Framing Human-Robot Task Communication as a Partially Observable Markov Decision Process

A dissertation presented
by

Mark P. Woodward

to
The School of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Computer Science

Harvard University
Cambridge, Massachusetts
April 2012

©2012 - Mark P. Woodward

All rights reserved.

Thesis advisor | Author

Robert J. Wood | **Mark P. Woodward**

Framing Human-Robot Task Communication as a Partially Observable Markov Decision Process

Abstract

As general purpose robots become more capable, pre-programming of all tasks at the factory will become less practical. We would like for non-technical human owners to be able to communicate, through interaction with their robot, the details of a new task; I call this interaction "task communication". During task communication the robot must infer the details of the task from unstructured human signals, and it must choose actions that facilitate this inference.

In this dissertation I propose the use of a partially observable Markov decision process (POMDP) for representing the task communication problem; with the unobservable task details and unobservable intentions of the human teacher captured in the state, with all signals from the human represented as observations, and with the cost function chosen to penalize uncertainty.

This dissertation presents the framework, works through an example of framing task communication as a POMDP, and presents results from a user experiment where subjects communicated a task to a POMDP-controlled virtual robot and to a human-controlled virtual robot. The task communicated in the experiment consisted of a single object movement and the communication in the experiment was limited to binary approval signals from the teacher.

This dissertation makes three contributions: 1) It frames human-robot task communication as a POMDP, a widely used framework. This enables the leveraging of techniques developed for other problems framed as a POMDP. 2) It provides an example of framing a task communication problem as a POMDP. 3) It validates the framework through results from a user experiment. The results suggest that the proposed POMDP framework produces robots that are robust to teacher error, that can accurately infer task details, and that are perceived to be intelligent.

Contents

Citations to Previously Published Work

Most of the work presented in this dissertation has been published in the following places:

> *Using Bayesian Inference to Learn High-Level Tasks from a Human Teacher*, Mark P. Woodward and Robert J. Wood, The International Conference on Artificial Intelligence and Pattern Recognition, AIPR 2009
>
> *Learning from Humans as an I-POMDP*, Mark P. Woodward and Robert J. Wood, Harvard University, 2012, arXiv:1204.0274v1 [cs.RO, cs.AI]
>
> *Framing Human-Robot Task Communication as a POMDP*, Mark P. Woodward and Robert J. Wood, Harvard University, 2012, arXiv:1204.0280v1 [cs.RO]

Acknowledgments

I would like to first thank my wife, Christine Skolfield Woodward, for encouraging me and for being a role model for getting things done. Secondly, I would like to thank my parents, O. James Woodward III and Dr. Judith Knapp Woodward, for giving me the life tools to reach this point.

I would like to particularly thank my advisor, Professor Robert J. Wood, for supporting and guiding me in all areas. He has been an exceptionally good advisor and will continue to be a valuable role model.

I thank my thesis committee members, Professor Radhika Nagpal and Professor David Parkes, for their insightful feedback, for allowing me to spout my views at their undergraduate students, and for providing valuable perspectives on the Ph.D. process.

It has been an honor to be a member of the Harvard Microrobotics Laboratory. The feedback from its members has encouraged and shaped my research. In particular, I would like to thank Peter Whitney, Nicholas Hoff, Michael Karpelson, Michael Petralea, and Benjamin Finio.

Several individuals from my time at Stanford University have had a profound influence on my research. For their instruction, their research, their advising, and their conversations, I would like to thank Andrew Ng, Sebastian Thrun, Pieter Abbeel (UC Berkeley), and Oussama Khatib.

There are several authors, with whom I have not been acquainted, but whose research has greatly influenced my own. In particular, I would like to thank Nicholas Roy (MIT), Jason Williams (AT&T), Piotr Gmytrasiewicz (UIC), and Joelle Pineau (McGill).

Lastly, I would like to thank the Wyss Institute for Biologically Inspired Engineering for their generous fellowship that has supported my research.

Chapter 1

Introduction

General purpose robots such as Willow Garage's PR2[1] and Stanford's STAIR robot[2] are capable of performing a wide range of tasks such as folding laundry [45], and unloading the dishwasher [31] (figure 1.1). While many of these tasks will come pre-programmed from the factory, we would also like the robots to acquire new tasks from their human owners. For the general population, this demands a simple and robust method of communicating new tasks. Through this dissertation I hope to promote the use of the partially observable Markov decision processes (POMDP) as a framework for controlling the robot during these task communication phases. The idea is that we represent the unknown task as a set of hidden random variables. Then, if the robot is given appropriate models of the human, it can choose actions that elicit informative responses from the human, allowing it to infer the value of these hidden random variables. I formalize this idea in chapter 2. This approach makes the robot

[1]http://www.willowgarage.com/pages/pr2/overview

[2]http://stair.stanford.edu/

an active participant in task communication.[3]

Note that I distinguish "task communication" from "task execution". Once a task has been communicated it might then be associated with a trigger for later task execution. This dissertation deals with communicating the details of a task, not commanding the robot to execute a task; i.e. task communication not task execution.

1.1 Dissertation Contents and Contributions

In the following section we review work related to human-robot task-communication and to communication using POMDPs. Chapter 2 presents the framework, with a review of partially observable Markov decision processes (POMDPs), including Bayesian Inference, Belman's equation, and an overview of POMDP solvers. Chapter 3 works through an example of encoding task communication as a POMDP for a simple task. Chapter 4 describes results from a user experiment, which evaluates the proposed POMDP framework. Chapter 5 summarizes the dissertation and outlines future research directions. Finally, the appendices present the full state and transition model used in the experiment.

This dissertation makes the following contributions:

- It frames human-robot task communication as a POMDP, a widely used framework. This enables the leveraging of techniques developed for the many problems framed as a POMDP.
- It provides an example of framing a task communication problem as a POMDP.

[3]Though related, this is different from an *active learning* problem [34], since the interaction in task communication is less structured than in the supervised learning setting.

- It validates the framework through results from a user experiment. The results suggest that the proposed POMDP framework produces robots that are robust to teacher error, that can accurately infer task details, and that are perceived to be intelligent.

1.2 Related Work

1.2.1 Demonstration

Many researchers have addressed the problem of task communication. A common approach is to control the robot during the teaching process, and demonstrate the desired task [10, 30, 23]. The problem for the robot is then to infer the task from examples. My proposed framework addresses the general case in which the robot must actively participated in the communication, choosing actions to facilitate task inference. That said, demonstration is a common and efficient method of communication. Many of these approaches are compatible with the general framework proposed in this dissertation, and would be appropriate when the robot chooses to observe a demonstration (see section 1.3).

1.2.2 Action Selection During Communication

In other work, as in mine, the task communication is more hands off, requiring the robot to choose actions during the communication, with much of the work using binary approval feedback as in my experiment below [44, 2, 13]. The approach proposed in this thesis differs in that it proposes the use of a POMDP representation, while prior work has created custom representations, and inference and action selection procedures. This work does introduce interesting task domains, and the task representations may be useful as representations of hypotheses as more complex tasks are considered (see section 5.2.2).

The Sophie's Kitchen work used the widely accepted MDP representation [40]. An important difference from the approach presented in this dissertation is in the way that actions are selected during task communication. In their work the robot repeatedly executes the task, with some noise, as best it currently knows it. In my proposed approach the robot chooses actions to become more certain about the task. Intuitively, if the goal of the interaction is to communicate a task as quickly as possible, then repeatedly executing the full task as you currently believe it, is likely not the best policy. Instead, the robot should be acting to reduce uncertainty specifically about the details of the task that it is unclear on. In order to generate these uncertainty reducing actions I feel that a representation allowing for hidden state is needed, and I have proposed the POMDP. Unlike an MDP, with a POMDP there can be a distribution over details of the task, and actions can be generated to reduce the uncertainty in this distribution. The purpose of their work was to report on how humans act during the teaching process. As such, it, and much of the work from Social Robotics (section 1.2.4), is relevant for the human models needed in the proposed POMDP.

1.2.3 Control

Substantial work has also been done on human assisted learning of low level control policies, such as the mountain car experiments, where the car must learn a throttle policy for getting out of a ravine [16]. While the mode of input is the same as is used in the demonstration of chapter 3 (a simple rewarding input signal), we are addressing different problems and different solutions are appropriate. They are addressing the problem of transferring a control policy from a human to the robot, where explicit conversation actions to reduce uncertainty would be inefficient, and treating the human input as part of an environmental reward is appropriate. In contrast I am addressing the problem of communicating higher level tasks, such as setting the table, in which case, explicitly modeling the human and taking communication actions to reduce uncertainty is beneficial, and treating the human input as observations carrying information about the task details is appropriate. The tasks that would be communicated with the proposed POMDP approach do assume solutions to these control problems; such as avoiding obstacles and manipulating objects. In a deployed setting, a robot will need to acquire these control skills in the field. Since a human is present, hopefully these techniques can be employed to help the robot acquire these control skills.

1.2.4 Social Robotics

The area of social robotics, which includes the Sophie's Kitchen work discussed above, is relevant and provides many insights for the problem of human-robot task communication. Social robotics deals with the class of robots "that people apply a social model to in order to interact with and to understand", Cynthia Breazeal [4]. The focus of social robotics research is on identifying important social interactions and demonstrating that a robot can participate in those interactions.

Three examples of these interactions are vocal turn taking, shared attention, and maintaining hidden beliefs about the partner. By encoding rules of vocal turn taking, involving vocal pauses, eye movement, and head movement, Breazeal demonstrated that a robot can converse with a human, in a babble language, smoothly and with few "hiccups" in the flow [3]. Breazeal et al., and Scassellati motivated and demonstrated shared attention, in which the robot looks at the human's eyes to determine the object of their focus and then looks at that object [5, 33]. Gray et al. demonstrated that a robot can maintain beliefs about the goals and the world state as seen from the conversation partner (these are not beliefs in the probabilistic sense, see section 2.1, the robot tracks the deterministic observable state of the world and the changes that the partner was present to observe; the goals are a shrinking list of the possible states that the partner is attempting to reach.) [9, 6].

The work in social robotics provides a guide for desirable interactions and human models. The hope is to develop robot controllers for which the robot's actions in these interactions are not scripted rules, triggered by observable state, but are chosen to minimize a global cost function and operate in uncertain environments. The disadvantages of a set of action rules (situation $\rightarrow$ action) are that the set is unwieldy to specify and maintain, it can have conflicting rules, and the long term effects of the rules can be hard to predict (no global objective). For an introduction to social robotics, see [4].

1.2.5 Spoken Dialog Managers

A spoken dialog manager is an important component within a spoken dialog system, such as an automated telephone weather information service. The dialog manager receives "speech act" inputs from the natural language understanding component, tracks the state of the conversation, and outputs speech acts to the spoken language generator component. Like task communication in robotics, a spoken dialog manager often seeks to fill in details of interest from noisy observations, and it can direct the conversation through actions. The current state of the art systems use POMDPs as the representation. As such, the techniques which allow these systems to scale are relevant to human-robot task communication. The two main components that resist scaling in a POMDP implementation are belief tracking and action planning.

Spoken dialog manager researchers have scaled belief tracking through two techniques: factoring and partitioning. In factoring, the details of interest are divided into sets that can be tracked independently [50, 42]. If $|A_1|$ is the number of answers to question one and $|A_2|$ is the number of answers to question two, then without factoring we have $|A_1||A_2|$ hypotheses to track, with factoring this is reduced to $|A_1| + |A_2|$ hypotheses. Unfortunately, there is often a dependency between details of interest which precludes factoring. Partitioning, on the other hand, can handle these dependencies. It lumps hypotheses into partitions, each partition contains one or more hypotheses, and tracks the probability of the partitions [46, 51]. For example, if we are interested in the city to report weather for, based on the input so far, the agent might be tracking four hypotheses (Boston, Austin, Houston, and !(Boston, Austin, or Houston)). Partitioning is effective because we can wait to enumerate hypothesis until there is evidence to support them. It also scales with the availability of processing and memory; with more processing and memory we can more finely partition the hypothesis space, allowing for more accurate tracking.

The planning problem has been addressed by reducing the problem space over which planning occurs. This is done by mapping the problem into a smaller feature space, perform planning in this space, and mapping the solution back to the original problem space [48, 49]. Using the telephone weather agent as an example of this mapping, the only reasonable confirmation action is to ask confirmation for the most likely city. Thus, the probability of all cities could be mapped to the two element feature vector which contains the probability of the most likely city and, perhaps, the entropy of the remaining cities, vastly simplifying the problem. These techniques have led to spoken dialog systems that can handle very large problem spaces [15, 52].

While these and other techniques are relevant, there are two important distinctions between spoken dialog management and human-robot task communication. The first is that the observations in a spoken dialog system are usually in one-to-one correspondence with details of interest, which allows for simplified inference through techniques like factoring. In human-robot task communication it is often unclear which task details the observation is relevant to (e.g. a pointing gesture could mean the task involves moving the object you are holding to that location, or it could mean that the task involves picking up another object at that location). The second distinction relates to the the termination of the communication. Most spoken dialog systems seek to submit the details of interest quickly to another system, which makes the cost function reasonably easy to specify (penalize an incorrect submission, reward a correct submission, and lightly penalize all non submit actions). Although outside the work presented in this dissertation, in human-robot task communication, the robot's operation is broader than a single task communication exchange. A task communication exchange is situated within the continuous operation of the robot, and the robot's actions should factor in the human's desire to communicate yet another task or to start the robot executing a task. Thus, the choice of a cost functions is less obvious. See section 5.2.5 for a discussion of good cost functions for human robot interaction. For an excellent overview of spoken dialog management, see [47].

1.3 Why Control the Communication?

In the task communication framework proposed below the robot plans its actions; i.e. the robot is in control of its actions and chooses those actions in accordance with an objective function. Since this planning adds a significant computational cost, why is it important? The alternative would be for the human to provide demonstrations of the task, either with their own body or by controlling the robot's body. The robot would still be required to infer the task from the demonstrations, but this would eliminate the additional need for planning.

The benefit of planning is that it makes task communication faster and more accurate:

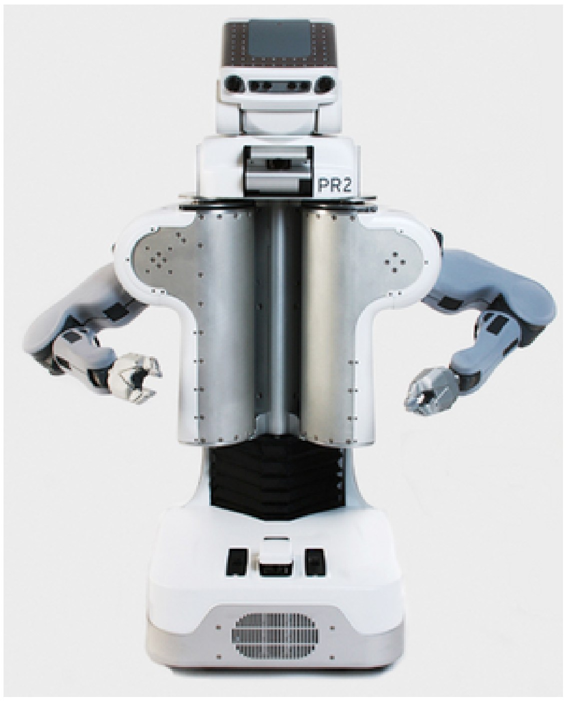
- **faster** — With planning, the robot can direct the communication away from details that are obvious to it (perhaps from related tasks), eliminating the time needed to demonstrate those details. Without planning, the human would need to fully demonstrate all of the details for every task that they teach the robot.

- **more accurate** — With planning, the robot can direct the communication towards details of the task that are not yet clear. Without planning, the teacher can easily omit demonstrations that might clarify a task detail. For example, if the human is teaching the "pour a glass of milk" task, they could easily provide all demonstrations with the glass roughly one foot from the sink, leaving the robot uncertain about the importance of this distance. With planning, the robot could plan to clarify the importance of the distance to from sink.

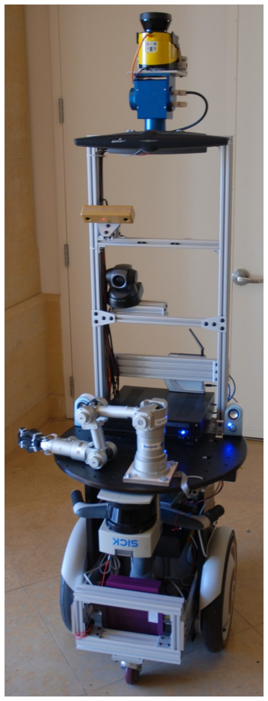
Both of these benefits have at their core the fact that only the robot knows what the robot knows, and planning can leverage this knowledge.

Note that planning does not preclude demonstrations, but the act of observing the demonstration should be an action that, through planning, is expected to improve communication. If the observed action loses its benefit over time, the robot can interrupt the observation and take a more productive action.

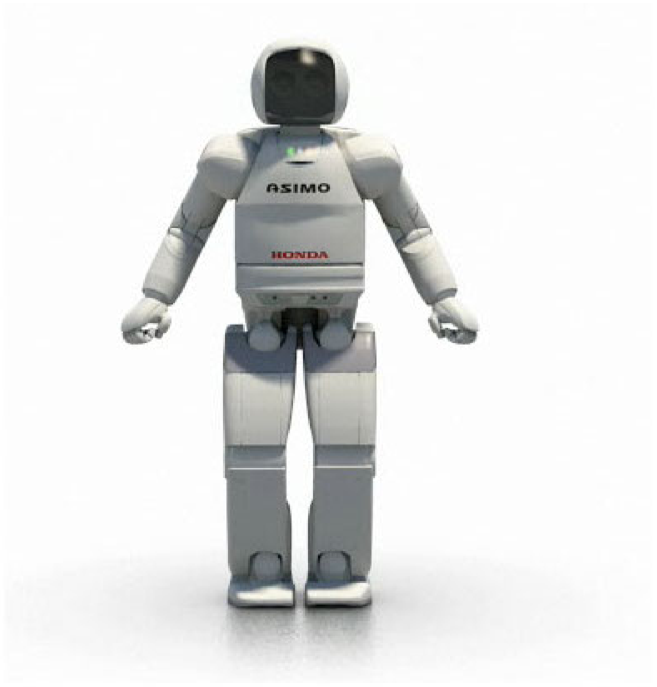
As an example of the benefit of planning in a familiar human setting, we can look at a professor's office hours. A student may choose to listen to their professor's explanation, but they are still free to interrupt the professor and direct the communication; perhaps informing the professor that they are clear on the aspect that the professor is explaining, but are unclear on another aspect. The ability of the student to direct the communication makes office hours more efficient.



(a) PR2 from Willow Garage



(b) STAIR from Stanford



(c) ASIMO from Honda

Figure 1.1: Three examples of modern general purpose robots. PR2 image from http://www.willowgarage.com/pages/pr2/overview, ©Willow Garage. STAIR image from http://stair.stanford.edu/ ©Stanford University. ASIMO image from http://world.honda.com/ASIMO/ ©Honda Motor Co.

Chapter 2

Framework

2.1 POMDP Review

A partially observable Markov decision process (POMDP) provides a standard way of representing sequential decision problems where the world state transitions stochastically and the agent perceives the world state through stochastic observations. A standard representation allows for the decoupling of problem specification and problem solvers. Once a problem is represented as a POMDP, any number of POMDP solvers can be applied to solve the problem. A POMDP solver takes the POMDP specification and returns a "policy", which is a mapping from belief states to actions. POMDPs have been successfully applied to problems as varied as autonomous helicopter flight [21], mobile robot navigation [29], and action selection in nursing robots [26].

In this section we review the POMDP, including Bayesian inference, Belman's equation, and the state of the art in POMDP solvers. For additional reading on

POMDPs see [36], [43], and [12].

2.1.1 Definitions

Random variables will be designated with a capital letter or a capitalized word; e.g. X. The values of the random variable will be written in lowercase; e.g. $X = x$. If the value of a random variable can be directly observed then we will call it an "observable" random variable. If the value of a random variable cannot be directly observed then we will call it a "hidden" random variable, also known as a "latent" random variable. If the random variable is "sequential", meaning it changes with time, then we will provide a subscript to refer to the time index; e.g. M_t below. A random variable can be multidimensional; e.g. the state S below is made up of other random variables: Mov, M_t, etc. If a set of random variables contains at least one hidden random variable then we will call it "partially observable".

$P(X)$ is the probability distribution defined over the domain of the random variable X. $P(x)$ is the value of this distribution for the assignment of x to X. $P(X|Y)$ is a "conditional" probability distribution, and defines the probability of a value of X given a value of Y. A "marginal" probability distribution is a probability distribution that results from summing or integrating out other random variables; e.g. $P(X, Y) : \forall_{x \in X, y \in Y} P(x, y) = \sum_{z \in Z} P(x, y, z)$. When a probability distribution has a specific name, such as the "transition model" or the "observation model" we will use associated letters for the probability distribution; e.g. $T(...)$ or $\Omega(...)$.

2.1.2 POMDP Specification

A POMDP specification is an eight element tuple: $\langle S, A, O, T, \Omega, C, \gamma, b_0 \rangle$. S is the set of possible world states; A is the set of possible actions; O is the set of possible observations that the agent may measure; T is the transition model defining the stochastic evolution of the world (the probability of reaching state $s' \in S$ given that the action $a \in A$ was taken in state $s \in S$); Ω is the observation model that gives the probability of measuring an observation $o \in O$ given that the world is in a state $s \in S$; C is a cost function which evaluates the penalty of a state $s \in S$ or the penalty of a probability distribution over states (called a belief state b, $b : \forall_{s \in S} \left[0 \leq b(s) \leq 1 \textit{ and } \sum_{s \in S} b(s) = 1\right]$); γ is the discount rate for the cost function; and, finally, b_0 is the initial belief state for the robot, i.e. the initial probability distribution over world states. Given a POMDP representation, a POMDP solver seeks a policy π, $\pi(b) : b \rightarrow a$, that minimizes the expected sum of discounted cost.[1], The cost is given by C, the discounting is given by γ, and the expectation is computed using T and Ω.

The state of the robot S should capture all quantities relevant to the decision making process. For example, the state for a path planning robot might consist of the two dimensional position of the robot ($S = (X, Y)$). An assignment to all of the quantities in S is often called a hypothesis. The number of hypotheses is the number of joint assignments to all quantities in S. A belief state b is a probability distribution over hypotheses; i.e. it assigns a probability to every hypothesis. Often

[1]Often a reward function is used instead of a cost function, but these are interchangeable; minimizing the cost function C is the same as maximizing the reward function -C.

this is explicitly represented as an array of probabilities, where each element is the probability of one assignment to S. One of these arrays represents one belief state. b_0 might be initialized to a uniform distribution; i.e. each element of the array has the same value: $\left(\frac{1}{size(array)}\right)$.

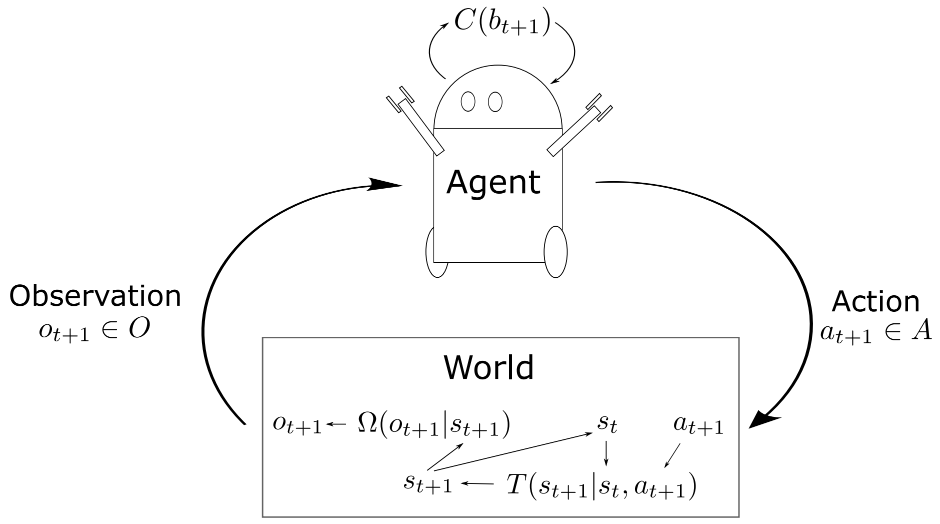

$C(b_{t+1})$
Agent
Observation
$o_{t+1} \in O$
Action
$a_{t+1} \in A$
World
$o_{t+1} \leftarrow \Omega(o_{t+1}|s_{t+1})$
s_t
a_{t+1}
$s_{t+1} \longleftarrow T(s_{t+1}|s_t, a_{t+1})$

Figure 2.1: The POMDP world view. In one timestep the agent selects an action and receives an observations from the world. The agent incurs costs associated with its updated belief based the action and the observation. The agent models the world by T and Ω; the new state is sampled from T, and the observation received is sampled from Ω.

Figure 2.1 depicts the problem that a POMDP represents. An agent performs an action $a \in A$, given this action the world state changes according to the transition model T. Given the new world state, an observation $o \in O$ is generated according to the observation model Ω. The agent receives this observation, updates its internal belief about the true world state, and incurs a cost $C(b)$ associated with this new belief. The goal of the agent is to choose actions that minimize the sum of costs over its lifetime, discounted by γ.

In the next two sections I show mathematically how the agent updates its belief

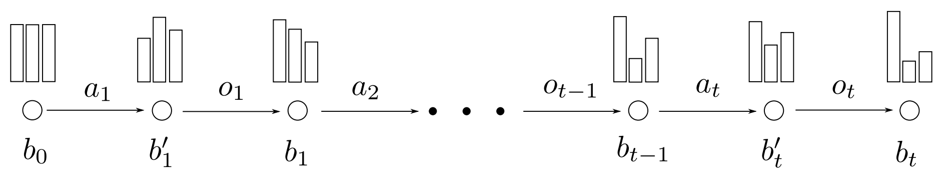


Figure 2.2: An illustration of history as seen from the POMDP perspective. Circles represent beliefs based on a history of actions and observations. The label of the belief is shown below a circle, a cartoon belief histogram is shown above the circle, and arrows are marked by the action or observation that effected the new belief. In one time step the robot receives an action a_t and an observation o_t; the action a_t moves the robot's belief state from b_{t-1} to the intermediate belief state b'_t, and the observation o_t moves the robot's belief state from the intermediate belief state b'_t to the new belief state b_t.

from one timestep to the next and I formally define the equation that the agent seeks to minimize. Note that due to the complexity of literally implementing this update and minimization, nearly all POMDP solvers approximate the update and/or the minimization.

2.1.3 Bayes Filtering (Inference)

The agent starts each timestep with a belief (b_0 for timestep zero), it then takes an action and receives a measurement related to the world state at the next timestep. These two pieces of information a_{t+1} and o_{t+1} are all the agent has to update its belief about the world from b_t to b_{t+1}. If we introduce an intermediate belief state b'_{t+1}, which captures the belief after incorporating a_{t+1}, but before receiving o_{t+1}, we get the graphically depicted scene in figure 2.2.

The beliefs can be updated recursively using the following two formulas, which are the Bayes filter update equations.

Update Equations:

$$\begin{aligned} b'_{t+1}(s_{t+1}) &= \sum_{s_t \in S} T(s_{t+1}|a_{t+1}, s_t) b_t(s_t) && (2.1) \\ b_{t+1}(s_{t+1}) &= \eta \Omega(o_{t+1}|s_{t+1}) b'_{t+1}(s_{t+1}) && (2.2) \end{aligned}$$

$b_0(s_0)$ is defined to be the probability of the state at time zero; $b_0(S_0) = P(s_0)$. This is called the prior distribution for the system's state and is specified ahead of time.

This update from b_t to b_{t+1}, given a_{t+1} and o_{t+1}, is called the Bayes filter. Most filtering algorithms are Bayes filters, notably the Kalman filter and the particle filter [43].

I will now derive equations 2.1 and 2.2, but first an additional notation is helpful. For a temporal random variable X, we denote $x_{t:1}$ to be an assignment of values to X for each of the timesteps from 1 to t; i.e. $(x_t, x_{t-1}, x_{t-2}, \ldots, x_2, x_1)$.

The recursive expression for the belief $b_{t+1}(s_{t+1})$ in terms of the belief $b_t(s_t)$ is derived as follows:

$$\begin{aligned} b_{t+1}(s_{t+1}) &= P(s_{t+1}|a_{t+1:1}, o_{t+1:1}) && (2.3) \\ &= \frac{P(o_{t+1}|s_{t+1}) P(s_{t+1}|a_{t+1:1}, o_{t:1})}{P(o_{t+1}|a_{t+1:1}, o_{t:1})} && (2.4) \\ &= \frac{P(o_{t+1}|s_{t+1}) \sum_{s_t \in S} P(s_{t+1}, s_t|a_{t+1:1}, o_{t:1})}{P(o_{t+1}|a_{t+1:1}, o_{t:1})} && (2.5) \\ &= \frac{P(o_{t+1}|s_{t+1}) \sum_{s_t \in S} P(s_{t+1}|a_{t+1}, s_t) P(s_t|a_{t+1:1}, o_{t:1})}{P(o_{t+1}|a_{t+1:1}, o_{t:1})} && (2.6) \\ &= \eta P(o_{t+1}|s_{t+1}) \sum_{s_t \in S} P(s_{t+1}|a_{t+1}, s_t) P(s_t|a_{t+1:1}, o_{t:1}) && (2.7) \\ &= \eta P(o_{t+1}|s_{t+1}) \sum_{s_t \in S} P(s_{t+1}|a_{t+1}, s_t) P(s_t|a_{t:1}, o_{t:1}) && (2.8) \\ &= \eta \Omega(o_{t+1}|s_{t+1}) \sum_{s_t \in S} T(s_{t+1}|a_{t+1}, s_t) b_t(s_t) && (2.9) \end{aligned}$$

Line 2.3 is the definition of the belief state $b_{t+1}(s_{t+1})$; i.e. the probability distribution over states given the full action and observation history. Line 2.4 uses Bayes rule to pull out o_{t+1} from the history and the fact that an observation o_{t+1} is independent of the history, given the current state s_{t+1}. Line 2.5 introduces s_t using the law of total probability. Line 2.6 uses the definition of conditional probability and the fact that the next state s_{t+1} is independent of the history, given the action taken a_{t+1} and the previous state s_t (Markov property). Line 2.7 uses the fact that the denominator is not a function of the variable of interest for the probability distribution (s_{t+1}), thus it is constant for all assignments to s_{t+1} and we can recover its value after the update; it is one over the sum of the unnormalized distribution. In line 2.8 the a_{t+1} is dropped. This is typically justified for pure filtering problems by saying that future actions are randomly chosen. In a control problem actions are determined by a policy $(a_{t+1} = \pi(b_t))$. So the explanation is more complicated,

$$\begin{aligned} P(s_t|a_{t+1:1}, o_{t:1}) &= \frac{P(a_{t+1}|s_t, a_{t:1}, o_{t:1})P(s_t|a_{t:1}, o_{t:1})}{P(a_{t+1}|a_{t:1}, o_{t:1})} && (2.10) \\ &= \frac{P(a_{t+1}|a_{t:1}, o_{t:1})P(s_t|a_{t:1}, o_{t:1})}{P(a_{t+1}|a_{t:1}, o_{t:1})} && (2.11) \\ &= P(s_t|a_{t:1}, o_{t:1}) && (2.12) \end{aligned}$$

Line 2.10 is from Bayes Rule and 2.11 is because the action a_{t+1} is independent of the true state, since it is chosen based on the belief state b_t which is a function only of the history. Finally, in the derivation of $b_{t+1}(s_{t+1})$, line 2.9 substitutes Ω, T, and b_t in place of their definitions.

For implementation we do this update in two steps, one for the action, which leads to an intermediate belief state $b'_{t+1}(s_{t+1})$. This intermediate belief state is the belief after incorporating the action but before incorporating the measurement.

$$\begin{aligned}
b'_{t+1}(s_{t+1}) &= P(s_{t+1}|a_{t+1}, a_{t:1}, o_{t:1}) && (2.13)\\
&= \sum_{s_t \in S} P(s_{t+1}|a_{t+1}, s_t) P(s_t|a_{t+1}, a_{t:1}, o_{t:1}) && (2.14)\\
&= \sum_{s_t \in S} P(s_{t+1}|a_{t+1}, s_t) P(s_t|a_{t:1}, o_{t:1}) && (2.15)\\
&= \sum_{s_t \in S} T(s_{t+1}|a_{t+1}, s_t) b_t(s_t) && (2.16)
\end{aligned}$$

To incorporate the observation into the belief we plug equation $b'_{t+1}(s_{t+1})$ into equation (2.9). This gives us our two recursive update equations mentioned above:

$$b'_{t+1}(s_{t+1}) = \sum_{s_t \in S} T(s_{t+1}|a_{t+1}, s_t) b_t(s_t) \quad (2.1)$$

$$b_{t+1}(s_{t+1}) = \eta \Omega(o_{t+1}|s_{t+1}) b'_{t+1}(s_{t+1}) \quad (2.2)$$

2.1.4 Belman's Equation (Planning)

Intuitively, certain belief states are more attractive to the agent then others. Not just because they receive a low immediate cost $C(b)$ but because they are on a path that will have a low sum of costs.

Let $EC(b)$, formally defined below, represent how much the agent dislikes a belief; i.e. the immediate cost plus the long run cost. Given $EC(b_{t+1})$ for each belief state one time step away, i.e. reachable by one action and one observation, depicted in figure 2.3, we can ask two important questions: 1) what action should the robot take in the current state?, and 2) what is $EC(b_t)$ for the current belief b_t?

Referencing figure 2.3, these questions assume that we are given the four $EC(b_{t+1})$ for each of the four leaf nodes. We can compute the $EC(b'_{t+1})$ for the two b'_{t+1} by

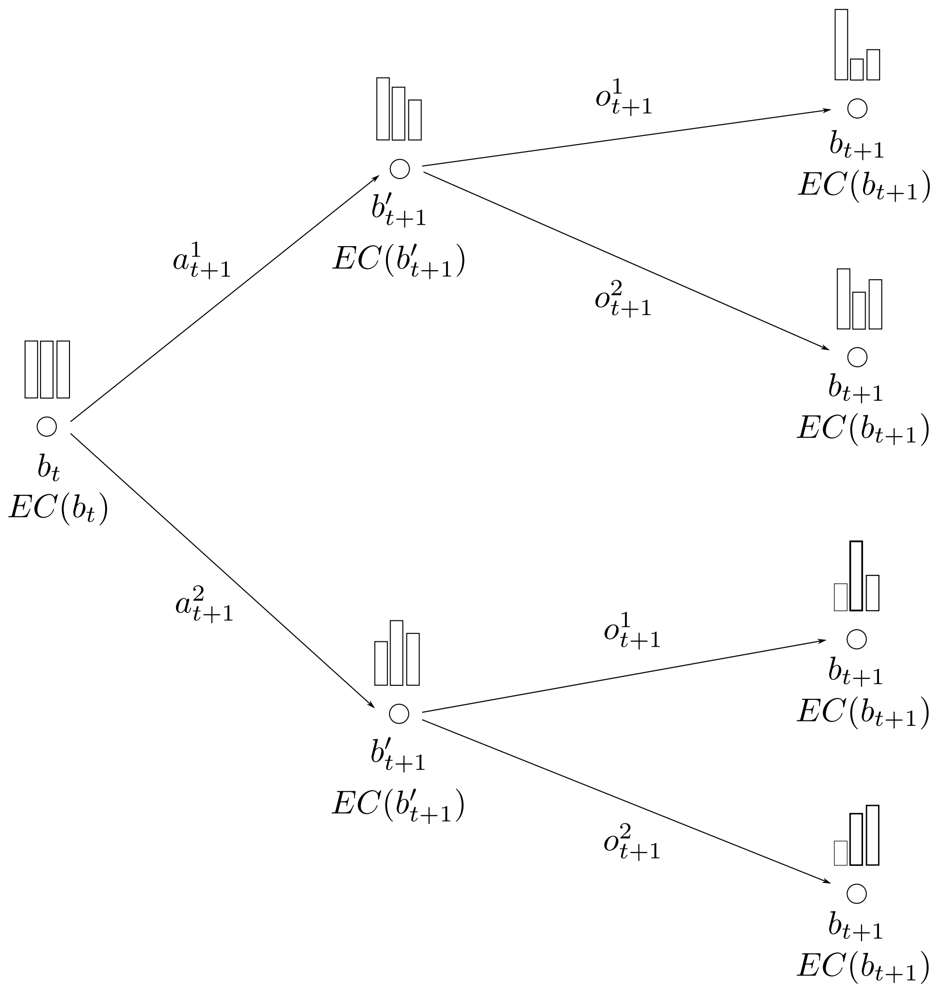


Figure 2.3: A belief tree expanded one time step into the future for a POMDP with two actions (a^1, a^2) and two observations (o^1, o^2). The belief label b_i is shown below the node, a cartoon histogram of the belief is shown above the node, and the expected sum of discounted costs $EC(b_i)$ from the belief b_i onward is shown below the belief. The $EC(b_i)$ are useful for choosing optimal actions. The actions and observations that effect the beliefs are shown on their arrows. Beliefs are propagated from the left to the right according to the Bayes filter equations (2.1 and 2.2). $EC(b_i)$ are propagated from right to left using Belman's equation (2.22)).

weighting each $EC(b_{t+1})$ by the probability of the observation that led to its belief node.

$$\begin{aligned} EC(b'_{t+1}) &= \sum_{o_{t+1}} p(o_{t+1}|a_{t+1}, a_{t:1}, o_{t:1}) EC(b_{t+1}) && (2.17) \\ &= \mathop{\mathbb{E}}_{o_{t+1}} EC(b_{t+1}) && (2.18) \end{aligned}$$

The optimal action is then just the action that leads to the b'_{t+1} with the smallest $EC(b'_{t+1})$.

$$\begin{aligned} a_{t+1} &= \arg\min_{a_{t+1}} EC(b'_{t+1}) && (2.19) \\ &= \arg\min_{a_{t+1}} \mathop{\mathbb{E}}_{o_{t+1}} EC(b_{t+1}) && (2.20) \end{aligned}$$

And $EC(b_t)$ is the immediate cost $C(b_t)$ plus the $EC(b'_{t+1})$ under the optimal action, discounted:

$$\begin{aligned} EC(b_t) &= C(b_t) + \gamma \min_{a_{t+1}} EC(b'_{t+1}) && (2.21) \\ &= C(b_t) + \gamma \min_{a_{t+1}} \mathop{\mathbb{E}}_{o_{t+1}} EC(b_{t+1}) && (2.22) \end{aligned}$$

Equation 2.22 is called Belman's equation and is the recursive constraint that guarantees optimal action selection.

An intuitive, though computationally demanding, POMDP solver would, starting at the current belief, roll out the action selection tree of figure 2.3 to a finite horizon T. For each leaf b_T it could approximate $EC(b_T)$ as

$$EC(b_T) = C(b_T). \qquad (2.23)$$

It could then back up the EC using Belman's equation (equation 2.22; taking expectations of observation branches and minimums of action branches), until $EC(b_{t+1})$ for all beliefs b_{t+1} had been computed. Finally it would select the optimal action using equation 2.20.

Derivation

I now derive why Belman's equation enforces optimal action selection, and I formally define EC in the process. By definition, the optimal next action is the action that minimizes the expected sum of discounted costs over action and observation futures:

$$a_{t+1} = \underset{a_{t+1}}{\arg\min} \underset{o_{t+1}}{\mathbb{E}} \left[C(b_{t+1}) + \gamma \min_{a_{t+2}} \underset{o_{t+2}}{\mathbb{E}} \left[C(b_{t+2}) + \gamma \min_{a_{t+3}} \underset{o_{t+3}}{\mathbb{E}} [\dots] \right] \right] \tag{2.24}$$

We define $EC(b_{t+1})$ as the quantity in the outer square brackets,

$$EC(b_{t+1}) = C(b_{t+1}) + \gamma \min_{a_{t+2}} \underset{o_{t+2}}{\mathbb{E}} \left[C(b_{t+2}) + \gamma \min_{a_{t+3}} \underset{o_{t+3}}{\mathbb{E}} [\dots] \right]. \tag{2.25}$$

We can express $EC(b_{t+1})$ recursively in terms of $EC(b_{t+2})$ by substituting $EC(b_{t+2})$ into equation 2.25 to get,

$$EC(b_{t+1}) = C(b_{t+1}) + \gamma \min_{a_{t+2}} \underset{o_{t+2}}{\mathbb{E}} EC(b_{t+2}). \tag{2.22}$$

This is Bellman's equation. Substituting $EC(b_{t+1})$ into the optimal action equation 2.24, gives,

$$a_{t+1} = \underset{a_{t+1}}{\arg\min} \underset{o_{t+1}}{\mathbb{E}} EC(b_{t+1}). \tag{2.20}$$

Thus, if we have $EC(b_i)$ for which Belman's equation holds, then the actions selected by equation 2.20 are optimal.

Lastly, in Belman's equation we take an expectation over observations ($\mathbb{E}_{o_{t+1}} EC(b_{t+1})$), I now express this expectation in terms from the previous section on Bayes filtering.

$$\underset{o_{t+1}}{\mathbb{E}} EC(b_{t+1}) = \sum_{o_{t+1} \in O} p(o_{t+1}|a_{t+1}, o_{t:1}, a_{t:1}) EC(b_{t+1}) \tag{2.26}$$

where

$$\begin{aligned} p(o_{t+1}|a_{t+1}, o_{t:1}, a_{t:1}) &= \sum_{s' \in S} P(o_{t+1}|s') P(s'|a_{t+1}, o_{t:1}, a_{t:1}) && (2.27) \\ &= \sum_{s' \in S} P(o_{t+1}|s') \sum_{s_t \in S} P(s'|a_{t+1}, s_t) P(s_t|o_{t:1}, a_{t:1}) && (2.28) \\ &= \sum_{s' \in S} \Omega(o_{t+1}|s') \sum_{s_t \in S} T(s'|a_{t+1}, s_t) b(s_t) && (2.29) \end{aligned}$$

2.1.5 POMDP Solvers

The goal of a POMDP solver is to choose an action a for a belief state b that minimizes the expected sum of discounted rewards (equation 2.24). POMDP solvers can be classified into two broad categories, *offline* or *online*. An offline solver does all of its processing before the agent is run and produces a policy $\pi(b)$, which maps every belief state b to an action a. An online solver uses the time between actions to compute the next action a_{t+1} given the current belief state b_t.

Both offline and online solvers have their tradeoffs. An offline solver generally has more time for computation but the computation must be spent on a range of belief states, since the policy must specify an action for any belief state. Also, since the policy returned by an offline solver is typically a simple mapping, it can be rapidly evaluated by the running agent, which can be important if the processing time between actions is limited. In contrast, an online solver has less time for computation (only the time between actions) but it can focus this processing on the immediately relevant belief states.

There is strong overlap between offline and online approaches. Advances in one can often be applied to others. And, in general, offline processing policies can be used to improve the quality of online policies. The best performing systems make use of all of the online processing available and augment this with a policy from an offline solver, see the heuristic solver below. Here is a brief overview of several offline and online POMDP solvers.

Offline Solvers

Most offline solvers (included all but one of the reviewed solvers) solve the POMDP by seeking the expected cost for all belief states $EC(b)$ (equation 2.25). The optimal action can then be determined by either direct lookup (often the optimal action that lead to minimizing $EC(b)$ is stored), or by using equation 2.20 to compute the optimal action in terms of $EC(b)$.

- **exact expected cost** — Early on it was shown that the expected cost of a belief state b_t can be expressed as a concave linear function of b_t, where the parameters are derived from the expected cost for one time step in the future EC_{t+1} (which is also a convex linear function of the belief state b_{t+1} [35]). By starting with $EC(b) = C(b)$, and repeatedly computing the expected cost one timestep early, as the number of updates goes to infinity, the expected cost approaches the true expected cost (equation 2.25). Unfortunately the number of linear equations that make up the expected cost grows exponentially with each update, thus this approach is only appropriate in extremely simple domains.

- **point based** — Point based POMDP solvers also solve Belman's equation (equation 2.22), but only for a small set of beliefs [18]. Implementations vary on how they select the belief set. The distribution of the belief set is critical to the accuracy of the solution. In general, the more beliefs in the set, the more accurate the estimate of expected cost, but, also, the more processing required. Two recent algorithms using the point based approach are PBVI [25] and Perseaus [37].

- **upper bound** — Some solvers return strict upper or lower bounds for the expected cost function $EC(b)$. These can be useful as heuristics for online solvers. One example of an upper bound solver evaluates the expected cost of always executing the same action, called a "blind" policy [11]. Since these policies are independent of observations, their expected cost can be solved with an MDP-like value iteration. $EC(b)$ is then computed in the same way as the MDP lower bound example below. Even a bound as loose as this can be helpful as a heuristic [28]. A tighter upper bound can be achieved using a point based solver, but this comes at the cost of more computation.

- **lower bound** — A lower bound solver computes a strict lower bound on $EC(b)$. One example of a lower bound solver is to solve the underlying MDP, which make the assumption that the state is observable [17]. Let $EC^{MDP}(s)$ be the expected cost under this assumption for the state s. We then compute $EC(b)$ as $EC(b) = \sum_s EC^{MDP}(s)b(s)$. Solving the underlying MDP results in a lower bound because it ignores uncertainty, and is thus overly optimistic. Recent lower bound POMDP solvers include QMDP [17], and FIB [11].

- **policy search** — A policy search method directly modifies a parameterized policy. If we can efficiently compute the expected cost of a policy, and if the parameterized policy is differentiable, then we can apply gradient descent methods directly on the policy [1]. Applications where a differentiable policy is appropriate are more common in control than in artificial intelligence. If the conditions are met, a policy search algorithm can be an efficient solver.

- **permutable POMDPs** — A permutable POMDP is a sub-class of POMDPs [7]. In many applications the optimal policy only depends on the shape of the current belief and not on the value of a state variable. For example, for a telephone directory agent, the agent may be seeking the first name of the person you want to reach. The optimal policy is independent of the value of the first name. If the belief were in a particular shape, the optimal policy would "ask confirmation of the most probable value for the first name"; whether to ask this question would not depend on the value of the most probable first name. Solvers can take advantage of the permutable property by computing expected costs only for a sorted belief state. Because the states are permutable, this also provides the expected cost for any permutation of that belief state. Simple transformations to and from the sorted belief state are used online to extract the expected cost for the current belief state. Doshi and Roy showed that this results in an exponential reduction in the belief space, making the POMDP easier to solve [7]. In their implementation, they wrapped these transformations within a point based value iteration solver, but it should be broadly applicable to most POMDP solvers when the POMDP has the requisite permutable structure.

Online Solvers

Most online solvers, including those presented here, recommend an action by expanding the belief tree (figure 2.3), evaluating the expected cost of leaf nodes, and backing them up, using Belman's equation (equation 2.22), to the current node. The action recommended is the very next action that resulted in the current belief node's minimum value. These approaches differ in how they expand this tree, as described below.

- **branch and bound** — Branch and bound techniques maintain a lower bound and an upper bound on EC for each each node in the tree [24]. If the lower bound for one action node a' is higher than the upper bound for another action, then we can stop exploring all branches below a', since the other action is guaranteed to result in a lower EC. The full process is as follows: the tree is expanded to a depth; the upper and lower bounds are computed for the leaf nodes (typically using an offline solution); these bounds are propagated up the tree; branches are then pruned beyond actions that will never be taken; and the the process repeats, expanding the tree from the remaining leaf nodes. This pruning saves significant computation, but we can do even better; see the heuristic solver below.

- **monte carlo** — A monte carlo solver also expands the belief tree, but stochastically traverses observation branches based on the probability of that observation [19]. This is effective because it steers the search towards observations that are more likely.

- **heuristic** — Heuristic solvers are similar to branch and bound solvers in that they maintain an upper and lower bound for each node's expected cost, but unlike branch and bound they do not uniformly expand leaf nodes. Heuristic solvers apply a heuristic function to all leaf nodes and expand the node with the best value. One effective heuristic is the contribution of that leaf nodes error (upper bound - lower bound) to the root node's error [27]. A leaf node's error contributes to the root's error in proportion to the discounted probability of reaching that leaf. This heuristic encourages the expansion of leaf nodes that will aid in the immediate decision of which action to take.

A good overview with references to further reading on POMDP solvers can be found in Ross et al. [28]. The current state of the art in POMDP solvers are heuristic methods with simple upper and lower bounds computed by an offline solver. For human-robot task communication in complex task domains, a reasonable option for a POMDP solver would be the combination of a heuristic solver (using "blind" and QMDP for bounds) with the approach of mapping to a reduced planning space (from the spoken dialog manager work in section 1.2). If the number of observations makes the evaluation of all leaf nodes intractable, then the monte carlo approach could be used to select a subset of leaf nodes to evaluate for expansion.

2.2 Task Communication as a POMDP

This dissertation proposes the use of a POMDP for representing the problem of a human communicating a task to a robot. Specifically, for the elements of the POMDP tuple $\langle S, A, O, T, \Omega, C, \gamma, b_0 \rangle$, the partially observable state S captures the details of the task along with the mental state of the human (helpful for interpreting ambiguous signals); the set of actions A capture all possible actions of the robot during communication (for example words, gestures, body positions, etc.); the set of observations O capture all signals from the human (be they words, gestures, buttons, etc.); and the cost function C should encode the desire to minimize uncertainty over the task details in S. The transition model T, the observation model Ω, the discount rate γ, and the initial belief state b_0 fill their usual POMDP roles.

The next chapter provides a demonstration of representing a human-robot task communication problem as a POMDP, including examples of T, Ω, and b_0.

2.2.1 Choice of Cost Function

I say above that the cost function C should encode the desire to minimize uncertainty over the task details in S; i.e. the cost function should penalize uncertainty. As mentioned in section 1.2.5, in the field of spoken dialog managers, the cost function is chosen to penalize communication time and incorrect submission of the quantities being communicated, and reward correct submissions. The system still explores (reducing uncertainty about the quantities), but only in pursuit of timely, correct submissions. Unlike an uncertainty penalizing cost function, this cost function has the added benefit of being linear in the belief state, which is a requirement of some POMDP solvers. Eventually, the robot will be in a situation where communication must be terminated in order to perform another function, such as task execution; but for this dissertation, the setting is solely task communication. As such, a terminal submit action is not appropriate, since it would end the robot's actions and prevent further communication. An uncertainty penalizing cost function is promoted because it focuses the robot's actions on task communication, which is the problem at hand, with the added benefit of being parameter free. That said, I do view this as a temporary cost function until a more encompassing cost function is developed for the broader problem of task communication and task execution (see section 5.2.5).

Chapter 3

Demonstration

In this chapter I provide a demonstration of representing a human-robot task communication problem as a POMDP. The representation is what is used in the experiment in chapter 4. The task to be communicated relates to a simulated environment shown in figure 3.1. As such we begin with a description of the simulator and its virtual world.

3.1 Simulator

The virtual world is shown in figure 3.1. It consists of 3 balls and the robot, displayed as circles and a square (3.1.a). The robot can "gesture" at balls by lighting them (3.1.b), it can pick up a ball (3.1.d), it can slide one ball to one of four distances from another ball (3.1.e), and it can signal that it knows the task by displaying "Final" (3.1.f). The experiment in chapter 4 will contain trials in which a human acts as the robot. For these comparison trials the robot actions are controlled by left

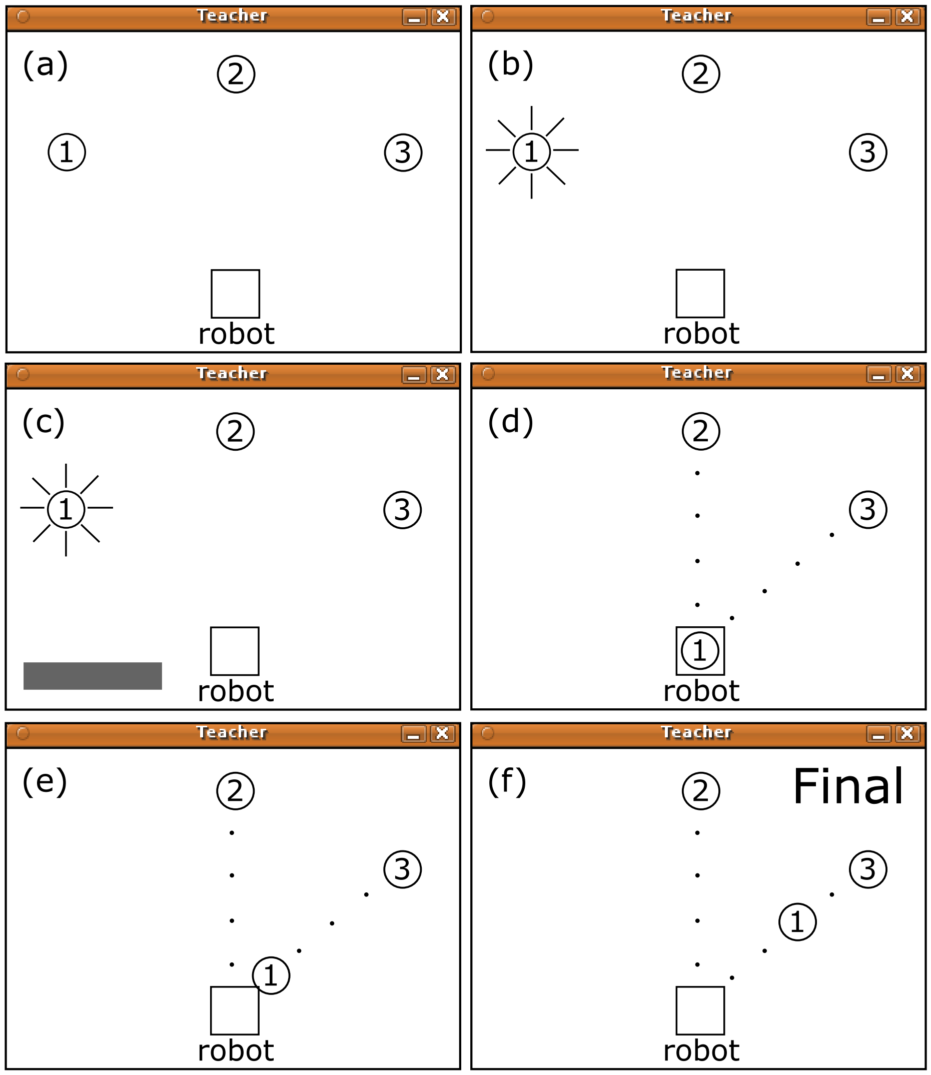


Figure 3.1: Typical human-robot interactions on the simulator. (a) The state all trials start in; the square robot, holding no balls, surrounded by the three balls. (b) The robot lighting one of the balls, as if to ask, "move this?" (c) The human teacher pressing the keyboard spacebar, displayed in the simulator as a rectangle, and used to indicate approval with something the robot has done. (d) The robot holding one of the balls, the four relative distances are displayed to the remaining two balls. (e) The robot has slid one of the balls to the furthest distance from another ball. (f) the robot has displayed "Final", indicating that it knows the task, and that the world is currently in a state consistent with that task.

and right mouse clicks on the objects involved in the action; e.g. right click on a ball to light it or turn off the light, left click on a ball to pick it up, etc. In this simulator the teacher input is highly constrained; the teacher has only one control and that is the keyboard spacebar, visually indicated by a one half second rectangle (3.1.c), and is used to indicate approval with something that the robot is doing or has done. The simulator is networked so that two views can be opened at once; this is important for the comparison trials, where the human controlling the robot must be hidden from view.

Timesteps are 0.5 second long, i.e. the robot receives an observation and must generate an action every 0.5 seconds. The simulator is free running, so, in the comparison trials where the human controls the robot, if the human does not select an action, then the “no action” action is taken. “no action” actions are still taken in the case of the POMDP-controlled robot, but they are always intentional actions that have been selected by the robot. I provide enough processing power so that the POMDP-controlled robot always has an action ready in the allotted 0.5 seconds. The simulated world is discrete, observable, and deterministic.

3.2 Toy Problem

The problem we wish to encode is as follows. A human teacher will try to communicate, through only spacebar presses, that a specific ball should be at a specific distance from another specific ball. Spacebar presses from teacher should be interpreted by the robot as approval of something that it is doing. The robot has to infer the relationship the teacher is trying to communicate from the spacebar presses. When the robot thinks it knows the relationship, it should move the world to that relationship and display “Final” to the teacher.

Figure 3.1 shows snap shots from a possible communication trial. The robot "questions" which ball to move (3.1.b), the teacher indicates approval (3.1.c), the robot picks up the ball (3.1.d), the robot "questions" which ball to move the one it is holding to (not shown), the robot slides the ball toward another ball (3.1.e), the teacher approves a distance or progress toward a distance (not shown), and, after further exploration, the robot indicates that it knew the task by displaying "Final" (3.1.f).

Although this problem is simplistic, a robot whose behaviors consist of chainings of these simple two object relationship tasks could be useful; e.g. for the "set the table" task: move the plate to zero inches from the placemat, move the fork to one inch from the plate, move the spoon to one inch from the fork, etc.

I chose the spacebar press as the input signal for the demonstration and for the experiment because it carries very little information, requiring the robot to infer meaning from context, which is a strength of this approach. For a production robot, this constrained interface should likely be relaxed to include signals such as speech, gestures, or body language. These other signals are also ambiguous, but the simplicity of a spacebar press made the uncertainty obvious for the demonstration.

3.3 Formulation

In this section I formulate this problem as a POMDP. This is only one of many possible formulations. It is perhaps useful to note that the formulation presented here, and used in the user experiment below, was the first attempt; neither the structure nor the parameters needed to be adjusted from my initial guesses. This suggests that the proposed approach is reasonably insensitive to modeling decisions.

3.3.1 State (S)

The state S is composed of hidden and observable random variables. The task that the human wishes to communicate is captured in three hidden random variables Mov, WRT, and $Dist$. Mov is the index of the ball to move $(1-3)$. WRT is the index of the ball to move ball Mov with respect to. $Dist$ is the distance that ball Mov should be from ball WRT.

The state also includes a sequential hidden random variable, M_t, for interpreting the observations O_t. M_t takes on one of five values: (*waiting*, *mistake*, *that_mov*, *that_wrt*, or *that_dist*). A value of *waiting* implies that the human is waiting for some reason to press the spacebar. A value of *mistake* implies that the human accidentally pressed the spacebar. A value of *that_mov* implies that the human pressed the spacebar to indicate approval of the ball to move. A value of *that_wrt* implies that the human pressed the spacebar to indicate approval of the ball to move ball Mov with respect to. A value of *that_dist* implies that the human pressed the spacebar to indicate approval of the distance that ball Mov should be from ball WRT. In addition to these hidden random variables, the state also includes observable random variables for the physical state of the world; e.g. which ball is lit, which ball is being held, etc.

Finally, the state includes "memory" random variables for capturing historical information, e.g. the last time step that each of the balls were lit, or the last time step that $M = that\_mov$. The historical information is important for the transition model T. For example, humans typically wait one to five seconds before pressing the spacebar a second time. In order to model this accurately we need the time step of the last spacebar press. See appendix B for a detailed description of the full state along with examples of the observable state variables for several configurations of the world.

state	Mov
	WRT
	Dist
	M
	⟨world state variables⟩
	⟨historical variables⟩

3.3.2 Actions (A)

There are six parameterized actions that the robot may perform. Certain actions may be invalid depending on the state of the world. The actions are: *noa*, for performing no action and leaving the world in the current state; $light\_on(index)$ or $light\_off(index)$, for turning the light on or off for the ball indicated by *index*; $pick\_up(index)$, for picking up the ball indicated by index; $release(index)$, for putting down the ball indicated by index; and $slide(index\_1, distance, index\_2)$, for sliding the ball indicated by $index\_1$ to the distance indicated by *distance* relative to the ball indicated by $index\_2$. Note that only a few actions are valid in any world state; for example, $slide(index\_1, distance, index\_2)$ is only valid if ball $index\_1$ is currently held or currently at a distance from ball $index\_2$ and if *distance* is only one step away from the current distance. See appendix C.1 for effects of these actions.

actions	noa
	light_on(index)
	light_of(index)
	pick_up(index)
	release(index)
	slide(index_1, distance, index_2)

3.3.3 Observations (O)

An observation takes place at each time step and there are two valid observations: *spacebar* or *no_spacebar*, corresponding to whether the human pressed the spacebar on that time step.

observations	spacebar
	no_spacebar

3.3.4 Transition Model (T)

A transition model gives the probability of reaching a new state, given an old state and an action. In this example, Mov, WRT, and $Dist$ are non-sequential random variables, meaning they do not change with time, so $T(Mov = i, ...|Mov = i, ...) = 1.0$. The transition model for the physical state of the virtual world is also trivial, since the virtual world is deterministic.

The variable of interest in this example for the transition model is the sequential random variable M that captures the mental state of the human (*waiting*, *mistake*, *that_mov*, *that_wrt*, or *that_dist*). The transition model was specified from intuition, but in practice I envision that it would either be specified by psychological experts, or

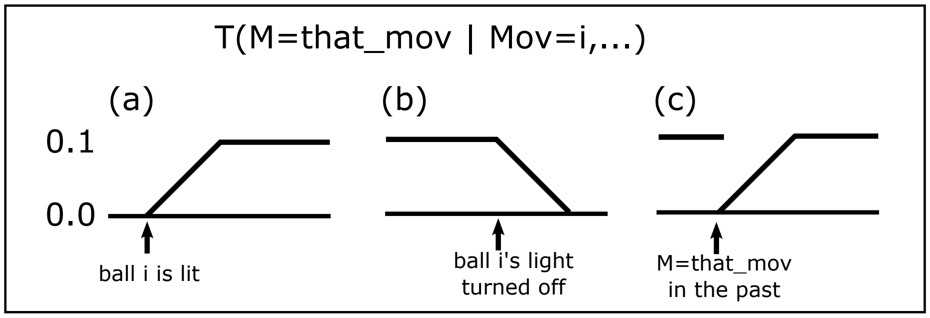


Figure 3.2: This is an illustration of part of the transition model T. Here I show the probability that the human will signal their approval (via a spacebar press) of the ball to be moved, $T(M = that\_mov|Mov = i, ...)$, where, in this hypothesis, the ball to be moved is ball i. (a) once the robot has lit ball i, the probability increases from zero to a peak of 0.1 over 2 seconds. (b) after the light has turned off there is still probability of an approving spacebar press, but decreasing over 2 seconds. (c) If the teacher has signaled their approval ($M = that\_mov$), then the probability resets. The structure and shape of these models was set from intuition.

learned from human-human or human-robot observations [41, 39]. For the experiment I set the probability that M transitions to *mistake* from any state to a fixed value of 0.005, meaning that at any time there is a 0.5% chance that the human will *mistakenly* press the spacebar indicating approval. I define the probability that M transitions to *that_mov*, *that_wrt*, or *that_dist* as a table-top function, as shown in figure: 3.2. I set the probability that M transitions to *waiting* to the remaining probability; $T(M = waiting) = 1 - T(M = mistake \vee that\_mov \vee that\_wrt \vee that\_dist)$. See appendix C for the full transition model.

3.3.5 Observation Model (Ω)

The observation model I have chosen for this problem is a many to one deterministic mapping:

$$P(O = \mathit{spacebar}|M) = \begin{cases} 0.0 & \text{if } M = \mathit{waiting} \\ 1.0 & \text{otherwise} \end{cases}$$

Note that this deterministic model does not imply that the state is observable since, given $O = \mathit{spacebar}$, we do not know why the human pressed the spacebar, $M \overset{?}{=} (\mathit{mistake} \vee \mathit{that\_mov} \vee \mathit{that\_wrt} \vee \mathit{that\_dist})$. [1]

3.3.6 Cost Function (C)

As mentioned earlier, the cost function should be chosen to motivate the robot to quickly and accurately infer what the human is trying to communicate. In our case this is a task captured in the random variables Mov, WRT, and $Dist$. The cost function I have chosen is the entropy of the marginal distribution over Mov, WRT, and $DIST$:

$$C(p) = -\sum_{x} p(x) \cdot log(p(x)). \tag{3.1}$$

Where p is the marginal probability distribution over Mov, WRT, and $Dist$, and x takes on all permutations of the value assignments to Mov, WRT, and $Dist$.

Since entropy is a measure of the uncertainty in a probability distribution, this cost function will motivate the robot to reduce its uncertainty over Mov, WRT, and $Dist$, which is what we want.

[1] If there were noise in the spacebar key then this would not be a deterministic mapping.

3.3.7 Discount Rate (γ)

γ is set to 1.0 in the experiment, meaning that uncertainty later is just as bad as uncertainty now. The valid range of γ for a POMDP solver evaluating actions to an infinite horizon is $0 \leq \gamma < 1.0$, but the solver only evaluates to a 2.5 second horizon. In practice, the larger the value of γ, the more willing the robot is to defer smaller gains now for larger gains later.

3.3.8 Initial Belief (b_0)

The initial distribution b_0 over the joint values of the hidden random variables Mov, WRT, $Dist$, and M are set as follows. M_0 is assumed to be equal to *waiting*. All 24 ($3 \times 2 \times 4 \times 1$) hypotheses, constructed from the permutations of the hidden random variables ($Mov = (1, 2, 3), WRT = (1, 2, 3), Dist = (20, 40, 60, 80), M =$ *waiting*), are set to the uniform probability of 1/24.

3.3.9 Action Selection

The problem of action selection is the problem of solving the POMDP. As described in section 2.1.5, there are many established techniques for solving POMDPs [20, 28]. Given the simplicity of the world and the problem, I can take a direct approach. The robot expands the action-observation tree (figure 2.3) out 2.5 seconds into the future, and takes the action that minimizes the sum of expected entropy over this tree. This solution is approximate, since the system only looks ahead 2.5 seconds, but, as I will show in chapter 4, it results in reasonable action selections for the toy problem used in the demonstration and experiment.

When the marginal probability of one of the assignments to Mov, WRT, and $Dist$ is greater than 0.98 (over 98% confident in that assignment), the robot moves the world to that assignment and displays "Final".[2]

[2]This termination is outside of the proposed POMDP approach of the dissertation. It was implemented in order to collect data in the experiment. The dissertation deals only with task communication, not with termination of communication to perform some other function. In a strict implementation of the proposed approach, the robot would never stop acting to reduce its uncertainty about the task. See section 5.2.5 for future work on the integration of task communication and task execution modes

Chapter 4

Performance

4.1 Experiment

The experiment consisted of multiple trials run on the simulator described in section 3.1, where each trial was one instance of the problem described in section 3.2. In half of the trials the virtual robot was controlled by the POMDP described in section 3.3, and in the other half the virtual robot was controlled by a human hidden from view. At the beginning of each trial the teacher was shown a card designating the ball relationship to teach. The robot, either POMDP or human controlled, had to infer the relationship from spacebar presses. When the robot was confident about the desired relationship it would move the world to that relationship and end the trial by displaying "Final" to the teacher. The teacher would then indicate on paper whether the robot was correct and how intelligent they felt the robot in that trial was.

The experiment involved 26 participants, consisting of undergraduate and graduate students ranging in age from 18 to 31 with a mean age of 22. Four of the participants were randomly selected for the "human robot" role, leaving 22 participants for the "teacher" role. The participants rated their familiarity with artificial intelligence software and systems on a scale from 1 to 7; the mean score was 3.4 with a standard deviation of 1.9. Participants were paid $10.00 for their time in the experiment. See appendix A for the raw data from the experiment.

4.2 Calibration of the Teacher to a Human Robot

The data below is reported on 44 teaching trials: 2 trials for each of the 22 teachers, one teaching the human-controlled robot and one teaching the POMDP-controlled robot. In early trials we realized that the human teacher was not teaching in a way that either the human robot or the POMDP robot expected. Both the human-controlled robot and the POMDP-controlled robot had the model that the teacher would first teach it which ball to move (ball Mov), and then which ball to move it to (ball WRT), but the teacher would often press the spacebar the first time that ball WRT was lit. Both the human and POMDP robot would then pick up ball WRT, thinking it was the ball to move. This would lead to a long trial before the human or POMDP-controlled robot recovered. Research has shown that an inconsistency in models is only temporary; over time humans will adjust to their partner's models [4]. We believe that there was an inconsistency because the spacebar interaction was novel to the teacher. As we move to more natural interactions I expect that the human teacher would be well calibrated to the model of a human student. To achieve calibration in the experiment, each teacher was given three calibration trials with the human-controlled robot (the robot identity was hidden from the teacher).

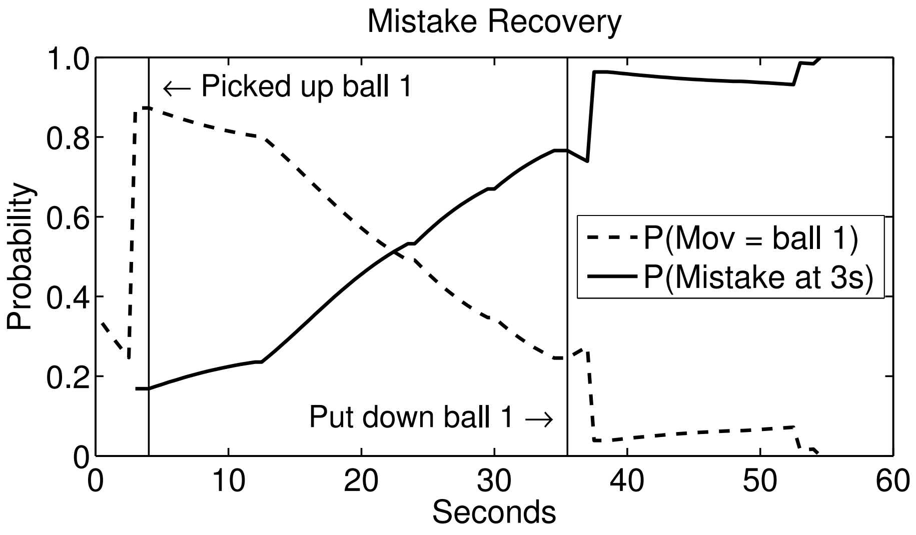


Figure 4.1: This figure shows a typical recovery from a mistaken spacebar press. In this figure the teacher mistakenly pressed the spacebar at the three second mark while the robot was lighting ball 1. The probability that ball 1 was the ball to be moved immediately spiked. At the same time there was a low probability that the spacebar press was a mistake. At 4 seconds the robot picked up ball 1 and started moving it, exploring tasks involving the movement of ball 1. As the trial progressed without further spacebar presses, the probability that the spacebar press at 3 seconds was a mistake increased and the probability that ball 1 was the ball to move decreased. Finally, at 36 seconds the approximately optimal policy was to put down ball 1 and reassess which object was to be moved.

All 22 teachers showed calibration to the human-controlled robot after the first two calibration trials. The three calibration trials were followed by the two experiment trials, one with the human-controlled robot and one with the POMDP-controlled robot (the controller order was randomized).

4.3 Robustness to Teacher Error

The strength of using a probabilistic approach such as a POMDP is in its robustness to noise. In the experiment, noise came in the form of mistaken spacebar presses. Figure 4.1 illustrates a typical mistaken spacebar press. In this trial, at the three second mark, the human mistakenly pressed the spacebar while ball 1 was lit, when in fact ball 1 was not involved in the task. As expected, the robot's marginal probability that ball 1 was the ball to move immediately spiked. Yet there was still a small probability that the random variable M equaled *mistake* at the three second mark. The trial proceeded with the robot making use of the strong belief that ball 1 was the ball to be moved: it picked up ball 1 at 4 seconds and lit ball 2 and ball 3. As time progressed, and the robot did not receive further spacebar presses that would be consistent with a task involving ball 1, the probability that the human mistakenly pressed the spacebar increased and the probability that ball 1 was the ball to move decreased. At thirty six seconds, the belief that a mistake occurred was strong enough that the action which minimized the expected entropy was to put down ball 1 and continue seeking another ball to move.

4.4 Ability to Infer the Task

The second result from the experiment is that the robot accurately inferred the hidden task and the hidden state of the teacher. In all trials the human teachers reported that the robot was correct about the task being communicated. Figure 4.2 shows a look at the robot's marginal probabilities, for one of the trials, of the random variables Mov, WRT, and $Dist$. In this trial, as was typical of the trials, the robot first grew its certainty about Mov followed by WRT and then $Dist$. Figure 4.3 shows the probability of the true assignment to M at the time of the spacebar press and at the end of the trial, for four assignments to the variable M.[1] This shows that as each trial progressed the robot became correctly certain about what the human meant by each spacebar press.

4.5 Quality of Resulting Actions: POMDP vs. Human Controlled Robot

Three metrics were captured in an effort to evaluate the quality of the POMDP selected actions: a subjective rating of the robot's intelligence, the time the trial took, and the value of the cost function v.s. time.

4.5.1 Perceived Intelligence

After each trial, the teacher rated the robot's intelligence. Figure 4.4 shows the ratings for the human-controlled robot and the ratings for the POMDP-controlled robot. The human received higher intelligence ratings, but not significantly; I believe that this gap can be improved with better modeling (see section 5.2.1).

4.5.2 Communication Time

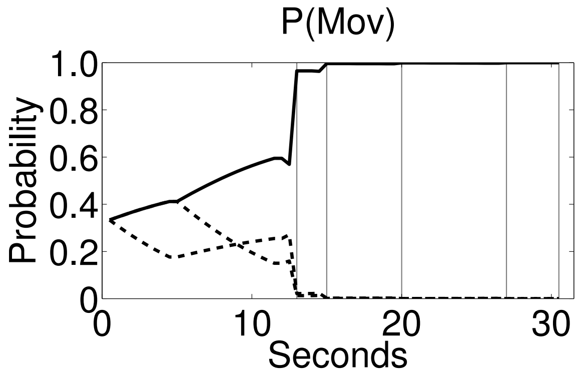
The communication time was measured as the time until the robot displayed "Final". Figure 4.5 is a histogram of the time until the robot displayed "Final" for the POMDP robot and for the human-controlled robot. Here again the human-controlled

[1] I did not include *before* and *after* for $M = waiting$ because the observation model Ω makes this assignment deterministic.

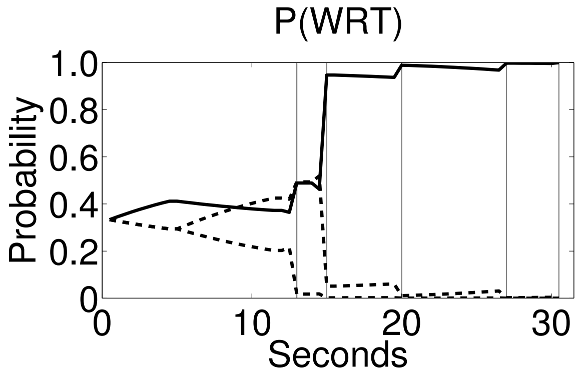
robot outperformed the POMDP-controlled robot, but the POMDP-controlled robot performed reasonably well. Part of this discrepancy could be due to inaccurate models, as in the intelligence ratings, but in this case I believe that the threshold for displaying “Final” was higher for the POMDP robot (over 98% confident) than for the human. Notably, I often observed the human-controlled robot displaying “Final” after a single spacebar press at the final location. In contrast, the POMDP robot always explored other distances; presumably to rule out the possibility that the first spacebar press was a mistake. Only after a second spacebar press would the POMDP robot display “Final”.

4.5.3 Reduction of Cost Function

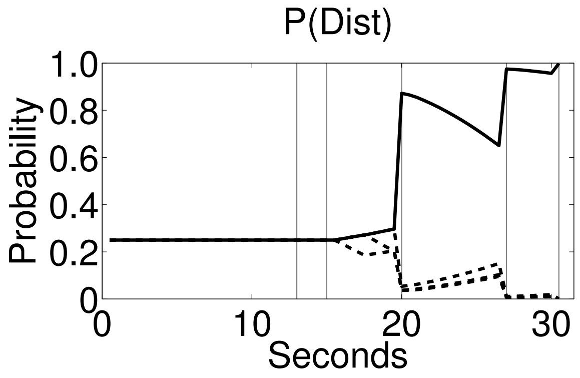
Of interest as well is the POMDP robot’s ability to drive down the cost function over each trial. Figure 4.6 plots the cost function (entropy) as a function of time for each of the trials with the POMDP-controlled robot. During several trials the entropy increased significantly before dropping again. This corresponds to the trials in which the teacher mistakenly pressed the spacebar; the POMDP robot initially believed that there was information in the key press, but over time realized that it was a mistake and carried no information. The figure shows the reduction of entropy in all trials to near zero.



(a) Ball to mov



(b) Ball to mov with respect to



(c) Distance from ball WRT

Figure 4.2: This figure shows the robot's inference of the ball to mov (a), the ball to move it with respect to (b), and the distance between the balls (c) for one of the trials. The vertical lines designate spacebar presses. The solid line in each figure shows the marginal probability of the true assignment for that random variable. The marginal probabilities for the true assignments are driven to near 1.0 by information gathered from the spacebar presses elicited by the robot's actions.

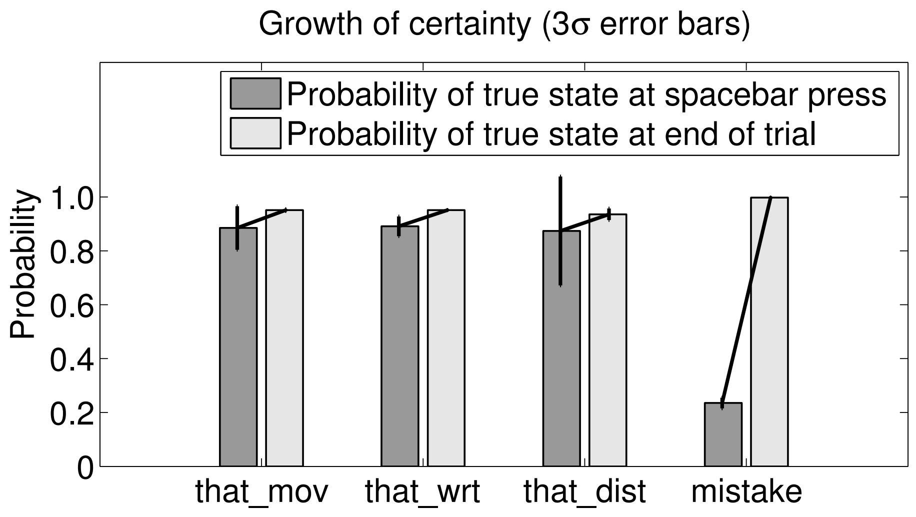


Figure 4.3: This figure shows the marginal probability of the four approval mental states at the time the spacebar was pressed (dark gray) and at the end of the trial (light gray). The true states were labeled in a post processing step. All spacebar presses from all 22 POMDP trials are included. This shows that, for each of the mental states, the marginal probability of the correct state increases as the trial progresses and ends at near certainty. This is most pronounced in the case of $M = mistake$, in which the initial probability that the spacebar was a mistake is low, but increases dramatically as the trial progresses.

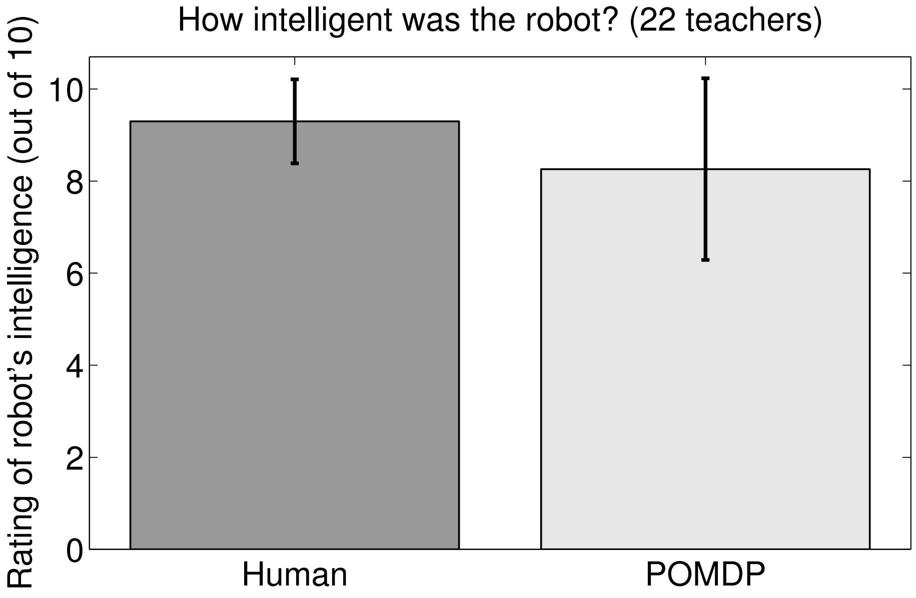


Figure 4.4: The 22 human teachers each participated in two trials, one teaching the human-controlled robot and one teaching the POMDP-controlled robot. The order of human or POMDP robot was randomized, and the true identity of the robot controller was hidden from the teacher. Following each trial the teacher rated the intelligence of the robot on a scale from 1 to 10, with 10 being the most intelligent. With the exception of one teacher, all teachers rated the human-controlled robot the same or more intelligent than the POMDP-controlled robot (mean of 9.30 vs. 8.26).

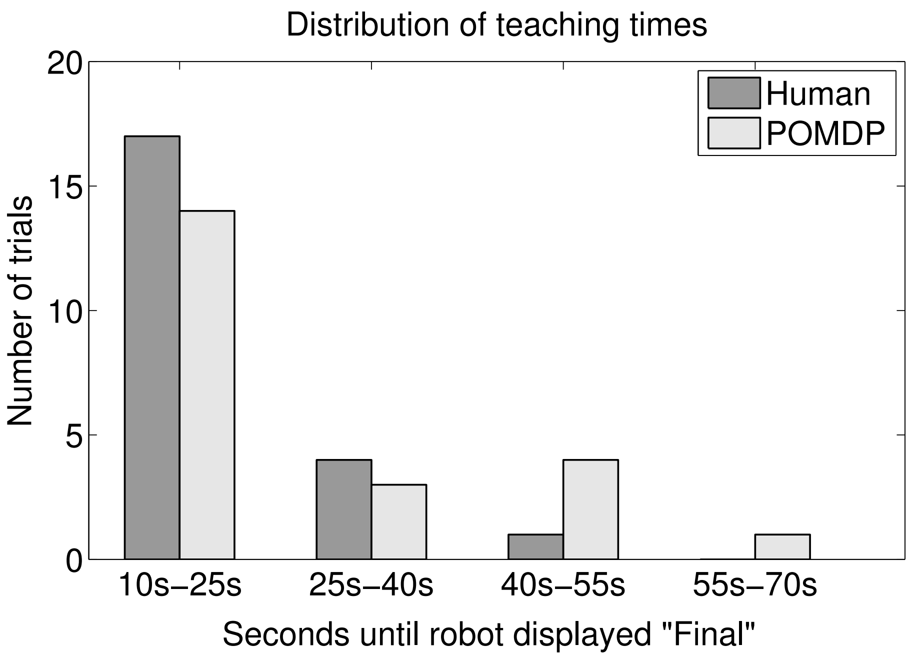


Figure 4.5: A histogram of the times until the robot, human or POMDP controlled, displayed "Final". The robot displayed "Final" to signal that they knew the task and that the world was displaying the task. The POMDP-controlled robot displayed "Final" when the marginal probability for a particular task, $P(Mov = i, WRT = j, Dist = k)$, was greater than 0.98. In all trials the robot, human or POMDP-controlled, correctly inferred the task. Task communication, as expected, took longer for the POMDP-controlled robot than for the human-controlled robot.

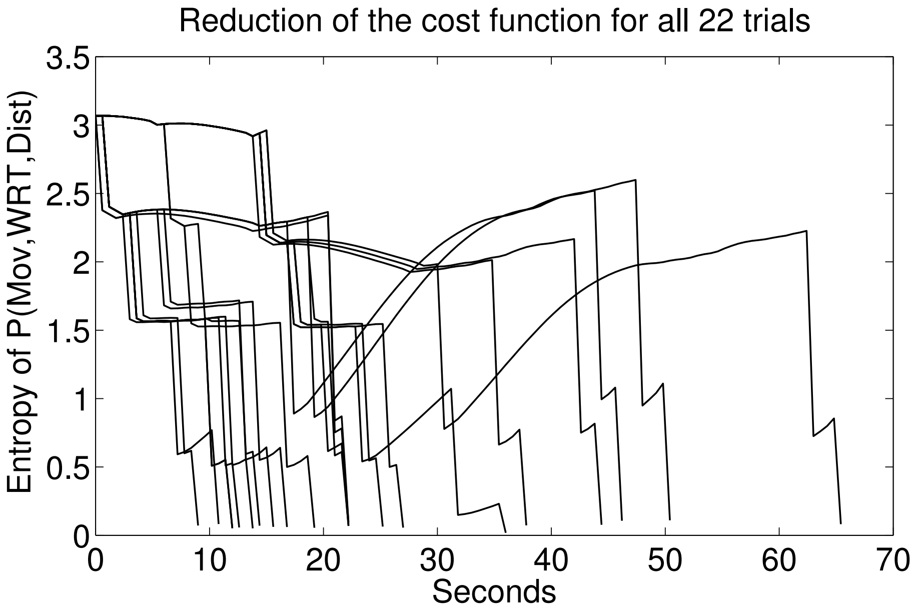


Figure 4.6: This figure shows the decrease in the cost function over time for all 22 trials of the POMDP-controlled robot. The cost function used was the entropy of the marginal distribution over the Mov, WRT, and $Dist$ random variables. All trials began at the same entropy, the entropy of the uniform distribution over Mov, WRT, and $Dist$. In all trials the entropy was driven to near zero in less than 70 seconds. Rapid drops correlate with spacebar presses, while large increases correspond to trials where the teacher mistakenly pressed the spacebar.

Chapter 5

Conclusion

5.1 Summary

This dissertation proposed the use of a POMDP for representing the human-robot task communication problem, reviewed the POMDP, demonstrated the representation on an example problem, and evaluated the approach through a user experiment. The experiment suggested that this representation results in robots that are robust to teacher error, that can accurately infer task details, and that are perceived to be intelligent. Relevant work related to human-robot task communication was reviewed, and an in depth review of POMDPs was provided, including Bayes filtering, Belman's equation, and a review of cutting edge POMDP solvers.

5.2 Future Work

5.2.1 Learning Model Structure and Model Parameters

In the POMDP representation described in chapter 3 the structure and parameters of T and Ω were set from intuition. I believe that both the structure and the parameters of the models can be "learned", in the machine learning sense. The models could be learned either from observations of humans communicating with other humans or from observations of humans communicating with robots. This is an important area of research as it may be unrealistic to expect social scientists to accurately and exhaustively model human teachers.

5.2.2 Complex Tasks

In the experiment presented in chapter 4, the task communicated consisted of a single object movement. Future work should aim to communicate more complex tasks, with chains of primitive task and ordering constraints (allowing the robot to select the optimal order of execution). These complex tasks could be represented as a directed graphs, where each node is a task that must be performed, and links links would capture task ordering constraints. Gray et al., describes a task representation that could be used for this purpose [9]. Just as in the demonstration of section 3, the robot would maintain a distribution over hypotheses, except here each hypothesis would be a fully specified task graph. Through communication the probability of the true hypothesis (the true task graph that the human is trying to communicate) would increase.

5.2.3 Complex Signals

Also in the experiment presented in chapter 4, the observations were limited to spacebar key presses. As research moves to tasks involving object movements in the real world, further observations should be incorporated, such as the gaze direction of the teacher and pointing gestures from the teacher, perhaps using a laser pointer [14]. Note that social behavior such as shared attention, argued for in [32], where the robot looks at the human to see where they are looking, would naturally emerge once the teacher's gaze direction, along with the appropriate models, is added as an observation to the system; knowing what object the human is looking at is informative (reduces entropy), so actions leading to the observation of the gaze direction would have low expected entropy and would likely be chosen.

5.2.4 Processing

As described in section 2.1.5, substantial progress has been made towards efficient solutions of POMDPs, yet processing remains a significant problem for POMDPs with complex domains. Further research is warranted, perhaps leveraging and extending techniques used in spoken dialog managers.

5.2.5 Smooth Task Communication and Task Execution Transitions

This dissertation focused on task communication, but a robot will also spend time executing communicated tasks. The formulation should be extended to apply to the entire operation of the robot; with optimal transitions between task communication and task execution. A choice of a broader cost function will be an important first step. One choice for this cost function might be the cost to the human under the human's cost function. The human's cost function would be captured in random variables, perhaps through a non-parametric model. The POMDP solver could then choose actions which would inform it about the human's cost function, which would aid in minimizing the cost to the human. Note that task communication would still occur under this cost function; for example, the robot might infer that doing a task is painful to the human, and communication would allow the robot to do this task for the human, thus performing communication actions would be attractive.[1]

5.2.6 IPOMDPs

In a classical POMDP the world is modeled as stochastic, but not actively rational; e.g. days transition from sunny to cloudy with a certain probability, but not as the result of the actions of an intelligent agent. In a POMDP the agent is the only intelligence in the world. An Interactive POMDP (IPOMDP) is one of several approaches that extend the POMDP to multiple intelligent agents [8]. It differs from game theoretic approaches in that it takes the perspective of an individual agent, rather than analyzing all agents globally; the individual agent knows that there are other intelligent agents in the world acting to minimize some cost function, but the actions of those agents and their cost functions may be only partially observable. I feel that the task communication problem falls into this category. The human teacher has objectives and reasons for communicating the task, knowing those reasons

[1]Research has shown that inferring another agent's cost function is possible (see inverse reinforcement learning)[22].

could allow the robot to better serve the human. Understanding the human and their objectives is important to the smooth communication and execution transitions described before. Thus future work should extend the proposed framework from the POMDP representation to the IPOMDP representation.

Unfortunately, an IPOMDP adds exponential branching of inter-agent beliefs to the already exponential branching of probability space and action-observations in a POMDP. Thus, while it is a more accurate representation it does make a hard problem even harder. That said, an IPOMDP may serve as a good formulation that we then seek approximate solutions for.

Chapter 6

Comparisons and Generalizations

This chapter compares the proposed task communication as a POMDP approach to other algorithms and generalizes the approach to other problems. For the comparisons I apply the Q-learning algorithm [38] and the TAMER algorithm [16] to the problem from chapter 3. Q-learning and TAMER are two algorithms used in recent literature to address the problem of learning from a human. I then generalize the proposed approach to the Sophies Kitchen problem [40]. The Sophies Kitchen problem has recently been used to evaluate reinforcement learning algorithms [40]. The comparisons and generalizations are presented without experimental results. The goal of this section is 1) to allow practitioners who are familiar with Q-learning and TAMER to quickly compare those algorithms with the proposed approach on the problem from chapter 3, and 2) to use Sophies Kitchen to provide an example of applying the proposed approach to new problems.

6.1 Comparisons

In this section I describe how to apply the Q-learning algorithm and the TAMER algorithm to the task communication problem from chapter 3.

6.1.1 Q-learning

The Q-learning algorithm learns a Q function for all state-action pairs, where $Q[s, a]$ is the expected sum of discounted rewards for executing action a in state s and thereafter executing a specific policy, π [38]. In a world with a finite set of actions and observations, Q can be represented with a table, where $Q[s, a]$ indexes the entry for state s and action a. If we knew Q under the optimal policy, π^*, then the optimal action, a, in a state s is:

$$a = \arg\max_a Q[s, a]. \tag{6.1}$$

Q-learning attempts to learn the true value of Q under the optimal policy by encorporating rewards as they are received. Assuming the robot executes action a in state s, transitions to state s' and receives reward R, then Q-learning would update $Q[s, a]$ as follows:

$$Q[s, a] = Q[s, a] + \alpha(R + \gamma max_{a'} Q[s', a'] - Q[s, a]). \tag{6.2}$$

From the robot's experience, the quantity $(R + \gamma max_{a'} Q[s', a'])$ is a good estimate of $Q[s, a]$; i.e. $Q[s, a]$ is the expected sum of discounted rewards, which is the immediate reward, R, plus the expected sum of discounted rewards going forward, $\gamma Q[s', a']$. The equation takes a gradiant decent step towards this estimate, where the step size is controlled by α.

As mentioned, the optimal action is apparant if Q-learning has converged to the optimal Q values, but how should the robot choose actions during this convergence? There are many schemes for choosing actions during this phase. A common action policy is to randomly select the next action in proportion to the current values of $Q[s,:]$:

$$a \sim P_{Q[s,:]}. \tag{6.3}$$

This is the policy used by researchers applying Q-learning to human-robot interaction [40]. After enough time, or after the average change in Q drops below some threshold, the robot could switch to the optimal policy given by equation 6.1. For details on the Q-learning algorithm, see [38].

In order to apply Q-learning to the task communication problem from chapter 3 we need to specify the States S, the actions A, and the rewards R. For S we will use the world state described in appendix B, consisting of: *holding*, *lit*, *relative_ball_mov*, *relative_ball_wrt*, and *relative_dist*. For A we will use the actions from section 3.3.2, consisting of: *noa*, *light_on*, *light_of*, *pick_up*, *release*, and *slide*. As is consistent with prior work applying Q-learning to the problem of learning from a human [40], the reward, R, will be the human input; 1 or 0 for the space-bar presses described in section 3.2.

A table-based Q function will be used. The Q value for the last state-action pair will be updated according to equation 6.2 after each time step. We will set $\alpha = 0.3$ and $\gamma = 0.75$, as was done in [40]. Actions will be chosen stochastically according to equation 6.3, as was the policy in [40].

I will now describe some advantages and dissadvantages of applying Q-learning to this problem. Since no experiments were performed, this is only speculation. Q-learning has many advantages: it is easy to implement (see equation 6.2); it is easy to apply to new problems, merely specify the states and actions (no need for time consuming modeling); Q-learning action selection typically requires very little processing power; and Q-learning updates as well typically require very little processing power. The dissadvantages of Q-learning would be: slow communication times, inability to infer hidden state, and difficulty of incorporating non-reward signals, such as gestures or spoken language.[1]

For the task communication problem from chapter 3, the main dissadvantage of Q-learning would be slower communication time. In the literature, Q-learning applied to a similar human robot communication problem resulted in an average communication time of twenty seven minutes, as apposed to sub-one minute in our experiments [40].[2] I believe that the speedup of the POMDP implementation is due mainly to a reduction of the problem complexity; Q-learning must learn a value for every permutation of the physical world state and the actions, while the POMDP only needs to learn the three values (Mov, WRT,$Dist$). Unfortunately these three relevant variables are hidden, so Q-learning cannot directly learn them. The speed of communication for the POMDP implementation comes at the cost of modeling and extra computation.

The POMDP approach excels where there is useful hidden state and where there are reliable models that link the hidden state to observations. If there is useful hidden

[1]As described in section 2.2, with the proposed POMDP approach, non-reward signals are incorporated in the same way that reward signals are incorporated. Namely, we add states and models that link the new signals to the hidden states that are relevant to the problem at hand.

[2]The authors did not directly report the average communication time. They did report the average number of actions per communication to be 816. With two seconds per action, we can infer that the average communication time was twenty seven minutes.

state, but no way to reliably link the hidden state to observations, then we cannot take advantage of the hidden state. Human robot communication is a good example of a case where we have useful hidden state and reliable models for linking that hidden state to observations.

For an example of the speed penalty due to Q-learning not modeling hidden state, we can look at the senario where the robot lights one of the balls and the human then presses the spacebar. Even though there is useful information in this spacebar press, this information is lost to Q-learning. Q-learning would just as readily pick up another ball, as it would pick up the ball that was lit when the spacebar was pressed. Thus, much more exploration, and time, would be needed for the Q-learning algorithm.[3]

6.1.2 TAMER

The TAMER algorithm is another algorithm that has been recently applied to the problem of learning from a human [16]. As with Q-learning, TAMER learns the function Q. The difference is in the treatment of the reward from the human. TAMER views the reward as an estimate of Q, rather than as a part of the sum that makes up Q. Accordingly, the Q-learning update equation 6.2, becomes:

$$Q[s,a] = Q[s,a] + \alpha(R - Q[s,a]). \tag{6.4}$$

This is because R is viewed as a direct estimate for $Q[s,a]$.

The TAMER algorithm also provides a credit assignment mechanism, since the

[3]In addition, Q-learning would generate a policy that takes actions which are irrelevant to task execution; such as lighting balls.

human's feedback may be delayed in settings where the timestep is short. The algorithm maintains a list of state-action pairs, (s_t, a_t), recently visited. After each timestep, the Q value for each state-action pair in the current list is updated as follows:

$$Q[s_t, a_t] = Q[s_t, a_t] + c(t)\alpha(R - Q[s_t, a_t]). \tag{6.5}$$

Where $c(t)$ indexes into a probability distribution modeling the human's feedback delay.[4]

The authors of TAMER recommend that actions be selected according to the optimal policy equation 6.1 [16], repeated here:

$$a = \arg\max_a Q[s, a]. \tag{6.1}$$

The task communication problem from chapter 3 would be formulated for TAMER as it was for Q-learning in section 6.1.1; with the same S, A, R, and α. I would use the action selection policy recommended for TAMER, equation 6.1. Also, since the time step in this problem is short enough to question which timestep the human was giving feedback for, I would make use of TAMER's credit assignment mechanism, with a $Gamma(k = 2, \Omega = 0.5)$ distribution. This distribution has a mean of one second, and a reasonable shape for feedback arrival times.

As they are very similar algorithms, TAMER has the same advantages and dissadvantages as Q-learning: easy of implementation, broad applicability, and low computational demands, but slow communication times, inability to infer hidden state, and difficulty when incorporating non-reward signals. With TAMER, since feedback only

[4]This credit assignment approach is equivalent to eligibility traces from reinforcement learning [38], but with a non-exponential probability distribution, and a discount rate, γ, set to one.

updates the current state, the problem of slow communication would be worse. The credit assignment mechanism would help to spread the approval back to states leading to the approval state, although this is not it's purpose. Also, with binary feedback, as seen in this problem, TAMER may not be conceptually appropriate; when the user presses the spacebar, issueing a *one* to the robot, it is not clear that *one* is the user's example of Q, the expected sum of discounted rewards from this state on. That said, due to the use of TAMER's credit assignment mechanism, I would expect TAMER to perform similarly to Q-learning on this problem.

6.2 Generalizations

In this section I apply the proposed framework to the Sophie's kitchen problem [40].

6.2.1 Sophie's Kitchen

Problem

Figure 6.1 is a screenshot from the Sophie's Kitchen world. The world consists of six objects: the $Agent$, $Flour$, a $Bowl$, a $Tray$, $Eggs$, and a $Spoon$. The objects are parameterized by their location: $Shelf$, $Table$, $Oven$, or $Agent$. The $Bowl$ has an additional parameter describing its state: $Empty$, $Flour$, $Eggs$, $Both$, or $Mixed$. The $Tray$ also has an additional parameter describing its state: $Empty$, $Batter$, or $Baked$. Figure 6.1 shows the world in the following state: $\langle Agent.loc = Shelf, Flour.loc = Shelf, Bowl.loc = Shelf, Bowl.state = Empty, Tray.loc = Table, Tray.state =$

$Empty, Eggs.loc = Table, Spoon.loc = Agent\rangle$. All objects start with their location set to $Shelf$.

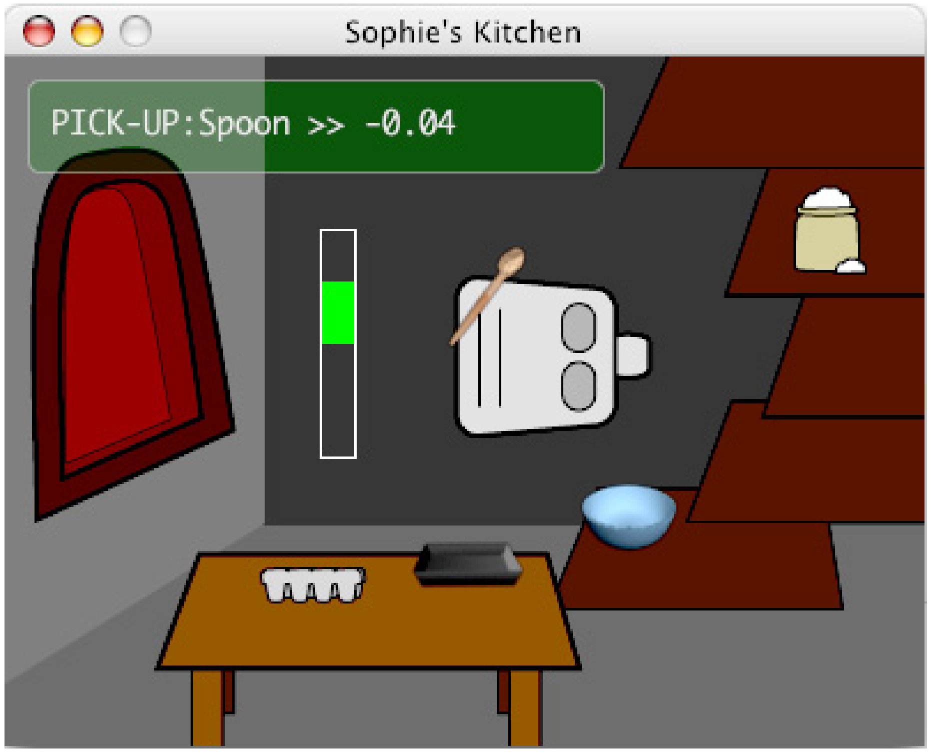


Figure 6.1: This is an image of the Sophie's Kitchen simulator, created by Andrea Thomaz at MIT [40]. The goal is to bake a cake. The human can provide feedback via the green slider. See the text for a description of this world.

The task for the agent is to bake a cake. Towards that end, the agent can perform four parameterized actions, the effects are shown in parenthesies: $Go(right|left)$ (moves the Agent.loc one step clockwise or counterclockwise), $Pick\text{-}Up(object)$ (if $object$ and $Agent$ are at the same location, then $object.loc = Agent$), $Put\text{-}Down(object)$ (if $object.loc = Agent$ then $object.loc = Agent.loc$), and $Use(object_1, object_2)$ (If $object_1.loc = Agent$, then $object_1$ is "used" on $object_2$; using $Flour$ or $Eggs$ on $Bowl$ changes $Bowl.state$ to $Eggs$, $Flour$, or $Both$; If $Bowl.loc = Agent$ and $Bowl.state = mixed$ and $Agent.loc = Tray.loc$, then $Use(Bowl, Tray)$ results in $Tray.state =$

batter).

The following is a sequence of actions which would accomplish the task of baking the cake, starting from the initial state of all objects at the shelf:

$$\begin{array}{l} PickUp(Eggs) \\ Use(Eggs, Bowl) \\ PutDown(Eggs) \\ PickUp(Flour) \\ Use(Flour, Bowl) \\ PutDown(Flour) \\ PickUp(Spoon) \\ Use(Spoon, Bowl) \\ PutDown(Spoon) \\ PickUp(Bowl) \\ Use(Bowl, Tray) \\ PutDown(Bowl) \\ PickUp(Tray) \\ Go(left) \\ PutDown(Tray) \end{array}$$

The agent receives feedback from the human as a real number between -1 and 1, potentially associated with an object. The human provides feedback by clicking with the mouse and dragging up or down. The feedback is sent when the mouse is released. If the mouse is clicked over an object then the agent is notified of the object that the mouse was clicked over, otherwise the feedback is general. The green bar shows the human the value of the current feedback, if they were to release the mouse button.

The simulator is free running. Though not specified in the text, the timestep between actions is assumed to be two seconds[5]. Many state-action pairs reset the simulator. The state-action pairs that reset the simular are not fully specified in the

[5]In [40], they report that the number of feedbacks per action at one point exceeds 1. This implies that their is enough time between actions to more than once decide on a feedback and use the mouse to administer it. Roughly two seconds seems to fit this information.

literature. For this section, we will assume that any *Put-Down*() action while the agents location is *Oven* resets the simulator.

POMDP Encoding

I will now represent this problem as a POMDP as proposed in this thesis. To do this we need to specify the eight elements of the tuple: $\langle S, A, O, T, \Omega, C, \gamma, b_0 \rangle$. This is a first pass at the encoding, if issues came up, I would modify the encoding.

State (S)

As with the example from chapter 3, the state S consists of the physical world state, the hidden state for the task to be learned, and the hidden mental state of the human. The physical world consists of 8 discrete variables: the location of each of the six objects, plus the "state" of the *Bowl* and the "state" of the $Tray$.

The hidden goal state that the human is trying to communicate will be the tripple, $Goal = \langle Goal\_S, Goal\_I, Goal\_A \rangle$. $Goal\_S$ and $Goal\_A$ are the desired final state-action pair. $Goal\_I$ is an eight dimensional indicator random variable disignating which of the eight state variables are important. The Goal state for baking a cake would be:

$$Goal = \langle Goal\_S, Goal\_I, Goal\_A \rangle \tag{6.6}$$

$$Goal\_S = \langle Agent.loc = Oven, Flour.loc =?, Bowl.loc =?, \tag{6.7}$$

$$Bowl.state =?, Tray.loc = Agent, Tray.state = Batter, \tag{6.8}$$

$$Eggs.loc =?, Spoon.loc =?\rangle \tag{6.9}$$

$$Goal\_I = \langle 1,0,0,0,1,1,0,0 \rangle \tag{6.10}$$

$$Goal\_A = Put\text{-}Down(Tray) \tag{6.11}$$

The agent is facing the oven, holding a tray with batter on it, and then puts down the tray. Since the agent is given the transition function for the physical world, it can always reach *Goal* by planning.[6]

As with the representation from chapter 3, the mental state of the human, M, is a dynamic random variable (depends on time) and explains the feedback given at the current time step. Here are the values that I have defined for M to take on: *waiting*, *mistake_pos*, *mistake_neg*, *good_ob_fut_*(*ob*), *good_ob_past_*(*ob*), *bad_ob_past_*(*ob*), *good_past*, or *bad_past*. There are twenty of these assignments; *good_ob_fut_*(*ob*), *good_ob_past_*(*ob*), and *bad_ob_past_*(*ob*) are each placeholders for the five values of *ob*: *good_ob_past_flour*, *good_ob_past_bowl*, *good_ob_past_tray*, etc. ***waiting*:** The *waiting* state corresponds to no feedback being received. ***mistake_pos* and *mistake_neg*:** Unlike the example in chapter 3, the feedback here is both positive and negative. Thus there is a positive and a negative mistake state. ***good_ob_fut_*(*ob*):** Our experiments have shown that humans will give feedback to direct future actions. My intuition

[6]If more than the final state-action pair is important, then additional variables would be created for the possibly infinite set and orderings of state-action pairs.

is that this feedback 1) will always be positive, and 2) will always be directed at a specific object. Thus the only mental state corresponding to directing future actions is $good\_ob\_fut\_(ob)$. ***good_ob_past_(ob), bad_ob_past_(ob), good_past, and bad_past:*** For feedback relating to the past, it seems reasonable that general and object specific positive and negative feedback will be given. Thus there are four mental states corresponding to feedback relating to the past: $good\_ob\_past\_(ob)$, $bad\_ob\_past\_(ob)$, $good\_past$, and bad_past.

The transition model for the mental state needs to know the object involved in the last action perfomed. So we add a single history variable to the state: $last\_ob$.

Actions (A)

The four parameterized actions are identical to those described in section 6.2.1:Problem.

Observations (O)

There are two observations for this representation, both capture the human feedback. The first, $o.feedback$, represents the sign of the feedback: $\{none, -1, 1\}$. $o.feedback = none$, corresponds to a timestep when the human did not give feedback. My intuition is that the magnitute of the feedback carries very little information in this domain. Thus, only the sign of the feedback is used in this encoding. The second observation, $o.ob$, represents the value of the object that the feedback corresponded to: $none$, $Agent$, $Flour$, $Bowl$, $Tray$, $Eggs$, or $Spoon$. $o.ob = none$ would correspond to a general (i.e. non-object) feedback.

Transition Model (T)

The transition model corresponding to the physical world is deterministic; e.g. $P(Spoon.loc{=}Agent|Pick\text{-}Up(Spoon)){=}1.0$. As with the encoding from chapter 3, the complexity of the Sophie's Kitchen encoding is in the transition model for the mental state, M, of the human. Here I specify the probability for each of the eight parameterized values of M given the state S, which includes *Goal*.

The following is my conceptual tree for distributing probability among the types of feedback. This forms a probability tree, where branches for a given node sum to one and the probability of a leaf is the product of the branches leading to that leaf:

- 0.10 *waiting*
- 0.01 *mistake_pos*
- 0.01 *mistake_neg*
- 0.88 GUIDANCE (0.88)
  - 0.5 FUTURE (0.44) → $good\_ob\_fut\_(ob)$
  - 0.5 PAST (0.44)
    - 0.2 OBJECT SPECIFIC (0.088)
      - 0.5 POSITIVE (0.044) → $good\_ob\_past\_(ob)$
      - 0.5 NEGATIVE (0.044) → $bad\_ob\_past\_(ob)$
    - 0.8 GENERAL (0.352)
      - 0.5 POSITIVE (0.176) → $good\_past$
      - 0.5 NEGATIVE (0.176) → bad_past

Here are the probabilities for each of the eight mental states, conditional on the world state.

$$P(M = waiting|S) = 0.1$$

$$P(M = mistake\_pos|S) = 0.01$$

$$P(M = mistake\_neg|S) = 0.01$$

For a given *Goal*, the robot can solve for the set of optimal paths to *Goal*. Let *next_ob* be the set of objects involved in the next action for all optimal paths. Similarly, let *nnext_ob* and *nnnext_ob*, be the set of objects involved two and three time steps away. Note that the optimal path takes into consideration *Goal_I*, so only the relevant state needs to be reached.

$$P(M = good\_ob\_fut\_(ob)|S)$$
$$= 0.44 \times \begin{cases} 0.6 & \text{if } ob \in next\_ob \\ 0.3 & \text{if } ob \notin next\_ob \wedge ob \in nnext\_ob \\ 0.1 & \text{if } ob \notin (next\_ob \cup nnext\_ob) \wedge ob \in nnnext\_ob \\ 0.0 & \text{otherwise} \end{cases}$$

An object is "on" an optimal path if it is involved in an important action to the achievement of *Goal*.

$$P(M = good\_ob\_past\_(ob)|S) = 0.044 \times \begin{cases} 1.0 & \text{if } ob = S.last\_ob \wedge ob \text{ is on path} \\ 0.0 & \text{otherwise} \end{cases}$$

$$P(M = bad\_ob\_past\_(ob)|S) = 0.044 \times \begin{cases} 1.0 & \text{if } ob = S.last\_ob \wedge ob \text{ is not on path} \\ 0.0 & \text{otherwise} \end{cases}$$

$$P(M = good\_past|S) = 0.176 \times \begin{cases} 1.0 & \text{if } S.last\_ob \text{ is on path} \\ 0.0 & \text{otherwise} \end{cases}$$

$$P(M = bad\_past|S) = 0.176 \times \begin{cases} 1.0 & \text{if } S.last\_ob \text{ is not on path} \\ 0.0 & \text{otherwise} \end{cases}$$

As with the transition model from chapter 3 we need to make sure that, for a given S, this is a proper probability distribution. This is done by computing the probabibility for each value of M and then normalizing by the sum of all probabilities.

Observation Model (Ω)

The observation model is as follows:

```
switch(s.M)
```

case $s.M = waiting$:

$$P(o|s) = \begin{cases} 1.0 & \text{if } o.feedback = none \wedge o.ob = none \\ 0.0 & \text{otherwise} \end{cases}$$

case $s.M = mistake\_pos$:

$$P(o|s) = \begin{cases} 1/7 & \text{if } o.feedback = 1\text{, for each of the seven values of o.ob} \\ 0.0 & \text{otherwise} \end{cases}$$

case $s.M = mistake\_neg$:

$$P(o|s) = \begin{cases} 1/7 & \text{if } o.feedback = -1\text{, for each of the seven values of o.ob} \\ 0.0 & \text{otherwise} \end{cases}$$

case $s.M = good\_ob\_fut\_(ob)$:

$$P(o|s) = \begin{cases} 1.0 & \text{if } o.feedback = 1 \wedge o.ob = ob \\ 0.0 & \text{otherwise} \end{cases}$$

case $s.M = good\_ob\_past\_(ob)$:

$$P(o|s) = \begin{cases} 1.0 & \text{if } o.feedback = 1 \wedge o.ob = ob \\ 0.0 & \text{otherwise} \end{cases}$$

case $s.M = bad\_ob\_past\_(ob)$:

$$P(o|s) = \begin{cases} 1.0 & \text{if } o.feedback = -1 \wedge o.ob = ob \\ 0.0 & \text{otherwise} \end{cases}$$

case $s.M = good\_past$:

$$P(o|s) = \begin{cases} 1.0 & \text{if } o.feedback = 1 \wedge o.ob = none \\ 0.0 & \text{otherwise} \end{cases}$$

case $s.M = bad\_past$:

$$P(o|s) = \begin{cases} 1.0 & \text{if } o.feedback = -1 \wedge o.ob = none \\ 0.0 & \text{otherwise} \end{cases}$$

As with the encoding from chapter 3, this observation model is deterministic. Again, if there was noise in the positive or negative feedback then this would be captured here. Stochastic observation models would be appropriate for signals that inherently contain noise, for example, spoken language, where $M = Boston$ sometimes comes through as $O = Austin$.

Cost Function (C)

The cost function is $Entropy(Goal)$ as in chapter 3. Thus the agent will act to reduce uncertainty about the components of *Goal*: the goal state, the goal state indicators, and the goal action.

Discount Rate (γ)

$\gamma = 1$, as in chapter 3.

Initial Belief (b_0)

The initial belief is set to a uniform distribution over all permutations of the elements of *Goal*.

Expected Performance

The original research on the Sophie's Kitchen problem applied the Q-learning algorithm. I described the tradeoffs of Q-learning in section 6.1.1. As in that analysis, for this problem, I expect that the POMDP approach would result in faster communication times. Also, I would expect to see more deliberate behavior with the POMDP approach. For example, if the robot were exploring around non-recoverable states, such as stiring the eggs and flour, I would expect to see the robot deliberately heading for reset states, such as putting anything in the over. These reset states would allow the robot to quickly get back to exploring the area that is most immediately informative. Finally, I would expect to see consequences of feedback, which would not be present in a Q-learning approach. For example, for tasks where the final goal does not involve the spoon, actions such as picking up the spoon then immediately putting down the spoon would be unecessary. So, if the robot received an approval immediately after picking up the spoon, then we would expect to see the robot exploring tasks that involve the spoon, and certainly, the robot would not immediately put down the spoon. This type of consequence would not be seen with the Q-learning approach.

Bibliography

[1] Jonathan Baxter, Lex Weaver, and Peter Bartlett. Direct gradient-based reinforcement learning: Ii. gradient ascent algorithms and experiments. Technical report, Department of Computer Science, Australian National University, 1999.

[2] Bruce Blumberg, Marc Downie, Yuri Ivanov, Matt Berlin, Michael Patrick Johnson, and Bill Tomlinson. Integrated learning for interactive synthetic characters. In *Proc. International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2002.

[3] Cynthia Breazeal. Regulation and entrainment in human-robot interaction. *International Journal of Robotics Research*, 21:883–902, 2002.

[4] Cynthia Breazeal. Toward sociable robotics. *Robotics and Autonomous Systems*, 42:167–175, 2003.

[5] Cynthia Breazeal, Aaron Edsinger, Paul Fitzpatrick, and Brian Scassellati. Active vision systems for sociable robots. *IEEE Transactions on Systems, Man, and Cybernetics: Special Issue Part A. Systems and Humans*, 31:5:443–453, 2001.

[6] Cynthia Breazeal, Jesse Gray, and Matt Berlin. Mindreading as a foundational skill for socially intelligent robots. In *Proc. of the 2007 International Symposium on Robotics Research (ISRR)*, 2007.

[7] Finale Doshi and Nicholas Roy. The permutable POMDP: Fast solutions to POMDPs for preference elicitation. In *Proc. of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2008.

[8] Piotr J. Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24, 2005.

[9] Jesse Gray, Matt Berlin, and Cynthia Breazeal. Intention recognition with divergent beliefs for collaborative robots. In *Proc. of the 2007 Artificial Intelligence and Simulation of Behavior (AISB) workshop on Mindfull Environments*, 2007.

[10] Daniel H. Grollman and Odest Chadwicke Jenkins. Dogged learning for robots. In *Proc. of the International Conference on Robotics and Automation*, 2007.

[11] Milos Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[12] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101, 1998.

[13] F. Kaplan, P. Y. Oudeyer, and E. Kubinyi. Robotic clicker training. *Robotics and Autonomous Systems*, 38(3-4):197–206, 2002.

[14] Charles C. Kemp, Cressel D. Anderson, Alexander J. Trevor, and Zhe Xu. A point-and-click interface for the real world: laser designation of objects for mobile manipulation. In *Proc. of the International Conference on Human-Robot Interaction (HRI)*, 2008.

[15] Kyungduk Kim and Gary Geunbae Lee. Multimodal dialog system using hidden information state dialog manager. In *Proc. of the night international conference on multimodal interfaces (ICMI)*, 2007.

[16] W. Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The TAMER framework. In *Proc. of the Fifth International Conference on Knowledge Capture (KCAP)*, September 2009.

[17] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: scaling up. In *Proc. of the 12th International Conference on Machine Learning (ICML)*, pages 362–370, 1995.

[18] William S. Lovejoy. Computationally feasible bounds for partially observable markov decision processes. *Operations Research*, 39:162–175, 1991.

[19] David A. McAllester and Satinder Singh. Approximate planning for factored POMDPs using belief state simplification. In *Proc. of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 409–416, 1999.

[20] Kevin P. Murphy. A survey of POMDP solution techniques. Technical report, 2000.

[21] Andrew Y. Ng, Jin Kim, Michael I. Jordan, and Shankar Sastry. Autonomous helicopter flight via reinforcement learning. In *Proc. of Conference on Neural Information Processing Systems (NIPS)*, 2003.

[22] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proc. of the International Conference on Machine Learning (ICML)*, 2000.

[23] Monica N. Nicolescu and Maja J. Mataric. Natural methods for robot task learning: Instructive demonstration, generalization and practice. In *Proc. of the Second International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 241–248, July 2003.

[24] Sebastien Paquet, Ludovic Tobin, and Brahim Chaib-draa. An online POMDP algorithm for complex multiagent environments. In *Proc. of the fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, pages 970–977, 2005.

[25] Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Point-based value iteration: an anytime algorithm for POMDPs. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032, 2003.

[26] Joelle Pineau, Michael Montemerlo, Martha Pollack, Nicholas Roy, and Sebastian Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems*, 42:271–281, 2003.

[27] Stephane Ross and Brahim Chaib-draa. AEMS: An anytime online search algorithm for approximate policy refinement in large pomdps. In *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2592–2598, 2007.

[28] Stephane Ross, Joelle Pineau, Sebastien Paquet, and Brahim Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[29] Nicholas Roy and Sebastian Thrun. Motion planning through policy search. In *in Proceedings of the Conference on Intelligent Robots and Systems (IROS)*, 2002.

[30] Joe Saunders, Chrystopher L Nehaniv, and Kerstin Dautenhahn. Teaching robots by moulding behavior and scaffolding the environment. In *Proc. ACM/IEEE International Conference on Human-Robot Interaction Interaction (HRI)*, pages 142–150, March 2006.

[31] Ashutosh Saxena, Justin Driemeyer, Justin Kearns, and Andrew Y. Ng. Robotic grasping of novel objects. In *Proc. of the Twenty-First Annual Conference on Neural Information Processing Systems (NIPS)*, 2007.

[32] Brian Scassellati. *Imitation and mechanisms of joint attention: A developmental structure for building social skills on a humanoid robot.* Springer-Verlag, 1999.

[33] Brian Scassellati. A theory of mind for a humanoid robot. In *Proc. of the first IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2000.

[34] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison, 2010.

[35] Richard D. Smallwood and Edward J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21:1071–1088, September 1973.

[36] Edward J. Sondik. *The Optimal Control of Partially Observable Markov Processes over a Finite Horizon*. PhD thesis, Stanford University, 1971.

[37] Matthijs T. J. Spaan and Nikos Vlassis. Perseus: randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[38] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[39] Umar Syed and Jason D Williams. Using automatically transcribed dialogs to learn user models in spoken dialog systems. In *Proc. of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (ACL-HLT)*, 2008.

[40] A. L. Thomaz, G. Hoffman, and C. Breazeal. Reinforcement learning with human teachers: Understanding how people want to teach robots. In *Proc. IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 352–357, September 2006.

[41] Blaise Thomson, Jost Schatzmann, Karl Welhammer, Hui Ye, and Steve Young. Training a real-world POMDP-based dialog system. In *Proc. Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2007.

[42] Blaise Thomson and Steve Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24:562–588, 2010.

[43] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[44] David S Touretzky and Lisa M Saksida. Operant conditioning in skinnerbots. *Adaptive Behavior*, 5:34, 1997.

[45] Jur van den Berg, Stephen Miller, Ken Goldberg, and Pieter Abbeel. Gravity-based robotic cloth folding. In *The 9th International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2010.

[46] Jason D Williams. Incremental partition recombination for efficient tracking of multiple dialog states. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2010.

[47] Jason D Williams. A case study of applying decision theory in the real world: POMDPs and spoken dialog systems. *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, pages 315–342, 2011.

[48] Jason D Williams and Steve Young. Scaling up POMDPs for dialog management: The "summary POMDP" method. In *Proc. of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 177–182, 2005.

[49] Jason D Williams and Steve Young. Scaling POMDPs for spoken dialog management. *IEEE Transactions on Audio, Speech, and Language Processing*, (7):2116–2129, 2007.

[50] Jason D Williams and Steve J Young. Scaling POMDPs for dialog management with composite summary and point-based value iteration (CSPBVI). In *Proc. of the American Association for Artificial Intelligence (AAAI) Workshop on Statistical and Empirical Approaches for Spoken Dialogue Systems*, 2006.

[51] Steve Young, Milica Gasic, Simon Keizer, Francois Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: a practical framework for POMDP-based spoken dialogue management. *Computer Speech and Language*, 24(2):150–174, 2010.

[52] Steve Young, Jost Schatzmann, Blaise Thomson, Karl Weilhammer, and Hui Ye. The hidden information state dialogue manager: A real-world POMDP-based system. In *Proc. of the Demonstration Session of Human Language Technologies: The Annual Confrence of the NOrth American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2007.

Appendix A

Raw User Experiment Data

Table A.1 and A.2 presents the raw data from the user experiment described in chapter 4.

run	subject id	gender: (m)ale, (f)emale	age	ai experience (1-7)	robot: (p)OMDP, (h)uman	robot intelligence? (1-10)	correct? (t)rue, (f)alse	trial time
1	1	m	65	1	p	7	t	46.5
2	1	m	65	1	h	10	t	24.5
3	2	m	21	1	h	10	t	19.0
4	2	m	21	1	p	10	t	12.5
5	3	m	22	3	p	10	t	36.5
6	3	m	22	3	h	10	t	18.0
7	4	m	20	4	p	8	t	45.0
8	4	m	20	4	h	9	t	36.0
9	5	m	21	4	h	10	t	43.0
10	5	m	21	4	p	10	t	17.0
11	6	f	20	4	p	5	t	38.0
12	6	f	20	4	h	10	t	21.0
13	7	m	20	4	p	9	t	13.0
14	7	m	20	4	h	9	t	12.0
15	8	m	24	4	p	9	t	22.5
16	8	m	24	4	h	8	t	18.0
17	9	m	31	4	h	10	t	17.0
18	9	m	31	4	p	10	t	15.0
19	10	m	26	3	h	8	t	19.5
20	10	m	26	3	p	8	t	19.5
21	11	f	28	1	p	7	t	22.5
22	11	f	28	1	h	7	t	20.0
continued in table A.2								

Table A.1: This is the raw data from the experiment described chapter 4

run	subject id	gender: (m)ale, (f)emale	age	ai experience (1-7)	robot: (p)OMDP, (h)uman	robot intelligence? (1-10)	correct? (t)rue, (f)alse	trial time
continued from table A.1								
23	12	f	21	7	h	10	t	22.5
24	12	f	21	7	p	10	t	9.5
25	13	m	22	7	p	8	t	27.5
26	13	m	22	7	h	10	t	24.5
27	14	f	19	1	h	9	t	33.5
28	14	f	19	1	p	9	t	22.5
29	15	f	25	5	p	10	t	22.5
30	15	f	25	5	h	10	t	22.0
31	16	f	23	6	h	8	t	17.0
32	16	f	23	6	p	8	t	16.0
33	17	f	22	3	p	9	t	11.0
34	17	f	22	3	h	10	t	18.5
35	18	f	19	1	p	10	t	51.0
36	18	f	19	1	h	10	t	16.5
37	19	f	18	1	h	10	t	17.5
38	19	f	18	1	p	10	t	12.5
39	20	m	22	2	h	9	t	14.5
40	20	m	22	2	p	10	t	25.5
41	21	f	21	4	p	8	t	14.0
42	21	f	21	4	h	8	t	16.5
43	22	m	18	4	h	10	t	14.0
44	22	m	18	4	p	10	t	66.0

Table A.2: This is a continuation from table A.1 of the raw data from the experiment described chapter 4

Appendix B

Full Experiment State (S)

In the tables below I list all of the variables that make up the system state for the POMDP representation described in chapter 3. The variables are presented in groups according to their types. It is helpful to introduce some terminology:

- static vs. dynamic: A static variable is a variable that does not change with time. A dynamic variable does change with time.
- hidden vs. observable: A hidden variable cannot be directly measured by the robot. An observable random variable can be directly measured. Information about hidden variables is only provided through observable variables.
- deterministic vs. random: A deterministic variable is a variable whose value does not depend on a probability. A random variable is a variable whose value does depend on probability. A deterministic variable could depend on the value of a random variable, but only deterministically; i.e. If we knew the value of the random variable it would fully determine the value of the deterministic variable.

The need for inference, and the motivation for task communication, is due to the variables that are hidden and random. In the POMDP representation described in chapter 3, the variables that capture the details of the task are static hidden random variables (they do not depend on time), and the variables that capture the mental state of the teacher are dynamic hidden random variables (they do depend on time). The other variables are either observable, so we can directly measure their values, or deterministic, so we know their value given other variables.

Task Variables (static hidden random variables)	
Mov	The ball (1-3) the human wishes the robot to move.
WRT	The ball (1-3) the human wishes the robot to move ball Mov with respect to.
$Dist$	The distance (20,40,60,80) the human wishes the robot to move ball Mov with respect to ball WRT.

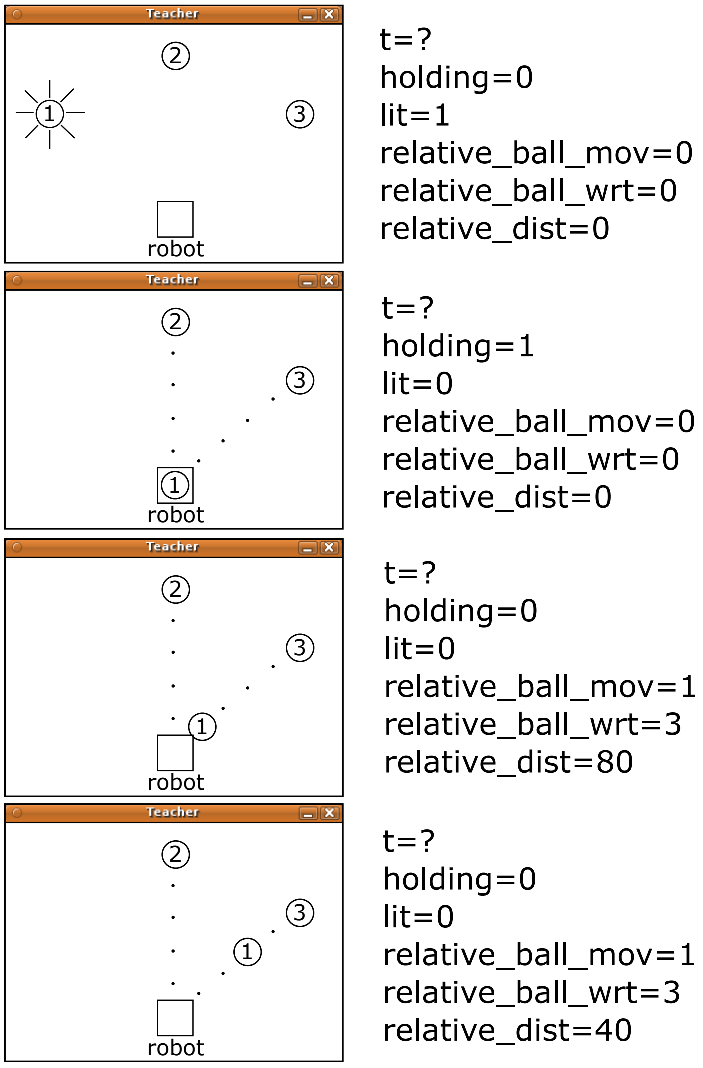
Human State (dynamic hidden random variable)	
M	The mental state of the teacher (*waiting*, *mistake*, *that_mov*, *that_wrt*, or *that_dist*) at the present time.

World State (dynamic observable deterministic variables)	
t	The current time step (1-∞).
holding	The ball (1-3) the robot is holding, 0 if none.
lit	The ball (1-3) the robot has lit, 0 if none.
relative_ball_mov	The ball (1-3) that is in a relative pose, 0 if none.
relative_ball_wrt	The ball (1-3) that *relative_ball_mov* is a distance from, 0 if no ball is in a relative pose
relative_dist	The distance, in percent (20,40,60,80), that ball *relative_ball_mov* is from ball *relative_ball_wrt*, 0 if no ball is in a relative pose

World State History (dynamic observable deterministic variables)	
$last\_lit\_on\_holding(b_1, b_2)$	The last timestep when the light of ball b_1 changed from off to on while holding ball b_2, -1 if this event has never occurred. If $b_2 = 0$ then this indicates the last time ball b_1 was lit while the robot was not holding any ball.
$last\_lit\_off\_holding(b_1, b_2)$	The last timestep when the light of ball b_1 changed from on to off while holding ball b_2, -1 if this event has never occurred. If $b_2 = 0$ then this indicates the last time that the light on ball b_1 went from on to off while the robot was not holding any ball.
$last\_dist(b_1, b_2, d)$	The last timestep that ball b_1 arrived at the distance d from ball b_2, -1 if this event has never occurred.
$last\_not\_dist(b_1, b_2, d)$	The last timestep that ball b_1 left the distance d from ball b_2, -1 if this event has never occurred.

Human State History (dynamic hidden deterministic variables	
$last\_M(m)$	The last timestep that the teacher was in the mental state m, -1 if the teacher was never in this state.

Figure B.1 shows the values of the world state variables (dynamic, observable, and deterministic) for four world configurations. The assignments are shown next to the corresponding simulator scene.



t=?
holding=0
lit=1
relative_ball_mov=0
relative_ball_wrt=0
relative_dist=0

t=?
holding=1
lit=0
relative_ball_mov=0
relative_ball_wrt=0
relative_dist=0

t=?
holding=0
lit=0
relative_ball_mov=1
relative_ball_wrt=3
relative_dist=80

t=?
holding=0
lit=0
relative_ball_mov=1
relative_ball_wrt=3
relative_dist=40

Figure B.1: Four assignments to the world state variables.

Appendix C

Full Experiment Transitions Model (T)

A transition model provides the probability of transitioning to all states in S after executing a specific action a while in a specific state s.

Here I present the full transition model for the POMDP representation described in chapter 3 in two sections. The first section presents the transition model for the world state and the second section presents the transition model for the task variables and the human mental state. These can be presented separately since the world state transitions are independent of the human's state.

C.1 World State Transition Model

The world state is deterministic, i.e. the position and lighting of the balls is fully determined by the previous position and lighting of the balls and the action that was performed. In this case it is most intuitive to define the transition model in terms of the effects of actions on the world state. This translates to a probabilistic transition model where all of the probability mass is on the determined state. e.g. if $\hat{s}$ is the deterministic state that results from executing action a in state s then the transition model is:

$$T(s'|a,s) = P(s'|a,s) = \begin{cases} 1 & \text{if } s' = \hat{s} \\ 0 & \text{otherwise} \end{cases} \tag{C.1}$$

The robot can perform six parameterized actions: noa, light_on(b), light_off(b), pick_up(b), release(b), and slide(b_1, d, b_2). For a description of these actions see section 3.3.2. Here I present the preconditions and effects of each of these actions.

In the effects below, if a variable of the world state is not mentioned, then it is unchanged from the previous timestep. Variables used on the right hand side of an assignment or anywhere in the conditions are from the previous time step.

---

action: noa

preconditions:

effects:

description:

The world substate is unchanged

---

action: light_on(b)

preconditions:

$lit == 0$

effects:

$lit = b$

$last\_lit\_on\_holding(b, holding) = t$

description:

Turns on the light for ball b. The event is recorded in the history state. Only one ball's light can be on at a time.

---

action: light_off(b)

preconditions:

$lit == b$

effects:

$lit = 0$

$last\_lit\_off\_holding(b, holding) = t$

description:

Turns off the light for ball b. The event is recorded the history state

---

action: pick_up(b)

preconditions:

$lit == 0$

$holding == 0$

$relative\_ball\_mov == 0 \lor (relative\_ball\_mov == b \land relative\_ball\_dist == 80)$

effects:

$holding = b$

$relative\_ball\_mov = relative\_ball\_wrt = relative\_ball\_dist = 0$

description:

A ball can be picked up if it is in its original pose or if it is at the position closest to the robot (i.e. the position furthest from another ball d=80). Not allowed if another ball is being held.

---

action: release(b)

preconditions:

$lit == 0$

$holding == b$

effects:

$holding = 0$

description:

Returns the ball to its starting location. No ball can be lit. The robot must currently be holding the ball.

---

action: slide(b_1, d, b_2)

preconditions:

$lit == 0$

$((relative\_mov == b_1 \wedge relative\_wrt == b_2 \wedge relative\_dist == d \pm 20)$

$\vee(holding == b_1 \wedge d == 80))$

effects:

$holding = 0$

$relative\_mov = b_1$

$relative\_wrt = b_2$

$relative\_dist = \max(\min(d, 80), 20)$

description:

Slides ball b_1 one step relative to ball b_2. No ball can be lit. Distance d must be one step away from the current distance; i.e. either the current distance plus or minus 20, or 80 if the robot is currently holding ball b_1.

C.2 Task and Human State Transition Model

The task and teacher mental state are captured in the random variables Mov, WRT, $Dist$, and M. Since Mov, WRT, and $Dist$ are static random variables (they do not change with time), their transition model is trivial.

$$P(mov', wrt', dist' | mov, wrt, dist, \dots) = \begin{cases} 1 & \text{if } (mov', wrt', dist') = (mov, wrt, dist) \\ 0 & \text{otherwise} \end{cases} \tag{C.2}$$

The teacher mental state M is a dynamic random variable (it changes with time) taking on values *that_mov*, *that_wrt*, *that_dist*, *waiting*, and *mistake*. To specify the transition model for M we must specify the probability of transitioning to each of the values (*that_mov*, *that_wrt*, *that_dist*, *waiting*, and *mistake*) for any state of the world.

The complexity of the transition model for M comes from the fact that there is often delay in the human's response to the robot's actions (light_on, light_off, slide, etc).

If the robot lights the ball that the human wishes them to move, it may take a few time steps for the human to push the spacebar (i.e. the probability of transitioning to

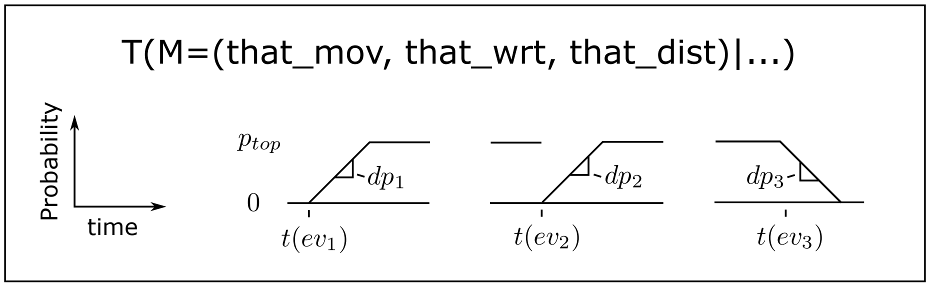


Figure C.1: The transition model for three of the assignments to M (that_mov, that_wrt, and that_dist) each use this model, containing three distinct phases. 1) The transition becoming more likely after an event ev_1 and maxing out at p_{top}, the transition probability resetting upon event ev_2, and the transition probability decreasing after an event ev_3. Each of these three values of M has their own set of the seven parameters (p_{top}, $t(ev_1)$, dp_1, $t(ev_2)$, dp_2, $t(ev_3)$, dp_3).

that_mov from *waiting* should start at zero after ball *mov* is lit, but quickly increases). Furthermore, the human may push the spacebar a timestep after the robot has turned off the light, since they didn't have time to stop themselves from pressing the spacebar (i.e. the probability of transitioning to *that_mov* from *waiting* should decrease after the light has been turned off, but not immediately be set to zero).

Figure C.1 shows the function used for $M = (that\_mov, that\_wrt,$ and $that\_dist)$. The functions differ in their events and the parameters, but the structure is the same: some event occurs, ev_1, followed by a rising probability dp_1 to a maximum p_{max}; if the spacebar is pressed under a certain mental state, ev_2, then the probability resets to zero and increases at the rate of dp_2 to p_{max}; if the supporting event is removed, ev_3, then the probability decreases at the rate of dp_3. Let $t(ev_i)$ be the time when event ev_i occurred.

If $[t(ev_1) > t(ev_2) \wedge t(ev_3)]$, then

$$P = min(dp_1 \cdot (t - t(ev_1)), p_{max}).$$

If $[t(ev_2) > t(ev_1) \wedge t(ev_3)]$, then

$$P = min(dp_2 \cdot (t - t(ev_2)), p_{max}).$$

If $[t(ev_3) > t(ev_1) \wedge t(ev_2)]$, then

$$P = max(dp_3 \cdot (t - t(ev_3)) + p_{max}, 0).$$

M=that_mov:

$$\begin{aligned}
t(ev_1) &= last\_lit\_on\_holding(Mov, 0) \\
dp_1 &= 0.02 \\
p_{top} &= 0.1 \\
t(ev_2) &= last\_M(that\_mov) \\
dp_2 &= 0.02 \\
t(ev_3) &= last\_lit\_off\_holding(Mov, 0) \\
dp_3 &= \text{-0.04}
\end{aligned}$$

M=that_wrt:

$$\begin{aligned}
t(ev_1) &= last\_lit\_on\_holding(WRT, Mov) \\
dp_1 &= 0.02 \\
p_{top} &= 0.1 \\
t(ev_2) &= last\_M(that\_wrt) \\
dp_2 &= 0.02 \\
t(ev_3) &= last\_lit\_off\_holding(WRT, Mov) \\
dp_3 &= \text{-0.04}
\end{aligned}$$

M=that_dist:

$$
\begin{aligned}
t(ev_1) &= last\_dist(Mov, WRT, Dist) \\
dp_1 &= 0.02 \\
p_{top} &= 0.1 \\
t(ev_2) &= last\_M(that\_dist) \\
dp_2 &= 0.02 \\
t(ev_3) &= last\_not\_dist(Mov, WRT, Dist) \\
dp_3 &= \text{-0.04}
\end{aligned}
$$

M=mistake:

The probability of a mistake on any time step is 0.005:

$P(M = mistake|\dots) = 0.005$

M=waiting:

And the probability that the human transitions to *waiting* is the remaining probability:

$P(M = waiting|\dots) = 1 - P(M = (that\_ob \vee that\_wrt \vee that\_dist)|\dots)$

The only variables left that we need to specify a transition model for are the human state history: $last\_M(m)$. Since these are deterministic variables, given a hypothesis with $M = m$, where m could be (*that_mov*, *that_wrt*, *that_dist*, *mistake*, or *waiting*), the deterministic update is:

$$last\_M(m) = t.$$

For $m' \neq m$, $last\_M(m')$ remains unchanged. Expressed as a probability function this is:

$$P(last\_M(m) = t' | \dots) = \begin{cases} 1 & \text{if } (M = m \wedge t' = t) \vee \\ & \quad (M \neq m \wedge t' = last\_M(m)) \\ 0 & \text{otherwise} \end{cases} \tag{C.3}$$