Autonomous Robots
https://doi.org/10.1007/s10514-023-10098-5

Multimodal embodied attribute learning by robots for object-centric action policies

Xiaohan Zhang[1] · Saeid Amiri[1] · Jivko Sinapov[2] · Jesse Thomason[3] · Peter Stone[4,5] · Shiqi Zhang[1]

Received: 25 February 2022 / Accepted: 26 February 2023

Abstract

Robots frequently need to perceive object attributes, such as RED, HEAVY, and EMPTY, using multimodal exploratory behaviors, such as *look*, *lift*, and *shake*. One possible way for robots to do so is to learn a classifier for each perceivable attribute given an exploratory behavior. Once the attribute classifiers are learned, they can be used by robots to select actions and identify attributes of new objects, answering questions, such as "*Is this object* RED *and* EMPTY?" In this article, we introduce a robot interactive perception problem, called **M**ultimodal **E**mbodied **A**ttribute **L**earning (MEAL), and explore solutions to this new problem. Under different assumptions, there are two classes of MEAL problems. OFFLINE- MEAL problems are defined in this article as learning attribute classifiers from pre-collected data, and sequencing actions towards attribute identification under the challenging trade-off between information gains and exploration action costs. For this purpose, we introduce **M**ixed **O**bservability **R**obot **C**ontrol (MORC), an algorithm for OFFLINE- MEAL problems, that dynamically constructs both fully and partially observable components of the state for multimodal attribute identification of objects. We further investigate a more challenging class of MEAL problems, called ONLINE- MEAL, where the robot assumes no pre-collected data, and works on both attribute classification and attribute identification at the same time. Based on MORC, we develop an algorithm called **I**nformation-**T**heoretic **R**eward **S**haping (MORC-ITRS) that actively addresses the trade-off between exploration and exploitation in ONLINE- MEAL problems. MORC and MORC-ITRS are evaluated in comparison with competitive MEAL baselines, and results demonstrate the superiority of our methods in learning efficiency and identification accuracy.

Keywords Interactive perception · Sequential decision making · Robot learning · Cognitive robotics

1 Introduction

Intelligent robots are able to interact with objects through exploratory actions in real-world environments. For instance, a robot can use a *look* action to figure out if an object is RED using computer vision methods. However, vision is not sufficient to answer if an opaque bottle is FULL or not, and actions that support other sensory modalities, such as *lift* and *shake*, become necessary. Given the sensing capabilities of robots and the perceivable properties of objects, it is important to develop algorithms to enable robots to use multimodal exploratory actions to identify object properties, answering questions such as "*Is this object* RED *and* EMPTY?" In this article, we use **attribute** to refer to a perceivable property of

✉ Xiaohan Zhang
xzhan244@binghamton.edu

Saeid Amiri
samiri1@binghamton.edu

Jivko Sinapov
jivko.sinapov@tufts.edu

Jesse Thomason
jessetho@usc.edu

Peter Stone
pstone@cs.utexas.edu

Shiqi Zhang
zhangs@binghamton.edu

[1] Department of Computer Science, The State University of New York at Binghamton, Binghamton, NY 13902, USA

[2] Department of Computer Science, Tufts University, Medford, MA 02155, USA

[3] Department of Computer Science, University of Southern California, Los Angeles, CA 90007, USA

[4] Department of Computer Science, The University of Texas at Austin, Austin, TX 78712, USA

[5] Sony AI, Austin, TX, USA

Published online: 29 March 2023

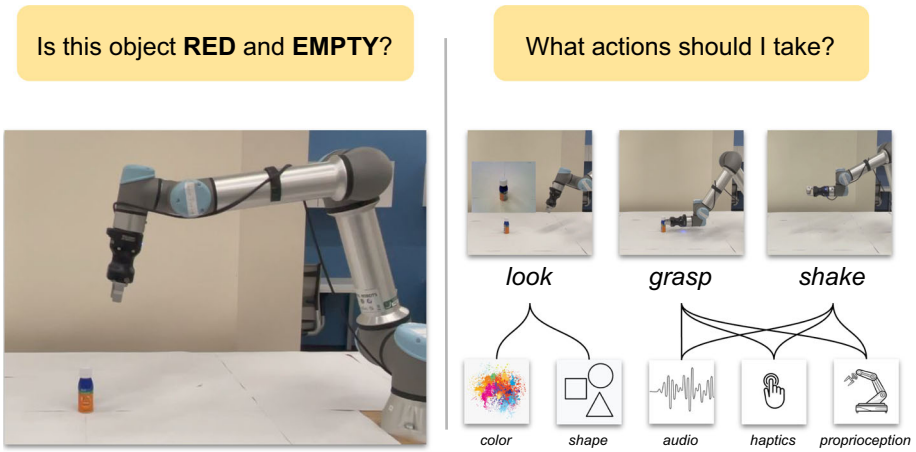


Fig. 1 A robot is tasked with identifying if an object is RED and EMPTY. Given the various sensory modalities produced by exploratory behaviors, the robot must decide what behavior(s) to perform to gain maximum information

an object and use **behavior**[1] to refer to an *exploratory action* that a robot can take to interact with the object. [2]

Given multimodal perception capabilities, a robot still needs to decide which of its many exploratory behaviors to perform on an object. In other words, the robot needs to generate an action policy for each given language request, as illustrated in Fig. 1. For instance, to obtain an object's color, a robot could adjust the pose of its camera, whereas sensing the content of an opaque container requires two behaviors: *grasp* and *shake*. The robot has to select actions in such a way that the information gain about object attributes is maximized while the cost of behaviors is minimized. Sequential reasoning is required in this action selection process. For example, *shake* would make sense only if *grasp* has been successfully executed. Also, robot perception capabilities are imperfect, so the robot sometimes needs to take the same behavior more than once. The above-mentioned observations motivate the development of this article focusing on the **M**ultimodal **E**mbodied **A**ttribute **L**earning (MEAL) problem. We informally define MEAL as follows:

> Algorithms for MEAL problems aim to learn a policy for sequentially selecting exploratory behaviors to efficiently and accurately identify perceivable attributes of objects. Those behaviors might involve multiple sensory modalities and are not necessarily always successful.

The capability of solving MEAL problems is important for robot multimodal perception. In this article, we introduce and investigate two types of MEAL problems: offline and online. Algorithms for **OFFLINE-MEAL** problems aim to learn robot behavioral exploration policies from a previously collected dataset. Probabilistic planning algorithms aim at computing action policies to help select actions toward maximizing long-term utility such as information gain in our case, while considering the uncertainty resulting from nondeterministic action outcomes. Markov decision processes (MDPs) [1] and partially observable MDPs (POMDPs) [2] enable an agent to plan under uncertainty with full and partial observability respectively. However, the observability of real-world domains is frequently mixed: some components of the current state can be fully observable while others are not. A mixed observability Markov decision process (MOMDP) is a special form of POMDP that accounts for both fully and partially observable components of the state [3]. In the **ONLINE-MEAL** setting, the robot needs to learn an accurate and efficient action policy for interacting with objects without pre-collected data. As a result, algorithms for **ONLINE-MEAL** problems are required to work on attribute classification and identification at the same time. ONLINE-MEAL raises the fundamental trade-off between exploration and exploitation. The robot has a short-term goal of identifying object attributes in the current task, and a long-term goal of improving its identification accuracy over multiple tasks. An extreme solution is to let the robot optimize its actions focusing on only the short-term goal. In doing so, the robot still improves its performance in identification tasks over a long term as the robot collects data along the way. However, the learning process in this extreme solution can be poor with respect to regret minimization, because it lacks a mechanism for actively improving its long-term attribute identification performance.

- This article introduces **M**ixed **O**bservability **R**obot **C**ontrol (MORC), as the **first algorithmic contribution** of this article, where we model OFFLINE- MEAL problems using MOMDPs because of the mixed observability of the world that the robot interacts with. For example, whether an object is in the robot's hand or not is fully

[1] The terms of "behavior" and "action" are widely used in developmental robotics and sequential decision-making communities respectively. In this article, the two terms are used interchangeably.

[2] Project webpage: https://sites.google.com/view/attribute-learning-robotics/

observable, but object attributes such as color and weight are not.

- The **second algorithmic contribution** of this article is an algorithm which is called **MORC** with **I**nformation-**T**heoretic **R**eward **S**haping (MORC-ITRS) for ONLINE-MEAL problems. MORC-ITRS, for the first time, equips a robot with the capability of optimizing its sequential action selection toward efficiently and accurately classifying and identifying attributes at the same time.

MORC and MORC-ITRS are evaluated using three datasets: **ISPY32** [4] which contains 32 objects with eight exploratory behaviors and six types of sensory modalities; **ROC36** [5] which includes 36 objects with eleven behaviors and four types of modalities; **CY101** [6] which has 101 objects with ten behaviors and seven types of modalities. These datasets have previously been used for a variety of tasks including language grounding [7,8], object recognition [9], object categorization [10], and sensorimotor learning [11]. Experiments on OFFLINE- MEAL problems show that the policies from MORC improve accuracy for recognizing new objects' attributes while reducing exploration cost, in comparison to baseline strategies that deterministically or randomly use predefined sequences of behaviors. For ONLINE- MEAL settings, compared with existing methods from the MEAL literature which also include a variant of MORC, MORC-ITRS reduces the overall cost of exploration in the long term while reaching a higher accuracy of attribute identification.

Initial versions of the MORC and MORC-ITRS algorithms were introduced in two separate conference papers [12,13]. Both papers aimed to enable a robot manipulator to identify object attributes using multiple exploratory behaviors and the produced multimodal sensory data. Both papers modeled the non-determinism of action outcomes, and exploration costs using a sequential decision-making framework under partial observability. Despite the shared goals, the two conference papers were developed under different assumptions. Next, we describe the relationship between this article and the two previous papers in the three dimensions of "problem," "algorithm," and "evaluation."

- *Problems* This article aims to solve two problems called OFFLINE- MEAL and ONLINE- MEAL. The two problems were initially defined under the respective names of Robot Attribute Identification (RAI) and Online Robot Attribute Learning (On-RAL) [13]. We observed that the initial problem statements [13] were incomplete, because neither the objective function nor the reward function was included in the definitions. We have fixed the issues in this article, and renamed the problems for better clarity.
- *Algorithms* This article develops two algorithms called MORC and MORC-ITRS. Our naming strategy highlights that MORC-ITRS is based on MORC. The key idea of MORC was initially presented in a conference paper [12], where it was only informally described. MORC-ITRS was initially presented in another [13] under the name of ITRS. This article formally defines both algorithms using the terminology specified in the corresponding problem statements.
- *Evaluations* The evaluation of MORC [12] was based on the two datasets of **ISPY32** and **ROC36**. The evaluation of MORC-ITRS [13] was based on the two datasets of **ISPY32** and **CY101**. The different datasets in the two conference papers made it hard to directly compare the different methods. In this article, new experiments were performed to include all three datasets in the evaluations.

In addition, we discuss related work more comprehensively and point out the limitations of both algorithms while identifying research directions for future work.

The remainder of this article is organized as follows. Section 2 discusses existing research on MEAL-related topics. Section 3 formally defines three MEAL problems, including the OFFLINE- MEAL and the ONLINE- MEAL. Section 4 presents existing research on **A**ction-**C**onditioned **A**ttribute **C**lassification (ACAC), which serves as a building block of algorithms and systems developed in this article. Sections 5 and 6 describe two algorithms respectively: MORC for OFFLINE- MEAL and MORC-ITRS for ONLINE- MEAL. Experimental results and demonstrations are detailed in Sect. 7. The article is concluded in Sect. 8, including discussions about the limitations of our algorithms and directions for future work.

2 Related work

This section summarizes a few research areas that are relevant to Multimodal Embodied Attribute Learning (MEAL), the focus of this article. We first briefly describe the concept of "attribute", which is widely used in the computer vision community, and then discuss existing research on robot perception (unimodal and multimodal). After that, a representative sample of algorithms for planning under uncertainty is described, based on which we develop MORC and MORC-ITRS. Finally, we present existing work on object-centric robot perception, which is the application domain of this article.

Visual attribute learning

The word "attribute" refers to as "an inherent characteristic" of an object in the computer vision community [14]. Attributes are also semantic and machine-understandable properties that are used by people to describe images [15]. Early vision-based attribute learning methods used image segments to learn visual attributes as patterns [16]. Later, researchers studied visual attribute learning in the context of

generalization across object categories [17], and considered transferring visual attributes for previously unseen object classes [18]. More recent approaches focused on zero-shot or few-shot learning for specifying attributes [19–21], relative attributes to enable visual comparisons between images [15,22], and attributes for an unconstrained set of objects by providing large-scale datasets [23–25]. In contrast to traditional attribute learning using these vision methods, we focus on Multimodal Embodied Attribute Learning (MEAL), where objects are explored and attributes are detected by a physical robot. Modern robots have the capability to actively interact with objects, producing rich sensory signals that go beyond vision.

Early research on symbol grounding introduced the term of "category" as an invariant feature of objects and events from their sensory projections [26]. In this article, we focus on object-centric robot perception, and use "attribute" to refer to the categorical representations of object features that are perceivable through a robot's multimodal exploratory behaviors.

Unimodal perception in robotics

Among various modalities that have been researched in robotics, visual perception has been widely studied, including visual manipulation [27] and navigation [28]. Language is another important modality that robotics researchers frequently focus on, solving the challenge of grounding natural language into noisy percepts and physical actions (as reviewed in a recent article [29]). Tactile sensing is traditionally studied in the area of sensors, but recent papers have investigated more about correlations with robot actions, as reviewed in articles [30,31]. There are also a few works that demonstrate the usefulness of smell for robots, especially in the application domain of gas source localization [32]. A limited number of papers have even mimicked the sense of taste for the interaction and cognitive abilities of modern robots [33]. Every single modality has been shown its effectiveness towards improving the perception capability of intelligent robots. Our agent learns from sensory modalities such as vision, audio, and haptics, and works on the problem of robot multimodal perception, which is reviewed in detail in the next paragraph.

Multimodal perception in robotics

Significant advances have been achieved recently in computer vision [34,35] and natural language processing [36,37]. While language and vision are important communication channels for robotic perception, many object attributes cannot be detected using vision alone [38] and people are not always available to verbally provide guidance in exploration tasks. Therefore, researchers have jointly modeled language and visual information for multimodal text-vision tasks [39]. However, many of the most common nouns and adjectives such as SOFT and EMPTY have a strong non-visual component [40] and thus, robots need to perceive objects using additional sensory modalities to reason about and perceive such linguistic descriptors. To address this problem, several lines of research have shown that incorporating a variety of sensory modalities is the key to further enhancing the robotic capabilities in recognizing multisensory object attributes, as reviewed in recent articles [31,41]. For example, visual and physical interaction data yields more accurate haptic classification for objects [42], and non-visual sensory modalities, such as audio and haptics, coupled with exploratory behaviors such as *touch* or *grasp*, have been shown useful for object recognition [43–45]. Grounding natural language descriptors that people use to refer to objects has also been a promising method for attribute recognition problems [4,46]. More recently, researchers have developed end-to-end systems to enable robots to learn to perceive the environment and perform actions at the same time [47,48]. A major limitation of these and other existing methods is that they require large amounts of object exploration data, which is much more expensive to collect as compared to vision-only datasets. A few approaches have been proposed to actively select actions at test time, for instance, when recognizing an object [10,49]. One recent work has also shown that robots can bias which behavior to perform at training time, that is, when learning a model grounded in multiple sensory modalities and behaviors, but they did not learn an actual policy for doing so [8]. Different from existing work, MORC for OFFLINE- MEAL problems is for learning an action policy when deciding whether a set of attributes hold true for an object. MORC-ITRS for ONLINE- MEAL problems learns an action policy for object exploration that a robot can use when learning to ground the semantics of attributes.

Planning under uncertainty

Decision-theoretic methods have been developed to help agents plan actions and address uncertainty in non-deterministic action outcomes [1,50]. Existing planning models such as partially observable Markov decision process (POMDP) [2], belief space planning [51] and Bayesian approaches [52] have shown great advantages for planning robot perception behaviors, because robots need to use exploratory behaviors to estimate the current world state. To learn semantic attributes, robots frequently need to choose multiple actions, so POMDP which is useful for long-term planning is particularly suitable. Many of the POMDP-based robot perception methods are vision-based [53–56]. Compared to those methods, our robot takes advantage of non-visual sensory modalities, such as audio and haptics. Our proposed algorithms both modeled the mixed observability [3] in domains of a robot interacting with objects. The main difference between OFFLINE- and ONLINE- MEAL is that the former assumed that sufficient training data and annotations are available for the robot to learn the perception models of its exploratory behaviors, and the latter deals with data collection and task completion processes simultaneously.

Object-centric robot perception

To select actions to identify objects' perceivable attributes such as HEAVY, RED, FULL, and SHINY, robots need observation models for their exploratory behaviors. Researchers have developed algorithms to help robots determine the presence of possibly new attributes [57] and learn observation models of objects' perceivable attributes given different exploratory behaviors [4,7,58]. In the case where the object attributes refer to the object's function, they are then referred to as 0-order affordances [59]. Task and motion planning methods have been applied to object-centric perception while leveraging physics simulation [68]. Those methods focused on learning to improve the robots' perception capabilities. Once the learning process is complete, a robot can use the learned attributes to perform tasks, such as attribute identification, for example, to tell if a bottle is HEAVY and RED. Compared to those learning methods, we consider both OFFLINE- and ONLINE- MEAL, where the robot learns the attribute classifiers (an exploration process) and uses the learned classifiers to identify object attributes (an exploitation process). Because ONLINE- MEAL agents iteratively learn and identify attributes, the exploration-exploitation trade-off is a fundamental decision-making challenge in this unknown robot perception environment. While the problem has been studied in multi-armed bandit [60] and reinforcement learning settings [50], it has not been studied in MEAL contexts.

3 Problem definitions

In this section, we first introduce the terminology of our work (Table 1). Then we formally define three types of robot multimodal perception problems.

A robot has a set of behaviors, such as *look*, *push*, and *lift*, that can be used for interacting with everyday objects as shown in Fig. 2. Let $o \in Obj$ be an object and $a \in \mathcal{A}^e$ be an *exploratory behavior*. Examples of a robot executing some of the exploration behaviors are shown in Fig. 3.

Each exploratory behavior is coupled with a set of sensory modalities, e.g., vision, haptics, and audio. We use $m \in \mathcal{M}$ to refer to a sensory modality. This behavior-modality coupling is formulated using function Γ:

$$\mathcal{M}_a = \Gamma(a) \tag{1}$$

where $\mathcal{M}_a \subseteq \mathcal{M}$. For example, $\{audio, haptics, vision\} = \Gamma(push)$ means that behavior *push* produces signals from audio, haptics, and vision modalities. When a is performed on an object o, for each $m \in \mathcal{M}_a$, the robot is able to record a data instance $f_a^m \in \mathbb{R}^{N^m}$, where N^m is the dimensionality for the modality m. Table 2 shows the set of viable combinations of behavior-modality pairs for one of the datasets used in our experiments (detailed in Sect. 7), along with the feature dimensionality N^m. We use f_a to represent a set of sensory data instances from all modalities ($m \in \mathcal{M}_a$) that a robot receives after performing a.

Let $\mathcal{P}$ specify a set of attributes that are used to describe objects in a domain. Given an object o, v^p is either *true* or *false*, depending on if p applies to o, where we use $ID(p, o)$ to refer to this attribute identification function. Here we "override" the function ID to use it to process a set of attributes:

$$\mathbf{v} = ID(\mathbf{p}, o) \tag{2}$$

where $\mathbf{v} = [v^{p_0}, v^{p_1}, \cdots]$, $\mathbf{p} = [p_0, p_1, \cdots]$, and v^{p_i} is the value of the i^{th} attribute of object o. For instance, given a red empty object (i.e., o) and two attributes [BLUE, EMPTY] (i.e., $\mathbf{p}$), ID([BLUE, EMPTY], *a red empty object*) outputs $[false, true]$. The reporting action set $\mathcal{A}^r$ includes actions that are used for the robot to report $\mathbf{v}$ to the human user. For the same red empty object, there will be two binary variables specifying whether each of the attributes is true or false. As a result, there will be four reporting actions corresponding to the four combinations of the attributes' values. The action set $\mathcal{A} = \mathcal{A}^e \cup \mathcal{A}^r$.

The robot state space is mixed observable and has two components, $\mathcal{X}$ and $\mathcal{Y}$. The global state space S includes a Cartesian product of $\mathcal{X}$ and $\mathcal{Y}$,

$$S = \{(x, y) \mid x \in \mathcal{X} \text{ and } y \in \mathcal{Y}\} \tag{3}$$

$\mathcal{X}$ is the state set specified by fully observable domain variables. In our case, $x \in \mathcal{X}$ represents the current state of the robot-object system, e.g., whether *lift* and *drop* behaviors are successful or not, or whether the object is in hand or not (i.e., the effect of behavior *grasp*). $\mathcal{Y}$ is the state set specified by partially observable domain variables. In our case, these variables correspond to the values of object attributes that are queried about, i.e., $\mathbf{v}$. Thus, $\mid \mathcal{Y} \mid = 2^{N^p}$, where N^p is the number of queried attributes, i.e., $\mid \mathbf{p} \mid$. For instance, given an object description that includes three attributes (e.g., [RED, EMPTY, BOTTLE]), $\mathcal{Y}$ includes $2^3 = 8$ states. Since $y \in \mathcal{Y}$ is partially observable, it needs to be estimated through observations, which will be defined in the next paragraph. Note that there is no state transition in the space of $\mathcal{Y}$, as we assume object attributes do not change over executions of robot actions.

Let $\mathcal{Z} = \mathcal{Z}^h \cup \emptyset$ be a set of observations. Elements in $\mathcal{Z}^h$ include all possible combinations of object attributes (i.e., $\mathbf{v}$) and have one-to-one correspondence with elements in $\mathcal{A}^r$. For instance, when $\mathbf{p}$ = [RED, EMPTY, BOTTLE], there exists an observation $z \in \mathcal{Z}^h$ that is $[true, false, true]$ that represents "the object's color is red; it is not empty, and it is a bottle." Behaviors that produce no information gain (including those failed behaviors) and reporting actions in $\mathcal{A}^r$ result in a $\emptyset$ (none) observation.

Table 1 Table of Notation

Symbol/Notation	Definition
o	An object
Obj	The set of all objects
a	A behavior
$\mathcal{A}^e$	The set of all exploratory behaviors
m	A modality
$\mathcal{M}$	The set of all modalities
$\Gamma(a)$	A behavior and modality coupling function
$\mathcal{M}_a$	A set of modalities that a behavior a produces
f_a^m	A sensory data instance of a modality m being recorded when a behavior a is applied
f_a	A set of sensory data instances from all modalities ($m \in \mathcal{M}_a$) that a robot receives after performing a
N^m	Dimensionality for a modality m
P	The set of all attributes
p	An attribute
v^p	The Boolean value of an attribute p indicating if p applies to an object
$ID(p, o)$	An attribute identification function
$\mathbf{v}$	Values of a set of attributes
$\mathbf{p}$	A set of attributes
$\mathcal{A}^r$	The set of all reporting actions
$\mathcal{A}$	The set of all actions including exploratory behaviors and reporting actions
$\mathcal{S}$	The global state space
$\mathcal{X}$	The state set specified by fully observable domain variables
x	A fully observable state
$\mathcal{Y}$	The state set specified by partially observable domain variables
y	A partially observable state
N^p	The number of queried attributes
$\mathcal{Z}$	A set of observations including $\emptyset$ (none) observation
$\mathcal{Z}^h$	A set of observations excluding $\emptyset$ (none) observation
$R(s, a)$	The reward function
t_a	Time length for executing a behavior a
r^-	Negative reward by given an incorrect report action
r^+	Positive reward by given a correct report action
ξ	An episode for representing a single attribute identification task
$s \oslash a$	An operation that outputs true (or false) when the identification task is successful (or not)
$Rst(\xi)$	A function that outputs if a task is successful
$Cst(\xi)$	A function that outputs the accumulative action cost in a task
$\mathcal{D}$	A dataset where each instance is in the form of $(f_a, p) : v^p$
$\Psi(f_a, p)$	A binary classifier. A robot collect data instance f_a on an object after performing action a, and $\Psi(f_a, p)$ outputs if attribute p applies to this object
$\hat{C}$	An action cost budget
$\Phi(f_a^m, p)$	A binary classifier that is similar to $\Psi(f_a, p)$ but with modality specifications
α	A normalization constant

Table 1 continued

Symbol/Notation	Definition
w_a^m	A weight for a binary classifier $\Phi(f_a^m, p)$
Θ_p^a	A confusion matrix that are computed by cross-validation at training stage
$T_{\mathcal{X}}$	Transition functions for fully observable states in MOMDP
$T_{\mathcal{Y}}$	Transition functions for partially observable states in MOMDP
γ	A discount factor in MOMDP
$O(s, a, z)$	An observation function that specifies the probability of observing z when action a is executed in state s
$v_s^{p_i}$	The true value of the i^{th} attribute in state s
$v_z^{p_i}$	The observed value of the i^{th} attribute in observation z
$Ent(s, a)$	The entropy that is used for indicating the perception quality
$IE(s, a)$	The interaction experience of applying action a to identify attribute p
δ	A sufficiently large integer to ensure $IE(s, a)$ is the range of [0,1)
α	A natural number for adjusting $Ent(s, a)$
β	A natural number for adjusting $IE(s, a)$

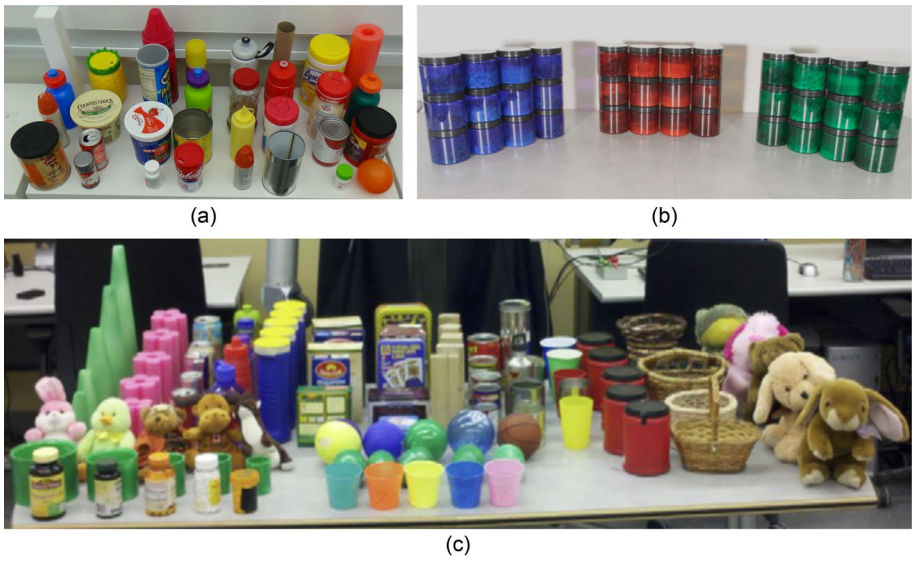


Fig. 2 Everyday objects in the three datasets that are used in the experiments of this article: **a** ISPY32 [4] **b** ROC36 [5] **c** CY101 [6]

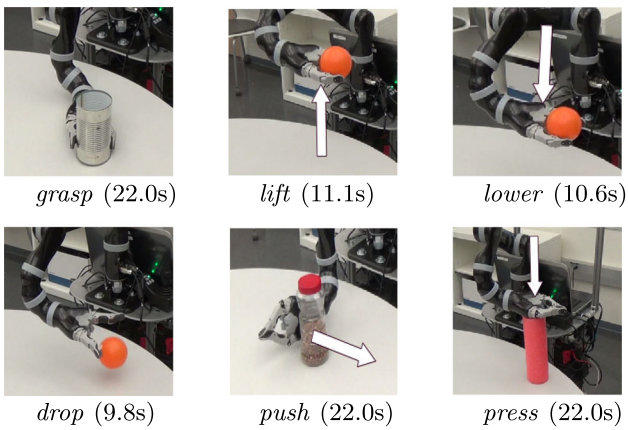


Fig. 3 Examples of behaviors and their durations in seconds (behaviors are from the ISPY32 dataset detailed in Sect. 7)

Table 2 The number of features extracted from each combination of robot behavior and perceptual modality for ISPY32. "VGG" modality is computed from 2D image of the object and is deep visual features from the 16-layer VGG network [4]

Behavior	Modality		
	Color	Shape	VGG
look	64	308	4096
	Audio	Haptics	Proprioception
grasp	100	60	20
drop, hold, lift, lower, press, push	100	60	

$R \rightarrow \mathbb{R}$ is the reward function. Each *exploration behavior*, $a^e \in \mathcal{A}^e$, has a cost that is determined by the time required to complete the behavior. These costs are empirically assigned

according to the datasets used in this research. The costs of *reporting actions* depend on whether the report is correct.

$$R(s,a) = \begin{cases} -t_a, \text{if } s \in S, a \in \mathcal{A}^e \\ r^-, \text{if } s \in S, a \in \mathcal{A}^r,\ s \oslash a = false \\ r^+, \text{if } s \in S, a \in \mathcal{A}^r,\ s \oslash a = true \end{cases} \tag{4}$$

where t_a is the time length for executing the behavior a, r^- (or r^+) is negative (or positive) given an incorrect (or correct) report. $s \oslash a$ outputs true (or false) when the identification task is successful (or not). We further define an episode as $\xi = [s_0, a_0, r_0, s_1, a_1, r_1, \cdots]$ for representing a single attribute identification task, where $s_i \in S, a_i \in \mathcal{A}$, and $r_i \in R$. Function $Rst(\xi)$ outputs if a task ξ is successful:

$$Rst(\xi) = \begin{cases} 1, \text{if there exists } s_i, a_i \in \xi, s_i \in S, a_i \in \mathcal{A}^r, \\ s_i \oslash a_i = true \\ 0, \text{otherwise} \end{cases} \tag{5}$$

Function $Cst(\xi)$ outputs the accumulative action cost in task ξ:

$$Cst(\xi) = \sum_{a_i \in \mathcal{A}^e} t_{a_i} = -\sum_{a_i \in \mathcal{A}^e} r_i(s, a_i) \tag{6}$$

Definition 1 (**ACAC**) At training time, the input includes a set of labeled sensory data instances, each in the form of $(f_a, p) : v^p$. This training set is sufficiently large and denoted as $\mathcal{D}$. Solving an **Action-Conditioned Attribute Classification**[3] (ACAC) problem produces a binary classifier:

$$\Psi(f_a, p), \text{ for each pair of } a \in \mathcal{A}^e \text{ and } p \in \mathcal{P}$$

At testing time, given an object o, a robot collects data instances f_a after performing action a and $\Psi(f_a, p)$ outputs *true* or *false* estimating if attribute p applies to o.

Definition 2 (**OFFLINE-MEAL**) Solving an **OFFLINE-MEAL** problem produces a policy π that sequentially selects action $a \in \mathcal{A}$ to identify the value of:

$$ID(\mathbf{p}, o), \text{ given } \mathcal{D}, R(s,a)$$

where the objective is to maximize the number of successful task completions within a cost budget $\hat{C}$. Let $[\xi_0, \xi_1, \cdots, \xi_n]$ be a set of n attribute identification tasks, then the objective function can be defined as follows:

$$\arg\max_{\pi} \left(E\left(\sum_{i \le n} Rst(\xi_i) \right) \right), \text{ where } \sum_{i \le n} Cst(\xi_i) < \hat{C} \tag{7}$$

Definition 3 (**ONLINE-MEAL**) Solving an **ONLINE-MEAL** problem produces a policy π that sequentially selects action $a \in \mathcal{A}$ to identify the value of:

$$ID(\mathbf{p}, o), \text{ given } R(s,a)$$

OFFLINE- and ONLINE- MEAL share the same objective function (Eq. 7), while algorithms for ONLINE- MEAL are *not* provided with pre-collected data $\mathcal{D}$. At execution time, after performing a to identify $\mathbf{p}$, the robot collects data f_a. After each identification task, the robot receives $\mathbf{v}$, the values of attributes $\mathbf{p}$.

Remark 1 Rational OFFLINE- MEAL agents treat individual attribute identification tasks independently, whereas rational ONLINE- MEAL agents learn from the data collected in early tasks, trading off early-phase performance for long-term performance.

Remark 2 One might have noticed the difference between the objective function defined in Eq. 7 and the reward function in Eq. 4. There can be the question of whether our reward function supports the robot in achieving the objective or not. Note that the objective function presented in Eq. 7 includes two dimensions: identification accuracy and exploration cost. The reward function presented in Eq. 4 includes three terms, where action execution time t_a corresponds to the exploration cost, and rewards r^- and r^+ correspond to the identification accuracy. Altogether, this reward function is able to motivate the robot to maximize identification accuracy and minimize exploration costs at the same time. Thus, our current reward function supports computing policies towards achieving the objective.

4 Preliminaries

The algorithms developed in this article rely on existing research on Action-Conditioned Attribute Classification (ACAC). For the sake of completeness, we summarize the technical approach of previous ACAC work [4,5], which is used as a building block for defining algorithms for MEAL problems.

We define **individual classifiers** for connecting data instances f_a^m to each attribute p, denoted as $\Phi(f_a^m, p)$. We

[3] We use attribute classification to refer to the problem of learning the attribute classifiers, which is a supervised machine learning problem. We use attribute identification to refer to the task of identifying whether an object has a set of attributes or not, which corresponds to a sequential decision-making problem.

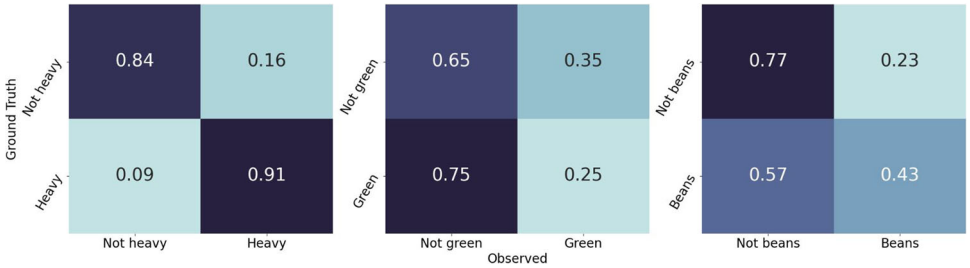


Fig. 4 Example confusion matrices training from 36 objects (ROC36) showing the TP, FP, TN, and FN rates for three of the attributes when using the robot's *shake* behavior. The behavior is good at recognizing HEAVY due to the rich haptic feedback produced when shaking an object, somewhat good at recognizing BEANS (referring to the objects' contents) due to the sound produced by the contents, and poor at recognizing GREEN as no visual input is processed when performing this behavior

assume that the outputs of Φ can be mapped to probabilities, i.e., $\Phi(f_a^m, p)$ can estimate $\mathbf{Pr}_a^m(v^p = true \mid f_a^m)$. In line with prior research [4,5], individual classifiers Φ were structured using support-vector machines with a polynomial kernel in this article. The binary classifier Ψ (introduced in Sect. 1) is for connecting a set of $\Phi(f_a^m, p)$ regardless of modality specifications. Similarly, the outputs of Ψ can be mapped to probabilities, i.e., $\Psi(f_a, p)$ can estimate $\mathbf{Pr}_a(v^p = true \mid f_a)$. It should be noted that for each behavior, different modalities are not equally preferred when the robot identifies certain attributes. For instance, color is more useful than shape when identifying RED. Thus, the probability estimates of Φ are combined using weighted combination and normalized again to compute the final probability estimates of Ψ:

$$\mathbf{Pr}_a(v^p{=}true \mid f_a){=}\alpha \times \sum w_a^m \times \mathbf{Pr}_a^m(v^p{=}true \mid f_a^m)$$

where α is a normalization constant to ensure the probabilities sum up to 1.0 and $w_a^m \in [0.0, 1.0]$ is a reliability weight indicating how good the classifier associates with the behavior a and the modality m is at recognizing attribute p. In other words, each behavior acts as a classifier ensemble where each individual classifier's output is combined using a weighted combination. The weights are estimated by performing cross-validation of the classifier specific to that modality and behavior.

At the end of the training stage, cross-validation at the behavior level is used to compute the confusion matrix $\Theta_p^a \in \mathbb{R}^{2\times 2}$ for each pair of attribute p and behavior a. These confusion matrices are normalized to compute the True Positive, True Negative, False Positive, and False Negative rates for each behavior-attribute pair. Cross-validation is not a general step for ACAC, but will later be used in MORC. Example confusion matrices are shown in Fig. 4. Next, we describe our MORC approach to address the problem of OFFLINE- MEAL which is defined in Sect. 1.

5 An algorithm for OFFLINE-MEAL: MORC

In this section, we describe the theoretical framework of mixed observability robot control (MORC) for solving OFFLINE- MEAL problems. Behaviors such as *look* and *drop*, have different costs and different accuracies in attribute recognition. At each step, the robot has to decide whether more exploration behaviors are needed, and, if so, select the exploration behavior that produces the most information. In order to sequence these behaviors toward maximizing information gain, subject to the cost of each behavior (e.g., the time it takes to execute it), it is necessary to further consider preconditions and non-deterministic outcomes of the behaviors. For instance, *shake* and *drop* behaviors make sense only if a preceding unreliable *grasp* behavior succeeds.

In this article, we assume action outcomes are fully observable and object attributes are not. For instance, a robot can reliably sense if a *grasp* behavior is successful, but it cannot reliably sense the color of a bottle or if that bottle is FULL. Due to this mixed observability and unreliable action outcomes, we use mixed observability MDPs (MOMDPs) [3] to model the sequential decision-making problem for object exploration.

A MOMDP has mixed state variables. The fully observable state components are represented as a single state variable x (in our case, the *robot-object status*, e.g., the object is in hand or not), while the partially observable components are represented as state variable y (in our case, the *object attributes*, e.g., the object is HEAVY or not). As a result, (x, y) specifies the complete system state, and the state space is factored as $S = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ is the space for fully observable variables and $\mathcal{Y}$ is the space for partially observable variables.

Formally, a MOMDP model is specified as a tuple,

$$(\mathcal{X},\ \mathcal{Y},\ \mathcal{A},\ T_{\mathcal{X}},\ T_{\mathcal{Y}},\ R,\ \mathcal{Z},\ O,\ \gamma),$$

where $\mathcal{A}$ is the action set, $T_{\mathcal{X}}$ and $T_{\mathcal{Y}}$ are the transition functions for fully and partially observable variables respectively, R is the reward function, $\mathcal{Z}$ is the observation set, O is the

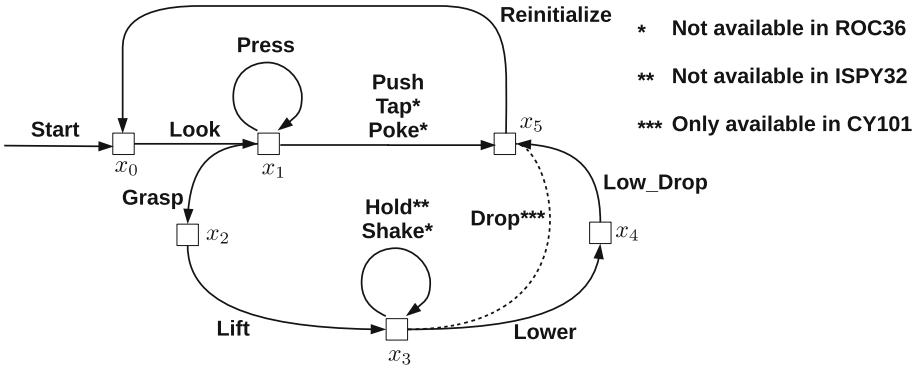


Fig. 5 A simplified version of the transition diagram in space $\mathcal{X}$ for object exploration. This figure only shows the probabilistic transitions led by *exploration behaviors*. *Report actions* that deterministically lead transitions from $x_i \in \mathcal{X}$ to *term* (terminal state) are not included

observation function, and γ is the discount factor that specifies the planning horizon.

5.1 Action transition system

$T_{\mathcal{X}} : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \rightarrow [0, 1]$ is the state transition function in the fully observable component of the current state. $T_{\mathcal{X}}$ includes a set of conditional probabilities of transitions from $x \in \mathcal{X}$—the fully observable component of the current state—to $x' \in \mathcal{X}$, the component of the next state, given $a \in \mathcal{A}$ the current action. The transition diagram is shown in Fig. 5. Reporting actions and illegal exploration behaviors (e.g., *drop* an object in state x_1—before a successful *grasp*) lead state transitions to *term* (terminal state) with 1.0 probability.

Most exploration behaviors are unreliable and succeed probabilistically. For instance, $T_{\mathcal{X}}(x_4, drop, x_5) = 0.95$ in our case, indicating there is a small probability the object is stuck in the robot's hand (detailed in Sect. 7.1). Such non-deterministic action outcomes are considered in our experiments. The success rate of the behavior *look* is 1.0 in our case since without changing positions of either the camera or the object it does not make sense to keep running the same vision algorithms.

$T_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{A} \times \mathcal{Y} \rightarrow [0, 1]$ is the state transition function in the partially observable component of the current state. It is an identity matrix in our case, (we assume) because object attributes do not change during the process of the robot's exploration behaviors.

5.2 Observation function

$O : S \times \mathcal{A} \times \mathcal{Z} \rightarrow [0, 1]$ is the observation function that specifies the probability of observing $z \in \mathcal{Z}$ when action a is executed in state s: $O(s, a, z)$. In this article, the probabilities are learned from performing cross-validation on the robot's training data. As described in Sect. 4, ACAC produces confusion matrix classifiers $\Theta^a_p \in \mathbb{R}^{2\times 2}$ for each attribute p and each action a.

$$O(s, a, z) = \mathbf{Pr}(\mathbf{v}_z \mid \mathbf{v}_s, a) = \prod_{i=0}^{N^p-1} \Theta^a_{p_i}(v_s^{p_i}, v_z^{p_i}) \quad (8)$$

where $\Theta^a_p \in R^{2\times 2}$ is a confusion matrix for attribute p and action a; $\mathbf{v}_s$ and $\mathbf{v}_z$ are the true and observed values of the attributes; $v_s^{p_i}$ (or $v_z^{p_i}$) is the true (or observed) value of the i^{th} attribute; and N^p is the total number of attributes in the query. The robot might fail in exploratory actions. In that case, the robot receives an empty observation, which causes no belief change.

So far, we have specified all components of MORC. Next, we discuss a way of computing high-quality policies for OFFLINE- MEAL that include large numbers of attributes.

5.3 Dynamically-learned controllers

The OFFLINE- MEAL problem can include a prohibitively large number of attributes. One of the datasets in our experiments contains 81 attributes, resulting in 2^{81} possible states in $\mathcal{Y}$. It is computationally intractable to generate a far-sighted policy while considering all the attributes. A strategy called iCORPP was introduced for dynamically constructing minimal (PO)MDPs to model domain attributes for robot planning [61]. Behind iCORPP is a family of algorithms for Integrated commonsense Reasoning and probabilistic Planning (IRP) [62]. Those algorithms aim to decomposing a sequential decision-making problem into two tractable subproblems that focus on high-dimensional reasoning (e.g., objects with many attributes) and long-horizon planning (e.g., tasks that require many actions). In line with iCORPP, we model only those attributes necessary to the current query in MORC, where the goal is to include a relatively small set of attributes in our MOMDPs while maintaining the quality of the generated policies.

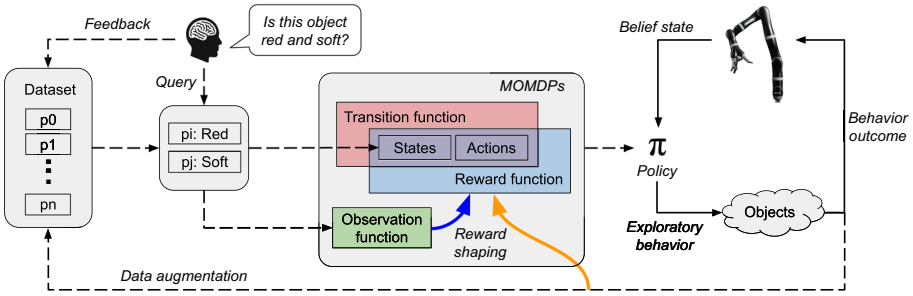


Fig. 6 An overview of the MORC-ITRS algorithm. A human user will choose an object and ask a query such as "*Is this object* RED *and* SOFT*?*". The robot will generate a perception model on the specified attributes, i.e., RED and SOFT. Queried attributes and the corresponding perception model then will be used to construct states and the observation function of the MOMDP model respectively. The reward function will be shaped by the quality of the observation function and the robot's experience. The robot uses the generated MOMDP model to compute a policy π and interacts with the queried object. Newly-perceived feature data will be used to update the robot's experience and augment the dataset. Humans will give feedback to the robot's answer and attach labels to the feature data points

Algorithm 1 MORC

Require: $\mathcal{P}$; $T_{\mathcal{X}}$; $\mathcal{A}^e$; Sol; $R(s,a)$; $\mathcal{D}$
1: Take queried attribute(s) $\mathbf{p}$ from human, where $\mathbf{p} \subseteq \mathcal{P}$
2: Generate $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{A}^r$, $T_{\mathcal{Y}}$ and $\mathcal{Z}$ using $\mathbf{p}$
3: Compute confusion matrix Θ^a_p using $\mathcal{D}$ where $p \in \mathbf{p}$, $a \in \mathcal{A}$
4: Generate $O(s,a,z)$ with Θ^a_p for $p \in \mathbf{p}$ using Eqn. 8
5: Compute policy π using Sol for $(\mathcal{X}, \mathcal{Y}, \mathcal{A}, T_{\mathcal{X}}, T_{\mathcal{Y}}, R, \mathcal{Z}, O, \gamma)$
6: Uniformly initialize belief b
7: **while** Current state s is not *term* **do**
8: Select action a with π based on b, and execute a
9: Make an observation z where $z \in \mathcal{Z}$
10: Update b with z and a using Bayesian rule
11: **end while**

Algorithm 1 shows the complete process of MORC.[4] We dynamically construct MOMDP controllers by specifying the following components in order: 1) State set $\mathcal{Y}$ that includes only the attributes that are mentioned in the query (e.g., BLUE, HEAVY, and BOTTLE, given that a user asks whether an object is *a blue heavy bottle* or not); 2) State set S, the Cartesian product of $\mathcal{X}$ (predefined) and $\mathcal{Y}$; 3) Action set $\mathcal{A}^r$, where each reporting action $a^r \in \mathcal{A}^r$ corresponds to a state in $\mathcal{Y}$; 4) Action set $\mathcal{A}$, union of $\mathcal{A}^e$ (predefined) and $\mathcal{A}^r$; 5) $\mathcal{Z}^h$, object attribute combinations; 6) $\mathcal{Z}$, union of $\mathcal{Z}^h$ and $\emptyset$. The components together form a complete MOMDP that is relatively very small, and typically includes fewer than 100 states at runtime. Our approach enables automatic generation of complete MOMDP models, which can be encoded, as in our experiments, such that existing planning algorithms (e.g., [63]) can be used to generate policies.

The implementation of the transition system of MORC is introduced in Sect. 5.1. Figure 5 also presents how we manually specify transition diagrams for different datasets. The observation function (detailed in Sect. 5.2) depends on classifiers produced by ACAC. When learning those classifiers, MORC assumes that sufficient training data and annotations are available for the robot. However, a large amount of object exploration data is very expensive for robots to collect in the real world. Our solution is to interleave ACAC and attribute identification where the robot interacts with the current object, collects exploration data, and uses the data to improve attribute classifiers for future identifications. This online process is referred to as ONLINE- MEAL, which will be discussed in the next section.

[4] Source code: https://github.com/keke-220/Predicate_Learning

6 An algorithm for ONLINE-MEAL: MORC-ITRS

In this section, we describe MORC with information-theoretic reward shaping (MORC-ITRS), which is a novel algorithm that is built on MORC but focuses on balancing exploration and exploitation in ONLINE- MEAL problems. An overview of MORC-ITRS as applied to ONLINE- MEAL is presented in Fig. 6.

6.1 Information-theoretic reward

We first introduce the shaped information-theoretic reward function in MORC-ITRS. In ONLINE- MEAL problems, the robot needs to optimize its actions toward not only improving the accuracy of attribute identification but also minimizing the cost of exploratory behaviors. We introduce the two factors of *perception quality* and *interaction experience* into the reward design of MOMDPs to achieve the trade-off between exploration (actively collecting data for attribute learning) and exploitation (using the learned attributes for identification tasks).

Let $Ent(s,a)$ be the entropy of the distribution over $\mathcal{Z}$, given s and a, which is used for indicating the perception quality of exploratory behavior a over the y component of s:

$$Ent(s,a) = -\sum_{z_i \in \mathcal{Z}} O(s,a,z_i) \log_2 O(s,a,z_i) \tag{9}$$

where z_i is the i^{th} observation and $O(s, a, z_i)$ is the probability of observing z_i in state s after taking action a. $O(s, a, z_i)$ is computed using the data instances gathered in the ONLINE- MEAL process.

Let $IE(p, a)$ be the interaction experience of applying action a to identify attribute p, which is in the form of:

$$IE(p, a) = \frac{1}{\delta} \cdot | \{f \in f_a, labeled(f, p) = true\} | \quad (10)$$

where f_a is a set of instances that a robot has collected so far, and $labeled(f, p)$ returns *true* if f has been labeled w.r.t. p, where the value v^p is *true* or *false*. δ is a sufficiently large integer to ensure $IE(p, a)$ is in the range of [0,1). A lower value of $IE(p, a)$ reflects a higher need of further exploring (p, a).

Building on the concepts of perception quality and interaction experience, our information-theoretic reward function is defined as follows:

$$R^{IT}(s, a) = R(s, a) + \alpha \cdot Ent(s, a) - \beta \cdot IE(p, a) \quad (11)$$

where α and β are natural numbers and used for adjusting how much perception quality and interaction experience are considered in reward shaping. Informally, when $O(s, a, z)$ is close to being uniform, the perception model of (s, a) is poor, and the value of $Ent(s, a)$ is high. As a result, our new reward function will encourage the robot to take action a by offering extra reward $\alpha \cdot Ent(s, a)$. When the robot is experienced in applying a to identify attribute p, $IE(p, a)$ will be high. In this case, an extra penalty of $\beta \cdot IE(p, a)$ will discourage the robot from taking those well-explored behaviors. In comparison to standard MOMDPs, where reward and observation functions are independently developed, MORC-ITRS enables the reward function to adapt to the changes of the observation function.

6.2 Algorithm description

Algorithm 2 presents MORC-ITRS for active ONLINE- MEAL problems. The inputs of MORC-ITRS include attribute set $\mathcal{P}$, transition function $T_{\mathcal{X}}$, action set $\mathcal{A}^e$, MOMDP solver Sol, parameters α and β, naive reward function $R(s, a)$. MORC-ITRS does not have a termination condition.

MORC-ITRS starts with initializing the interaction experience function with zeros for all (p, a) pairs, and then initializes dataset $\mathcal{D}$ that will be later augmented as the robot interacts with objects (Lines 1 and 2). In each iteration of the main loop (Lines 3-24), MORC-ITRS takes an identification query from people (Line 4), constructs a MOMDP model (Lines 5-10), computes its policy, uses the policy to interact with an object (Lines 13–18), and augments its dataset for improving the MOMDP model in the next iteration (Lines 20-23).

Algorithm 2 MORC-ITRS

Require: $\mathcal{P}$; $T_{\mathcal{X}}$; $\mathcal{A}$; Sol; α; β; $R(s, a)$
1: Initialize $IE(p, a) = 0$ for each action $a \in \mathcal{A}$ and $p \in \mathcal{P}$
2: Initialize online training dataset $\mathcal{D} = \emptyset$
3: **repeat**
4: Take queried attribute(s) **p** from human, where **p** $\subseteq \mathcal{P}$
5: Generate $\mathcal{X}$, $\mathcal{Y}$, $T_{\mathcal{Y}}$, and $\mathcal{Z}$ using **p**
6: Compute confusion matrix Θ_p^a using $\mathcal{D}$ where $p \in$ **p**, $a \in \mathcal{A}$
7: Generate $O(s, a, z)$ with Θ_p^a for $p \in$ **p** using Eqn. 8
8: Compute $Ent(s, a)$ using Eqn. 9
9: Generate $R^{IT}(s, a)$ with $R(s, a)$ using Eqn. 11
10: Compute policy π using Sol for $(\mathcal{X}, \mathcal{Y}, \mathcal{A}, T_{\mathcal{X}}, T_{\mathcal{Y}}, R^{IT}, \mathcal{Z}, O, \gamma)$
11: Initialize action set $\mathcal{A}^{select}$ and feature set $\mathcal{F}$ with $\emptyset$
12: Uniformly initialize belief b
13: **while** Current state s is not $term$ **do**
14: Select action a with π based on b, append a to $\mathcal{A}^{select}$, and execute a
15: Record data instances f_a, and $\mathcal{F} \leftarrow \mathcal{F} \cup \{f_a\}$
16: Make an observation z where $z \in \mathcal{Z}$
17: Update b with z and a using Bayesian rule
18: **end while**
19: Ask human to provide **v** for **p** as label(s) for $\mathcal{F}$
20: **for** each a in $\mathcal{A}^{select}$ **do**
21: Update $\mathcal{D}$ using $\mathcal{F}$ and **v**
22: Update $IE(p, a)$ for $a \in \mathcal{A}$ and $p \in$ **p** using Eqn. 10
23: **end for**
24: **until** end of interactions

In the first inner loop (Lines 13–18), the robot interacts with an object based on the generated policy. π suggests an action at each state b. The robot then executes the action and makes an observation. Based on the action and observation, the robot updates its belief using the Bayesian rule. After selecting each action, MORC-ITRS records this action along with its collected feature data (Lines 14 and 15). In Line 19, we ask people to provide the label y for the collected data. The final step is to iterate over all selected actions to augment $\mathcal{D}$, and calculate the new interaction experience (Lines 20-23).

Intuitively, we aim to encourage the robot to select exploratory behavior $a \in \mathcal{A}^e$ from those behaviors, where the perception model of (s, a) is of poor quality, and there is relatively limited experience of applying a to attribute p, i.e., the experience of (p, a) is limited.

7 Experiments

In this section, we present the experiment setup and experimental results from the evaluation of our MORC and MORC-ITRS algorithms. MORC assumes the availability of training data for learning action-conditioned attribute classifiers. Accordingly, the baselines for evaluating MORC includes:

- *Random*: Actions are randomly selected from both reporting and legal exploration actions. A trial is terminated by any of the reporting actions.

- *Random Legal*: Actions are randomly selected from legal exploration actions. Under an exploration budget, one selects the reporting action corresponding to y with the highest belief. This baseline corresponds to the algorithm for MEAL problems of [8] (we did not use their linguistic component).
- *Predefined*: An action sequence is strictly followed: *ask*, *look*, *press*, *grasp*, *lift*, *lower*, and *drop*. Under an exploration budget or in early terminations caused by illegal actions, the robot selects the reporting action that makes the best sense.
- *Predefined Plus*: The same as Predefined except that unsuccessful actions are repeated until achieving the desired result(s).

MORC-ITRS assumes no prior data, so the robot must learn perception models and perform attribute identification at the same time. We compare MORC-ITRS with the baselines of:

- *Iterative Random Legal*: It is an iterative version of one of the baselines for solving OFFLINE- MEAL problems. Just like Random Legal, the robot considers only the "legal" behaviors (e.g., *lift* is legal only after a successful *grasp* behavior), and then randomly selects one from the legal actions. The only difference is that the robot collects more data after identifications, and uses the data for future tasks. With an exploration budget for each trial (50 s and 80 s for one-attribute trials, i.e., $N^p = 1$, and two-attribute trials, i.e., $N^p = 2$, respectively), the robot is forced to report $y \in \mathcal{Y}$ of the highest belief.
- *Iterative* MORC: It iteratively runs MORC using all data collected so far. This process is repeated after each batch. Iterative MORC enables ONLINE- MEAL actions by *passively* collecting data and training attribute classifiers.

7.1 Experiment setup

Dataset description Three public datasets of **ISPY32** [4], **ROC36** [5], and **CY101** [6] are used in our experiments. CY101 (an updated version of the dataset [10]) contains many more household objects and attributes.

In the ISPY32 dataset, a robot from the Building-Wide Intelligence project [64] explored 32 objects using 8 exploratory behaviors: *look*, *grasp*, *lift*, *hold*, *lower*, *drop*, *push*, and *press* (Fig. 3). The *hold* behavior was performed by holding the object in place. The *look* behavior was performed by taking a visual snapshot of the object using the robot's sensors prior to exploration. Each behavior was performed 5 times on each object in the dataset. Features of VGG, color, SURF, auditory, finger, and haptics were recorded in ISPY32.

In ROC36, the robot explored 36 different objects using 11 prototypical exploratory behaviors: *look*, *grasp*, *lift*, *shake*, *shake-fast*, *lower*, *drop*, *push*, *poke*, *tap*, and *press* 10 different times per object. The objects are lidded containers with the same shape and varied along 3 different attributes: (1) color: RED, GREEN, BLUE; (2) weight: LIGHT, MEDIUM, HEAVY; and (3) contents: BEANS, RICE, GLASS, SCREWS. These variations result in the $3 \times 3 \times 4 = 36$ objects bearing combinations of these attributes in the set $\mathcal{P}$ that the robot is tasked with learning.

For CY101 dataset, an uppertorso humanoid robot with 7-DOF arm explored 101 objects belonging to 20 different categories using 10 exploratory behaviors: *look*, *grasp*, *lift*, *hold*, *shake*, *drop*, *push*, *tap*, *poke*, and *press*. Seven different types of features including auditory, vibrotactile, finger, color, optical flow, SURF, and haptics (i.e., joint forces) were considered in CY101. Each behavior was performed 5 times per object.

Every individual classifier (introduced in Sect. 4) corresponds to an attribute-behavior pair. The exact numbers of the classifiers needed by the robot depend on the datasets that provide different numbers of attributes. For instance, in experiments using the **ROC36** dataset, there were a total of $9 \times 11 = 99$ classifiers.

Action costs and action failures Each exploratory behavior a has a cost (planning and execution) in the range of [0.5, 22.0] that came with the datasets, and is modeled in $R(s, a)$. For instance, the cost of behavior *press* (22.0) is much higher than the cost of behavior *look* (0.5). The costs of behaviors in the three datasets are different because the datasets were collected using different robots. Additionally, action *ask* has the cost of 100.0.[5]

In MORC, the reward of the reporting action was +500.0 (or −500.0) when the robot's attribute identification is correct (or incorrect). In MORC-ITRS, we set the reward to be +300.0 (or −300.0). Most actions are considered unreliable to some degree in our MOMDP model and we uniformly set the failure probability to 0.05 which we did not refine in OFFLINE- or ONLINE- MEAL. For instance, an unsuccessful *drop* behavior models the situation that the object is stuck in the robot's hand. We used an off-the-shelf system for solving MOMDPs [63]. γ is a discount factor and $\gamma = 0.99$ in our case. This setting gives the robot an unspecified, relatively long planning horizon.

We observed different behaviors of the robot given different success bonuses and failure penalties. Intuitively, increasing the success bonus (positive reward) and the failure penalty (negative reward) encourages the robot to spend more time exploring objects for higher success rates, i.e., being risk-averse. We also observed that a very large success bonus frequently produced optimistic, risk-seeking behaviors, such as reporting before taking any exploration behaviors. Such discussions point to the research area of reward engineering,

[5] Action *ask* was used only in the ISPY32 experiments, because other exploration behaviors are not as effective as in ROC36 and CY101.

which is a long-standing challenge to researchers working on planning under uncertainty and reinforcement learning. In this article, the success bonus and failure penalty values are manually specified.

7.2 MORC evaluation

Next, we describe the experiments we conducted to evaluate MORC. We aim to answer the following questions:

- *How does* MORC *perform in efficiency and accuracy*? If MORC is more efficient and more accurate than the baselines, then we can claim MORC's superiority in achieving the objective of solving OFFLINE- MEAL problems.
- *Can we build a "super" model in* MORC? A super model includes all potential attributes into the sequential decision maker. In comparison, MORC dynamically constructs query-dependent MOMDPs.

The training process of MORC follows the principle of "leave one object out." In other words, an object was randomly selected, and all data instances corresponding to the particular object were excluded in training the classifiers. The excluded object was then used for evaluating the classifiers' performance. We iterated over all the objects in the experiments for evaluating MORC in solving OFFLINE- MEAL problems. Specifically, for dataset **ISPY32** which includes 32 objects, each classifier was trained using $31 \times 5 = 155$ data samples, where one object was excluded, and each behavior was repeated five times. For dataset **ROC36**, there were $35 \times 10 = 350$ data instances used for training each classifier.

7.2.1 Illustrative trial of MORC

We now describe an example in which a robot works on an OFFLINE- MEAL task. We randomly selected an object from the ISPY32 dataset: a blue and red bottle full of water. We then randomly selected attributes, in this case YELLOW and METALLIC, and asked the robot to identify whether the object has each of the attributes or not. The selected object was not part of the robot's training set used to learn the attribute classifiers and the MOMDP observation model. The robot should report negative to both attributes while minimizing the overall cost of exploration behaviors.

Given this user query, the state space of MORC includes 25 states. We then generate an action policy using past work's methods [63]. Currently, building the model takes almost no time, and we uniformly gave five seconds for policy generation using the model (same in all experiments). The time for computing the policy is insignificant relative to the time for exploratory behaviors (which is what we are really trying to minimize).

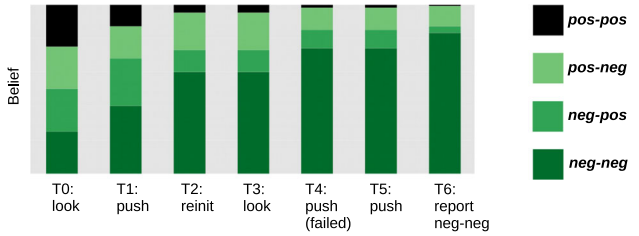


Fig. 7 Action selection and belief change in the exploration of a red and blue bottle full of water using MORC, given a query of "is this object YELLOW and METALLIC?"

Figure 7 shows the belief change in this process. The initial distributions over $\mathcal{X}$ and $\mathcal{Y}$ are $[1.0, 0.0, \cdots]$ and $[0.25, 0.25, 0.25, 0.25]$ respectively. The policy suggests to *look* first. We queried the dataset to make an observation, *neg-neg* in this case. The belief over $\mathcal{Y}$ is updated based on this observation: $[0.41, 0.28, 0.19, 0.13]$, where the entries represent *neg-neg*, *neg-pos*, *pos-neg*, and *pos-pos* respectively. There is a (fully observable) state transition in $\mathcal{X}$, from x_0 to x_1, so the belief over $\mathcal{X}$ becomes $[0.0, 1.0, 0.0, \cdots]$. Based on the updated beliefs, the policy suggests taking the *push* behavior, which results in another *neg-neg* observation. Accordingly, the belief over $\mathcal{Y}$ is updated to $[0.60, 0.13, 0.22, 0.05]$, which indicates that the robot is more confident that the object is neither YELLOW nor METALLIC. After behaviors of *reinitialize*, *look*, *push*, and *push* (this first *push* behavior was unsuccessful, and produced the $\emptyset$ observation), the belief over $\mathcal{Y}$ becomes $[0.84, 0.04, 0.12, 0.01]$. The policy finally suggests reporting *neg-neg*, making it a successful trial with an overall cost of 167 seconds, which results in an overall reward of $500 - 167 = 333$ (an incorrect report would have resulted in -667 reward).

Remarks: It should be noted that the classifiers associated with each action and word will produce an output even in cases where the sensory signals from that action are irrelevant to the word. For instance, although the sensory signals relevant to *push* are haptics and audio, the first *push* behavior results in an observation of YELLOW. It was "YELLOW:neg", because most objects in the prior training set are not yellow. The robot favors behaviors that distinguish "easy" attributes (*look* distinguishes YELLOW well in this case). If a behavior is useful, the robot will prefer taking it early. The more the behavior is delayed, the more the expected reward is discounted (we use a discount factor of 0.99 in our experiments).

7.2.2 Results of applying MORC to OFFLINE-MEAL problems

How does MORC perform in efficiency and accuracy on ROC36? In each trial, we place an object that has three attributes (color, weight, and content) on a table and then

Table 3 Performances of MORC and two baseline planners in cost and accuracy on the ROC36 dataset. Numbers in parentheses denote the Standard Deviations over 400 trials

N^p	Method	Overall cost (std)	Accuracy
2	Random	17.56 (30.00)	0.245
	Predefined Plus	37.10 (0.00)	0.583
	MORC (Ours)	29.85 (12.87)	0.860
3	Random	10.12 (21.77)	0.130
	Predefined Plus	37.10 (0.00)	0.373
	MORC (Ours)	33.87 (8.78)	0.903

generate an object description that includes the values of two or three attributes. This description matches the object in only half of the trials. When two (or three) attributes are queried, $\mathcal{Y}$ includes four (or eight) states plus the *term* state, resulting in $\mathcal{S}$ that includes 25 (or 49) states. The other components of MORC grow accordingly, given an increasing number of queried attributes.

Experimental results are reported in Table 3. Not surprisingly, randomly selecting actions produces low accuracy. The overall cost is smaller in more challenging trials (all three attributes are questioned), because in these trials there are relatively fewer exploratory behaviors (more attributes produce more reporting actions), making the agent more likely to take a reporting action. MORC reduces the overall action cost while significantly improving the reporting accuracy. Our performance improvement is achieved by repeating actions as needed, selecting legal actions (e.g., *lift* is legal only if the current state is x_2) that produce the most information or have the potential of doing so in the future, and even arbitrarily reporting without "wasting" exploratory behaviors given queries where the exploratory behaviors are not effective.

How does MORC perform in efficiency and accuracy on ISPY32? In this set of experiments, a user query is specified by randomly selecting one object and N^p attributes ($1 \leq N^p \leq 3$), on which the robot is questioned. Each data point is an average of 200 trials, where we conducted pairwise comparisons over the five strategies, i.e., the strategies were evaluated using the same set of user queries. A trial is successful only if the robot reports correctly on all attributes. It should be noted that most of the contexts are misleading in this dataset due to the large number of object attributes, so more exploratory behaviors confuse the robot if the behaviors are not carefully selected.

Figure 8 shows the experimental results. The overall reward is computed by subtracting the overall action cost from the reward yielded by the reporting action (either a big bonus or a big penalty). We do not compute standard deviations in this dataset, because the diversity of the tasks results in problems of very different difficulties. We can see MORC consistently performs the best in terms of the overall reward and overall accuracy. When more attributes are queried, MORC enables the robot to take more exploratory behaviors (Middle subfigure), whereas the baselines could not adjust their question-asking strategy accordingly.

Can we build a "super" model in MORC? The last experiment aims to evaluate the need for dynamically constructed controllers, answering the question "*Can we build a 'super' controller that models all attributes?*" We constructed MOMDP controllers including two relevant and an increasing number of irrelevant attributes (i.e., the ones that are not queried). Our dynamically learned controllers include only the relevant attributes and correspond to the curves' left ends. Results are shown in Fig. 9. We can see, the quality of the generated action policies decreases soon, e.g., from > 150 to < 25 in reward, when more irrelevant attributes are included in MORC. The right two subfigures show that MORC first tries to achieve higher accuracy by taking more exploration behaviors and then "gives up" due to the growing number of irrelevant attributes. The results show the infeasibility of "super" controllers in MORC that model all attributes and justify the need for dynamic controllers.

7.3 MORC-ITRS evaluation

Regarding the evaluation of MORC-ITRS for ONLINE- MEAL problems, we aim to answer the following questions:

- *How does* MORC-ITRS *perform in efficiency and accuracy?*

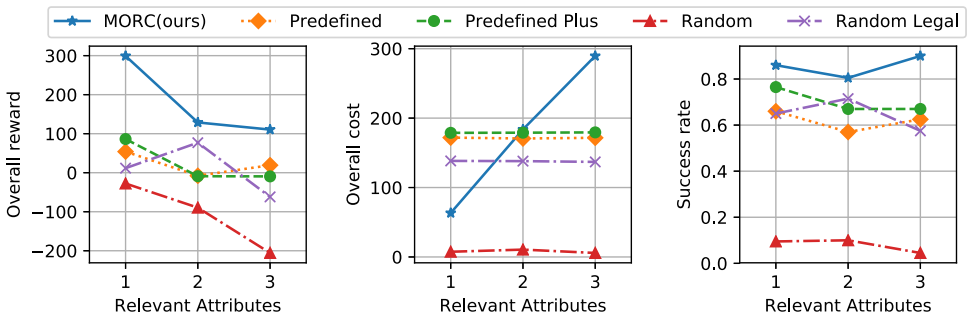


Fig. 8 Evaluations of five action strategies (including MORC) on the ISPY32 dataset. Comparisons are made in three categories of *overall reward* (Left), *overall exploration cost* (Middle), and *success rate* (Right)

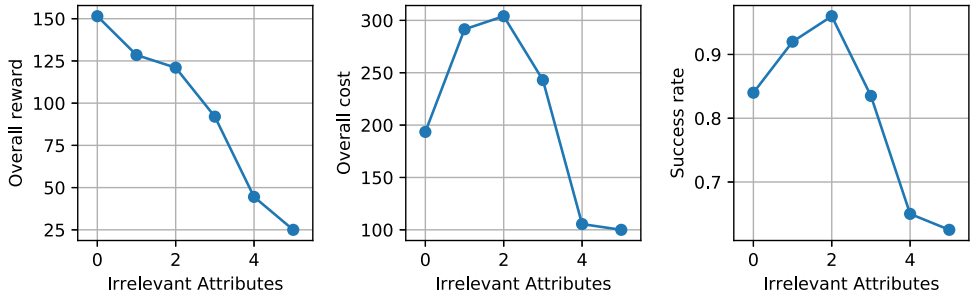


Fig. 9 A "super" MORC framework that models two relevant attributes, and an increasing number of irrelevant attributes (x-axis). Our dynamically learned controllers correspond to the left end of each curve, and model only the relevant attributes. The three subfigures correspond to three different dimensions for evaluation: *overall reward* (Left), *overall exploration cost* (Middle), and *success rate* (Right)

- *Does* MORC-ITRS *outperform baselines for individual attribute?*
- *How sensitive is* MORC-ITRS *to the parameters?*

Attributes: In order to select attributes that are learnable given the robot's exploratory behaviors, evaluations of all attributes in the two datasets were performed prior to the experiments. We set $|\ \mathcal{P}\ |$ to 10 and picked the attributes that have enough positive examples for training and those are most learnable.

Queries: At the beginning of each trial, N^p was either 1 or 2. At the end of each trial, the robot is told if the identification was correct. In the case of $N^p = 1$, the robot could learn the attribute's ground-truth value from the human's feedback. In the case of $N^p = 2$, the robot could do so, only if the 2D identification was correct.

Batch-based Learning: In both datasets, we randomly split the objects into three subsets of equal sizes. The subsets are used for pretraining (Obj^{pre}), training (Obj^{train}), and testing (Obj^{test}) respectively. In the pretraining phase, the robot started with a handcrafted policy where each action is forced to be applied on the queried object once. We collected feature instances with labels from those interactions and built a pretraining dataset $\mathcal{D}^{pre}$ that represents the robot's prior knowledge.

In principle, we do not need a pretraining dataset in MORC-ITRS. In practice, however, without a small amount of data for "warm up," the robot might take a large number of interactions with objects for exploration in order to identify object attributes and learn meaningful observation models. This number is particularly large at the early learning phase. A practical challenge is that the datasets used in this article can provide only a limited number of samples for each attribute-behavior pair. As a result, we would have to reuse the same samples from the dataset when the robot performs the same actions more than N times ($N = 5$ in our case), which is detrimental to the quality of the experiments. To alleviate this practical issue, we provide a small amount of data for pretraining, though the ONLINE- MEAL algorithm does not require that.

Table 4 Early and late observation models for behavior *press*

	Early phase		Late phase	
	Not soft (Observed)	Soft (Observed)	Not soft (Observed)	Soft (Observed)
Not soft (Ground truth)	0.68	0.32	0.82	0.17
Soft (Ground truth)	0.50	0.50	0.20	0.80

7.3.1 Illustrative trials of MORC-ITRS

From the robot's many trials of the learning experience, we selected two trials (T_1 and T_2), where the robot faced the same object (a Coke can that has attributes METAL, EMPTY, and CONTAINER) and needed to answer the same question "*Is this object* SOFT?" From the dataset, we know that the correct answer should be "no" (the robot did not know it). T_1 appeared at the second batch of training, and T_2 appeared at the ninth. We present both trials and explain how the robot performed better in T_2.

In T_1 (early learning phase), the robot first performed the *look* behavior. Then, the robot had the following options: *grasp*, *tap*, *push*, *poke* and *press* according to Fig. 5. Specifically, for *press*, the confusion matrix $\Theta^{press}_{\text{SOFT}}$ (shown in Table 4) was nearly uniform, which is typical in the early learning phase. Among those "less useful" behaviors, the robot chose *grasp*. The distribution over $\mathcal{Y}$ was changed from [0.37, 0.63] to [0.46, 0.54], where the entries represent "not soft" and "soft" respectively. After *press*, MORC-ITRS sequentially suggested *grasp*, *lift*, *hold* and *hold*. Finally, the robot reported *pos* that resulted in a failed trial with a total cost of 55.5 s.

In T_2 (late learning phase), behavior *press* became more useful for identifying attribute SOFT compared to T_1, as shown in Table 4. For *grasp*, $IE(\text{SOFT}, grasp) = 0.67$ and $\Theta^{grasp}_{\text{SOFT}}$ was [0.66, 0,33, 0.61, 0.38] (TN, FN, FP, TP), which meant that the robot was experienced with behavior *grasp* and

considered *grasp* was not as useful as *press*. Accordingly, MORC-ITRS suggested *press* instead of *grasp* after taking *look*. The belief over $\mathcal{Y}$ changed from [0.57, 0.43] to [0.67, 0.33]. After only *look* and *press*, the robot was able to quickly report *neg*, resulting in a successful trial with a total cost of 22.5 s.

From the above two trials (same query and object in different learning phases), we see how the improved perception model of (*press*, SOFT) helped the robot correctly identify SOFT with a low cost.

7.3.2 Results of applying MORC-ITRS to ONLINE-MEAL problems

How does MORC-ITRS perform in efficiency and accuracy? Fig. 10 shows the learning curve for one-attribute and two-attribute identification queries evaluated on the three datasets, where we conducted experiments over three different strategies (two baselines and MORC-ITRS). x-axis is the accumulative cost of all trials at the training phase. Since the cost is determined by the time required to complete each action, we can regard the x-axis as training time. y-axis reflects the identification accuracy at the testing phase. The proposed method consistently performs better in task completion rate along the whole training process and achieves higher accuracy than baselines.

Although we provided the same pretraining data, three curves in the two subfigures (for each of the three datasets) do not start from the same point. That is because pretraining data only affects the observation model for the robot, but it is not directly related to the policy for attribute identification. Three strategies have the same observation model but they use different methods to select exploratory behaviors. As a result, the task-completion accuracy is not the same for them at the starting point. MORC-ITRS assigns extra rewards for exploration at the very beginning of the training phase. It resulted in not only a bigger cost but also a higher accuracy.
Does MORC-ITRS outperform baselines for individual attribute? At an exploration cost budget of 2 h, we further evaluated the performance of each individual attribute on CY101 using the three strategies we mentioned, as shown in Fig. 11, where 10 attributes are ranked by the identification accuracy of our method, i.e. MORC-ITRS. The robot has a higher identification accuracy for most of the attributes using MORC-ITRS, while the Iterative Random Legal baseline produces a relatively weak result compared to the other two strategies. Attributes such as PLASTIC, HARD, and EMPTY are more difficult to learn since the accuracy is no more than 80% for all three methods. And attributes such as BLUE, FULL and CONTAINER are easier, where Iterative MORC and MORC-ITRS both offer pretty good results.
How sensitive is MORC-ITRS to the α and β parameters? In Eq. 11, we have two parameters α and β. We conducted experiments on CY101 with different α and β combinations, as shown in Fig. 12. One observation is that the selections of α and β affect the performance of MORC-ITRS. A small α value leads to a higher identification accuracy in the beginning, but the accuracy does not improve much when it reaches the middle or late learning phase. A lower β value encourages the robot to explore no matter whether it is experienced or not, while a higher β value affects the robot to compute the optimal policy. Thus, when β is within the middle range, the robot has the best identification performance. We leave the auto-learning of the parameters to future work. Another observation is that the overall accuracy becomes higher in the middle (Middle) and late (Right) learning phases than early (Left) learning phase, which is expected and verifies the robustness of MORC-ITRS to α and β selections.

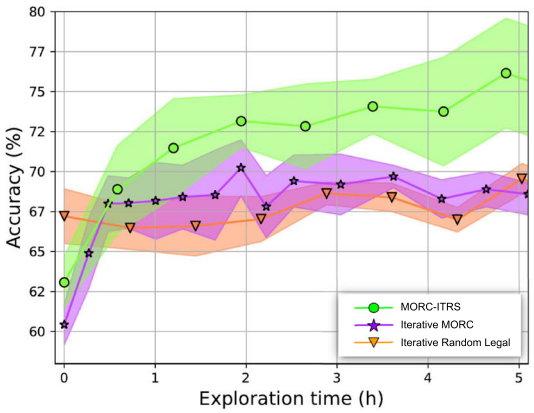
7.4 Real robot demonstration of MORC-ITRS

We have demonstrated the learned action policy using a real robot (UR5e arm from Universal Robots). It should be noted that the three datasets we used in this research were collected on robots that are different from the robot in the demonstration. It is a major challenge in robotics of transferring skills learned from one robot to another. To alleviate the effect caused by the heterogeneity of robot platforms, after performing each action, we sampled a data instance from CY101 according to $x \in \mathcal{X}$, the fully observable component of the current state.

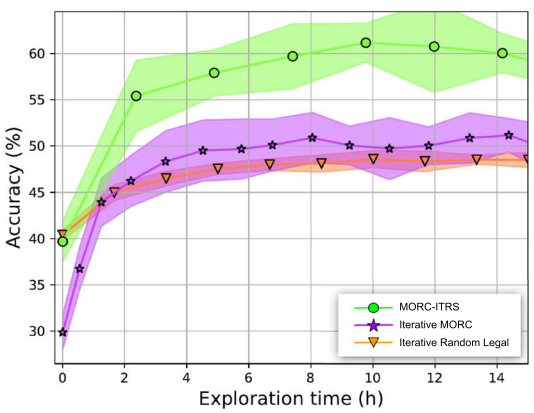
In the demonstration trial, our robot was given an object—a pill bottle half-full of beans. The one-attribute query was "*Is this object* EMPTY?" The robot performed a sequence of exploratory behaviors, as shown in Table 5, where we also listed the observation and the belief after each behavior. For instance, a *pos* observation means that the robot perceives that the object is EMPTY. Figure 13 shows a sequence of screenshots of the UR5e robot completing the task using a learned action policy.

Note that at step 3 in the demonstrated trial, lift was not very useful for identifying whether the object was empty or not. That was because the training set contained objects that were of various weights causing a single *lift* action that could not distinguish between attributes such as EMPTY and LIGHT- WEIGHTED. In contrast, the *shake* actions produced a distinctive sound that seemed to be beneficial for leading the robot to a correct identification.

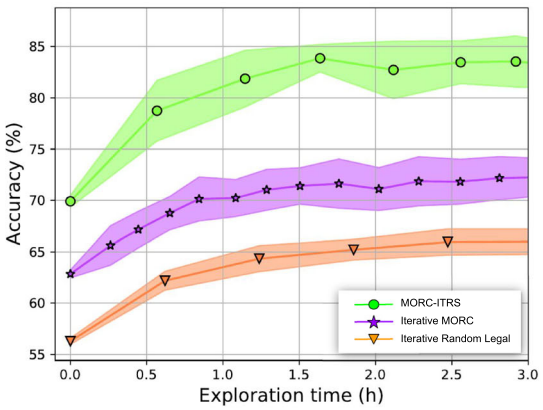
There were drastic changes in the belief after the two shake actions in Steps 5 and 6 in Table 5. At step 5, shake was executed successfully and led the system to the next state. However, there was uncertainty in the perceived sensory data which caused inaccurate outputs from the classifier and undesired belief changes from the robot. Intuitively, how the belief evolves over time depends on how the robot trusts its actions on different attributes. When the robot believes an action is useful for detecting an attribute, this action might cause a



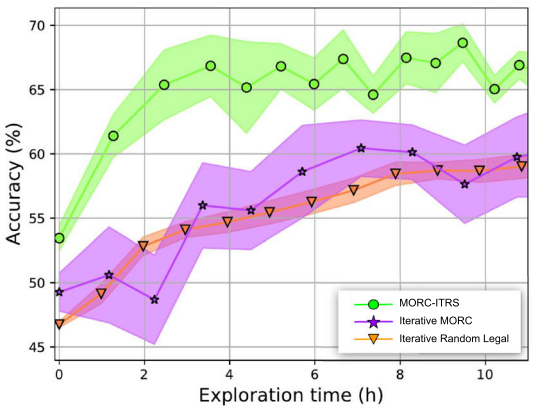
(a) One-attribute queries on **ISPY32**.



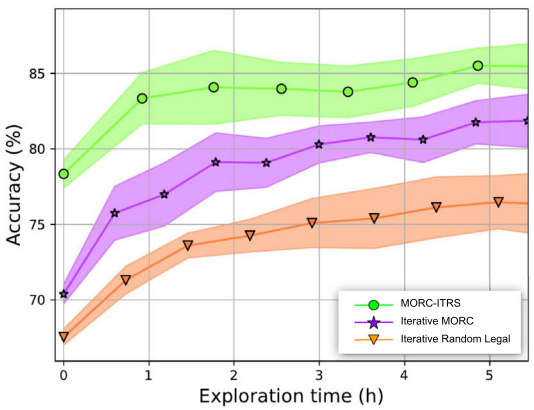
(b) Two-attribute queries on **ISPY32**.



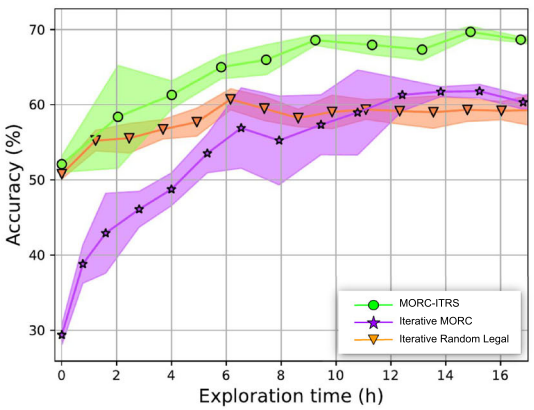
(c) One-attribute queries on **ROC36**.



(d) Two-attribute queries on **ROC36**.



(e) One-attribute queries on **CY101**.



(f) Two-attribute queries on **CY101**.

Fig. 10 Time length of conducting exploratory actions in hours, and identification accuracy of ONLINE- MEAL tasks, where we compared MORC-ITRS (ours) to two baseline strategies including *Iterative Random Legal*, and *Iterative* MORC

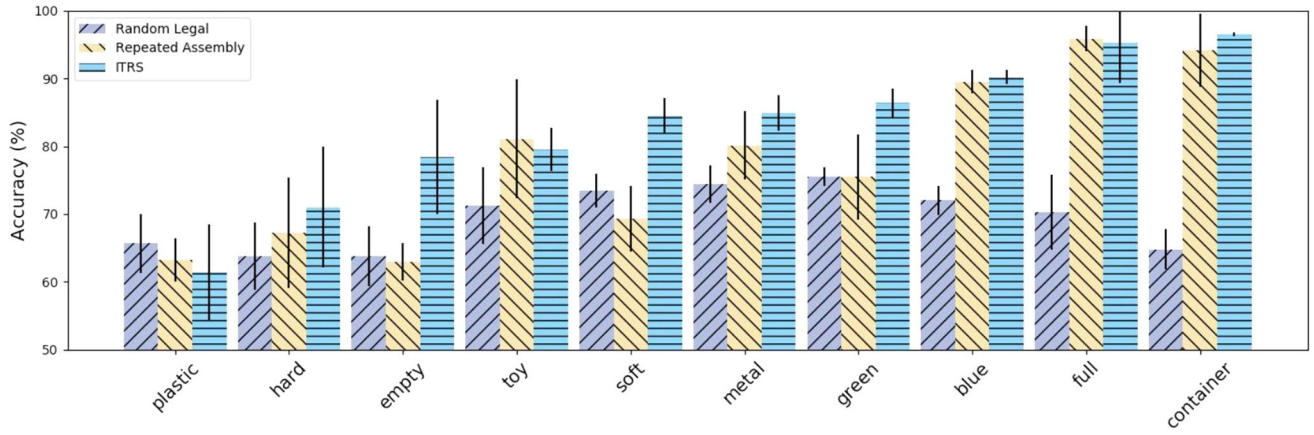
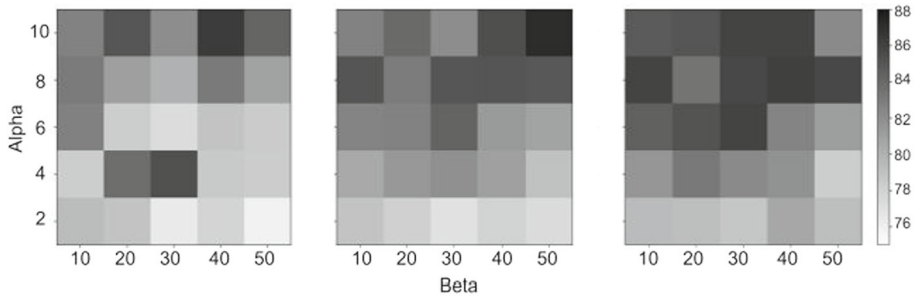


Fig. 11 Accuracy of attribute identification tasks. The attributes (x-axis) are ranked based on MORC-ITRS' performance. MORC-ITRS performed the best on seven out of the ten attributes



(a) Early (b) Middle (c) Late

Fig. 12 We empirically evaluated the identification accuracies of MORC-ITRS in early (**a**), middle (**b**), and late (**c**) learning phase using different values of α and β, which are two parameters of our reward shaping approach (Eq. 11)

Table 5 Behaviors, observations, and belief updates in the demonstration trial of MORC-ITRS

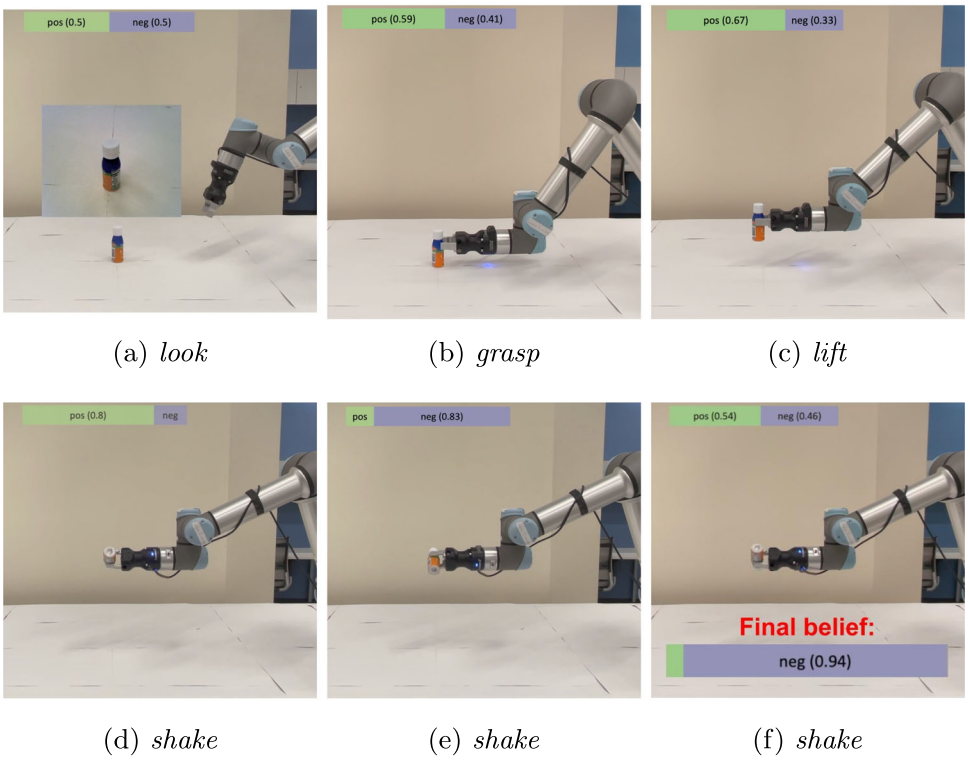
Step	Behavior	Observation	Belief (Initial belief: [0.5, 0.5])
1	*look*	*pos*	[0.41, 0.59]
2	*grasp*	*pos*	[0.33, 0.67]
3	*lift*	*pos*	[0.20, 0.80]
4	*shake*	*neg*	[0.83, 0.17]
5	*shake*	*pos*	[0.46, 0.54]
6	*shake*	*neg*	[0.94, 0.06]

drastic change in its belief; otherwise, the beliefs before and after the action might look similar. In this example, the robot believed *shake* is useful for detecting EMPTY, so the belief updates were significant after each of the three consecutive *shake* behaviors.

8 Conclusion and future work

In this article, we introduce two Multimodal Embodied Attribute Learning (MEAL) problems that both require a robot to compute a policy of leveraging multimodal exploratory behaviors to identify object attributes. In OFFLINE- MEAL problems, the robot is provided data for learning action-conditioned attribute classifiers, whereas the robot does not have such data in ONLINE- MEAL domains. Accordingly, we have developed two algorithms called mixed observability robot control (MORC) and MORC with information-theoretic reward shaping (MORC-ITRS) for addressing OFFLINE- and ONLINE- MEAL problems respectively.

MORC uses mixed observability Markov decision processes (MOMDPs) to solve OFFLINE- MEAL problems, where a robot selects actions for multimodal perception in object exploration tasks. Our approach can dynamically construct a MOMDP model given an object description from a human user, compute a high-quality policy for this model, and use the policy to guide robot behaviors (such as *look* and *shake*) toward maximizing information gain. The dynamically built models in MORC enable the robot to focus on a minimum set of domain variables that are relevant to the current object and query. Attribute classifiers in MORC are learned using existing datasets collected with robots interacting with objects in the real world. Experimental results show that MORC enables the robot to identify object attributes more accurately without introducing extra cost from exploratory behaviors compared to a baseline that suggests actions following a predefined action sequence.



(a) *look* (b) *grasp* (c) *lift*

(d) *shake* (e) *shake* (f) *shake*

Fig. 13 A demonstration of the learned action policy. The robot performed six actions in a row. In the beginning, the robot started with a uniform distribution (it evenly believed the object can be EMPTY or not). After completing the six actions, the belief converged to "negative" (0.94 probability). Finally, the robot selected a reporting action to report that the object is not EMPTY"

MORC-ITRS selects exploratory behaviors toward simultaneous attribute classification and attribute identification. This algorithm is built on MORC, and provoides an information-theoretic reward function for the exploration-exploitation trade off in ONLINE- MEAL problems. The proposed method and baseline methods are evaluated using three real-world datasets. Experimental results show that MORC-ITRS enables the robot to complete attribute identification tasks at a higher accuracy using the same amount of training time compared to baselines.

This research primarily focuses on a robot exploring objects in a tabletop scenario. For future work, one interesting direction will be applying this approach to tasks that involve mobile robot platforms, where exploration would require navigation actions and perceptual modalities such as human-robot dialog. In this article, we empirically evaluated the performance of both MORC and MORC-ITRS, however there is room to improve the evaluation through formal analysis. One common limitation of the two algorithms is that the attribute classifiers are learned by a single robot and cannot directly be used by another robot that has different behaviors, morphology, and sensory modalities. It may be possible to use sensorimotor transfer learning (e.g., [65,66,69]) in future work to scale up our framework to allow multiple different robots to learn such models and share their knowledge to further speed up learning. In addition, considering correlations between attributes and handling fuzzy attributes can potentially improve the performance of ONLINE- MEAL. Handling unseen attributes could be another interesting focus. Another direction for the future is to learn the world dynamics through the task completion process (currently the transition function is provided and the observation function is learned), where reinforcement learning methods potentially can be used. It is also important to consider human-robot dialogue to acquire attribute labels for objects in MEAL problems.

Finally, we would like to explore the possibility of formulating ONLINE- MEAL as Bayes-Adaptive POMDPs (BAPOMDP) [67] where observation probabilities (functions) can be considered as unobserved parameters in the state space over which we maintain beliefs. During the learning process, the robot will continually gather data for approximating the BAPOMDP model while maintaining attribute identification performance guarantees. Fundamentally, a BAPOMDP model enables sequencing actions with the optimal trade-off between exploration and exploitation, which is exactly the underlying challenge of ONLINE- MEAL. There can be practical challenges in applying BAPOMDP to our ONLINE- MEAL problems, such as the lack of systems for learning BAPOMDP policies and the necessity for more training data. BAPOMDP assumes the reward function is known and stationary. In our current formulation of ONLINE-MEAL, however, the reward function dynamically changes based on the learned observation function. Thus, a principled solution would require a new version of POMDP (and corresponding algorithms and systems), where both reward and observation functions are learned over time. Developing

such a general-purpose framework would add a strong theoretical contribution to the literature, though in this article we chose to develop solutions (MORC and MORC-ITRS) focusing on the specific MEAL problems. There can be interesting future research to answer those questions.

Funding AIR research is supported in part by the National Science Foundation (NRI-1925044), Ford Motor Company, OPPO, and SUNY Research Foundation. MuLIP lab research is supported in part by the National Science Foundation (IIS-2132887, IIS-2119174), DARPA (W911NF-20-2-0006), the Air Force Research Laboratory (FA8750-22-C-0501), Amazon Robotics, and the Verizon Foundation. GLAMOR research is supported in part by the Laboratory for Analytic Sciences (LAS), the Army Research Laboratory (ARL, W911NF-17-S-0003), and the Amazon AWS Public Sector Cloud Credit for Research Program. LARG research is supported in part by the National Science Foundation (CPS-1739964, IIS-1724157, FAIN-2019844), the Office of Naval Research (N00014-18-2243), Army Research Office (W911NF-19-2-0333), DARPA, General Motors, Bosch, and Good Systems, a research grand challenge at the University of Texas at Austin.

Declarations

Conflict of Interest This work has taken place in the Autonomous Intelligent Robotics (AIR) group at The State University of New York at Binghamton, the Multimodal Learning, Interaction, and Perception (MuLIP) laboratory at Tufts University, the Grounding Language in Actions, Multimodal Observations, and Robots (GLAMOR) lab at The University of Southern California, and the Learning Agents Research Group (LARG) at the Artificial Intelligence Laboratory, The University of Texas at Austin. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research. The views and conclusions contained in this document are those of the authors alone.

References

1. Puterman, M. L. (2014). *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons.
2. Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence, 101*(1–2), 99–134.
3. Ong, S. C., Png, S. W., Hsu, D., & Lee, W. S. (2010). Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research, 29*(8), 1053–1068.
4. Thomason, J., Sinapov, J., Svetlik, M., Stone, P., & Mooney, R. J. (2016). Learning multi-modal grounded linguistic semantics by playing "I Spy". In: *IJCAI* (pp. 3477–3483).
5. Sinapov, J., Schenck, C., & Stoytchev, A. (2014). Learning relational object categories using behavioral exploration and multimodal perception. In: *2014 IEEE international conference on robotics and automation (ICRA)* (pp. 5691–5698). IEEE.
6. Tatiya, G., & Sinapov, J. (2019). Deep multi-sensory object category recognition using interactive behavioral exploration. In: *2019 international conference on robotics and automation (ICRA)* (pp. 7872–7878). IEEE.
7. Thomason, J., Padmakumar, A., Sinapov, J., Hart, J., Stone, P., & Mooney, R.J. (2017). Opportunistic active learning for grounding natural language descriptions. In: *Conference on robot learning* (pp. 67–76). PMLR.
8. Thomason, J., Sinapov, J., Mooney, R., & Stone, P. (2018). Guiding exploratory behaviors for multi-modal grounding of linguistic descriptions. In: *Proceedings of the AAAI conference on artificial intelligence* (vol. 32).
9. Sinapov, J., & Stoytchev, A. (2013). Grounded object individuation by a humanoid robot. In: *2013 IEEE international conference on robotics and automation* (pp. 4981–4988). IEEE.
10. Sinapov, J., Schenck, C., Staley, K., Sukhoy, V., & Stoytchev, A. (2014). Grounding semantic categories in behavioral interactions: Experiments with 100 objects. *Robotics and Autonomous Systems, 62*(5), 632–645.
11. Chen, X., Hosseini, R., Panetta, K., & Sinapov, J. (2021). A framework for multisensory foresight for embodied agents. In: *2021 IEEE international conference on robotics and automation*. IEEE.
12. Amiri, S., Wei, S., Zhang, S., Sinapov, J., Thomason, J., & Stone, P. (2018). Multi-modal predicate identification using dynamically learned robot controllers. In: *Proceedings of the 27th international joint conference on artificial intelligence (IJCAI-18)*.
13. Zhang, X., Sinapov, J., & Zhang, S. (2021). Planning multimodal exploratory actions for online robot attribute learning. In: *Robotics: Science and Systems (RSS)*.
14. Russakovsky, O., & Fei-Fei, L. (2010). Attribute learning in large-scale datasets. In: *European conference on computer vision* (pp. 1–14). Springer.
15. Chen, S., & Grauman, K. (2018). Compare and contrast: Learning prominent visual differences. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1267–1276).
16. Ferrari, V., & Zisserman, A. (2007). Learning visual attributes. *Advances in Neural Information Processing Systems, 20*, 433–440.
17. Farhadi, A., Endres, I., Hoiem, D., & Forsyth, D. (2009). Describing objects by their attributes. In: *2009 IEEE conference on computer vision and pattern recognition* (pp. 1778–1785). IEEE.
18. Lampert, C. H., Nickisch, H., & Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In: *2009 IEEE conference on computer vision and pattern recognition* (pp. 951–958). IEEE.
19. Jayaraman, D., & Grauman, K. (2014). Zero shot recognition with unreliable attributes. In: *Advances in neural information processing systems*.
20. Al-Halah, Z., Tapaswi, M., & Stiefelhagen, R. (2016). Recovering the missing link: Predicting class-attribute associations for unsupervised zero-shot learning. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5975–5984).
21. Ren, M., Triantafillou, E., Wang, K. -C., Lucas, J., Snell, J., Pitkow, X., Tolias, A. S., & Zemel, R. (2020). Flexible few-shot learning with contextual similarity. In: *4th Workshop on Meta-Learning at NeurIPS*.
22. Parikh, D., & Grauman, K. (2011). Relative attributes. In: *2011 International conference on computer vision* (pp. 503–510). IEEE.
23. Patterson, G., & Hays, J. (2016). Coco attributes: Attributes for people, animals, and objects. In: *European conference on computer vision* (pp. 85–100). Springer.
24. Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., et al. (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision, 123*(1), 32–73.
25. Pham, K., Kafle, K., Lin, Z., Ding, Z., Cohen, S., Tran, Q., & Shrivastava, A. (2021). Learning to predict visual attributes in the wild. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 13018–13028).
26. Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena, 42*(1–3), 335–346.
27. Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., & Vanhoucke, V., et al.

(2018). Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In: *Conference on Robot Learning*.
28. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., & Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: *2017 IEEE international conference on robotics and automation (ICRA)* (pp. 3357–3364). IEEE.
29. Tellex, S., Gopalan, N., Kress-Gazit, H., & Matuszek, C. (2020). Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems, 3*, 25–55.
30. Dahiya, R. S., Metta, G., Valle, M., & Sandini, G. (2009). Tactile sensing-from humans to humanoids. *IEEE Transactions on Robotics, 26*(1), 1–20.
31. Li, Q., Kroemer, O., Su, Z., Veiga, F. F., Kaboli, M., & Ritter, H. J. (2020). A review of tactile information: Perception and action through touch. *IEEE Transactions on Robotics, 36*(6), 1619–1634.
32. Monroy, J., Ruiz-Sarmiento, J.-R., Moreno, F.-A., Melendez-Fernandez, F., Galindo, C., & Gonzalez-Jimenez, J. (2018). A semantic-based gas source localization with a mobile robot combining vision and chemical sensing. *Sensors, 18*(12), 4174.
33. Ciui, B., Martin, A., Mishra, R. K., Nakagawa, T., Dawkins, T. J., Lyu, M., Cristea, C., Sandulescu, R., & Wang, J. (2018). Chemical sensing at the robot fingertips: Toward automated taste discrimination in food samples. *ACS sensors, 3*(11), 2375–2384.
34. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems, 25*, 1097–1105.
35. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).
36. Devlin, J., Chang, M. -W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. In: *The 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.
37. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., & Askell, A., et al. (2020). Language models are few-shot learners. In: *Advances in Neural Information Processing Systems*.
38. Gibson, E. J. (1988). Exploratory behavior in the development of perceiving, acting, and the acquiring of knowledge. *Annual review of psychology, 39*(1), 1–42.
39. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., & Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In: International conference on machine learning.
40. Lynott, D., & Connell, L. (2009). Modality exclusivity norms for 423 object properties. *Behavior Research Methods, 41*(2), 558–564.
41. Bohg, J., Hausman, K., Sankaran, B., Brock, O., Kragic, D., Schaal, S., & Sukhatme, G. S. (2017). Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics, 33*(6), 1273–1291.
42. Gao, Y., Hendricks, L. A., Kuchenbecker, K. J., & Darrell, T. (2016). Deep learning for tactile understanding from visual and haptic data. In: *2016 IEEE international conference on robotics and automation (ICRA)* (pp. 536–543). IEEE
43. Kerzel, M., Strahl, E., Gaede, C., Gasanov, E., & Wermter, S. (2019). Neuro-robotic haptic object classification by active exploration on a novel dataset. In: *2019 International joint conference on neural networks (IJCNN)* (pp. 1–8). IEEE.
44. Gandhi, D., Gupta, A., & Pinto, L. (2020). Swoosh! rattle! thump!–actions that sound. In: *Robotics: Science and Systems (RSS)*.
45. Braud, R., Giagkos, A., Shaw, P., Lee, M., & Shen, Q. (2020). Robot multi-modal object perception and recognition: synthetic maturation of sensorimotor learning in embodied systems. *IEEE Transactions on Cognitive and Developmental Systems, 13*(2), 416–428.
46. Arkin, J., Park, D., Roy, S., Walter, M. R., Roy, N., Howard, T. M., & Paul, R. (2020). Multimodal estimation and communication of latent semantic knowledge for robust execution of robot instructions. *The International Journal of Robotics Research, 39*(10–11), 1279–1304.
47. Lee, M. A., Zhu, Y., Srinivasan, K., Shah, P., Savarese, S., Fei-Fei, L., Garg, A., & Bohg, J. (2019). Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks. In: *2019 International conference on robotics and automation (ICRA)* (pp. 8943–8950). IEEE.
48. Wang, C., Wang, S., Romero, B., Veiga, F., & Adelson, E. (2020). Swingbot: Learning physical features from in-hand tactile exploration for dynamic swing-up manipulation. In: *IEEE/RSJ International conference on intelligent robots and systems* (pp. 5633–5640).
49. Fishel, J. A., & Loeb, G. E. (2012). Bayesian exploration for intelligent identification of textures. *Frontiers in neurorobotics, 6*, 4.
50. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
51. Platt Jr, R., Tedrake, R., Kaelbling, L., & Lozano-Perez, T. (2010). Belief space planning assuming maximum likelihood observations.
52. Ross, S., Pineau, J., Chaib-draa, B., & Kreitmann, P. (2011). A Bayesian approach for learning and planning in partially observable Markov decision processes. Journal of Machine Learning Research **12**(5).
53. Sridharan, M., Wyatt, J., & Dearden, R. (2010). Planning to see: A hierarchical approach to planning visual actions on a robot using POMDPs. *Artificial Intelligence, 174*(11), 704–725.
54. Eidenberger, R., & Scharinger, J. (2010). Active perception and scene modeling by planning with probabilistic 6d object poses. In: *2010 IEEE/RSJ international conference on intelligent robots and systems* (pp. 1036–1043). IEEE.
55. Zheng, K., Sung, Y., Konidaris, G., & Tellex, S. (2021). Multi-resolution pomdp planning for multi-object search in 3d. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
56. Zhang, S., Sridharan, M., & Washington, C. (2013). Active visual planning for mobile robot teams using hierarchical POMDPs. *IEEE Transactions on Robotics, 29*(4), 975–985.
57. Konidaris, G., Kaelbling, L. P., & Lozano-Perez, T. (2018). From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research, 61*, 215–289.
58. Sinapov, J., Khante, P., Svetlik, M., & Stone, P. (2016). Learning to order objects using haptic and proprioceptive exploratory behaviors. In: *IJCAI* (pp. 3462–3468).
59. Aldoma, A., Tombari, F., & Vincze, M. (2012). Supervised learning of hidden and non-hidden 0-order affordances and detection in real scenes. In: *2012 IEEE international conference on robotics and automation* (pp. 1732–1739). IEEE.
60. Katehakis, M. N., & Veinott, A. F., Jr. (1987). The multi-armed bandit problem: decomposition and computation. *Mathematics of Operations Research, 12*(2), 262–268.
61. Zhang, S., Khandelwal, P., & Stone, P. (2017). Dynamically constructed (po) MDPs for adaptive robot planning. In: *Proceedings of the AAAI conference on artificial intelligence* (vol. 31).
62. Zhang, S., & Stone, P. (2020). icorpp: Interleaved commonsense reasoning and probabilistic planning on robots. arXiv preprint arXiv:2004.08672.
63. Kurniawati, H., Hsu, D., & Lee, W. S. (2008). Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In: *Robotics: science and systems* (vol. 2008). Citeseer.

64. Khandelwal, P., Zhang, S., Sinapov, J., Leonetti, M., Thomason, J., Yang, F., Gori, I., Svetlik, M., Khante, P., Lifschitz, V., et al. (2017). Bwibots: A platform for bridging the gap between ai and human-robot interaction research. *The International Journal of Robotics Research, 36*(5–7), 635–659.
65. Tatiya, G., Shukla, Y., Edegware, M., & Sinapov, J. (2020). Haptic knowledge transfer between heterogeneous robots using kernel manifold alignment. In: *2020 IEEE/RSJ international conference on intelligent robots and systems*.
66. Tatiya, G., Hosseini, R., Hughes, M. C., & Sinapov, J. (2020). A framework for sensorimotor cross-perception and cross-behavior knowledge transfer for object categorization. *Frontiers in Robotics and AI, 7*, 137.
67. Ross, S., Chaib-draa, B., & Pineau, J. (2007). Bayes-adaptive pomdps. Advances in neural information processing systems **20**.
68. Ding, Y., Zhang, X., Zhan, Xingyu., Zhang, S. (2022). Learning to ground objects for robot task and motion planning. *IEEE Robotics and Automation Letters. 7*(2),5536–5543.
69. Tatiya, G., Francis, J., Sinapov, J. (2023). Transferring Implicit Knowledge of Non-Visual Object Properties Across Heterogeneous Robot Morphologies. *IEEE International Conference on Robotics and Automation (ICRA)*.

Xiaohan Zhang is a Ph.D. student in the department of computer science at The State University of New York at Binghamton, starting from Fall 2020. Xiaohan received a B.S. degree at Renmin University of China in 2019. His research goal is to develop algorithms that solve complicated and long-horizon tasks for autonomous robots. His research touches robot planning (e.g., probabilistic planning, task and motion planning) and learning (e.g., classical machine learning, deep learning, reinforcement learning).

Saeid Amiri is a Ph.D. student of Computer Science at SUNY Binghamton. He received his MS.c. and BS.c in Mechanical Engineering at University of Houston (2015) and Sharif University of Technology (2013) where his primary focus was on classical control. During Ph.D., his main research has been on sequential decision making algorithms under partial observability that leverage human knowledge and past experiences. He has applied these algorithms to robotic spoken dialogue systems, human-robot interaction, and target search domains. In 2020, he interned at ABB Robotics. He is interested in using mobile and manipulator robotic platforms.

Jivko Sinapov is an Assistant Professor at the Department of Computer Science at Tufts University. Jivko Sinapov received his Ph.D. in computer science and human-computer interaction from Iowa State University (ISU). While working toward his Ph.D. at ISU's Developmental Robotics Lab, he developed novel methods for behavioral object exploration and multi-modal perception. He went on to be a clinical assistant professor with the Texas Institute for Discovery, Education, and Science at UT Austin and a postdoctoral associate working with Peter Stone at the Artificial Intelligence lab. Sinapov's research interests include developmental robotics, computational perception, autonomous manipulation, and human-robot interaction.

Jesse Thomason is an Assistant Professor at USC starting a research group focused on grounded language learning. Recently, he was a visiting scholar with Amazon Alexa AI and a postdoctoral researcher at the University of Washington. He received his PhD from the University of Texas at Austin in 2018 working with Raymond Mooney. Dr. Thomason's research area brings together natural language processing and robotics to connect language to the world (RoboNLP), and centers around three efforts to bring language skills to robot and embodied simulation agents: connecting language to agent perception, including visual, tactile, and auditory senses, connecting language to agent actions by inferring goals from language, and lifelong learning through dialog with people.

Peter Stone holds the Truchard Foundation Chair in Computer Science at the University of Texas at Austin. He is Associate Chair of the Computer Science Department, as well as Director of Texas Robotics. In 2013 he was awarded the University of Texas System Regents' Outstanding Te aching Award and in 2014 he was inducted into the UT Austin Academy of Distinguished Teachers, earning him the title of University Distinguished Teaching Professor. Professor Stone's research interests in Artificial Intelligence incl-ude machine learning (especially reinforcement learning), multiagent systems, and robotics. Professor Stone received his Ph.D in Computer Science in 1998 from Carnegie Mellon University. From 1999 to 2002 he was a Senior Technical Staff

Member in the Artificial Intelligence Principles Research Department at AT&T Labs - Research. He is an Alfred P. Sloan Research Fellow, Guggenheim Fellow, AAAI Fellow, IEEE Fellow, AAAS Fellow, ACM Fellow, Fulbright Scholar, and 2004 ONR Young Investigator. In 2007 he received the prestigious IJCAI Computers and Thought Award, given biannually to the top AI researcher under the age of 35, and in 2016 he was awarded the ACM/SIGAI Autonomous Agents Research Award. Professor Stone co-founded Cogitai, Inc., a startup company focused on continual learning, in 2015, and currently serves as Executive Director of Sony AI America.

Shiqi Zhang is an Assistant Professor of Computer Science, The State University of New York at Binghamton (SUNY Binghamton). From 2014 to 2016, he was a Postdoctoral Fellow working on a team of mobile service robots at UT Austin. He worked at Cleveland State University from 2016-2018. He received his Ph.D. in Computer Science (2013) from Texas Tech University, and received his M.S. (2008) and B.S. (20-06)degrees from Harbin Institute of Technology. Dr. Zhang's research lies in the intersection of artificial intelligence and robotics. He works on developing intelligent mobile robots that are able to interact with people, provide services to people, and learn from this experience, in human-robot collaborative environments.