Multi-modal Predicate Identification using Dynamically Learned Robot Controllers

Saeid Amiri[1], Suhua Wei[1], Shiqi Zhang[1], Jivko Sinapov[2], Jesse Thomason[3], and Peter Stone[3]
[1] Cleveland State University, Cleveland, OH 44115 USA
[2] Tufts University, Medford, MA 02155 USA
[3] The University of Texas at Austin, Austin, TX 78712 USA
{s.amiri@vikes.; s.wei@vikes.; s.zhang9@}csuohio.edu; jivko.sinapov@tufts.edu;
{jesse; pstone}@cs.utexas.edu

Abstract

Intelligent robots frequently need to explore the objects in their working environments. Modern sensors have enabled robots to learn object properties via perception of multiple modalities. However, object exploration in the real world poses a challenging trade-off between information gains and exploration action costs. Mixed observability Markov decision process (MOMDP) is a framework for planning under uncertainty, while accounting for both fully and partially observable components of the state. Robot perception frequently has to face such mixed observability. This work enables a robot equipped with an arm to dynamically construct query-oriented MOMDPs for multi-modal predicate identification (MPI) of objects. The robot's behavioral policy is learned from two datasets collected using real robots. Our approach enables a robot to explore object properties in a way that is significantly faster while improving accuracies in comparison to existing methods that rely on hand-coded exploration strategies.

1 Introduction

Service robots are increasingly present in everyday environments, such as homes, offices, airports, and hospitals, where a common task is to retrieve an object for a user. Consider the request, "*Please fetch me the red, empty bottle*." A key problem for the robot is to decide whether a particular candidate object matches the properties (or *predicate*) in the request, which we refer to as the *multi-modal predicate identification* (MPI) problem. For certain words (e.g., *heavy*, *soft*, etc.), visual classification of the object is insufficient. The robot would need to perform an action (e.g., lift the object to determine whether it is heavy or not). Multi-modal perception research has focused on combining information arising from such multiple sensory modalities.

Given multi-modal perception capabilities, a robot needs to decide which actions (possibly out of many) to perform on an object, i.e., generate a behavioral policy for a given request. 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 actions: grasping and shaking. The robot has to select actions in such a way that the information gain about object properties is maximized while the cost of actions is minimized. Sequential reasoning is required in this action selection process, e.g., a shaking action would make sense only if a grasping action has been successfully executed. Also, robot perception capabilities are imperfect, so the robot sometimes needs to take the same action more than once. Probabilistic planning algorithms aim at computing action policies to help select actions toward maximizing long-term utility (information gain in our case), while considering the uncertainty in non-deterministic action outcomes.

Markov decision processes (MDPs) [Puterman, 1994] and partially observable MDPs (POMDPs) [Kaelbling *et al.*, 1998] 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 [Ong *et al.*, 2010]. In this work, we model robot MPI problems using MOMDPs because of the mixed observability of the world that the robot interacts with (e.g., whether an object is in hand or not is fully observable, but object properties such as color and weight are not).[1]

Robot behavioral exploration policies are used for suggesting exploration actions given the current world state estimation. In this work, the robot learns its policies from the experience of interacting with objects in the real world. We use datasets that include tens of objects and nearly one hundred properties. In such domains, it frequently takes a prohibitively long time to compute effective behavioral exploration policies. To tackle this issue, we dynamically learn MOMDP-

[1] Referring to our model as a MOMDP (as opposed to a POMDP) is not of practical importance in this paper. It is mainly for ease of describing the domain.

based controllers to model a minimum set of domain variables that are relevant to current user queries (e.g. "red, empty bottle"). This strategy ensures a small state set and enables us to generate high-quality robot action policies in a reasonable time (e.g., ≤ 5 seconds). Our experiments show that the policies of the learned controllers improve accuracy for recognizing new objects' properties while reducing exploration cost, in comparison to baseline strategies that deterministically or randomly use predefined sequences of actions.

2 Related Work

Recent research has shown that robots can learn to classify objects using computer vision methods as well as non-visual perception coupled with actions performed on the objects [Högman *et al.*, 2013; Sinapov *et al.*, 2014a; Thomason *et al.*, 2016]. For example, a robot can learn to determine whether a container is full or not based on the sounds produced when shaking the container [Sinapov and Stoytchev, 2009]; or learn whether an object is soft or hard based on the haptic sensations produced when pressing it [Chu *et al.*, 2015]. Past work has shown that robots can associate (or *ground*) these sensory perceptions with human language predicates in vision space [Alomari *et al.*, 2017; Whitney *et al.*, 2016; Krishnamurthy and Kollar, 2013; Matuszek *et al.*, 2012] and joint visual and haptic spaces [Gao *et al.*, 2016].

Nevertheless, there has been relatively little emphasis on enabling a robot to *efficiently* select actions at test time when it is tasked with classifying a new object. The few approaches for tackling action selection, e.g., [Rebguns *et al.*, 2011; Fishel and Loeb, 2012; Sinapov *et al.*, 2014a; Thomason *et al.*, 2018], assume that only one target property has to be identified (e.g., the object's identity in the case of object recognition). In contrast, we address the multi-modal predicate identification (MPI) problem where a robot needs to recognize multiple properties about an object, e.g., "is the object a *red empty bottle*?".

Sequential decision-making frameworks, such as MDPs, POMDPs and MOMDPs, can be used for probabilistic planning toward achieving long-term goals, while accounting for non-deterministic action outcomes and different observabilities [Kaelbling *et al.*, 1998; Ong *et al.*, 2010]. As a result, these frameworks have been applied to multi-modal predicate identification (MPI) problems on physical objects in robotics. For instance, hierarchical POMDPs were used for suggesting visual operators and regions of interests for exploring multiple objects on a tabletop scenario [Sridharan *et al.*, 2010]; the work of Eidenberger and Scharinger further enabled a robot to actively adjust its position to avoid occlusions [Eidenberger and Scharinger, 2010]. More recent work used a robotic arm to move objects enabling better visual analysis [Pajarinen and Kyrki, 2015]. Interaction with objects in these lines of research relies heavily on robot vision while other sensing modalities, such as audio and haptics, are not considered.

Behavioral policies for MPI problems have been learned in simulation using deep reinforcement learning methods [Denil *et al.*, 2017], where *force* was directly used in the interactions with objects. The simulation environment used in that work makes it possible to run large numbers of trials, but does not establish applicability on real robots.

Behavior	Modality		
	color	shape	deep
look	64	308	4096
	audio	haptics	proprioception
grasp	100	60	20
drop, hold, lift, lower, press, push	100	60	

Table 1: The number of features extracted from each *context* (i.e., combination of robot behavior and perceptual modality) for one of the datasets (**Thomason16**) used in our experiments.

3 Theoretical Framework

Next, we describe the theoretical framework used by the robot to learn multi-modal predicate identification (MPI) models and generate efficient policies when tasked with identifying whether a set of predicates hold true for a new object.

3.1 Multi-modal Predicate Learning

In this work, the robot learns predicate recognition models using the methodology described in [Sinapov *et al.*, 2014b; Thomason *et al.*, 2016], briefly summarized here. The robot interacts with objects using behaviors (e.g., *look*, *grasp*, *lift*) coupled with sensory modalities (e.g., *color*, *haptics*, *audio*). We refer to a combination of a behavior and modality as a sensorimotor context (e.g., *look-color*, *lift-haptics*, etc.), where $\mathcal{C}$ is the set of all such contexts. Table 1 shows the set of sensorimotor contexts for one of the datasets used in our experiments (discussed in more detail in Section 5), along with the feature dimensionality for each context. Note that not all modalities are produced by every behavior – for example, the *lift* action does not produce *color* features while the *look* action does not produce *haptic* features.

We connect these feature representations of objects to predicates by learning discriminative classifiers on the feature spaces for each predicate $p \in \mathcal{P}$, the set of all predicates. For each predicate p, and context $c \in \mathcal{C}$, the robot learns a binary classifier using data points $[x_i^c, y_i]$, where x_i^c is the i^{th} feature vector in context c (e.g., in the *look-color* context, the feature vector encodes a color histogram of the object), and $y_i = true$ if the predicate p holds true for the object in trial i, and $false$ otherwise. We assume that the classifiers' outputs can be mapped to probabilities, i.e., a classifier associated with context c for predicate p can estimate $\mathbf{Pr}_p^c(y_i = true|x_i^c)$.

Let $\mathcal{C}_b \subset \mathcal{C}$ be the set of sensorimotor contexts associated with behavior $b \in \mathcal{B}$. When executing action b, the robot detects a set of feature vectors, $\mathcal{X}_i$ (one vector per each context in $\mathcal{C}_b$), and uses them to query the classifiers associated with contexts $\mathcal{C}_b$. The probability estimates of the classifiers are combined using weighted combination and normalized again to compute the final predication:

$$\mathbf{Pr}_p(y_i = true|\mathcal{X}_i) = \alpha \times \sum_{x_i^c \in \mathcal{X}_i} w_c^p \times \mathbf{Pr}_p^c(y_i = true|x_i^c)$$

where α is a normalization constant to ensure the probabilities sum up to 1.0 and $w_c^p \in [0.0, 1.0]$ is a reliability weight indicating how good the classifier associate with context c is

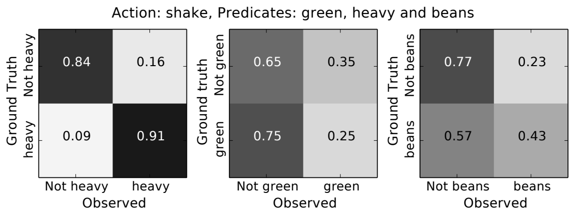

Figure 1: Example confusion matrices from one of the datasets (**Sinapov14**) showing the TP, FP, TN, and FN rates for three of the predicates when using the robot's *shake* action. The action is good at recognizing *heavy* due to the rich haptic feedback produced when shaking an object, somewhat good 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 action.

at recognizing predicate p, as estimated by performing cross-validation on the training data. In other words, each behavior acts as a classifier ensemble where each individual classifier's output is combined using weighted combination.

At the end of the training stage, cross-validation at the behavior level is used to compute the confusion matrix $C_p^b \in \mathbb{R}^{2\times 2}$ for predicate p and behavior b. These confusion matrices are normalized to compute the True Positive, True Negative, False Positive, and False Negative rates for each behavior-predicate pair, which are later used for dynamically constructing controllers. Example confusion matrices are shown in Figure 1. Next, we describe the problem of generating an action policy when identifying whether a set of predicates hold true for a novel object.

3.2 MOMDP-based Controllers

Behaviors (or actions[2]), such as *look* and *drop*, have different costs and different accuracies in predicate 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 actions. For instance, *shaking* and *dropping* actions make sense only if a preceding (unreliable) *grasping* action succeeds.

In this work, we assume action outcomes are fully observable and object properties are not. For instance, a robot can reliably sense whether a *grasping* action is successful, but it cannot reliably sense the color of a bottle or whether that bottle is full. Due to this mixed observability and unreliable action outcomes, we use mixed observability MDPs (MOMDPs) [Ong *et al.*, 2010] to model the sequential decision-making problem for object exploration.

A MOMDP is fundamentally a factored POMDP with 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

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

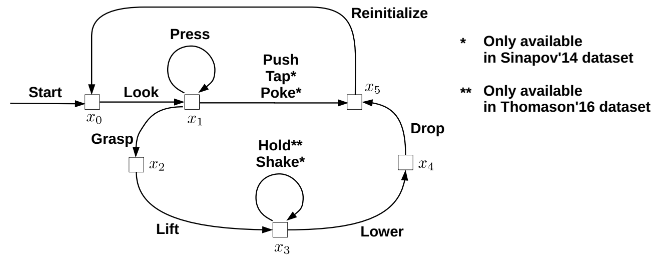

Figure 2: A simplified version of the transition diagram in space $\mathcal{X}$ for object exploration. This figure only shows the probabilistic transitions led by *exploration actions*. *Report actions* that deterministically lead transitions from $x_i \in \mathcal{X}$ to the *term* state are not included.

variable y (in our case, the *object properties*, 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,

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

where 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, Z is the observation set, $\mathcal{O}$ is the observation function, and γ is the discount factor.

The definitions of A, R, Z, $\mathcal{O}$, and γ of a MOMDP are identical to these of POMDPs [Kaelbling *et al.*, 1998], except that Z and $\mathcal{O}$ are only applicable to $\mathcal{Y}$, the partially observable components of the state space. γ is the discount factor that specifies the planning horizon.

Next, we present how each component of our MOMDP model is specified for our object exploration problem.

3.3 State Space Specification

The state space of our MOMDP-based controllers has two components of $\mathcal{X}$ and $\mathcal{Y}$. The global state space S includes a Cartesian product of $\mathcal{X}$ and $\mathcal{Y}$,

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

$\mathcal{X}$ is the state set specified by fully observable domain variables. In our case, $\mathcal{X}$ includes a set of six states $\{x_0, \cdots, x_5\}$, as shown in Figure 2, and a terminal state $term \in \mathcal{X}$ that identifies the end of an episode. $x \in \mathcal{X}$ is fully observable, and the robot knows the current state of the robot-object system, e.g., whether grasping and dropping actions are successful or not.

$\mathcal{Y}$ is the state set specified by partially observable domain variables. In our case, these variables correspond to N object properties that are queried about, $\{v_0,\ v_1,\ \cdots,\ v_{N-1}\}$, where the value of v_i is either *true* or *false*. Thus, $|\mathcal{Y}| = 2^N$.

For instance, given an object description that includes three properties (e.g., "a *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. It should be noted that there is no state transition in the space of $\mathcal{Y}$, as we assume object properties do not change over the course of robot action.

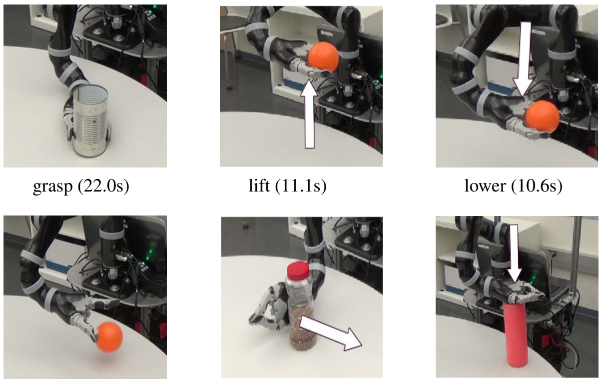

Figure 3: The behaviors, and their durations in seconds (behaviors are from the **Thomason16** dataset detailed in Sec. 5). In addition, the *hold* (1.0s) behavior was performed by holding the object in place. The *look* (0.5s) behavior was also performed by taking a visual snapshot of the object using the robot's sensors prior to exploration.

3.4 Actions and Transition System

We present the transition system of our MOMDP-based controllers by first introducing the action set and then the transition probabilities. $A : A^e \cup A^r$ is the action set. A^e includes the object *exploration* actions pulled from the literature of robot exploration, as shown in Figure 2, and A^r includes the *reporting* actions used for object property identification.

Exploration actions: Figure 2 shows all exploration actions except for action *ask* (i.e., ask a human operator) that is allowed in any state $x \in \mathcal{X}$. Among the actions, *tap*, *poke*, and *shake* are only available in the dataset of [Sinapov *et al.*, 2014b] and *hold* is only available in the dataset of [Thomason *et al.*, 2016]. As one of the main contributions, our approach enables a robot to automatically figure out what actions are useful given a user query by learning from the datasets. Examples of a robot executing some of the exploration actions are shown in Figure 3.

Reporting actions: A^r includes a set of actions that are used for reporting the object's properties and can deterministically lead the state transition to *term* (terminal state). For instance, if a user queries about "a blue, heavy can", there will be three binary variables specifying whether each of properties is true or false. As a result, there will be eight reporting actions. For $a \in A^r$, we use $s \odot a$ (or $y \odot a$) to represent that the report of a matches the underlying values of object properties (i.e., a correct report) and use $s \oslash a$ (or $y \oslash a$) otherwise.

$T_{\mathcal{X}} : \mathcal{X} \times A \times \mathcal{X} \to [0,1]$ 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 $a \in A$ the current action. Reporting actions and illegal exploration actions (e.g., *dropping* an object in state x_1—before a successful grasp) lead state transitions to *term* with 1.0 probability.

Most exploration actions are unreliable and succeed probabilistically. For instance, $p(x_4,\ drop,\ x_5) = 0.95$ in our case, indicating there is small probability the object is stuck in the robot's hand. Such non-deterministic action outcomes are considered in our experiments. The success rate of action *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.

$T_{\mathcal{Y}} : \mathcal{Y} \times A \times \mathcal{Y} \to [0,1]$ 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 properties do not change during the process of the robot's exploration actions.

3.5 Reward Function and Discount Factor

$R : S \times A \to \mathbb{R}$ is the reward function. Each *exploration action*, $a^e \in A^e$, has a cost that is determined by the time required to complete the action. 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.

$$R(s,a) = \begin{cases} r^-, & \textbf{if } s \in S,\ a \in A^r,\ s \oslash a \\ r^+, & \textbf{if } s \in S,\ a \in A^r,\ s \odot a \end{cases}$$

where r^- (or r^+) is negative (or positive) given an incorrect (or correct) report. Unless otherwise specified, $r^- = -500$ and $r^+ = 500$ in this paper.

Generally, a robot is more risk-seeking (e.g., preferring fewer exploration actions before taking the reporting action), when the penalty of incorrect reports is lower or the bonus of correct reports is higher. Prior research studied such parameters in a dialog system context [Zhang and Stone, 2015]. We set the values of r^- and r^+ heuristically in this work.

Costs of other exploration actions come from the datasets used in this research, and are within the range of $[0.5, 22.0]$ (corresponding reward is negative), except that action *ask* has the cost of 100.0. γ is a discount factor, and $\gamma = 0.99$ in our case. This setting gives the robot an unspecified, relatively long planning horizon.

3.6 Observations and Observation Function

$Z : Z^h \cup \emptyset$ is a set of observations. Elements in Z^h include all possible combinations of object properties and have one-to-one correspondence with elements in A^r and $\mathcal{Y}$. For instance, when the query is about "a *red empty bottle*", there exists an observation $z \in Z^h$ that represents "the object's color is red; it is not empty, and it is a bottle." Actions that produce no information gain (*reinitialize*, in our case), and reporting actions in A^r result in a $\emptyset$ (none) observation.

$O : S \times A \times Z \to [0,1]$ is the observation function that specifies the probability of observing $z \in Z$ when action a is executed in state s: $O(s,a,z)$. In this work, the probabilities are learned from performing cross-validation on the robot's training data. As described in Section 3.1, predicate learning produces confusion matrix $C_p^b \in \mathbb{R}^{2\times 2}$ for each predicate p and each behavior b, where b corresponds to one of the exploration actions shown in Figure 2.

$$\begin{aligned} O(s,a,z) &= \mathbf{Pr}(\mathbf{p}^s, b, \mathbf{p}^z) \\ &= C_{p_0}^b(p_0^s, p_0^z) \cdot C_{p_1}^b(p_1^s, p_1^z) \cdots C_{p_{N-1}}^b(p_{N-1}^s, p_{N-1}^z) \end{aligned}$$

where behavior b corresponds to action a; $\mathbf{p}^s$ and $\mathbf{p}^z$ are the vectors of *true* and *observed* values (0 or 1) of the predicates;

p_i^s (or p_i^z) is the true (or observed) value of the i^{th} predicate; and N is the total number of predicates in the query.

So far, we have specified all components of our MOMDP-based controller. It should be noted that there are other subclasses of POMDPs that can be used for formalizing the MPI problem (e.g., POMDP-lite [Chen *et al.*, 2016]). We leave the comparisons to future work.

Next, we discuss a way of computing high-quality policies for MPI problems that include large numbers of predicates.

4 Dynamically Learned Controllers

Multi-modal predicate identification (MPI) problems can include a prohibitively large number of predicates. One of the datasets in our experiments contains 81 predicates, resulting in 2^{81} possible states in $\mathcal{Y}$. It is computationally intractable to generate a far-sighted policy while considering all the predicates. The goal of dynamically learned controllers is to include a relatively small set of predicates in our MOMDPs while maintaining quality of the generated policies.

Recent research on Integrated commonsense Reasoning and probabilistic Planning (IRP) [Zhang and Stone, 2017] enables decomposing a sequential decision-making problem into two tractable subproblems that focus on high-dimensional reasoning (e.g., objects with many properties) and long-horizon planning (e.g., tasks that require many actions). Among the IRP algorithms, iCORPP [Zhang *et al.*, 2017] enables an agent to reason about all domain variables in a complete (static) world model, specify a transition system focusing on the current task, and plan under uncertainty toward achieving long-term goals.

We dynamically construct MOMDP controllers (inspired by iCORPP) by specifying the following components in order: 1) State set $\mathcal{Y}$ that includes only the predicates that are relevant to the query (e.g., "blue", "heavy", and "bottle", given a query of "a blue heavy bottle"); 2) State set $\mathcal{S}$, the Cartesian product of $\mathcal{X}$ (predefined) and $\mathcal{Y}$; 3) Action set A^r, where each reporting action $a^r \in A^r$ corresponds to a state in $\mathcal{Y}$; 4) Action set A, union of A^e (predefined) and A^r; 5) Z^h, object predicate combinations; 6) Z, union of Z^h and $\emptyset$.

Given the above components, observation function $\mathcal{O}$ is learned from datasets as described in Section 3.1, and transition function T and reward function R are constructed accordingly. The components together form a complete MOMDP that is relatively very small, and typically includes fewer than 100 states at runtime. It should be noted that we use MOMDP, as a special form of POMDP, to model our domain mainly for the ease of describing the mixed observability over $\mathcal{X}$ and $\mathcal{Y}$ (Section 3.3). Our approach enables automatic generation of complete MOMDP models, which can be encoded, as in our experiments, such that existing POMDP solvers (e.g., [Kurniawati *et al.*, 2009]) can be used to generate policies.

5 Experimental Results

We evaluate the proposed method using two datasets in which a robot explored a set of objects using a variety of exploratory behaviors and sensory modalities, and show that, for both, our proposed MOMDP model outperforms baseline models in exploration accuracy and overall exploration cost. Two datasets of **Sinapov14** and **Thomason16** are used in the experiments, where **Thomason16** has a much more diverse set of household objects and a larger number of predicates that arose naturally during human-robot interaction gameplay.

Figure 4: Objects in the **Thomason16** dataset (Left) and the one used in the illustrative example in Section 5.1 (Right).

Sinapov14 Dataset: In this dataset, 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 [Sinapov *et al.*, 2014b]. 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 P that the robot is tasked with learning. The costs of actions in the two datasets are different because the datasets were collected using different robots.

Thomason16 Dataset: In this dataset, the robot, part of the Building-Wide Intelligence project [Khandelwal *et al.*, 2017], explored 32 common household objects using 8 exploratory actions: *look*, *grasp*, *lift*, *hold*, *lower*, *drop*, *push*, and *press*. Each behavior was performed 5 times on each object. The dataset was originally produced for the task of learning how sets of objects can be ordered and is described in greater detail by [Sinapov *et al.*, 2016].

For the *look* behavior, *color*, *shape*, and *deep* features (the penultimate layer of the trained VGG network [Simonyan and Zisserman, 2014]) are available. For the remaining behaviors, the robot recorded *audio* and *haptic* (i.e., joint forces) features produced by the interaction with the object. Finally, for the *grasp* action, finger position features were also extracted. These modalities result in $|C| = 1 \times 3 + 7 \times 2 + 1 = 18$ sensorimotor contexts. The set of predicates $\mathcal{P}$ consisted of 81 words used by human participants to describe objects in this dataset during an interactive gameplay scenario [Thomason *et al.*, 2016]. Example predicates include the words *red*, *heavy*, *empty*, *full*, *cylindrical*, *round*, etc. Unlike the **Sinapov14** dataset, here the objects vary greatly, and the predicate recognition problem is much more difficult.

5.1 Illustrative Example

We now describe an example in which a robot works on the multi-modal predicate identification (MPI) task. We randomly selected an object from the **Thomason16** dataset: a blue and red bottle full of water (Figure 4). We then randomly selected properties, in this case "yellow" and "metallic," and asked the robot to identify whether the object has each of the properties or not. The selected object was not part of the robot's training set used to learn the predicate recognition models and the MOMDP observation model. The robot should report negative to both properties while minimizing the overall cost of exploration actions.

Given this user query, we generate a MOMDP model that includes 25 states. We then generate an action policy using past work's methods [Kurniawati *et al.*, 2009]. 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).

Figure 5 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 "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" action, 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 actions of *reinitialize*, *look*, *push*, and *push* (this first *push* action 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 a 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 behavior and word will produce an output even in cases where the sensory signals from that behavior are irrelevant to the word. For instance, although the sensory signals relevant to "push" are haptics and audio, the first "push" action results in an observation of "yellow". It was "yellow:neg", because the training set prior of most objects are not yellow. The robot favors actions that distinguish 'easy' predicates (*look* distinguishes *yellow* well in this case). If an action is useful, the robot will prefer taking it early. The more the action is delayed, the more the expected reward is discounted (we use a discount factor of 0.99 in our experiments).

5.2 Results

Next, we describe the experiments we conducted to evaluate the proposed MOMDP-based perception strategy using MPI problems. The goal was to increase the accuracy in identifying properties of a novel object while reducing the overall action costs required in this process. In all evaluation runs, the object that needs to be identified was not part of the robot's training set when learning the predicate recognition models or the MOMDP parameters. The following baseline action strategies are used in experiments, where belief is updated using Bayes' rule except for *Random*:

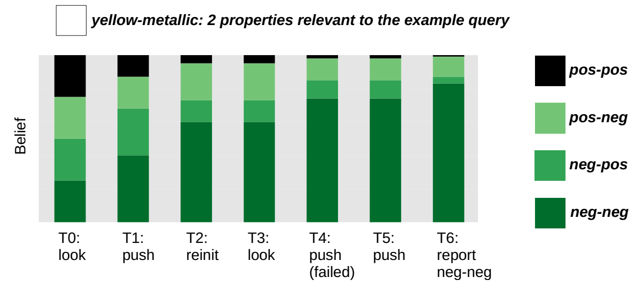

Figure 5: Action selection and belief change in the exploration of a red and blue bottle full of water, given a query of *yellow* and *metallic*.

Properties	Method	Overall cost (std)	Accuracy
Two	Random	17.56 (30.00)	0.245
	Predefined Plus	37.10 (0.00)	0.583
	MOMDP (Ours)	29.85 (12.87)	0.860
Three	Random	10.12 (21.77)	0.130
	Predefined Plus	37.10 (0.00)	0.373
	MOMDP (Ours)	33.87 (8.78)	0.903

Table 2: Performances of MOMDP-based and two baseline planners in cost (second) and accuracy on the **Sinapov14** dataset. Numbers in parenthesis denote the Standard Deviations over 400 trials.

- *Random*: Actions are randomly selected from both reporting and legal exploration actions. A trial is terminated by any of the reporting actions.
- *Random Plus*: Actions are randomly selected from legal exploration actions. Under an exploration budget, one selects the reporting action corresponding to y with the highest belief.
- *Predefined*: An action sequence is strictly followed: *ask, look, press, grasp, lift, lower* and *drop*.[3] 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).

Sinapov14 Dataset: In each trial, we place an object that has three attributes (color, weight and content) on a table and then 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 *term* state, resulting in $\mathcal{S}$ that includes 25 (or 49) states. The other components of the dynamically constructed MOMDPs grow accordingly, given an increasing number of queried attributes.

Experimental results are reported in Table 2. Not surprisingly, randomly selecting actions produces low accuracy. The overall cost is smaller in more challenging trials (all three properties are questioned), because in these trials there are relatively fewer exploration actions (more properties produce more reporting actions), making the agent more likely to take a reporting action. Our MOMDP-based multi-modal perception strategy reduces the overall action cost while signifi-

[3] Action *ask* was used only in the **Thomason16** experiments, because other exploration actions are not as effective as in **Sinapov14**.

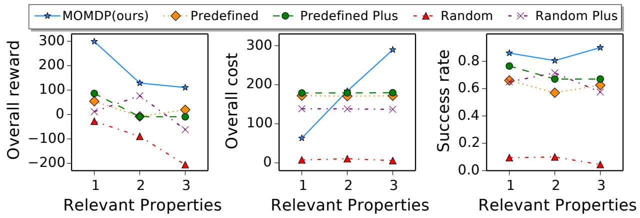

Figure 6: Evaluations of five actions strategies on the **Thomason16** dataset. Comparisons are made in three categories of *overall reward* (Left), *overall exploration cost* (Middle), and *success rate* (Right).

cantly 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" exploration actions given queries where the exploration actions are not effective.

Thomason16 Dataset: In this set of experiments, a user query is specified by randomly selecting one object and N properties ($1 \le N \le 3$), on which the robot is questioned. Each data point is an average over 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 properties. It should be noted that most of the contexts are misleading in this dataset due to the large number of object properties, so more exploration actions confuse the robot if the actions are not carefully selected. Figure 6 shows the experimental results. Overall reward is computed by subtracting 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 our MOMDP-based strategy consistently performs the best in terms of the overall reward and overall accuracy. When more properties are queried, the MOMDP-based controllers enable the robot to take more exploration actions (Middle subfigure), whereas the baselines could not adjust their question-asking strategy accordingly.

The last experiment aims to evaluate the need of dynamically constructed controllers, answering the question "*Can we build a 'super' controller that models all properties?*" We constructed MOMDP controllers including two relevant and an increasing number of irrelevant properties (i.e., the ones that are not queried). Our dynamically learned controllers include only the relevant properties and correspond to the curves' left ends. Results are shown in Figure 7. We can see, the quality of the generated action policies decreases soon, e.g., from >150 to <25 in reward (what we try to maximize), when more irrelevant properties are included in the controllers. The right two subfigures show that the controllers first try to achieve a higher accuracy by taking more exploration actions and then "give up" due to the growing number of irrelevant properties. The results show the infeasibility of "super" controllers that model all properties and justify the need of dynamic controllers.

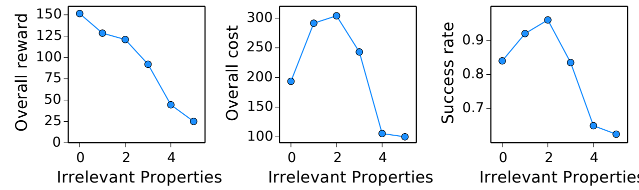

Figure 7: A "super" MOMDP that models two relevant plus increasing irrelevant properties, in comparison to our dynamically learned controllers that model only the relevant predicates and correspond to the left end of each curve.

6 Conclusions and Future Work

We investigate using mixed observability Markov decision processes (MOMDPs) to solve the multi-modal predicate identification (MPI) problem, where a robot selects actions for multi-modal perception in object exploration tasks. Our approach can dynamically construct a MOMDP model given an object description from a human user (e.g., "*a blue heavy bottle*"), 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 controllers enable the robot to focus on a minimum set of domain variables that are relevant to the current object and query. The MOMDP perception models are learned using two existing datasets collected with robots interacting with objects in the real world. Experimental results show that our object exploration approach enables the robot to identify object properties more accurately without introducing extra cost from exploration actions compared to a baseline that suggests actions following a predefined action sequence.

This research primarily focuses on a robot exploring objects in a tabletop scenario. In future work, we plan to investigate applying this approach to tasks that involve more human-robot interaction and mobile robot platforms, where exploration would require navigation actions and perceptual modalities such as human-robot dialog. Finally, in the two datasets used in this paper, the robot's manipulation actions were always successful. That is not the case in a real-world scenario (we model unsuccessful actions in a simple way in this work); therefore we plan to extend our framework to situations in which the robot's actions may cause undesirable outcomes (e.g., dropping an object may break it).

Acknowledgments

The authors thank the anonymous reviewers for their useful feedback. A portion of this work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-1637736, IIS-1651089, IIS-1724157), Intel, Raytheon, and Lockheed Martin. Peter Stone serves on the Board of Directors of Cogitai, Inc. 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.

References

[Alomari *et al.*, 2017] Muhannad Alomari, Paul Duckworth, David C. Hogg, and Anthony G. Cohn. Natural language acquisition and grounding for embodied robotic systems. In *AAAI*, February 2017.

[Chen *et al.*, 2016] Min Chen, Emilio Frazzoli, David Hsu, and Wee Sun Lee. Pomdp-lite for robust robot planning under uncertainty. In *ICRA*, pages 5427–5433, 2016.

[Chu *et al.*, 2015] Vivian Chu, Ian McMahon, Lorenzo Riano, Craig G McDonald, et al. Robotic learning of haptic adjectives through physical interaction. *Robotics and Autonomous Systems*, 63:279–292, 2015.

[Denil *et al.*, 2017] Misha Denil, Pulkit Agrawal, Tejas D Kulkarni, Tom Erez, Peter Battaglia, and Nando de Freitas. Learning to perform physics experiments via deep reinforcement learning. In *International Conference on Learning Representations*, 2017.

[Eidenberger and Scharinger, 2010] Robert Eidenberger and Josef Scharinger. Active perception and scene modeling by planning with probabilistic 6d object poses. In *IROS*, pages 1036–1043. IEEE, 2010.

[Fishel and Loeb, 2012] Jeremy Fishel and Gerald Loeb. Bayesian exploration for intelligent identification of textures. *Frontiers in Neurorobotics*, 6:4, 2012.

[Gao *et al.*, 2016] Yang Gao, Lisa Anne Hendricks, Katherine J Kuchenbecker, and Trevor Darrell. Deep learning for tactile understanding from visual and haptic data. In *ICRA*, pages 536–543. IEEE, 2016.

[Högman *et al.*, 2013] Virgile Högman, Marten Björkman, and Danica Kragic. Interactive object classification using sensorimotor contingencies. In *IROS*. IEEE, 2013.

[Kaelbling *et al.*, 1998] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

[Khandelwal *et al.*, 2017] Piyush Khandelwal, Shiqi Zhang, Jivko Sinapov, Matteo Leonetti, Jesse Thomason, et al. 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, 2017.

[Krishnamurthy and Kollar, 2013] Jayant Krishnamurthy and Thomas Kollar. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206, 2013.

[Kurniawati *et al.*, 2009] Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems Conference*, pages 65–72. The MIT Press, 2009.

[Matuszek *et al.*, 2012] Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. A joint model of language and perception for grounded attribute learning. In *ICML*, 2012.

[Ong *et al.*, 2010] Sylvie CW Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010.

[Pajarinen and Kyrki, 2015] Joni Pajarinen and Ville Kyrki. Robotic manipulation of multiple objects as a POMDP. *Artificial Intelligence*, 2015.

[Puterman, 1994] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

[Rebguns *et al.*, 2011] Antons Rebguns, Daniel Ford, and Ian R Fasel. Infomax control for acoustic exploration of objects by a mobile robot. In *Lifelong Learning*, 2011.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[Sinapov and Stoytchev, 2009] Jivko Sinapov and Alexander Stoytchev. From acoustic object recognition to object categorization by a humanoid robot. In *the RSS Workshop: Mobile Manipulation in Human Environments*, 2009.

[Sinapov *et al.*, 2014a] Jivko Sinapov, Connor Schenck, Kerrick Staley, Vladimir Sukhoy, and Alexander Stoytchev. Grounding semantic categories in behavioral interactions: Experiments with 100 objects. *Robotics and Autonomous Systems*, 62(5):632–645, 2014.

[Sinapov *et al.*, 2014b] Jivko Sinapov, Connor Schenck, and Alexander Stoytchev. Learning relational object categories using behavioral exploration and multimodal perception. In *ICRA*, pages 5691–5698, 2014.

[Sinapov *et al.*, 2016] Jivko Sinapov, Priyanka Khante, Maxwell Svetlik, and Peter Stone. Learning to order objects using haptic and proprioceptive exploratory behaviors. In *IJCAI*, 2016.

[Sridharan *et al.*, 2010] Mohan Sridharan, Jeremy Wyatt, and Richard Dearden. Planning to see: A hierarchical approach to planning visual actions on a robot using POMDPs. *Artificial Intelligence*, 174(11):704–725, 2010.

[Thomason *et al.*, 2016] Jesse Thomason, Jivko Sinapov, Maxwell Svetlik, Peter Stone, and Raymond J Mooney. Learning multi-modal grounded linguistic semantics by playing I Spy. In *IJCAI*, 2016.

[Thomason *et al.*, 2018] Jesse Thomason, Jivko Sinapov, Raymond Mooney, and Peter Stone. Guiding exploratory behaviors for multi-modal grounding of linguistic descriptions. In *AAAI*, 2018.

[Whitney *et al.*, 2016] David Whitney, Miles Eldon, John Oberlin, and Stefanie Tellex. Interpreting Multimodal Referring Expressions in Real Time. In *ICRA*, 2016.

[Zhang and Stone, 2015] Shiqi Zhang and Peter Stone. CORPP: Commonsense Reasoning and Probabilistic Planning, as Applied to Dialog with a Mobile Robot. In *AAAI*, 2015.

[Zhang and Stone, 2017] Shiqi Zhang and Peter Stone. Integrated commonsense reasoning and probabilistic planning. In *Proceedings of the 5th Workshop on Planning and Robotics (PlanRob)*, pages 111–114, 2017.

[Zhang *et al.*, 2017] Shiqi Zhang, Piyush Khandelwal, and Peter Stone. Dynamically constructed (PO)MDPs for adaptive robot planning. In *AAAI*, 2017.