation of the MbD algorithm (Fitzgerald et al. 2016) for an interactive robot
(2) A user study evaluating the effectiveness of interactive assistance for object mapping
(3) A confidence-based approach to maximizing autonomy during object mapping and minimizing the effect of interaction errors

2 RELATED WORKS

2.1 Generalization in Learning from Demonstration

Learning from Demonstration enables a robot to quickly learn skills from a human teacher (Akgun et al. 2012; Argall et al. 2009; Chernova and Thomaz 2014). Pastor et al. (2009) demonstrates how a robot can generalize Dynamic Movement Primitives (DMPs) (Schaal 2006) for environments containing different object configurations. While this approach does not directly generalize to environments containing different objects, it provides a useful application for object mapping; (Fitzgerald et al. 2015) describes how an object mapping could be used to identify DMP parameters in a target environment. A related problem is grounding a task representation (e.g., "task recipes" (Misra et al. 2016; Waibel et al. 2011)) in an observed set of objects. Bullard et al. (2016) describes a method for grounding objects and semantic locations in perception using demonstrations. Tenorth et al. (2010) grounds task recipes in perception by identifying the object that best matches the features

prototypical for that task step's object designator. Kulick et al. (2013) uses an active learning approach to train a classifiers for task symbol grounding. However, symbol grounding differs from object mapping in that it grounds abstract object references in specific object perception (whereas mapping directly correlates two object instances).

2.2 Cognitive Systems Approaches to Object Mapping

Research on cognitive systems has long addressed object mapping, particularly by identifying structural similarity between source and target objects; that is, objects sharing similar relations to their surroundings (Gentner 1983; Gentner and Markman 2006; Gick and Holyoak 1983). In the Structure Mapping Engine, the structure of a scene is represented as a graph; objects sharing similar structural relations within their own graphs are mapped (Falkenhainer et al. 1989). Liu and Stone (2006) adapt the Structure Mapping Engine to transfer strategies for RoboCup simulated soccer to map state variables and actions based on their relational structure (specified using qualitative Dynamic Bayes Networks). Holyoak and Thagard (1989) describes a constraint-satisfaction method for object mapping, comparing objects according to their pragmatic, spatial, and semantic features. Davies et al. (2008) describes a method for visual analogical reasoning in which transfer and mapping are constrained by each other; the objects utilized in the source solution inform which mapping constraints to enforce. While these approaches have been demonstrated in simulated domains, they have not been applied to identify analogies between physical objects with a rich set of perceptual features.

2.3 Machine Learning Approaches to Object Mapping

The goal of task transfer in reinforcement learning domains is often to use a source domain to bootstrap policy learning in the target domain (Taylor and Stone 2009). Taylor et al. (2007) has the robot explore its state space in the target domain to derive an inter-task mapping between source and target state variables, using it to transfer its learned policy. While effective, this approach requires significant additional experience in the target domain, which our approach aims to minimize. Object-oriented MDPs expressly encode the relation between objects as a part of its state space representation (Diuk et al. 2008); however, this approach assumes that objects within the same classification play the same role within the task. Chernova and Veloso (2007, 2009) introduces confidence-based autonomy for policy reuse, in which the robot assesses its confidence in applying its current policy to a new state; if its confidence is below a threshold, it requests additional demonstrations from the teacher, otherwise acting autonomously from its current policy.

Lee et al. (2015) address mapping physical objects by their perceived point clouds. Their method infers a warping function that transforms the source object point cloud such that it closely matches that of the target object. However, using this warping function as a measure of similarity limits object mapping to those with similar shape. Huang et al. (2015) address a similar problem of aligning object point clouds, but prioritize alignment of certain labeled features of the objects. These approaches are effective in identifying similar objects, but assume that (i) similar objects play the same role in the source and target environments and (ii) corresponding objects are comprised of the same parts or shapes. These assumptions do not hold when the agent does not know *a priori* which features to use in object mapping.

Overall, current approaches to object mapping assume (i) objects with the same classifications play the same role in any task, (ii) the robot knows *a priori* which object features to use for mapping, (iii) the robot may continue to explore/train in the target environment, and/or (iv) an abstraction of the object can be used to identify specific instances of that object symbol. However, these assumptions do not hold in situated mapping problems, where we would like the robot

to receive very limited new data/demonstrations of a task and then identify an object mapping without knowledge of the specific object features relevant to that task.

We adapt the MbD approach to situated mapping, in which an agent uses *mapping assistance* (in the form of a limited number of object correspondences) to infer the remainder of the mapping (Fitzgerald et al. 2016). However, Fitzgerald et al. (2016) does not apply or evaluate this approach on a physical robot. More importantly, it does not address how mapping assistance would be obtained by the agent. We present the first HRI study that uses the MbD algorithm, and in doing so, expand on this work by (i) evaluating a mode of interaction enabling the robot to receive mapping assistance in the target domain and (ii) demonstrating human-guided object mapping on a physical robot.

3 PROBLEM: TASK-DEPENDENT OBJECT MAPPING

In situated mapping, the robot must identify the mapping that maximizes similarity between source environment objects (S) and their equivalent objects in the target environment (T), as follows (Fitzgerald et al. 2016):

$$map(S,T) = \underset{m}{\operatorname{argmax}} \sum_{i=0}^{|S|} \delta(s_i, m(s_i,T)), \tag{1}$$

where $m(s_i,T)$ returns the object in T which corresponds to object s_i according to the mapping m. This strategy is dependent on a similarity metric $\delta(a,b)$ that returns a similarity score for objects a and b. However, defining this similarity metric (and the object features it considers relevant) is nontrivial. The similarity metric that is appropriate for one task may not represent the object features relevant to another task. Consider our previous example, with cups the robot learns to use in both a stacking task and a sort-by-color task. In the first task, the robot should identify an object mapping maximizing the similarity between mapped objects based on their *size* feature, whereas the second task requires objects to be mapped according to their *hue* feature.

We address this problem of *situated* (context-dependent) mapping, in which object mapping is dependent on the task being performed. The situated mapping problem has the following stages of interaction with a human teacher:

(1) The human teacher demonstrates a task in the source environment.
(2) The robot observes the object features and task steps involving those objects.
(3) The robot observes the target environment containing new objects, and is asked by the human teacher to repeat the newly learned task.
(4) The robot infers the mapping between objects in the source and target environments.
(5) The robot uses this mapping to transfer the learned task for execution in the target environment (Fitzgerald and Goel 2015).

This article is concerned with how additional interaction between the robot and the human teacher can facilitate step 4 (inferring the object mapping). We currently focus on object mapping for ordered tasks where the source and target contain the same number of objects (thus requiring an *n-to-n* mapping). This lays the groundwork to address other variations of the object mapping problem in future work, including *m-to-n* mapping and partial-ordered tasks.

4 APPROACH

Since the human teacher is aware of both (i) the goal of the task and (ii) the role that each object plays in achieving that goal, we propose a *human-guided object mapping* method (Figure 2), consisting of two interaction phases and two mapping phases. The interaction phases include a

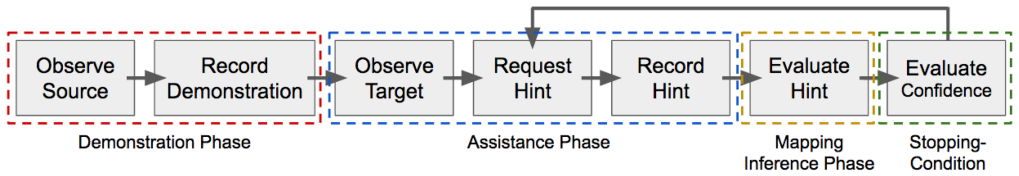


Fig. 2. Human-guided Mapping Process.

Table 1. Source of Each Object Feature's Value

Perceived Features	Derived Features	Knowledge-base Features
Location	Spatial relations	Affordances Properties
Hue	Hue-shift	
Size	Size-shift	

demonstration phase in which the robot learns the task steps from demonstration, and an **assistance** phase in which the robot records interactive assistance from the teacher. The assistance is used in the two mapping phases, first in a **mapping inference** phase, and then in the **confidence evaluation** phase that determines whether additional assistance should be requested. We now describe all four phases, later evaluating phases in isolation via simulated, interactive, and offline evaluations.

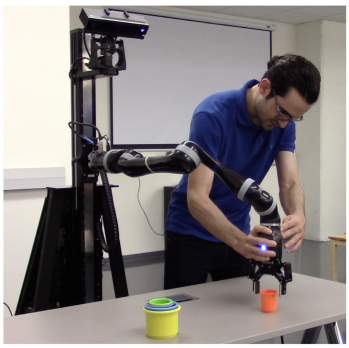
4.1 Demonstration Phase

At the start of the interaction, the robot observes the source environment using an RGBD sensor, and segments objects from the tabletop using the algorithm described in Trevor et al. (2013). After extracting the location, size, and hue features of each object, it derives the set of spatial relations between each object, where the spatial relations between two objects X and Y is: X above Y, X below Y, X left-of Y, and/or X right-of Y. Object size and color are used to look up *affordances* (the actions enabled by that object, e.g., *openable*, *pourable*) and *properties* (variables associated with an object's affordances, e.g., an *openable* object has the property *open* or *closed*). The robot uses a manually defined lookup table as a stand-in for a more complex process to derive this information from visual data.

In sum, these feature values are obtained from the sources listed in Table 1 and comprise the object representation $<x, y, z, c, d, s, a, p>$:

- x, y, z is the centroid location
- c is the average hue of the object's color, ranging from 0 to 360
- d is the bounding box volume (derived from object dimensions)
- $s = \{s_0, s_1, \ldots\}$ is the set of spatial relations between the object and all other objects, where each element $s_k \in$ {LEFT-OF, RIGHT-OF, ABOVE, BELOW}
- $a = \{a_0, a_1, \ldots\}$ is the set of that object's affordances
- $p = \{p_0, p_1, \ldots\}$ is the set of property values associated with that object and its affordances

After the robot observes objects in its environment, the human teacher interacts with the robot's arm to physically guide it through executing the task (e.g., Figure 3(a)). The robot records the trajectory of its end effector position in cartesian space, and then segments it into pick or place *task steps* according to the open/close actions of its gripper. The robot then identifies the object that was closest to the gripper at the end of each task step, recording it as the *primary object* for that task



(a) Demonstration Phase

(b) Mapping: Handoff

(c) Mapping: Indicate goal

Fig. 3. Demonstration and Mapping Interactions.

step. Following the demonstration, the task is represented as (i) the list of object representations and (ii) a list of task steps that indicate the primary object for each step.

To evaluate our algorithm in the *worst-case scenario* where the task steps contain all object features, we do not assume that the robot has learned which object features are salient nor a model of the goal of the task but rather include all object features in every task step. This ensures that our algorithm can complement a variety of learning algorithms that produce a list of object-directed task steps. Should the task learning algorithm also prune the list of candidate object features that are relevant for object mapping, we expect our method to demonstrate even better performance.

4.2 Assistance Phase

After the task demonstration, the robot may receive assistance from the human teacher to repeat the task in the target environment, later using this assistance during the mapping inference phase. Having the teacher provide assistance via natural interaction (e.g., pointing at or picking up an object) mitigates the need for human teachers to have knowledge of which features the robot has the capacity to observe (e.g., the robot can record object color, but not product brand), or how to express feature values (e.g., hue values).

Once the robot requests assistance (e.g., "Where do I go next?"), the human teacher provides a mapping assist by handing the robot the next object it should use to complete the task in the target environment (e.g., Figure 3(b)) or, if the robot is already holding an object, by pointing to where the robot should place the object in the target environment (e.g., Figure 3(c)). Each mapping assist indicates a correspondence between (i) the object referenced by the teacher in the target environment and (ii) the object that *would* have been used in the *next* step in the original task demonstration.

4.3 Mapping Inference Phase

Two goals must be addressed simultaneously to infer the object mapping: selection of (i) an object similarity function (and associated object feature) which is representative of the mapping assistance received thus far and (ii) an object mapping that maximizes object similarity according to the selected similarity function. As such, Algorithm 1 maintains both a (i) feature set space F containing all feature sets under consideration for the similarity metric and (ii) a mapping hypothesis space M containing all mapping hypotheses still under consideration. In Sections 4.3.1–4.3.4, we now summarize the MbD algorithm described in Fitzgerald et al. (2016).

ALGORITHM 1: Human-guided Mapping Algorithm

```
1:  function MappingInteraction(S)
2:      T ← observeEnvironment()
3:      M ← initializeHypothesisSpace(S, T)
4:      F ← initializeFeatureSpace()
5:      while target task is incomplete do
6:          h ← mapping assistance
7:          (M, F, p, c) ←InferMapping(M, F, S, T, h)
8:  function InferMapping(M, F, S, T, h)
9:      M ← pruneMappingHypotheses(M, h.src, h.tgt)
10:     F ← pruneFeatureSpace(F, S − {h.src})
11:     E ← evaluateHypotheses(M, F, S, T)
12:     p ← maximizeMapping(M, F, E)
13:     c ← evaluateConfidence(p)
14:     return (M, F, p, c)
```

4.3.1 Initialization. The mapping hypothesis space M is initialized as the set of all possible object mappings, and thus begins as a $n!$-sized set for an n-to-n mapping problem (see Algorithm 1, line 3). The feature set space is initialized as the set of object features from Section 4.1, plus two features derived from mapping assistance: size shift and hue shift (Algorithm 1, line 4). *dh* is the average *size* shift indicated over all assistance thus far. This feature is useful for tasks in which object size is relevant to the task, but the source and target objects are at a different scale (e.g., target objects are a scaled version of source objects). *ch* is the average *hue* shift indicated over all assists. Similarly, this feature is useful for tasks in which object hue is relevant to the task, but differs between the two environments (e.g., blue objects in the source environment correspond to purple objects in the target). After the robot has received its first mapping assist, *dh* and *ch* are initialized to the size and hue differences, respectively, between the two objects indicated in the mapping assist, and are updated after additional assists.

In the extensive simulated experiment (Experiment 1 described in Section 5), the feature set space is initialized as the power set of all object features, resulting in the initial feature set space including all 127 combinations of features as described in the original MbD implementation in Fitzgerald et al. (2016). We simplify this feature set space in our interactive implementation of MbD (used in Experiments 2 and 3), such that each feature set consists of a single feature. This results in an initial feature set space F consisting of only 7 feature sets, each containing one of the elements of the feature vector $< c, ch, d, dh, s, a, p >$ consisting of hue, hue shift, size, size shift, spatial relation, affordance, and property features. A n x n x 7 evaluation matrix E is generated during initialization, containing the evaluation score for every possible object correspondence according to each of the seven object features:

$$E_{ij} = \left\langle \Delta\mathbf{c}_{i,j}, \Delta\mathbf{ch}_{i,j}, \Delta\mathbf{d}_{i,j}, \Delta\mathbf{dh}_{i,j}, \Delta\mathbf{s}_{i,j}, \Delta\mathbf{a}_{i,j}, \Delta\mathbf{p}_{i,j} \right\rangle,$$

which evaluates the correspondence between objects at indices i and j according to each evaluation metric denoted as Δx based on a single object feature x. Each evaluation metric is based on a generic distance function defined in Equation (2) as the difference between two objects' value for that feature, measured along a Gaussian curve:

$$\mathrm{D}(v_1, v_2, r) = \frac{\mathcal{N}(v_2|v_1, 1\sigma r)}{\mathcal{N}(v_1|v_1, 1\sigma r)}. \tag{2}$$

Since we cannot weight evaluation metrics based on a feature's prevalence in previous instances of that task, normalization becomes a challenge. To address this, the evaluation metric for each feature is scaled based on that feature's value range, as follows:

- *Hue similarity:* $\Delta\mathbf{c}_{i,j} = D(0, \tan^{-1}(\frac{\sin(c_i - c_j)}{\cos(c_i - c_j)}), 360)$
- *Hue-shift similarity:* $\Delta\mathbf{ch}_{i,j} = D(c_j, c_i + \mu_c, 360)$
- *Size similarity:* $\Delta\mathbf{d}_{i,j} = D(1, \frac{d_j}{d_i}, d_{max})$
- *Size-shift similarity:* $\Delta\mathbf{dh}_{i,j} = D(\mu_d, \frac{d_j}{d_i}, 1)$
- *Spatial similarity:* $\Delta\mathbf{s}_{i,j} = D(|s_i|, |s_i \cap s_j|, |s_i|)$
- *Affordance similarity:* $\Delta\mathbf{a}_{i,j} = D(|a_i|, |a_i \cap a_j|, |a_i|)$
- *Property similarity:* $\Delta\mathbf{p}_{i,j} = D(|p_i|, |p_i \cap p_j|, |p_i|)$,

where d_{max} is the ratio between the largest and smallest source objects' sizes and μ_d and μ_c are the average differences in size ratio and hue, respectively, between object pairs in previous mapping assists. Note that this is clearly not an exhaustive list of features that may be relevant to a task; many tasks may require additional features that have not been addressed here. However, this method is intended to consider a variety of features for object mapping and is still applicable in such tasks. Additional object features can be incorporated by defining a new evaluation metric and expanding the evaluation matrix E to include the new metric.

After initialization, each mapping assist is used to (i) prune the mapping hypothesis space M, (ii) prune the feature set space F, and then (iii) select the highest-ranked mapping hypothesis $m \in M$ as the predicted mapping. We now describe each step in further detail.

4.3.2 Pruning Step. A mapping assist consists of an object in the source environment and its corresponding object in the target environment. The pruning step (Algorithm 1, line 9) ensures that each remaining object mapping in the hypothesis space M contains this correspondence. A mapping hypothesis is represented as the n x n binary matrix m, where each element $m_{ij} = 1$ if $S_i \mapsto T_j$, and $m_{ij} = 0$ otherwise. The pruned mapping hypothesis space only contains mappings $m \in M$ with $m_{ij} = 1$, where i and j are the corresponding object indices in the source (S) and target (T) environments, respectively. For example, if a mapping assist corresponds source object 1 with target object 5, the mapping hypothesis space is pruned such that all remaining mappings contain $m_{1,5} = 1$. Additionally, the feature set space F is pruned after each mapping assist (Algorithm 1, line 10). A feature set is removed if it has no variance over the remaining, unmapped objects in the source environment. For example, if all unmapped objects share the same affordances, then that feature is no longer discriminating, and is irrelevant to the object mapping.

4.3.3 Hypothesis Evaluation Step. The evaluation matrix only needs to be generated once (during initialization). Afterward, the evaluation matrix is referenced (see Algorithm 1, line 11) to evaluate a mapping hypothesis m according to a feature set f, as follows:

$$\mathrm{V}_{m,f} = \sum_{f_k \in f} \mathrm{sum}(m \circ E^{f_k}). \tag{3}$$

E^{f_k} is the n x n matrix containing the evaluation for every possible object correspondence according to feature f_k, such that every element $E^{f_k}_{ij} = E_{i,j,f_k}$. The function $m \circ E^{f_k}$ returns the entry-wise product of the object mapping m (represented as a n x n binary matrix) and the evaluation matrix based on feature f_k. This results in a matrix containing the evaluation of each object correspondence in mapping m, according to the feature f_k.

4.3.4 Mapping Maximization Step. Each combination of a mapping hypothesis $m \in M$ and feature set $f \in F$ is then evaluated:

$$m_0^* = \underset{m \in M}{\operatorname{argmax}} \left(\max_{f \in F} \left(\frac{1}{n} \mathrm{V}_{m,f} \right) \right), \tag{4}$$

where n is the number of source objects and $V_{m,f}$ is Equation (3). The highest-ranked mapping (m_0^*) and feature set combination is then returned as the predicted mapping (Algorithm 1, line 12).

4.4 Confidence Evaluation Phase

A single mapping assist may not provide enough information to infer the *correct* object mapping. Thus, after receiving a mapping assist and inferring the object mapping, a decision must be made to either request additional assistance or to complete the rest of the task autonomously using the most-recently inferred object mapping. Similar to the confidence-based autonomy (CBA) approach introduced by Chernova and Veloso (2007, 2009), our work aims to enable the robot to rely on confidence as a means to managing its assistance from the human teacher. However, since the robot does not know which features are relevant to the task, it is unable to select features to calculate its confidence in the same manner as CBA. We propose a variation of confident execution, utilizing two sources of information available during interactive object-mapping: (1) interaction from the teacher and (2) the resulting mapping hypothesis evaluations.

While the MbD algorithm only returns a mapping prediction after each assist, we propose that the evaluation matrix E can be used to determine the confidence of that predicted mapping (Algorithm 1, line 13). We now calculate confidence based on two feature sets:

- α is the feature set leading to the highest mapping evaluation:

$$\alpha = \underset{f \in F}{\operatorname{argmax}} \max_{m_0^* \in M} \mathrm{V}_{m_0^*, f} \tag{5}$$

  The resulting mapping is denoted as m_0^*.
- β is the feature set leading to the second-highest mapping evaluation such that $\alpha \neq \beta$ and the resulting mapping $m_1^* \neq m_0^*$:

$$\beta = \underset{f \in F \setminus \{\alpha\}}{\operatorname{argmax}} \left(\mathrm{eqv}(m_1^*) \cdot max_{m_1^* \in M} \mathrm{V}_{m_1^*, f} \right), \tag{6}$$

$$\mathrm{eqv}(m) = \begin{cases} 1 & \text{if } m \neq m_0^* \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

  This resulting, second-highest mapping is denoted as m_1^*.

As more assistance is received, it should more clearly support one mapping hypothesis over the rest. If it does not, then the assistance provided so far supports multiple mapping hypotheses, and additional assistance is necessary to arbitrate between the top hypotheses. Thus, we use decision margin as a proxy for confidence; the more separated the top-ranked mapping evaluation is from the remaining mapping hypotheses' evaluations, the more confidently it can be selected as the correct mapping. We define **confidence** as the decision margin between these two top-ranked mapping hypotheses' evaluations: $c = \mathrm{V}_{m_0^*, \alpha} - \mathrm{V}_{m_1^*, \beta}$.

5 EXPERIMENT 1: SIMULATED EVALUATION

We first perform an extensive evaluation of the *mapping inference* phase (Section 4.3) in simulation. Rather than obtain mapping assistance from interaction with a teacher, the system receives assistance for each task step based on the ground-truth mapping.

We evaluated the system with three categories of simulated tasks: containing $n = 5$, $n = 6$, or $n = 7$ objects in the source and target environments. These categories represent incrementally more difficult problems; as the number of objects increases, the mapping hypothesis space from which the system must choose a single mapping increases factorially. For each category of problem, 10 mapping problems (each representing a task and consisting of a source and target environment) were generated randomly, such that each object's perceived and knowledge-base features had random value assignments, with derived features automatically generated from perceived features. To simulate how some objects may be present in both the source and target environments (as in a realistic mapping problem), each source object had a 50% likelihood of being reused in the target environment, with the remaining objects being randomly generated. Reused objects retained intrinsic feature values (size, color, and affordances), but were given a randomly assigned location and properties, since the values of these features can be changed without replacing the object (e.g., moving a cup to a new location or emptying a cup so that its "is_filled" property changes).

Finally, ground-truth mappings for each task were generated corresponding to feature sets consisting of one or two features (except for redundant feature sets {size, size-change} and {hue, hue-change}), resulting in a set of 26 possible ground-truth mappings. The source and target environments had the same number of objects, so a bijective mapping was generated. This resulted in a total of 780 evaluations (3 categories $\times$ 10 tasks $\times$ 26 ground-truth mappings). Note that this does not result in 780 *unique* ground-truth mappings, since two feature sets may result in the same ground-truth mapping.

Mapping assistance was also generated for each mapping problem instance. In a realistic mapping problem, one assist will be provided for each step of the task as described in Section 4.2. As a result, the assistance ordering will be dictated by the order in which objects are used in the task plan. Since the task plan is undefined in our simulated tasks, we evaluated the MbD algorithm using every possible assistance ordering to observe the impact of this ordering on mapping performance. This results in evaluating over a permutation of ${}^{n}P_{n-2}$ potential assistance orderings. The ordering of the last two assists does not matter, since assist $n - 1$ leaves only one object remaining, resulting in the complete ground truth after $n - 1$ assists.

We ran the simulated evaluation as follows:

(1) Problem instances, ground truths, and assistance orderings are generated *a priori*.
(2) For each problem instance, the system iteratively retrieves the next assist to be provided by the teacher to the object mapping algorithm and checks its predicted solution.
(3) If the predicted mapping is correct, then the system halts and records the number of assists needed to get the correct solution using this assistance ordering. If the prediction is incorrect or if multiple predictions are returned, then the system repeats the process by retrieving the next assist.

Results. For each class of n-object problems, we collected data on the algorithm's performance over all possible assistance orderings, where we measure performance as the number of correct mappings after each assist is provided. For all of these problems, the full ground-truth mapping is provided at $n - 1$ assists, since only one source and target object remains to be mapped after assist $n - 1$. Figures 4–6 compare the expected performance of the MbD algorithm over all assistance orderings, with error bars denoting one standard deviation, to two baselines, (i) expected performance when selecting a random mapping without utilizing mapping assistance and (ii) expected performance when using mapping assistance to *only* prune the hypothesis space (described in the Pruning step in Section 4.3.2), and then choosing a random mapping from the remaining hypothesis space (rather than using the assistance to infer features on which mapping is based).

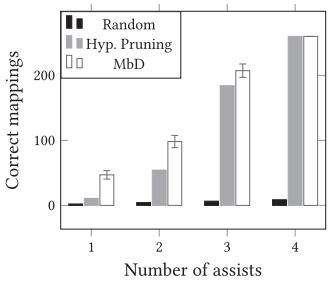


Fig. 4. Performance in 5-Object Tasks.

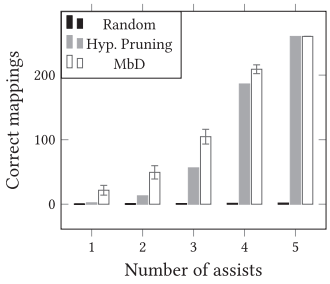


Fig. 5. Performance in 6-Object Tasks.

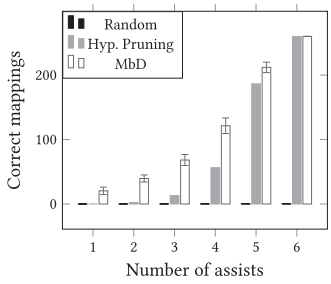


Fig. 6. Performance in 7-Object Tasks.

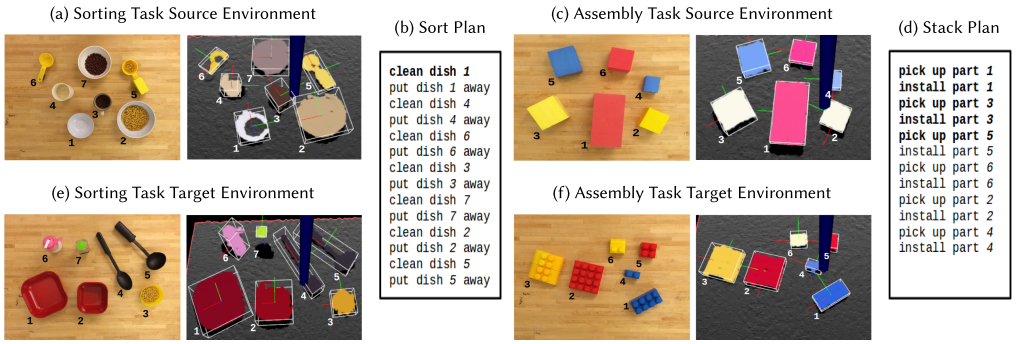


Fig. 7. Sorting and Assembly Task Environments

5.1 Real-World Case Studies

We have provided simulation results as a systematic analysis of the approach. The following real-world examples provide case studies demonstrating example tasks for which situated mapping problems exist and how they can be addressed using the MbD approach. As an initial evaluation of the suitability of the MbD algorithm for a robot learner, we tested it on two physical tasks: a dish sorting task (shown in Figure 7(a)) and a stacking assembly task (shown in Figure 7(c)). The robot passively observed its environment, and did not record or execute any actions. Similar to the simulated evaluation, we ran the real-world evaluation as follows:

(1) The mapping ground truth(s), task plan, and object affordances/properties are defined *a priori*. In the sorting task, there are several correct object mappings, whereas the assembly task has a single correct object mapping.
(2) We set up the source environment and recorded the robot's observation of the scene, repeating this step for the target environment.
(3) The algorithm accepts the assist corresponding to the next object used in the task plan and predicts a mapping solution.
(4) If correct, then the algorithm halts and records the number of assists needed to get a correct solution using this assistance ordering. If the prediction is incorrect, then the algorithm repeats the previous step and accepts the next mapping assist.

Rather than generate feature values as in the simulated evaluation, in this evaluation all objects' perceived features listed in Table 1 are observed from an RGBD sensor above the environment. We incorporate perception to provide an example of how our mapping strategy may be applied to

Table 2. Provided Object Knowledge Base

Object	Affordances	Properties
Cups	Fillable, Pourable	Empty, Full, Upright
Bowls	Fillable, Stackable	Empty, Full, Upright
Utensils	Scoopable, Pourable	Empty, Full, Upright
Blocks	Stackable	N/A

Table 3. Predicted Mappings after Each Assist

Task	Assist	Predicted Mapping
Sorting	$1 \mapsto 3$	$1 \mapsto 3, 2 \mapsto 1, 3 \mapsto 6, 4 \mapsto 7, 5 \mapsto 5, 6 \mapsto 4, 7 \mapsto 2$
Assembly	$1 \mapsto 2$	$1 \mapsto 2, 2 \mapsto 6, 3 \mapsto 3, 4 \mapsto 1, 5 \mapsto 4, 6 \mapsto 5$
	$3 \mapsto 3$	$1 \mapsto 2, 2 \mapsto 6, 3 \mapsto 3, 4 \mapsto 1, 5 \mapsto 4, 6 \mapsto 5$
	$\mathbf{5 \mapsto 1}$	$1 \mapsto 2, 2 \mapsto 6, 3 \mapsto 3, 4 \mapsto 4, 5 \mapsto 1, 6 \mapsto 5$

real-world problems, but do not consider the perception aspect itself to be a contribution of our work. Our main focus is on the underlying mapping strategy, and thus we indicate the perceptual features that our mapping algorithm uses and identify sources from sensors and a knowledge base.

Objects are perceived by abstracting a set of segmented objects from the point cloud using the algorithm described by Trevor et al. (2013). A bounding box is fitted to each segmented object to approximate its centroid $\langle x, y, z \rangle$ and dimensions $\langle width, depth, height \rangle$ as shown in Figure 7(a). The object's overall hue is derived from average hue of each pixel located at the surface of the object. Once perceived features are obtained, object locations and dimensions are used to derive a set of spatial relations between objects. Finally, size and color are used as a heuristic to assign an ID to each object, which is then used to retrieve its knowledge-base features: the affordances and properties associated with that object. Currently, we manually provide the affordances and properties associated with each object ID. In future work, we plan for the robot to autonomously retrieve this knowledge using an object classifier.

5.1.1 Cleaning and Sorting Task. In the first task, each environment consists of two utensils, two cups, and three bowls, which are to be cleaned and sorted separately based on their object type (indicated by their affordances). The source and target environments are shown in Figure 7(a) and (e), respectively. Table 2 lists the affordance and property values provided for these objects. The remaining features listed in Table 1 are derived from perception.

In contrast to simulated evaluations, there are several correct mappings for this task, since any bowl in the source environment can be mapped to any target bowl, either source utensil can be mapped to either target utensil, and either source cup can be mapped to either target cup. Whereas the task plan was undefined in the simulated evaluations, and thus the algorithm was evaluated over every assistance ordering, we use the task plan shown in Figure 7(b) to define the assistance ordering. Assistance was provided in the order in which objects would be used to complete the sorting task in the source environment: starting with the object closest to the robot's left hand and continuing in order of increasing distance from the robot's hand.

The algorithm returned eight mappings (all of which were correct for the task) after the first assist: the correspondence between the small white bowl and small yellow bowl. This assist, and one of the returned mappings, is listed in Table 3.

5.1.2 Assembly Task. In the second task, each environment consists of six colored blocks of various sizes. The goal of this task is to assemble a model by stacking the blocks in a repeating color sequence (red-yellow-blue) and in order of decreasing size (such that large blocks are placed first). Thus, objects should be mapped according to their hue and relative size. The source and target environments each contain a different kind of block, and are shown in Figure 7(c) and (f). Affordance and property values were provided as listed in Table 2, and remaining features were derived from perceptual information.

As in the cleaning and sorting task, object assistance was provided in the order in which objects would be used to complete the task in the source environment. Objects in this task are stacked in order of the required color sequence and in decreasing size as listed in Figure 7(d) (the task plan referencing objects in the source environment); thus, the object corresponding to the large red block was provided first, then the object corresponding to the large yellow block, and so forth. There is one correct mapping for this task, which was returned by the algorithm after three assists. Each of the provided assists and the predicted mapping after each assist is listed in Table 3, with the bolded assist being the final one provided before the algorithm returned the correct mapping.

5.2 Discussion

In any mapping problem, the number of possible object mappings increases factorially with the number of objects present. Graph isomorphism is an intractable problem in general, and thus this attribute is inherent to any technique for object mapping. This motivates situating mapping in the task environment. While all mapping hypotheses are considered, leveraging mapping assistance helps (i) prune the hypothesis space after each assist, (ii) prune the feature set space, and (iii) re-evaluate the remaining mapping hypotheses to produce a mapping prediction.

The results (Figures 4–6) indicate that using mapping assistance increases the robot's likelihood of predicting a correct mapping quickly. Even when mapping assistance is only used to prune the hypothesis space, as shown in the Hypothesis Pruning results in the simulated evaluation, the robot's likelihood of selecting a correct mapping is dramatically increased over that of choosing an object mapping at random. The MbD algorithm provides further benefit by inferring additional information from the assistance; rather than only use the mapping assist to prune the hypothesis space, it is also used to infer the feature(s) on which mapping may be performed. This benefit is especially evident as the hypothesis space increases. Particularly, Figure 6 shows that in the seven-object task (the category of mapping problems with the largest hypothesis space), the MbD algorithm correctly solves significantly more problems within the first one to four assists than either the Hypothesis Pruning or Random Mapping baselines.

The two perceived tasks (sorting and assembly) demonstrate the use of MbD on physical tasks such as those a robot would need to encounter. By assisting the robot with the initial steps of a task in the target environment, the robot would be able to complete the rest of the task autonomously once it has inferred a correct mapping. The cleaning and sorting task described in Section 5.1.1 would consist of at least two steps per object (clean dish x, put dish x away), resulting in a task containing a total of 14 steps. Only one mapping assist was needed for the robot to predict a correct mapping; as a result, the robot requires mapping assistance only during the first step of the cleaning and sorting task (shown in bold in Figure 7(b)) and would be able to execute the remaining 13 steps autonomously. Similarly, the assembly task described in Section 5.1.2 would consist of two steps per object (pick up part x, install part x), resulting in a task containing 12 steps. Three mapping assists were needed for the robot to predict the correct mapping; as such, the robot requires assistance only during the first five steps of the assembly task (shown in bold in Figure 7(d)) and would be able to execute the remaining seven steps autonomously.

5.2.1 Scaling Considerations. Two variables affect the scalability of the MbD algorithm: the number of objects that are to be mapped and the number of features that are considered when evaluating each mapping hypothesis. As Figures 4–6 indicate, as the number of objects to map increases from $n = 5$ to $n = 7$, more assists are needed to converge on the correct mapping hypothesis. We expect this trend to continue for problems containing $n > 7$ objects. Additionally, the initialization of the $n \times n \times 7$ evaluation matrix E (described in Section 4.3.1) will increase factorially with n and linearly with the number of features (in our evaluation, 7 features were used). Following initialization, this matrix is only updated for relative features (e.g., hue-shift and size-shift similarity).

In a practical application of this algorithm, we expect its scalability to be most challenged in noisy real-world environments, when there are objects that are visible to the robot (and thus increase the complexity of the mapping problem) but irrelevant to the task. For example, a robot that is sorting dishes into a kitchen cabinet is likely to observe other kitchen objects that are unrelated to the sorting task. The MbD algorithm could still be applied to these types of scenarios by considering correspondences between irrelevant objects in its mapping hypotheses; however, this would be an inefficient application of the algorithm. A more efficient approach would be to separate the filtering problem from mapping, using a more suitable algorithm for object filtering so that mapping can be performed only on objects known to be relevant to the context of the task.

While we have limited the scope of this work to n-to-n mapping, this filtering problem also poses an example of how some m-to-n mapping problems may be reduced to an n-to-n problem, where the $m - n$ additional objects are distractor objects that should not be included in the object mapping. Filtering these objects prior to mapping reduces the mapping problem to an n-to-n problem. Similarly, for m-to-n problems in which multiple objects in the source environment map to a single object in the target environment (or vice versa), objects may be clustered prior to mapping to reduce it to an n-to-n problem. An additional (albeit less computationally efficient) approach is to perform mapping for each n-to-n subset of the original m-to-n problem, selecting the mapping that results in the best evaluation score overall. Since the cause of the additional $m - n$ objects is contextual (such as whether it is due to the presence of distractor objects, object composition, object decomposition, or another reason), we leave this problem of m-to-n mapping to future work.

Finally, these results are obtained using simulated assistance and thus rely on the assumption that mapping assistance is always correct. In a realistic interaction, however, the robot would need to obtain this assistance from the human teacher. This introduces several potential sources for error, such as mis-interpretation of the human teacher's assistance or the teacher's misinterpretation of the task. In the next section, we evaluate the MbD algorithm in the interactive context, and explore the effects of interaction error on the algorithm's performance.

6 EXPERIMENT 2: USER STUDY DATA COLLECTION

We now explore the interactive use case of the MbD algorithm. We collect interaction data from a user study consisting of the *demonstration* and *assistance* phases described in Section 4, collecting mapping assistance from the study participant for *every* step of the task. We later perform an offline evaluation of the remaining two phases of our approach, *mapping inference* and the *confidence-based stopping condition,* in Sections 6.4–6.7. Collecting the participants' assistance with every step of the task allows us to later analyze (i) whether the mapping assistance provided by participants could be used to infer the correct object mapping, (ii) the minimum number of assistance requests that were necessary to correctly infer the object mapping, and (iii) the robot's ability to evaluate the confidence of its inferred object mapping.

Table 4. Comparison of User Study Tasks

Task	# of Objects	# of Mapping Hypotheses	# of Correct Solutions	# of Steps	Mapping Type	Source/Target Env. Differences	Affordances	Properties
Stacking	5	120	1	8	Size	Same object set	{pourable, stackable}	{upright}
Sorting	6	720	8	6	Color	Different object set between source/target	{pourable, stackable}	{upright}
Lunch-Packing	4	24	1	6	Affordance	Different object sets within environments	{fillable, openable, edible, pourable}	{open, closed, full, empty, upright}

6.1 Protocol

Eleven participants were recruited from a university campus, each teaching the robot three pick-and-place tasks (as in Figure 3). A recording error occurred during one participant's interaction, and so data from 10 of the participants (8 male, 2 female) could be utilized. To begin, we guided each participant through moving the robot's arm to a series of poses to gain familiarity with the arm's joint configuration, and then through demonstrating an example pick-and-place task to gain familiarity with trajectory demonstrations. The robot perceived its workspace with a RGBD camera and received demonstrations and manipulated objects using a six-degree-of-freedom Kinova Jaco2 arm and a Robotiq-85 gripper. A Wizard of Oz interface was used for gesture interpretation, indicating to the robot which object the participant handed to it, and which object the participant then pointed to. A similar interface was used for speech recognition to toggle opening/closing the robot's gripper when verbally indicated by the participant.

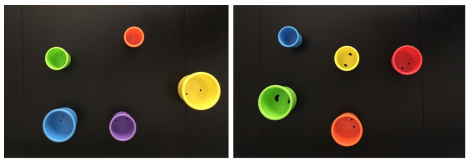
6.2 Interaction

In the first phase, the *demonstration phase*, the robot first observed its environment, recording the features of all objects present. Participants then physically guided the robot's arm to complete the task (as shown in Figure 3(a)), verbally indicating when the robot should open or close its gripper. After the demonstration, the robot's arm was placed into a tucked pose while the workspace was cleared.

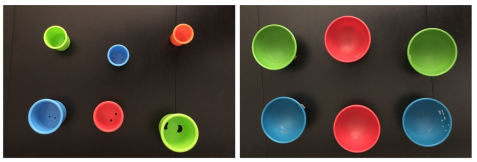
In the second phase, the *assistance phase*, the tabletop contained the target environment objects to be used to repeat the task. The robot again recorded the features of all objects present. After observing the environment, the robot moved its arm to a location centered above the workspace (shown in Figure 3(b)), asking "What do I use now?" Participants were instructed to place the first object used for the task in the robot's gripper. After grasping the object, the robot asked the participant "Where do I go next?," to which participants were to respond by pointing at the location where the object was to be placed (as shown in Figure 3(c)). The robot then placed the object at the indicated location, and then returned to the central location shown in Figure 3(b). While not all mapping assists are necessary for the robot to infer the correct object mapping, we recorded mapping assistance for all steps of the task to evaluate performance after *each* assist; as a result, the process of requesting an object, requesting a goal location, and placing the object was repeated until all task steps were completed successfully.

6.3 Tasks

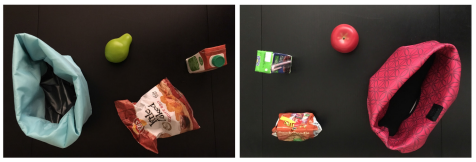
Participants performed both the demonstration and assistance phases for three fully-ordered pick-and-place tasks. These task explore several variables outlined in Table 4. In a **stacking task**, participants were instructed to teach the robot to stack the source and target objects (left and right



(a) Stacking: Source and Target Environments



(b) Sorting: Source and Target Environments



(c) Lunch: Source and Target Environments

Fig. 8. Objects in source and target environments for each task. Figures are best viewed in color.

images in Figure 8(a), respectively) one at a time. Objects in the source and target environment were obtained *from the same object set*, and thus had similar feature values (e.g., object dimensions and hue). In a **sorting task**, participants were to teach the robot to sort cups into a red, green, and blue stack (in that order), and then assist the robot in sorting bowls in the same order (Figure 8(b)). The source and target objects were obtained *from different object sets* (cups and bowls). This task has eight possible correct mappings, since either blue cup may be correctly mapped to either blue bowl, either green cup may be mapped to either green bowl, and either red cup may be mapped to either red bowl. Finally, in a **lunch-packing task**, participants were to teach the robot to pack the drink, fruit, and stack items into the lunch bag (Figure 8(c)), in that order. Later, participants were to assist the robot in packing the second set of lunch items in the same order. This task uses the most realistic objects, where objects within each environment were obtained *from different sources* (and thus the most likely to differ in their perceptual features). The initial object configurations were varied such that the initial configuration of objects in the source and target environments differed from one participant to the next.

6.4 Evaluating Interactive Mapping

After recording all mapping assists during the assistance phase, we provided each assist to the object mapping algorithm incrementally in an offline evaluation. We evaluated performance according to **two criteria**: (1) *correctness* at assist k was measured based on the number of mappings correctly predicted with $\leq k$ assists and (2) *interaction efficiency* was measured as the number of interactions needed to infer the correct mapping. We compared the results of the human-guided mapping approach to **two baseline methods**: (1) *"unguided mapping"* (Algorithm 2) where object mapping is performed without mapping assistance, and (2) *"task relearning"* where we record the theoretical performance if the teacher were to repeat the task demonstration in the target environment. We use this second baseline as a performance upper bound, assuming that relearning the task would require the same number of steps as was recorded in each participant's original task demonstration and that it would result in perfect execution of the task in the target environment.

6.4.1 Object Mapping Results. We first identify the worst-case performance of human-guided object mapping; that is, for each task, we find the upper-bound k number of assists such that the algorithm's best performance is reached with $0 \leq n \leq k$ assists on all instances of that task. These results are recorded in Figure 9. The unguided mapping baseline is represented as the results at 0 assists. The results of human-guided mapping is represented as its performance within assists 1–3;

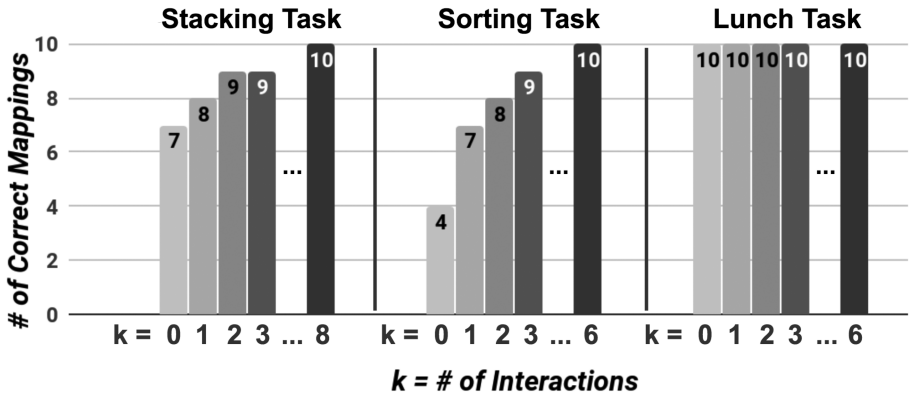


Fig. 9. Number of correct mappings inferred within n assists for each task. Performance at 0 assists represents the unguided baseline, and maximum # of assists ($k = 6$ or $k = 8$) represents the task relearning baseline. In 9 of 10 stacking tasks, assistance was needed for *at most* 2 steps (1/4 of the total number of task steps) before the correct mapping was inferred. In 9 of 10 sorting tasks, assistance was needed for at most 3 steps (1/2 of the total number of task steps). In the lunch task, assistance was not necessary to correctly infer the mapping in all 10 task instances.

ALGORITHM 2: Unguided Mapping Algorithm

```
function MappingInteraction(S)
    T ← observeEnvironment()
    M ← initializeHypothesisSpace(S, T)
    F ← initializeFeatureSpace()
    F ← pruneFeatureSpace(F, S)
    E ← evaluateHypotheses(M, F, S, T)
    p ← predictMapping(M, F, E)
    return p
```

results after assists 3 are not shown, because performance did not improve after the third assist. Task relearning is represented as expected performance after the number of steps that would need to be demonstrated for the task (6 or 8 steps, depending on the task).

Efficiency Implications. Figure 9 indicates that the human-guided method achieved its highest performance for the lunch-packing task without any additional assistance, highest performance for the stacking task with **at most two** assists, and highest performance for the sorting task at most **three** assists. Its performance for the stacking and sorting tasks resulted in approximately 1.3× and 2.25× as many correct mappings, respectively, as the unguided method. For the lunch-packing task, the unguided mapping algorithm was able to perform perfectly, since there was a clear similarity between the objects' affordance and property features in the two environments and little similarity between other features.

Overall, these results indicate that a robot using the human-guided mapping algorithm in real-time would require only modest assistance to infer the correct mapping and repeat the rest of the task autonomously. For 9/10 instances of the stacking task (consisting of eight steps) the robot would only need assistance with *at most* 25% of the task (two steps) to target the correct objects *autonomously* in the remaining 75% of the task. Similarly for 9/10 instances of the sorting task (consisting of six steps) the robot would need assistance with the first 50% of the task (three steps)

Table 5. Interaction Results

Task	# of Demos in Expected Order	# of Assists in Expected Order	# of Consistent Demos & Assists
Stacking	6/10	7/10	**4/10**
Sorting	6/10	6/10	**5/10**
Lunch	8/10	10/10	**8/10**

at most to target the remaining objects autonomously. Note that in a typical case, the robot would need even fewer assists (discussed in Section 7).

6.4.2 Interaction Results. Next, we specifically analyze the human teacher's *interactions* with the robot during the demonstration and assistance phases. While the results in Section 6.4.1 indicate that the mapping algorithm is able to efficiently infer the correct object mapping from assistance, there were several instances in which the teacher did not provide the expected assistance. Table 5 lists the number of demonstrations in which the task was recorded in the expected order in its entirety. In the remaining demonstrations, the objects used in each task step was either (i) unclear to the robot for a particular step (and thus the wrong object was recorded for that step) or (ii) provided to the robot in the wrong order by the human teacher's demonstration. Similarly, the second column lists the number of assists that were provided in the expected order during the assistance phase. The last column indicates the number of task instances in which the demonstration and assistance were provided and recorded in a consistent order. Overall, 43.3% of task instances had at least one inconsistency between the demonstration and assistance orderings, despite the task order being specified prior to the demonstration and assistance phases.

6.5 Discussion

We now discuss three implications of the inconsistencies between demonstration and assistance orderings: (1) their effect on mapping results, (2) the increased error risk incurred by interactive assistance, and (3) proposed strategies for reducing the number of inconsistent assists.

6.5.1 Effect on Mapping Results. There were two sources of error that led to inconsistencies between the demonstration and assistance: recording errors, and interaction errors. As an example of an interaction error, the very first mapping assist for a sorting task was provided in the wrong order, after which the mapping algorithm was unable to recover despite receiving additional assistance (still resulting in 9/10 performance at $k = 3$ assists, compared to the hypothetical 10/10 performance expected with the relearning upper-bound baseline shown at $k = 6$).

As an example of a recording error, one participant demonstrated the stacking task by having the robot first move the largest yellow cup to a central location, and then move all other cups to the yellow cup in its new location. Since the objects were moved to a location that the robot did not record during the initial observation, it recorded the task steps as stacking the cups into the wrong object, leading to incorrect mapping assistance being recorded.

Additional recording errors may occur as a result of mis-classifying semantic features (e.g., object affordances and properties). While these semantic features were manually defined in our evaluation, we expect that these features could be derived from a knowledge base or visual classifier, which may introduce additional error depending on how well they are suited/trained for the target domain. In tasks where objects should be mapped based on a particular semantic feature, this error could result in an incorrect mapping being selected if that feature is mis-classified or missing from the knowledge base.

6.5.2 Increased Error Risk. The effects of inconsistent assistance presents a downside to relying on assistance from the human teacher for object mapping: as the robot requests more assistance, the potential for incorrect assistance increases. Figure 10(b) illustrates this tradeoff; the number of correct mappings increases (or remains stable) with the increase in the number of assists, until a point at which the additional (incorrect) assistance actually causes the number of correct mappings to *decrease*.

We observe that this tradeoff does not occur when the algorithm is performed over a "consistent dataset": the subset of data in which the demonstration and all assistance were provided and recorded in a fully consistent order (4 instances of stacking, 5 instances of sorting, and 8 instances of lunch-packing). Rather, we observe that it correctly infers the object mapping in all 17 of these instances and that the algorithm performance does *not* decrease with additional assistance (in contrast to Figure 10(b)).

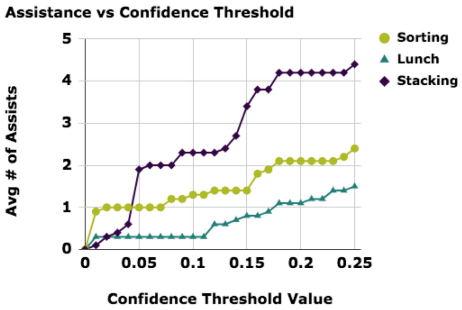
Thus, one method of addressing this tradeoff is to request just enough assistance for the robot to maximize mapping performance, while also minimizing the risk of receiving incorrect assistance. Our third experiment demonstrates a confidence-based threshold that aims to achieve this balance. While limiting the amount of assistance reduces the opportunity for error caused by inconsistent assistance, it does not eliminate it; interaction errors that occur early in the task would still cause the algorithm to converge quickly on the wrong mapping hypothesis.

6.5.3 Guidelines for Interactive Assistance. The consistent-dataset results suggest that performance could be further improved by refining the interaction to reduce the teacher's likelihood of providing inconsistent assistance. We note three factors for future work: (i) ensuring that the task constraints (including the task ordering) are clear to the teacher, (ii) maintaining consistency between the source and target task orderings, and (iii) ignoring mistaken or redundant mapping assists.

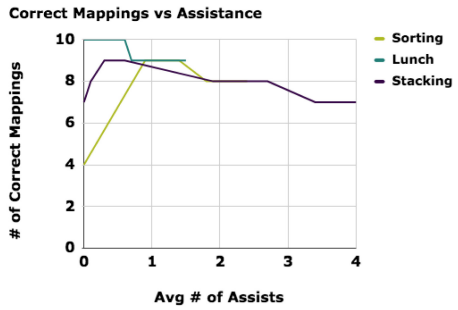
Clarifying Task Constraints. Regarding the first point, the teacher's understanding of the robot's limitations will affect the order in which they demonstrate the task, such as the robot's inability to observe the cup being moved to another location, or attempting to pick up multiple objects while the robot's gripper can only accommodate one. One solution to this challenge is to provide a visual list specifying the order in which object should be used to successfully complete the task. Another solution is for the teacher to be provided with practice trials of the task to become more familiar with it and better understand the task ordering constraints. This would also serve to address errors caused by the teacher mistakenly repeating the task in a different order than originally demonstrated.

Reducing Inconsistent Assistance. Providing additional transparency into the robot's reasoning process may also reduce these errors. While assistance is currently provided by the teacher via gestures, it may be beneficial for the teacher to be able to correct perceptual and/or interaction errors via another interaction method, such as speech (e.g., verbally indicating when to discard the previous assist) or a graphical interface (e.g., a visual representation of the assistance provided so far). One may also consider an alternate framework for obtaining assistance, in which the robot attempts to complete the task using its currently highest-ranked mapping hypothesis; the teacher would then provide assistance by interrupting and correcting the robot's motion, after which the robot would record the correction as an assist and re-rank its mapping hypotheses accordingly.

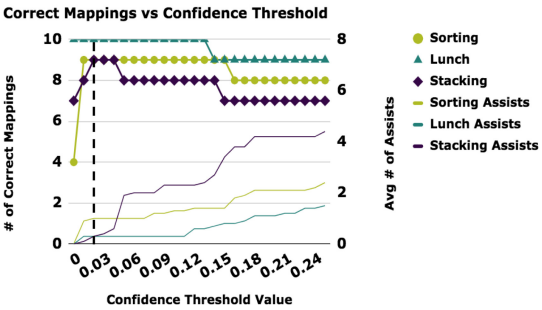
Accounting for Noisy Assistance. Finally, the robot may need to account for "noise" in the teacher's assistance. When providing the robot with mapping assistance, the robot may fail to grasp the intended object, or the teacher may change their mind about which object should be used next in the task. In either case, the robot should record the teacher's corrective actions, rather than



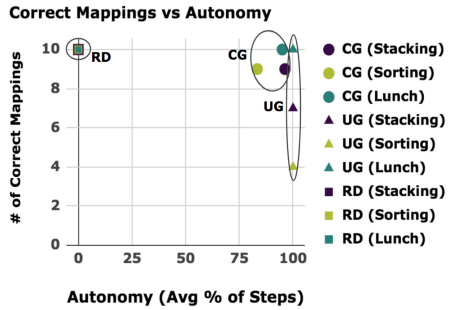
(a) As the confidence threshold value increases, so does the number of assists needed to attain that threshold



(b) Performance initially increases after assistance, but later decreases with further assistance



(c) Relation between confidence threshold (x-axis), average # of assists (bottom lines, sourced from Fig. 10a), and performance (top lines). Threshold of 0.02 (dashed line) maximizes performance and minimizes number of assists.



(d) With optimal threshold value (0.02), confident guided (CG) method results in high autonomy *and* mapping correctness, compared with unguided (UG) and relearning (RD) baselines.

Fig. 10. Efficiency and Correctness Results.

record the mistakes as assistance. This could be enabled by providing an interaction method for overwriting past actions or assistance (such as a speech command the teacher can use to indicate that they want to repeat a particular step of the task).

We also note that the use of assistance is intended to lower the teacher's effort in comparison to giving a full demonstration, by (i) using a simpler form of interaction (indicating an object rather than re-demonstrating a full task motion) and (ii) requiring interaction during only a subset of the task. An additional direction for future work is to compare the teachers' perceived cognitive effort when providing assistance versus a full re-demonstration.

7 EXPERIMENT 3: CONFIDENCE-BASED STOPPING CONDITION

The previous experiment presented a tradeoff between mapping performance and increased assistance. One solution to this tradeoff is for the robot to request just enough assistance to maximize mapping performance, while also minimizing the risk of receiving incorrect assistance. Minimizing the number of assists requested by the robot thus serves two purposes: (i) increasing the robot's autonomy and (ii) reducing the possibility of error introduced via incorrect mapping assistance. We now evaluate the confidence-based threshold described in Section 4.4, considering the number of assists that are needed to maximize mapping performance when the number of assists is *variable* and based on mapping confidence (rather than identify an upper bound as in Section 6.4.1). Confidence values (and thus the decision margin between them) are within the range [0.0, 1.0]. We analyze performance at a range of confidence thresholds between [0.0, 0.25].

We observe the effect of confidence thresholding on mapping performance in two steps. First, changing the confidence threshold affects the number of mapping assists that are needed for the robot to reach that confidence threshold. Thus, as the confidence threshold increases, the number of requested mapping assists also increases, as seen in Figure 10(a). The confidence threshold has a second effect: as more mapping assists are needed to attain the confidence threshold, the performance of the human-guided mapping algorithm also changes in response to the additional assists as shown in Figure 10(b). Figure 10(c) illustrates the resulting relation between the confidence threshold and the mapping algorithm's performance. The optimal confidence threshold ($c = 0.02$) is indicated by the dashed line, and maximizes the algorithm's performance (the line charts at the top of Figure 10(c)) while minimizing the number of assists needed to attain that performance (the line charts at the bottom of Figure 10(c)). Finally, we compare the performance of (i) human-guided mapping using this optimal confidence threshold and (ii) the unguided and relearning baselines. Figure 10(d) demonstrates how confident human-guided object mapping maximizes average autonomy and correctness across the three tasks, whereas the relearning baseline minimizes the robot's autonomy and the unguided baseline provides fewer correct mapping results on average.

Confidence Implications. Applying this threshold to real-time object mapping would enable the robot to repeat the rest of the task autonomously. In Section 6.4.1, we discussed the *maximum* number of assists needed to maximize the human-guided mapping algorithm's results. However, the results from implementing a confidence-based stopping condition on our evaluation data indicate that even fewer assists are necessary in a typical mapping problem. Our results indicate that with a properly selected confidence threshold, the robot could infer the correct object mapping in 93.3% of problems and requested 0 or 1 assists in the average case, thus maximizing both *autonomy* and *correctness*. Further evaluation should (i) incorporate confidence thresholding during a real-time interaction and (ii) test whether this confidence threshold is generalizable across a wider variety of tasks or is dependent on a particular feature of the task (e.g., overall object similarity or number of object features under consideration).

8 CONCLUSION

Without contextual knowledge about the task, the robot cannot weigh object features to identify an object mapping for task transfer. Prior work assumes that (i) the robot has access to multiple new demonstrations of the task or that (ii) the primary features for object mapping have been specified. Our method does not make either assumption; rather than requiring additional demonstrations of the task, it uses limited, structured interaction with a human teacher. Additionally, by having human teachers provide mapping assistance by indicating objects (rather than describing features), we mitigate the need for teachers to have knowledge of which features the robot can observe, or how to express feature values.

Our simulated evaluation demonstrates how the use of mapping assistance quickly reduces the mapping problem complexity, enabling the correct mapping to be identified within a small number of assists. The aim of our interactive evaluation was to test whether (i) participants could provide mapping assistance through natural interaction with the robot and its environment, (ii) the robot could record the objects involved in each step of the task, and (iii) the mapping assistance provided by participants could be used to efficiently infer the correct object mapping. The evaluation results demonstrate that by incorporating human interaction, our human-guided mapping approach meets these criteria, while providing mapping predictions that are accurate, confident, and obtained through a limited number of additional interactions with the human teacher. Finally, we have demonstrated how a confidence-based stopping condition can be used to moderate the robot's interaction with the human teacher, and thus find a balance between **autonomy** and **interaction**.

ACKNOWLEDGMENT

We thank Kalesha Bullard for her help in analyzing the simulated evaluation results.

REFERENCES

Baris Akgun, Maya Cakmak, Karl Jiang, and Andrea L. Thomaz. 2012. Keyframe-based learning from demonstration. *Int. J. Soc. Robot.* 4, 4 (2012), 343–355.

Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robot. Auton. Syst.* 57, 5 (2009), 469–483.

Kalesha Bullard, Baris Akgun, Sonia Chernova, and Andrea L. Thomaz. 2016. Grounding action parameters from demonstration. In *Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN'16)*.

Sonia Chernova and Andrea L. Thomaz. 2014. Robot learning from human teachers. *Synth. Lect. Artif. Intell. Mach. Learn.* 8, 3 (2014), 1–121.

Sonia Chernova and Manuela Veloso. 2007. Confidence-based policy learning from demonstration using Gaussian mixture models. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'07)*. ACM Press, New York, NY, 1. DOI: https://doi.org/10.1145/1329125.1329407

Sonia Chernova and Manuela Veloso. 2009. Interactive policy learning through confidence-based autonomy. *J. Artif. Intell. Res.* 34 (2009), 1–25. DOI: https://doi.org/10.1613/jair.2584

Jim Davies, Ashok K. Goel, and Patrick W. Yaner. 2008. Proteus: Visuospatial analogy in problem-solving. *Knowl.-Based Syst.* 21, 7 (2008), 636–654.

Carlos Diuk, Andre Cohen, and Michael L. Littman. 2008. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, 240–247.

Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. 1989. The structure-mapping engine: Algorithm and examples. *Artif. Intell.* 41, 1 (1989), 1–63.

Tesca Fitzgerald, Kalesha Bullard, Andrea Thomaz, and Ashok Goel. 2016. Situated mapping for transfer learning. In *Proceedings of the 4th Annual Conference on Advances in Cognitive Systems*.

Tesca Fitzgerald and Ashok Goel. 2015. A case-based framework for task demonstration storage and adaptation. In *Proceedings of the International Conference on Case-Based Reasoning Workshop on Case-Based Agents*.

Tesca Fitzgerald, Ashok Goel, and Andrea Thomaz. 2015. A similarity-based approach to skill transfer. In *Workshop on Women in Robotics at Robotics: Science and Systems*.

Dedre Gentner. 1983. Structure-mapping: A theoretical framework for analogy*. *Cogn. Sci.* 7, 2 (1983), 155–170.

Dedre Gentner and Arthur B. Markman. 2006. Defining structural similarity. *J. Cogn. Sci.* 6, 1 (2006), 1–20.

Mary L. Gick and Keith J. Holyoak. 1983. Schema induction and analogical transfer. *Cogn. Psychol.* 15, 1 (1983), 1–38.

Keith J. Holyoak and Paul Thagard. 1989. Analogical mapping by constraint satisfaction. *Cogn. Sci.* 13, 3 (1989), 295–355.

Sandy H. Huang, Jia Pan, George Mulcaire, and Pieter Abbeel. 2015. Leveraging appearance priors in non-rigid registration, with application to manipulation of deformable objects. In *Proceedings of the International Conference on Intelligent Robots and Systems*.

Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. 2013. Active learning for teaching a robot grounded relational symbols. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'13)*.

Alex X. Lee, Abhishek Gupta, Henry Lu, Sergey Levine, and Pieter Abbeel. 2015. Learning from multiple demonstrations using trajectory-aware non-rigid registration with applications to deformable object manipulation. *Proceedings of the Intelligent Robots and Systems (IROS'15)*.

Yaxin Liu and Peter Stone. 2006. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the National Conference on Artificial Intelligence*, Vol. 21. MIT Press, Cambridge, MA, 415.

Dipendra K. Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2016. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *Int. J. Robot. Res.* 35, 1-3 (Jan. 2016), 281–300. DOI: https://doi.org/10.1177/0278364915602060

Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. 2009. Learning and generalization of motor skills by learning from demonstration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'09)*. IEEE, 763–768.

Stefan Schaal. 2006. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*. Springer, 261–280.

Matthew E. Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.* 10 (2009), 1633–1685.

Matthew E. Taylor, Shimon Whiteson, and Peter Stone. 2007. Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM, 37.

Moritz Tenorth, Daniel Nyga, and Michael Beetz. 2010. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'10)*. IEEE, 1486–1491.

Alexander J. B. Trevor, Suat Gedikli, Radu B. Rusu, and Henrik I. Christensen. 2013. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration* (2013).

Markus Waibel, Michael Beetz, Javier Civera, Raffaello D'Andrea, Jos Elfring, Dorian Galvez-Lopez, Kai Haussermann, Rob Janssen, J. M. M. Montiel, Alexander Perzylo, and others. 2011. A world wide web for robots. *IEEE Robot. Autom. Mag.* 18, 2 (2011), 69–82.

Received April 2018; accepted August 2018