Human-Centered Planning for Effective Task Autonomy

Stephanie L. Rosenthal

CMU-CS-12-110
May 2012

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Manuela Veloso, Co-Chair
Anind K. Dey, Co-Chair
Manuel Blum
Eric Horvitz, Microsoft Research

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the National Science Foundation under grant numbers CNS-0910754, CCF-1029549, IIS-1012733; US Army Research Office under grant number W911NF0910273; Office of Naval Research under grant number N00014-09-1-1031; National Institutes of Health under grant numbers 5 T32 MH 19883 and 2 R01 MH 52808; and the National Science Foundation Graduate Research Fellowship; Siebel Scholarship; Google Anita Borg Scholarship; and National Physical Science Consortium. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Asking for Help, Task Planning, Robotics, Artificial Intelligence, Ubiquitous Computing, Human-Computer Interaction

Abstract

Increasingly available mobile devices (e.g., mobile robots, smart phones) are becoming more intelligent in their ability to autonomously perform tasks for users. However, when deployed in complex human environments, these devices still face many sensing, reasoning, and actuation limitations. To overcome limitations, we propose symbiotic relationships as those in which the device can request help from humans in the environment while it performs tasks for them. Because the devices are performing tasks for humans, humans have incentive to help the device complete its tasks effectively. However, they may not always be available or willing to help. We introduce human-centered planning to model and reason about humans in the environment in addition to their own state and goals to determine how to act and whether, who, and how to seek help.

The thesis first contributes an understanding of what and how to model humans in the environment through user studies. We first evaluate whether attributes such as availability and interruptibility affect willingness to help. Then, we contribute to the understanding of how to ask humans for help to increase the accuracy of their responses. We show that providing humans with device context, classification prediction and uncertainty, and additional feedback all increase the accuracy of human responses to device questions. Finally, we contribute algorithms to learn these models both through surveys and online while the device is performing tasks.

The thesis then introduces human-centered conditional, deliberative, and replanning algorithms that use models of humans. We contribute conditional plans that include asking actions to enable devices to perform tasks that they could not otherwise perform. We then contribute a human-centered deliberative planner for a robot to use to determine which navigational path to take that minimizes its uncertainty and maximizes the likelihood of finding available human helpers. Finally, we contribute a replanning algorithm for a robot to determine which helper to have travel to help in a particular location, such as elevators or kitchens.

Through extensive experiments and deployments, in particular with a mobile service robot, this thesis shows that human-centered algorithms trade off task performance with costs of asking and interrupting human helpers increase functionality while maintaining usability.

Acknowledgments

I would like to thank the many people who helped me over the years to complete this thesis.

First, I would like to thank my advisors Manuela Veloso and Anind Dey for their guidance, encouragement, and help. I am grateful for the opportunities and confidence they gave me to work on unique interdisciplinary projects. Their support has been invaluable throughout the last five years. I would also like to thank my committee members, Manuel Blum and Eric Horvitz, for their advice and feedback on my work. Additionally, I'd like to thank the Naval Research Laboratory for giving me the opportunity to explore and find my passion for robotics and research in high school, along with Mark Stehlik, Reid Simmons, Susan Finger, and Anind Dey for their guidance in undergraduate research.

I would like to thank all of my collaborators, my friends, and my lab-mates in the CORAL and Ubicomp groups for their support, time listening to ideas and practice talks, and understanding during the inevitable ups and downs of graduate school - Joydeep Biswas, Brian Coltin, Jenn Tam, Sean Hyde, Kanat Tangwongsan, Samir Sapra, James Tolbert, Tawanna Dillahunt, Matt Lee, Brian Lim, Scott Davidoff, Ian Li, Cetin Mericli, and Tom Kollar. I'd like to especially thank Joydeep Biswas and Brian Coltin, whose contributions on CoBot enabled me to perform my experiments for this thesis.

Finally, I'd like to thank my family for their love, confidence in me, hours and hours of listening to my research successes and challenges, and encouragement from a young age to always ask "why?". Last but not least, I would like to thank Joel. While he was not around at the beginning, he certainly made up for it at the end. Thank you for your never-ending support and patience to help me finish.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Intelligent devices, such as mobile phones and robots, are increasingly available to autonomously perform tasks for us in our environments. These devices sense the environment around them, predict their state within the environment, plan which actions would result in them completing their tasks, and then take those actions autonomously. However, when deployed in complex human environments, intelligent devices still face many limitations in performing tasks for humans.

We view device limitations as roughly divided into two categories - reasoning uncertainty and sensing/actuation limitations. A device with reasoning uncertainty senses the environment, but may not always be able to accurately determine its state or which action to perform. A device with sensing or actuation limitations does not have a sensor or actuator capable of performing an action necessary for its tasks. While a device may overcome its reasoning uncertainty and perform its tasks autonomously, a device with sensing or actuation limitations requires intervention to sense or perform the action for it.

In order to perform tasks in human environments and overcome these limitations during autonomous deployment, devices are typically constrained to tasks that they can perform completely and environments in which they have little uncertainty. Instead, this thesis argues that:

> **Intelligent devices can plan and execute tasks with reasoning uncertainty and sensing/actuation limitations using models of human helpers to determine whether, how, and who to proactively ask for help during execution.**

Because the devices are performing tasks for humans, the humans have incentive to help the device overcome its limitations to complete its tasks effectively. We define **sym-**

biotic relationships as those in which the device asks for help from humans in the environment while acting autonomously to perform tasks for them. In particular, the help can come in two forms:

- Overcoming Sensing/Action Limitations - the human performs an action for the device;
- Reducing Reasoning Uncertainty - the human increases the device's ability to execute the state-action policy;

While a robot could reduce its reasoning uncertainty and learn to perform tasks it requests help for, we do not expect any robot to be able to overcome sensing or actuation limitations autonomously. For example, a robot without arms cannot ever lift a cup of coffee. Through asking for help, a device with limitations can complete tasks that it could not otherwise perform.

Rather than limiting devices' tasks to those that only include actions that devices can perform autonomously, we instead contribute algorithms that reason about, plan for, and overcome device limitations by proactively asking humans in the environment for help. As often as possible, the device senses and acts autonomously. However, when a device has reasoning uncertainty that will affect its task performance, we contribute algorithms for it to ask a human to indicate its state or which action it should take. When a device's task requires an action that it does not have the capability to perform, we contribute algorithms for it to ask a human to perform that action for it.

In addition to increasing task completion, we believe that the successful deployment of devices with limitations also depends on the usability of their requests for help. Humans may not always be available to help their devices. While existing algorithms have taken advantage of humans in order to accomplish tasks, they do not take into account the timing (*e.g.,* availability, interaction history) and human cost (*e.g.,* interruption, time to help) which can affect the algorithms' deployability. This thesis introduces human-centered planning as an approach to determining which actions and help interactions to perform around humans during deployment, including whether, how, and who to ask for help.

The key to our human-centered planning approach is that we model humans as sensors and actuators that can be queried for information probabilistically and at a cost. Just as other sensors or actuators may be available at different times and with different accuracies, so are humans. Similarly, just as sensors and actuators have different associated costs of power or startup time, humans have different costs of interruption and about of time they are willing to help. This thesis argues that although humans are not oracles, they can be modeled and included in plans to help devices complete tasks.

Our human-centered algorithms model both the humans' states in the environment (*e.g.,* availability and interruption), the device's own state (*e.g.,* location, capabilities, and uncertainty) and task goals. Then, they autonomously choose the device's best actions by trading off the costs and benefits of different actions based on the joint human-device state. The thesis contributes both the formal method of developing human-centered planning algorithms and the evaluation of our human-centered planning approach on multiple devices, performing multiple and different tasks, for multiple types of human helpers in simulation and real deployments.

1.1 Human-Centered Planning Approach

In our human-centered planning approach, a device reasons about humans in the environment in addition to its own state and goals to determine which actions to take. In particular, it incorporates models of human state and interaction actions into its model of the environment with its own state and actions. Using this combined human-device model, our planning algorithms can proactively weigh the costs and benefits of different actions and interactions in different states to determine an action plan. Because the actions are based on both the human state and the device's state, the planner optimizes not only the task goals but also usability during deployment.

This thesis contributes a formal approach to human-centered planning that focuses on building human models that accurately represent the tradeoffs that humans make when interacting with their intelligent devices in the environment. We take a three step approach to human-centered planning in order to increase the likelihood that it is both usable and deployable (Figure 1.1). We first perform studies to understand how humans can help devices. The resulting models of humans and help are used in planning algorithms along with the device's own state to determine the best action policy. The action policy is deployed in the environment to trade off task goals with usability. The results of the deployment can then be fed back to improve the planning algorithm.

What human state should be modeled?

Rather than naïvely determining what factors affect whether humans will help devices, this thesis contributes studies of humans interacting with intelligent devices to understand how the human state affect their willingness to help the device overcome its limitations. Our studies include surveys, in-lab experiments, and remote-controlled *in-situ* experiments depending on the potential model requirements. We use the results of the studies to deter-

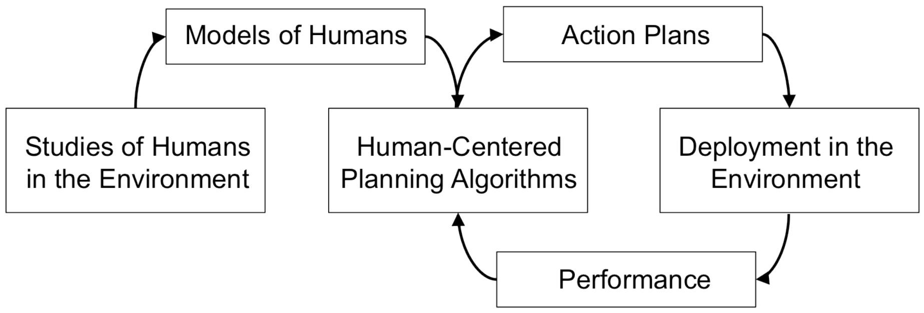

Models of Humans
Action Plans
Studies of Humans in the Environment
Human-Centered Planning Algorithms
Deployment in the Environment
Performance

Figure 1.1: This thesis contributes a three step human-centered approach to planning for limitations.

mine what human state to model and then develop algorithms to predict this human state and use it as input into the human-centered planning algorithms.

How can the human state models be incorporated into deployable planners?

There have been many proposed models of device state (*e.g.,* location) and uncertainty which, along with device actions, can be solved to find the optimal action policy. However, they are often not scalable for devices that act in large environments. This thesis contributes decision-theoretic algorithms that trade off costs and benefits of performing actions and interactions for help and that model large numbers of humans in environments. We demonstrate that our algorithms perform tasks with higher reward or lower task completion time than those that do not model humans or assume that they are always willing to help.

Additionally, while we know which state to model, it can be difficult to learn human models prior to deployment. We contribute an active learning algorithm to learn human models in only a few survey questions and show that it can also be extended to learning multi-utility cost functions such as those needed for our planning algorithms' tradeoffs. We also contribute an algorithm to learn human models online while the device is deployed. Because solving for optimal policies is NP-hard, our learning algorithm only recomputes the policy when there is a statistically significant change in the human model.

How do human-centered plans affect the device's deployment?

While we are able to show in simulation that our human-centered planners choose better actions than other algorithms, we also test how they plan and act under real environment, non-laboratory conditions. Additionally, we use the results from our deployments as feedback to refine our planners to improve task performance.

We apply our three-step human-centered planning approach on two different devices with limitations, as we describe next.

1.2 Devices and Limitations

The thesis introduces human-centered planning algorithms to improve the task performance and usability of two intelligent devices - a smart phone and a mobile robot. While these devices seem very different, they each have sensors and can predict their state and the state of humans around them, determine which actions to take (*e.g.,* when to ring aloud or where to navigate), and execute them. We will demonstrate that both smart phones and robots can use our human-centered algorithms to perform tasks for humans while overcoming their limitations.

1.2.1 Smart Phones

Smart phones are becoming increasing ubiquitous in our environments. Smart phones have a variety of sensors that applications can use to sense the environment and human users, such as accelerometers, GPS, and microphones. Additionally, phone makers also provide in-code access to other information about the user and contacts (*i.e.,* by phone or text message), such as the user's contact list, favorites list, and calendar. Table **??** lists the sensor and other data that we had access to on Google Android phones in this work.

Limitations: Despite the sensors and information that smart phones can access and the ability to autonomously change ring tone volume, they do not autonomously do so. Instead, they still require their users to set phone volume. Several research studies have shown that users often forget to set their phone volume, causing embarrassing interruptions and missed calls for the user (Milewski and Smith (2000); Toninelli et al. (2008)). However, even if smart phones did model volume preferences, they would have reasoning uncertainty about whether to ring aloud or silently and possibly change volume incorrectly at times.

This thesis contributes a smart phone application that senses the phone user's state (*e.g.,* GPS and accelerometer readings) and autonomously sets phone volume according to a user-personalized volume action policy. It learns the personalized policy using our human-centered planning approach in which the phone asks questions "Would you like your phone to ring out loud?" when the benefit of asking the question to learn a better volume policy outweighs the cost of interruption. By asking for help, the phone learns new preferences and reduces uncertainty over time.

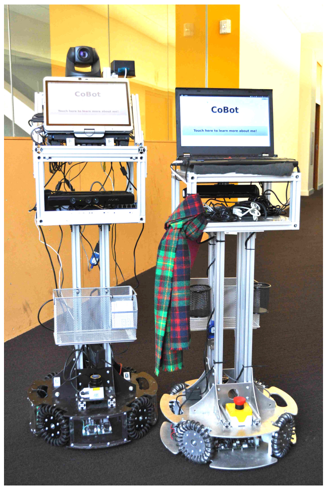
Models of Human Helpers: We focus on phone users' willingness to help their phone learn their personalized volume policy. The requests for volume preferences may happen at inopportune times exactly when they do not want to be interrupted by their phone ringing. Additionally, each person's preferences to answer questions may differ. Some users may never want to answer questions at work while other may. We model users' personalized preferences of when to ask questions.

1.2.2 CoBot - Autonomous Mobile Service Robot

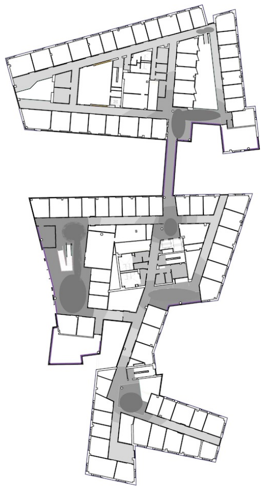
Our robots, all named CoBot (Fig. 1.2(a)), are capable of autonomous localization and navigation in our 9-floor building environment as well as dialog with humans. Each has a laptop with screen that people can use to interact with the robot while it performs tasks autonomously in our building. Tasks for the robot include traveling to offices, delivering messages or mail, and transporting objects from one office to another.

Limitations: Like many robots, CoBot has limitations. CoBot has high localization uncertainty in large open spaces (Fig. 1.2(b) - darker grey areas indicate more uncertainty) Biswas and Veloso (2010); Biswas, Coltin, and Veloso (2011). Commonly, robots with localization uncertainty autonomously overcome it by stopping to collect more sensor data and sometimes backtracking along its path. This thesis contributes two novel human-centered path planning algorithms that trade off localization uncertainty and human availability and interruption costs to determine where to navigate and when to proactively ask for help.

CoBot also has actuation limitations. CoBot does not have arms or the ability to manipulate objects to push chairs out of the way, pick up the mail to give to the building occupants, pour coffee, or push elevator buttons to travel between the building floors. In order to accomplish tasks that require these actions, CoBot must ask for help from someone in the environment. Rather than depending on human supervisors to watch the robot and perform these actions, this thesis contributes human-centered algorithms to model humans in the environment and determine who to proactively ask for help. Because these



(a) The CoBot Robot



(b) Areas of Uncertainty

Figure 1.2: (a) CoBot performs tasks in our multi-floor building but has actuation limitations and (b) localization uncertainty (the darker the grey the more uncertainty in that location) Biswas and Veloso (2010); Biswas, Coltin, and Veloso (2011).

actions are often spatially-situated in the environment at particular locations, the algorithms tradeoff task completion time with human interruption cost and travel distance to determine who to ask for help.

Models of Human Helpers: This thesis focuses people's willingness to help the robot with different limitations. These requests for help take different amounts of time to answer and some require people to travel down the hallway to perform the requested help. We model two types of humans in the environment as helpers - *device users* who are near the robot and receive task benefit and *environment occupants* of buildings who have predefined work spaces and conduct work that requires that they be present in the environment over a period of time but not monitor the robot's progress.

1.3 Evaluation

In order to demonstrate that our human-centered planning algorithms effectively plan for human interaction during autonomous deployments, we measure task performance and usability metrics and compare them against planners that do not model humans or that model humans as always available and willing to help.

Task Performance: Because devices are expected to perform tasks, we measure task performance to understand how our human-centered algorithms plan for and complete tasks compared to other algorithms. We measure *number of tasks completed* and *task completion time* during deployment. Particularly, for requests for help, we measure the *number of autonomous versus assisted tasks, number of asked and answered questions* (these numbers may be different), and the *number of replans during execution*.

Usability: We argue that our interactive intelligent devices must also be usable as they perform tasks for us. We measure the *perceived benefit of the device* compared to the *cost of answering questions, the response accuracy and frequency, and human annoyance and willingness to answer questions over time*.

We demonstrate that our human-centered planning algorithms perform more tasks total and with less uncertainty than algorithms that do not ask for help. Additionally, we show that our algorithms are more usable than those that ask for help but do not model their human helpers.

1.4 Thesis Assumptions

This thesis and the algorithms in it assume that models of humans are stationary during planning and execution time. In other words, we assume that the granularity of tasks is short enough that humans will not change their availability or interruption level in that time. While we do have time-dependent models in which availability changes throughout the day, they do not account for within-task changes. Fine-grained time-dependent models require accurate predictors of these changes (requiring a lot of data each minute of the day). Then, planning algorithms that employ these models exponentially increase planning time to consider different actions that take different amounts of time. While our algorithms can handle these models, we do not employ them in this thesis due to the practicality of collecting such data.

However, the thesis does not assume that humans do not change their preferences over time. We employ learning algorithms to capture human preferences and state over time including becoming more or less accurate or available during the deployment of the device.

We discuss directions to reduce the stationarity assumption in the Future Work Chapter.

1.5 Thesis Overview

This thesis is organized into two parts - Human-Centered Planning Algorithms and Studies of Human Helpers. We first contribute our human-centered conditional (Chapter 2), deliberative (Chapter 3) and replanning (Chapter 4) algorithms and the models of human state they use to determine whether, how, and who to ask for help (summarized in Table 1.1). These algorithms assume that we have precomputed states that can be used in planning and execution. In the second half of the thesis, we contribute studies of humans that demonstrate the need for modeling each of the human state attributes used in the algorithms (summarized in Table 1.2) as well as novel algorithms to learn these attributes from humans in the environments. In particular, the thesis contributes validation of the models used in the human-centered algorithms through user studies of what models to develop (Chapter 5), how to ask for help to improve response accuracy (Chapter 6), and how to learn these models (Chapter 7).

Human-Centered Algorithms: We divide our contributed algorithms along two dimensions - the type of help that the device must ask for and the type of human that the device has access to request help from (Table 1.1. When a device with reasoning uncertainty or actuation limitations asks for help from its user (*i.e.,* those who are nearby or benefitting from the device's task), it uses a human-centered conditional plan that weighs the cost and incentive for the user to answer to determine whether to ask (Chapter 2). If the user does not see incentive to answer or if there is a high cost to answering, they may abandon the device rather than helping it.

When there is no human nearby, we contribute algorithms for a device to instead ask for help from environment occupants (*i.e.,* those who are in the environment but do not directly benefit from the current task). Our human-centered algorithms to request help from occupants weigh the costs of asking each person to determine who to request help from. If the device has reasoning uncertainty, the our human-centered deliberative planner plans actions that minimize the uncertainty while also maximizing the likelihood that there will be a person nearby who is willing to help (Chapter 3). For actuation limitations, our algorithm must determine who to ask and where to navigate (Chapter 4).

Studies of Human Helpers: After contributing our human-centered algorithms that depend on models of human state, we then contribute several demonstrations, through studies of human helpers, that our used state attributes do indeed affect likelihood, cost, and accuracy of humans responding (Table 1.2). Additionally, these studies validate our hypotheses that models of humans are important in making devices that require help deployable. In

Type of Human	State Reasoning Uncertainty	Action Reasoning Uncertainty	Actuation Limitation
Device User	*Conditional Plan*: Ask when uncertain *Human State Model:* - Help Type - Availability - Cost of Help	*Conditional Plan*: Ask when benefit of asking is greater than cost *Human State Model:* - Accuracy - Cost of Help - Incentive to Answer	*Conditional Plan*: Ask at help location and send timeout email *Human State Model:* - Interruption - Cost of Help - Incentive to Answer
Environment Occupant	*Deliberative Planning Algorithm*: Determine where to navigate and who to ask *Human State Model:* - Availability - Accuracy - Cost of Help		*Replanning Algorithm*: Determine who to ask *Human State Model:* - Help Type - Availability - Cost of Help - Incentive to Answer - Interruptibility - Expertise - Distance to Help Location - Frequency of Questions - Recency of Last Question

Table 1.1: This thesis contributes conditional, deliberative, and replanning algorithms to determine whether, how, and who to ask for help with reasoning uncertainty and actuation limitations.

order to understand which state attributes affect both user and environment occupant willingness to help devices, we performed surveys and *in-situ* experiments (Chapter 5). In order to understand how devices can affect the accuracy of helper responses, we performed a series of in-lab studies (Chapter 6). Last, we contribute algorithms to learn helper models both with surveys and online while the device is performing tasks (Chapter 7).

Contributions: This thesis makes the following contributions:

- A symbiotic approach for devices to ask humans for help to overcome limitations during tasks;
- A human-centered planning approach in which we study how humans interact with

Human State Attribute	Study Chapter
Help Type	Chapter 5
Interruptibility	Chapter 5
Distance to Help Location	Chapter 5
Frequency of Questions	Chapter 5
Recency of Last Question	Chapter 5
Incentive to Answer	Chapter 5
Availability	Chapter 5 and 7
Cost of Help	Chapter 5 and 7
Accuracy/Expertise	Chapter 6 and 7

Table 1.2: This thesis validates each of the human attributes used in our human-centered algorithms through user studies. These attributes are used in combination in our human-centered planning algorithms.

devices, and then use those models of humans in planning algorithms;

- A classification of two types of help (reasoning uncertainty and actuation limitations) that devices need and two types of humans (device users and environment occupants) that devices can request help from;
- Human-centered planning algorithms (conditional, deliberative, and replanning) that model human helpers to plan device actions;
- An increased understanding of our two types of human helpers and how their human state affects their willingness to respond to device questions through multiple survey, in-lab, and in situ studies;
- Learning algorithms for capturing personalized models of human states during deployment for use in the planning algorithms;
- Use of human-centered planning algorithms to determine whether, how, and who to ask for help;
- Validation of the human-centered planning algorithms in simulation and deployment in the environment against both naïve models of humans that assume availability and accuracy as well as state-of-the-art algorithms that do not model humans;

Chapter 2

Conditional Planning with Device Users

We define three types of people that intelligent devices interact with while completing their tasks - device users, environment occupants, and bystanders. Device users receive direct benefit from the tasks that the devices are performing. These people interact with devices to request tasks, are near the devices as they are performing the tasks, and are the ones who determine and confirm whether the task has been completed. While users may also help devices complete their tasks, devices like robots may act autonomously without a user and may still need help. In these cases, environment occupants who are located in offices or spatially-situated in specific locations can help (*e.g.,* CoBot requires actuation help in the kitchen and at the elevator). Finally, there are other bystanders who interact with devices in the environment. They may be interested in what the device is doing or testing it to understand its capabilities.

In this chapter, we contribute conditional plans for devices to use to perform tasks that include interactions with each of these types of humans. First, we contribute conditional plans for completing each of CoBot's user-requested tasks in our multi-floor building. The tasks each require device user interaction to confirm their completion in addition to other possible interactions such as speaking messages and transporting objects from one location to another. The navigational plans for each task also include interaction with device users, environment occupants, or bystanders to use the elevator. These interactions must occur in order for tasks to be completed effectively. Finally, they also include possible interactions with bystanders along the way who may be interested in CoBot.

Second, we contribute a human-centered conditional planner for CoBot that requests help to reduce uncertainty and actuation limitations while navigating. This planner assumes that there is a device user being escorted by the robot to meetings in the building. We model the user as a helper in a symbiotic relationship with **incentive to help** in order for the task to be completed and to receive other benefits from the robot such as learning about the building and receiving coffee. However, they also have a **cost of help** associated with the number of questions asked. The planner chooses the best plan that minimizes costs and maximizes incentives to help while asking when the localization uncertainty exceeds a threshold.

Finally, we contribute a human-centered conditional planner for a smart phone to learn to reduce uncertainty about when it should ring and when it should be on silent mode. By trading off the personalized costs of help (interruption) and incentives to help (misclassification of volume later) of the phone's user, the algorithm determines when to ask for the user's volume preferences. We demonstrate that when users have a high cost of help, the algorithm correctly chooses to ask infrequently at the cost of higher volume error. When users have a low cost of help, the algorithm learns a very accurate volume classifier with few interruptions to the user. We show that our human-centered algorithm is more usable than other algorithms that do not model the user tradeoffs, in addition to building accurate volume classifiers.

2.1 Conditional Planning for Robot Tasks

CoBot performs several autonomous tasks for users in our building environment including:

- Go To Room - navigate to a designated room;
- Transport Object - navigate to pick up an object at a room and transport it to another room;
- Escort Person - navigate from the elevator to a room with a person;
- Visitor Companion - escort a person to each meeting on a schedule;
- Telepresence - navigate to locations that a remote user designates (Coltin et al. (2011));

In order to complete these tasks, CoBot must localize and navigate autonomously (Biswas and Veloso (2010); Biswas, Coltin, and Veloso (2011)) and interact with users, environment occupants, and bystanders in the environment. Figure 2.1 shows the interaction of picking up a folder from a room with the user.

We contribute conditional planners for CoBot that include both autonomous actions and interactions to model the control flow for each task. A flow diagram bests represent

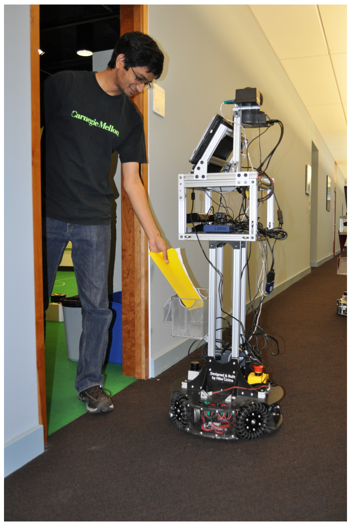


Figure 2.1: CoBot stops at a room to pick up a folder before navigating autonomously to drop it off (Biswas and Veloso (2010); Biswas, Coltin, and Veloso (2011)).

these conditional plans. Figure 2.2 shows the plan for escorting a visitor (device user) from one room to another. It requires one question to confirm the visitor's identity at the pickup location and other to signify that the visitor has reached their destination. The transport object task is similar to the escort task except that rather than confirming visitor identity, it asks a person to place an object in its basket (pictured in Figure 2.1).

Note that as the plan executes, it is reevaluated to determine if localization uncertainty questions are required. These localization questions are of the form "Can you point to where we are on this map?" and the pixel values from the click are translated into (x,y) coordinates for the robot to use to relocalize. This localization help is described further in Chapter 6 - localization study experiments with humans.

Figure 2.3 is a more specific conditional plan for executing the "navigate to destination" action between different floors of the building (found in the escort plan). Because CoBot does not have manipulators to press elevator buttons or vision algorithms to detect which the robot is on, it must ask for help to overcome these actuation limitations. When CoBot arrives at the elevator, it first asks to press the appropriate up/down button (See Figure 2.4). Then, it asks which elevator is open and going in that direction and navigates into that elevator. Finally, in the elevator, it asks to identify when it is on the correct floor and exits at the correct time. If it finds that it exited on the incorrect floor, it replans to navigate back to the elevator and try again.

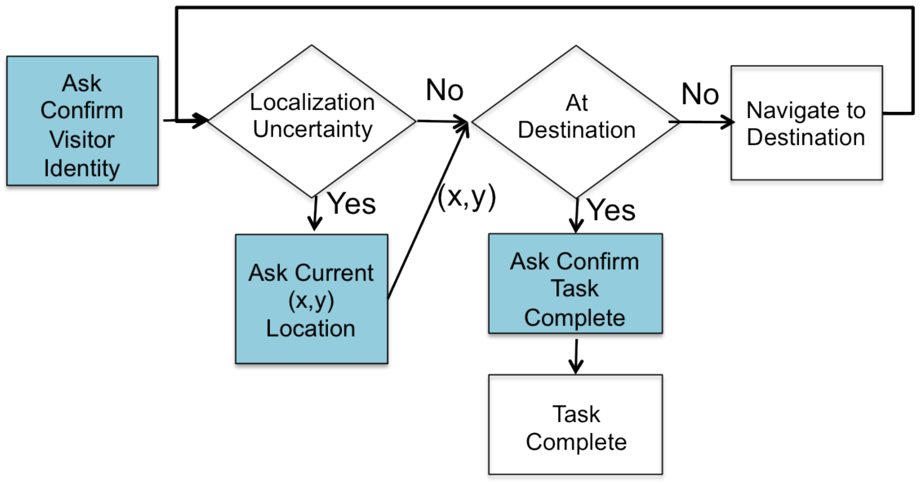


Figure 2.2: In order to escort a visitor to a destination, CoBot confirms their identity, then navigates with them to the destination, and finally confirms that it has completed its task. It asks for localization help if it is uncertain of its location.

Importantly, without including these help interactions in the task plan, CoBot would not be able to complete any task that requires elevator use. In the escort task or others with device users, the user helps the robot use the elevator because he/she also needs to use it and therefore there is low cost of helping. In the escort task, the visitor would not be able to be escorted by the robot if they do not help CoBot use the elevator. In other autonomous tasks where the robot is traveling alone, it needs to seek help from environment occupants to overcome actuation limitations. The conditional plan for using the elevator waits for a response at the elevator. We performed a study of elevator use and found that one person uses the elevator every 5-10 minutes depending on the floor of the building. Spending 10 minutes just waiting at the elevator will delay task performance and therefore usability for task requestors and users. Chapter 4 describes a deployment of the conditional plan to further illustrate this finding and then contributes a novel replanning algorithm to proactively navigate to environment occupants in offices for help to reduce the waiting time at the elevator.

Finally, as CoBot is navigating, it may encounter bystanders along the way. It knows that there are bystanders around when its path is blocked. At these times, CoBot repeats “Please excuse me” to the bystander to indicate that it needs to continue forward to complete its task. CoBot also has a button on its interface for bystanders to press while blocking the path to hear that CoBot “performs tasks on the 6th, 7th, 8th, and 9th floors. You can go online to schedule tasks for me [it] to perform” (Figure 2.5) We prefer that

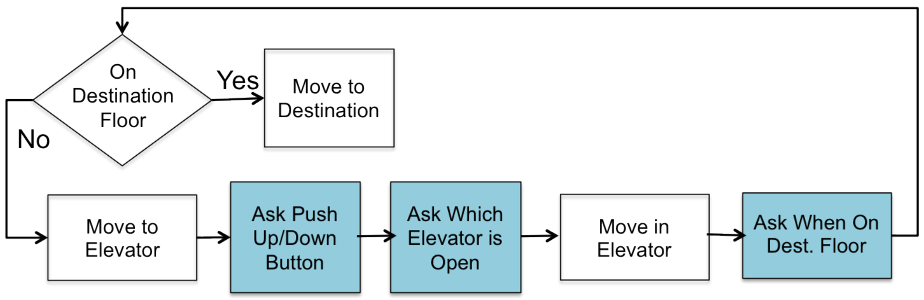
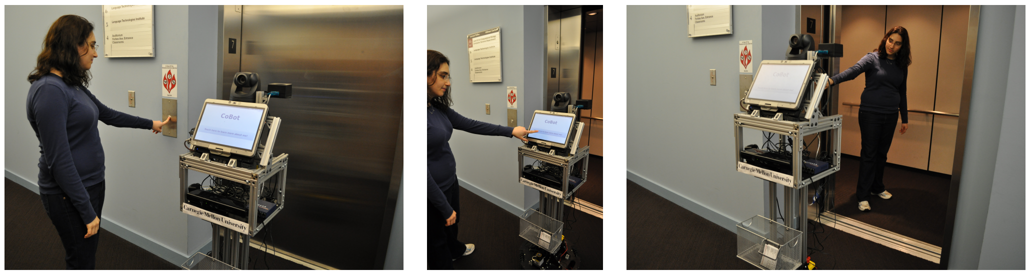


Figure 2.3: CoBot does not have manipulators or sensors to use the elevator but can navigate autonomously. It executes a series of ask actions in between navigation motion commands.



(a) Pressing the down button (b) Indicating the open elevator (c) Holding the door open

Figure 2.4: Humans help CoBot use the elevator.

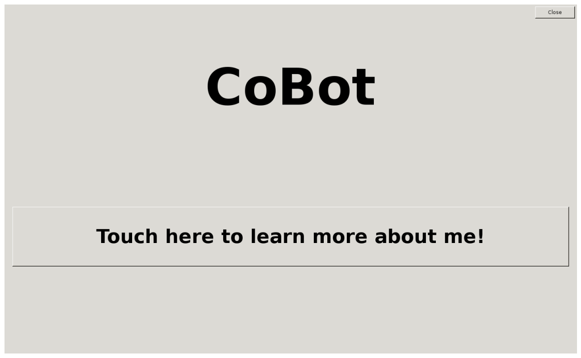


Figure 2.5: Bystanders can press the large button in the user interface on CoBot to learn more about its capabilities.

CoBot continue navigating to complete its task rather than stop for the bystanders in order to maintain high task performance and usability for task users.

To summarize, CoBot's conditional plans assume that there are people in the environment willing to help (device users, environment occupants, or bystanders). These helpers are in symbiotic relationships with their devices with **incentives to help** complete tasks and **costs of help** when they are interrupted. These plans assume that there is always an incentive for someone to help because otherwise CoBot could not be deployed to complete tasks for anyone. Additionally, they assume there is a static cost of asking a question and minimize the number of questions asked. For reasoning uncertainty help, CoBot only asks for localization help when it is very uncertain and it always attempts to perform autonomously before requesting help. For actuation help, CoBot must ask for help every time it needs to use the elevator.

2.2 Reducing Localization Uncertainty with Robot Users

In the previous section, we created static conditional plans to complete tasks and assumed that helpers always had incentive to help because otherwise no task could be completed. However, the Visitor-Companion (Escort) Task is extendable to provide more services to visitors as it escorts them to meetings throughout the day. In this section, we formally define this extended Visitor-Companion Task with states (Table 2.1) and actions (Table 2.2) from the robot's perspective. This formalism is written in the Planning Domain Definition

Language (PDDL) (Ghallab et al. (1998)) with extensions[1]. Because the robot does not know the actions the human can perform, we do not model them in the task. We then contribute our human-centered conditional planning algorithm that models all of the incentives and costs for the visitor to help based on all previous actions in the day in order to determine which actions to perform. The algorithm aims to act as autonomously as possible to reduce the costs of help while satisfying the visitor's and task goals.

2.2.1 Visitor-Companion Task Definition

In this Visitor-Companion Task, we allow the robot to provide information about particular locations during navigation, provide coffee during meetings, and answer other visitor requests. In order to perform the Visitor-Companion Task, CoBot must maintain state about where it is currently located, which meetings have already been attended, and what other tasks it has already performed such as informing the visitor about interesting locations along the way to meetings. Using the state information as *preconditions* or predicate true/false conditions that must be met in order for the robot to be able to perform the action, CoBot can determine which actions to take to satisfy task goal predicates.

States and Actions: While the robot maintains state mostly about itself and the actions it has performed, it also maintains some state about the visitor in order to evaluate the visitor's *costs and benefits* when deciding which action to take (Table 2.1). The robot chooses actions that maximize the benefits and minimize the costs to the visitor (Table 2.2).

We divide the actions into categories - asynchronous (in *italics*: Execute, Inform, Ask, Request) and synchronous (Respond, Process-Response, Process-Request, Notify). While the asynchronous actions can happen whenever the preconditions are met, the synchronous actions require a communication action (Table 2.3) be performed before they can be invoked and affect the state of the visitor. Both humans and the robot can perform both asynchronous and synchronous actions, asking/requesting and offering help to benefit each other.

Asynchronously, the robot can inform visitor about different locations such as labs that might be of particular interest. These interesting locations are initialized at the start of a visitor's day with the `still-interesting` state. When the robot is at a location that it knows about but hasn't already talked about, it can inform the visitor with action

[1]The extension that we propose is the "`prob-or`" operator in listing the effect of an action. When an action `:action a1` has effect `:effect (prob-or e1 e2)`, it indicates that the effect of the action is either `e1` or `e2` with probabilities unknown a-priori. The action `a1` has to internally decide on the effect.

Agent	State Predicate	Description
Robot	(robot-at-loc ?loc)	the robot's current location is ?loc
Robot	(known-loc ?loc)	the robot knows about location ?loc
Robot	(still-interesting ?type ?info)	the visitor does not know about info (type = who, where, when, location)
Robot	(schedule ?loc1 ?loc2)	the visitor's schedule includes the transition from ?loc1 to ?loc2
Robot	(nav-target ?loc)	the robot's current navigation target location
Robot	(success ?action)	the robot successfully completed ?action (navigate, open-door, put-coffee)
Robot	(failed ?action ?why)	failed ?action for reason ?why (whereAmI, openDoors, or getCoffee)
Robot	(next-to-robot ?human)	a human is next to the robot
Robot	(asked ?human ?ques)	asked a human for ?ques (?ques = whereAmI, openDoors, or getCoffee)
Robot	(h-responded ?human ?ques ?ans)	human responded to question with answer
Robot	(visitor-requested-info ?type)	visitor requested info about the next meeting or location
Robot	(visitor-requested-act ?act)	visitor requested the coffee or an email (act = reqCoffee or emailLate)
Robot	(notify-completed ?act)	robot notified the visitor that it completed the request
Visitor	(visited-loc ?loc)	the visitor's visited ?loc for a meeting
Visitor	(late ?loc)	the visitor was late to meeting at ?loc
Visitor	(visitor-informed ?type ?info)	the visitor was informed about info
Visitor	(req-satisfied ?act)	the visitor's request was satisfied

Table 2.1: The Visitor-Companion robot's state predicates and the predicates it holds for the visitor's state. Predicates are true/false variables.

Action Type	Actor (→ Actee)	Action(Parameters)	Description

Execute	robot	nextMeeting(?loc)
		navigate(?loc)
		open-door
		push-button
			robot gives next meeting ?loc
			robot moves to ?loc
			robot opens the door in front of it
			robot picks up coffee (to carry)

Inform	robot → visitor	inform-loc(?loc)
		inform-meet(?type ?info)
			robot informs visitor about ?loc
			robot informs visitor about meeting

Ask	robot → human	ask(?ques, ?action)	robot asks human ?ques (*e.g.,* whereAmI, push-button)
Respond	**human → robot**	respond(?ques)	visitor responds to the robot

Process-Response	robot	process-loc(?ans)
		process-act(?ques ?ans)
			robot processes visitor's loc answer
			robot processes visitor's action
			(*e.g.,* button-press)

Request	**visitor → robot**	request-info(?type)
		request-act(?act)
			visitor requests info
			visitor requests a robot action

Process-Request	robot	proc-req-info(?type)
		proc-req-act(?act)
			processes request about meeting
			robot processes request for action

Notify	robot → visitor	notify-IAmThere(?loc)
		notify-done(?act)
			?loc arrival notification
			task completion notification

Table 2.2: Visitor-Companion robot and visitor's actions. Action types in *italics* are actions the agents perform asynchronously. The visitor both requests help and responds to the robot's questions (in **bold**).

Comm. Type	Action	Effect	Description
Speak	ask proc-req-info inform-loc inform-meet notify-IAmThere notify-done(?act)	asked visitor-informed visitor-informed visitor-informed visited-loc req-satisfied	human is asked for help visitor is informed of meeting info. visitor is informed of the location visitor is informed of meeting information visitor's location is updated visitor's request has been satisfied
Email	proc-req-act(?act)	req-satisfied	meeting host emailed about lateness

Table 2.3: The Visitor-Companion robot communicates with humans in several ways.

Action	Parameters	Preconditions	Effects
inform-loc	?loc - location	(known-loc ?loc), (still-interesting ?loc), (robot-at-loc ?loc)	(visitor-informed ?loc), (not (still-interesting ?loc))
navigate	?loc - location	(nav-target ?loc)	(robot-at-loc ?loc) or (failed nav whereAmI)
ask	?ques - question, ?action - action	(failed ?action ?ques)	(asked ?ques)
notify-IAmThere	?loc - location	(robot-at-loc ?loc)	(visited-loc ?loc)

Table 2.4: CoBot models actions with preconditions and effects to determine its action.

`inform-loc` (See Table 2.4 for preconditions and effects).

The robot can `navigate` past these different locations around the building using the `nav-target` state to maintain knowledge of where it is going. This autonomous action as well as `push-button` include an action completion *uncertainty* or probability of failure based on the robot's limitations (discussed later), which can result in either a success or failure.

Based on the failure (*e.g.,* localization error from `navigate`), the robot can `ask` a human nearby for help. When the visitor responds to a location question, the robot processes the response and updates its location information to continue moving. Otherwise, the robot waits for the action to be taken, updates its state, and continues with its plan. Finally, when the robot arrives at meeting location with the visitor, it notifies him that he has arrived with the `notify-IAmThere` action.

Uncertainty: Unlike socially embedded learning (Asoh et al. (1997)) or other algorithms in which the robot's task is only to ask questions while learning, we expect the agents to perform tasks autonomously when possible. The robot should ask for help only when it lacks the ability to perform some action. In order to model these limitations, some actions have both success and failure effects that happen according to uncertainty - the probability of failure p. If there is no chance of completing an action, $p = 1$. For example, if the robot does not have arms, there is no change it could perform `push-button` itself. These actions will always result in failure and the robot will always request help from a human near the coffee maker with action `(ask ''Please press the up button.'' push-button)`. When $p < 1$, the robot may not complete an action successfully due to the reasoning uncertainty in the robot models. For example, in the `navigate` action, the robot may be uncertain of its location which contributes its successful completion.

Given these states and actions, our human-centered conditional planning algorithm will determine which actions to take.

2.2.2 Human-Centered Algorithm

One could use any planning algorithm to search for all of the sequences of actions that satisfy the Visitor-Companion task goals of taking visitors to their meetings. The plan is conditional because it depends on the precondition predicates being true in order to perform the next action in the sequence. If they are not true, then some other sequence of actions is performed.

There may be many conditional plans that equally satisfy the constraints of the plan, but not all of them are human-centered. In order to determine the best human-centered plan, we define costs and benefits that a robot may have on their state. Formally, we define a cost as a pair $\langle s, c\rangle$ where $s =$(and (pred-i)) the combination of state predicates and c is the cost of s being satisfied. The cost is incurred each time the state is satisfied. For example, the visitor may not want to be late to any meeting. In this case, we model this with the cost that the visitor has not requested the robot to email a meeting host about his lateness:

$$\langle\texttt{(and (visitor-requested-act emailLate)}\ \texttt{(late loc)),}c_{late}\rangle$$

The visitor expects a drink shortly after requesting it, so there is a cost of not receiving it:

$$\langle\texttt{(and (visitor-requested-act drink)}\ \texttt{(not (req-satisfied drink))),}c_{drink}\rangle$$

Additionally, the visitor might assign a cost each time he is asked and responds to question:

$$\langle\texttt{(and (h-responded visitor ?ques)}\ \texttt{(asked visitor ?ques)),}c_{ask}\rangle$$

If these state predicates are ever true, the robot incur a cost of c. Using these costs, the robot can choose the best plan, the best action, or the best time to take an action that minimizes the cost to the visitor. While the robot may not always be able to avoid asking for help, for example, it can ask raise the threshold of how uncertain it is to avoid asking questions if it may be able to perform the action itself. Additionally, it can perform more benefits for the visitor to incentivize the visitor to help and decrease the impact of the cost on the task. The plan with the lowest cost is the one that is executed for the visitor.

In summary, we have enumerated states and actions for a Visitor-Companion robot to create a symbiotic relationship with a human. The robot performs tasks for the visitor, helping him move between his meetings and satisfying other requests like informing him about locations in the building. In cases when the robot has limitations, the visitor helps the robot with the expectation that the robot completes the plan to his satisfaction with a minimal cost. In Chapter 5, we present a study that demonstrates the need for modeling both costs and benefits of visitors to determine whether to ask them for help.

2.3 Reducing Volume Uncertainty for Phone Users

In order to study the deployability of human-centered conditional planning algorithms that model the costs and benefits of device users to determine whether to ask them for help, we focused on smart phone applications. Unlike the Visitor Companion task in which the device user is assigned a robot for a few hours, many people carry smart phones around with them all the time. This platform gives us the opportunity to study the effects of human-centered planning algorithms over weeks rather than hours.

We focus on the particular problem of learning smart phone users' personalized preferences for when their phone rings and buzzes and beeps and when it should be silent. While users can characterize their own preferences by changing phone modes (*e.g.,* ring, vibrate, silent) to avoid unwanted phone calls (Toninelli et al. (2008)), they often forget to set and reset their phone modes, resulting in unwanted interruptions or potentially missing important calls, or SMS or calendar notifications due to silent notifications (Milewski and Smith (2000)). With a model of users' preferences for interruption, a phone could automatically set its volume to avoid inappropriate interruptions and important missed calls.

Phones today offer a variety of sensors such as accelerometers, microphones, and GPS that can be leveraged to classify a user's context and interruptibility preferences. Studies have shown that human interruption in offices can be captured accurately by simple sensors such as these (Horvitz and Apacible (2003); Fogarty et al. (2005)), and other studies have found that users decide whether to answer their phones based on their activity, location, and who is calling all of which are becoming more observable using current phone sensors

(Ho and Intille (2005); Khalil and Connelly (2005); Krishnan (2008)).

We contribute an autonomous phone volume changer which determines whether to ring or be on silent using a logistic regression (LR) classifier. We use LR because of its computational speed and efficiency on small platforms such as phones. The LR model distinguishes between two "classes" of interruption preferences - those in which the phone should audibly ring (LOUD = 1) and those in which it should not (SILENT = 0) - using the features F defined in Table **??**. In particular, for a new situation with features F, LR calculates the probability of those features being labeled as LOUD as:

$$p(\text{LOUD}|F) = \frac{1}{1 + \exp(w_0 + \sum w_i F_i)} \tag{2.1}$$

If the probability $p(\text{LOUD}|F)$ is greater than 0.5, then the prediction is LOUD. Otherwise, the prediction is SILENT.

The classifier defines the weights w_i by minimizing differences (errors) between the labels y^j of personalized volume preferences and the classifier's predicted label Y^j for each training example j. However, while it is possible for machine learning researchers to collect data and build classifiers that apply to all users in some applications, it is infeasible for creating personalized preference models such as those for interruption because different people have different preferences. Additionally, because users often forget to change their phone volumes, their current volume settings are not an accurate indication of their actual volume preferences and the labels cannot be captured automatically as in (Faulring et al. (2010)) to learn email classifiers. Horvitz *et. al*'s BayesPhone learned models of phone users' interruption preferences offline in a survey but asked for clarification of users' interruptibility online when determining phone volume during execution Horvitz et al. (2005). While BayesPhone was successful at modeling interruption costs and changing phone volume, Kern and Schiele argue that if the mobile device could use experience sampling (Csikszentmihalyi and Larson (1987); Shadbolt and Burton (1989)) to elicit preferences while the user is using the device, the resulting classifiers would be more accurate (Kern and Schiele (2006)).

In this section, we contribute our human-centered conditional planning algorithm to elicit volume preferences through experience sampling from phone users in a usable way. We use the phone volume preferences to train our LR classifier and additionally test the accuracy of final learned classifier.

2.3.1 Human-Centered Conditional Planning Algorithm

Experience sampling was originally introduced to intentionally interrupt study participants in order to have them make notes about their current situations (Csikszentmihalyi and Larson (1987)). These interruptions could happen at regular or random intervals with the expectation that participants would be more accurate in describing their current situations in the moment rather than later during interviews. Rather than depend on users to define their preferences before our study or recall them each evening, we use this approach to collect user preferences for training volume classifiers.

We want to use experience sampling to build and train personalized preference classifiers for mobile phone users without affecting the usability of our application. Unlike traditional experience sampling techniques in which the participant should be interrupted, we are interested in minimizing this interruption so that users are more likely to answer the questions over time (Scollon, Kim-Prieto, and Diener (2003)). Several techniques including Random Sampling and Uncertainty Sampling have been proposed for when to collect accurate data from users. However while some focused on minimizing the questions, they do not guarantee that questions minimize interruption.

Random Sampling: In random sampling, the decision to elicit the user's preferences is made irrespective of the classifier that is being built or the user who is responding (*e.g.,* McFarlane (1999)). It is likely that a preference may be asked for the same or very similar situations multiple times, making some of the elicitations extraneous. However, this sampler ensures that the there is a broad set of data to train a classifier with.

In our implementation of this technique, we would like to collect preferences approximately 1/3 of the time when the phone rings. To decide when to ask, the sampler uses the conditional plan that generates a random number p between 0 and 1 and asks if $p < 0.3$.

Uncertainty-Based Sampling: Unlike random sampling, uncertainty sampling builds the preference classifier using the data collected so far and then decides whether to ask for a new preference based on the classifier prediction (Cohn, Atlas, and Ladner (1994); Lewis and Catlett (1994)). The goal of uncertainty-based sampling is to reduce the number of labeled preferences by only asking in situations that have not previously been encountered. If a new situation is encountered, it may benefit the classifier to get the user's preferences in order to classify it correctly in the future. However, if a similar situation was already encountered, the user should not have to provide their preferences again.

Specifically, classifiers such as LR, output a real value p between 0 and 1 rather than the binary 0/1 classification with the rule that if $p <$ threshold of 0.5, then predict 0, otherwise predict 1. We use $p(\text{LOUD}|F)$, defined above, as our uncertainty measure p, where LOUD is defined as 1. The closer to 0.5, the less certain the classifier is of the user's actual preference and the less likely it is that there is a previously labeled situation that is similar to the current one. In our implementation of this technique, the sampler uses the conditional plan that asks for the user's volume preference if the current classifier outputs $0.3 \leq p \leq 0.7$.

Human-Centered Sampling Kapoor and Horvitz proposed a decision-theoretic sampling approach that trades off an interruption cost of helping and a future cost of misclassification (an incentive to help now) to limit the number of questions (Kapoor and Horvitz (2008)). When the uncertainty is high, this technique trades off a predefined cost of asking A (a user's cost of help) with the cost of misclassification M (user's incentive to help) with the aim of minimizing the cost of helping as well as classification error. If the cost of asking is higher than the cost of misclassification, the assumption is that the user is busy. If the cost of misclassification is higher, the assumption is that he is more willing to answer. Concretely, the conditional plan for the decision-theoretic sampler asks for a user's volume preference if $M > A$.

However, their costs are not personalized for each person. Some users may have very high cost of misclassification and therefore may be much more willing to answer questions to train an accurate classifier or vice versa. By more accurately estimating these costs and incentives for each user, we argue that it is possible to create a more personalized asking mechanism that is more usable for each user. We contribute our human-centered conditional planner that uses the decision-theoretic sampling technique with personalized costs and incentives that are learned for each person. We will compare the usability and accuracy of our augmented human-centered experience sampling approach against the other experience sampling techniques.

2.3.2 Experiment

In order to understand the impact of our human-centered conditional planning algorithm on the usability of experience sampling and the accuracy of the resulting phone volume classifier, we designed a four-week experiment. Participants in the study were given our phone application, which learned their volume preferences and actually changed the volume of the phone based on learned classifiers. The application used one of three experience sampling algorithms - random, uncertainty, or decision-theoretic sampling - which asked them about their interruptibility preferences for each of the notification types, and used those preferences to train the volume classifiers.

Study Design and Procedure In order to capture the personalized cost (interruption) and incentive (classifier misclassification) to help train volume classifiers, we first distributed surveys to twenty participants (survey described further in Chapter 5 and presented in Appendix) who had Android version 2.0 or higher phones. We learned 3 cost and incentive models for each participant - one for call ring volume, one for SMS text message volume, and one for calendar notification volume. Upon completion of the surveys, we used a linear regression model to learn personalized cost and incentive models for each participant (details described in Chapter 7). These models were saved in a text file on the participants' phones.

With the personalized models loaded on their phone with the applications, participants were told about the features that the application monitored and that it logged the features of each incoming notification, the classifier's prediction, and labels into a text file that we would collect once the study was complete. In addition to answering the application's questions, they were asked to fill out nightly online surveys on their phone about the accuracy of the three classifiers each day as well as the application's usability. In total, they were asked to train three classifiers for their application, providing their volume preferences when asked, for two weeks and then test the resulting models for another two weeks, each night filling out usability surveys.

Participants were randomly but evenly assigned to one of four conditions - including two for human-centered sampling - which determined the conditional of when to ask for their volume preferences. Because the two human-centered conditions were not significantly different, we do not discuss the details here. For the details on the differences between the two human-centered conditions, see (Rosenthal, Dey, and Veloso (2011)).

- Random Sampling
- Uncertainty Sampling
- Human-Centered Sampling

Because user preferences varied so greatly across participants, we did not test Decision-Theoretic (DT) sampling with a non-personalized cost of help model. Additionally, we do not test Kapoor and Horvitz's DT-dynamic condition (shown to be most accurate in highly changing domains) because we assume that users' preferences remain constant over the four weeks of the study.

Participants were asked to keep the application running at all times during the 4 weeks of the study and were notified via email if the application quit at any time. After two weeks, the application automatically switched from training mode, which asked users for preferences but did not change the phone volume, to testing mode, which used the prediction to turn on or off the volume of the phone for each type of notification. One participant left the study after the training phase because of a family emergency that required her to hear her phone all the time. After four weeks, researchers paid the participants $80, removed the application and collected the logs that were written to the phone over the course of the study.

Measures and Analysis: We measure four dependent variables: the number of questions asked, the accuracy of the classifier (collected each night over the 4 weeks) and the annoyance of both the asking and misclassification (the incentive to help the phone). The classifier accuracy is measured by comparing the classifier's predictions and the user's actual preferences collected from nightly surveys. We compare the experience sampling techniques using a repeated measures ANOVA of the accuracy, number and timeliness of responses over time. We collected annoyance ratings in the nightly surveys, but because participants did not have any other condition to compare to, they all rated their application as usable. Instead, we asked participants during their final interviews to recall specific situations when their application interrupted them, when the volume was incorrect as well as any other general impressions that they had about the application. We used these findings to distinguish the different sampling techniques.

2.3.3 Results

Overall, we found our human-centered conditional plan had a significant effect on the number of questions that participants were asked and the usability and accuracy of the application. Participants in both human-centered conditions reported that they were overall very satisfied with the timeliness of their questions and the resulting models were more accurate for most of the participants compared to the participants in random and uncertainty sampling conditions. We find that human-centered plan participants who predicted they would have high interruption costs had lower accuracy because they were asked fewer questions, but that we can use participants' survey results to add more training examples and increase the accuracy.

Number and Timeliness of Questions Participants received an average of 285 (min 32, max 717) phone calls, SMS notifications, and calendar alarms during the 14-day training period and received an average of 13 (s.d. 9.1), 41 (s.d. 59), and 3.2 (s.d. 5.8) questions respectively over the same period of time. Participants received far more SMS messages than phone calls and calendar alarms and the number of questions about them reflects this difference.

We compared the number of questions that participants received in each condition of the study for each type of notification (phone call, SMS message, calendar alarm) using a repeated measures test to understand whether the number of questions decreased over time and differed between conditions. We found that, for phone calls, both day of training ($F[13, 195] = 4.67, p < 0.01$) and condition ($F[3, 15] = 4.95, p = 0.01$) played a role in the number of questions participants received, but there was no interaction effect ($F[39, 195] = 1.0, p > 0.05$). For SMS messages, there was high variability in the number of questions by participant mainly because some participants received many more text messages than others so we found that there was only a significant effect of day of training on the number of questions ($F[13, 195] = 3.55, p < 0.01$). There were no significant effects on the calendar alarms as all participants received very few questions to learn an accurate classifier. Next, we analyzed the specific effects that the training day and experimental condition had on the number of questions.

A Tukey HSD test on the day of training for each of the phone and SMS messages showed that participants received statistically significantly more questions on days 1 and 2 (mean phone 2.33, SMS 6.96) compared to each of days 5-14 (all phone means less than 1.0 questions per day, SMS means less than 2.5). After day 2, the number of questions decreased for both phone and SMS notifications (2.6). The drop in notifications in the random condition is not significant.

Interestingly, a Tukey HSD test on the experimental condition for phone calls showed that one of the two Human-Centered Sampling conditions resulted in a statistically higher number of questions (mean 1.6 questions per day) compared to Uncertainty sampling (mean .47 questions) and the second Human-Centered Sampling condition (mean 0.65 questions). There was no statistical difference between Random sampling (mean 0.96) and any other condition. Upon further investigation, we found that 4/5 participants in the first Human-Centered condition reported low estimated costs of asking and were willing to answer questions - each had an average cost of less than 4 out of 7 - compared to only 2/5 with low costs of asking in the other condition. When we add an extra independent variable representing a binary high or low cost of asking in our analysis, we find (as expected) that participants in both Human-Centered conditions who indicated they had a low cost of asking were asked statistically significantly more questions per day compared to those with a high cost - on average 1.45 compared to 0.52 (F[1,6] = 6.51, p ¡ 0.05). This cost accounts for the differences in these two conditions.

Despite the higher number of questions for 6 out of 10 of the Human-Centered condition participants, all participants in this condition reported that they were very satisfied

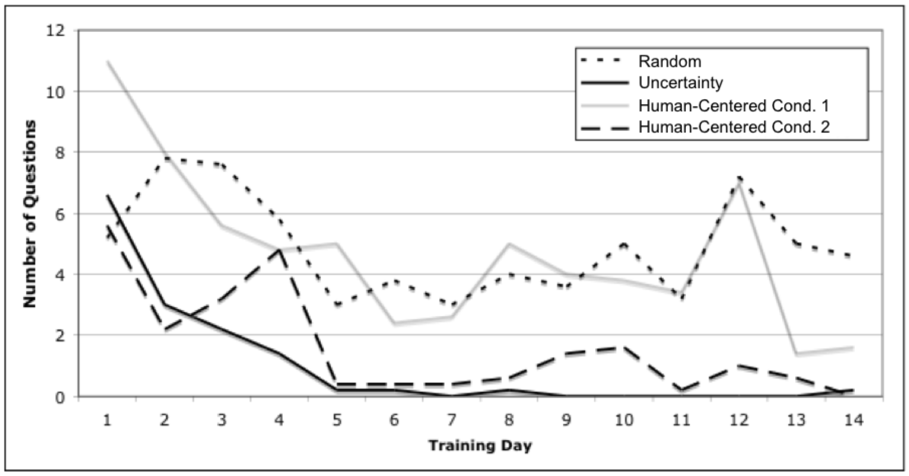


Figure 2.6: As the classifier uncertainty decreased through training, the number of questions decreased for Uncertainty and both Human-Centered conditions. However, it did not decrease for Human-Centered participants who said they were willing to answer more questions to increase accuracy.

with the timeliness of the experience sampling questions. Many participants in the random and uncertainty sampling conditions said they "eventually got used to the questions" but were annoyed by them before that. This indicates that human-centered models had the effect we intended, in reducing the number of questions when users had high interruption costs and asking at more appropriate times for all participants including those who received questions everyday.

Accuracy: Thirteen out of nineteen participants reported at the end of the study that they were happy with the accuracy of their application. Three requested to see the application in the Android app store to download again. The accuracies of the conditions were 0.83 (s.d. 0.1) for random sampling, 0.85 (s.d. 0.1) for uncertainty, 0.88 (s.d. 0.22) for the human-centered sampling. The difference in accuracy between conditions is not statistically significant.

We combine the Human-Centered conditions to show the differences in accuracy between the 6 participants with low costs of asking compared to the 4 with high costs (Figure 2.7). Three of the four high cost participants in the Human-Centered conditions had accuracy lower than 0.8 for phone calls and text messages (mean 0.66, s.d. 0.16) compared to an average accuracy of 0.98 for participants with low cost of asking. Our human-centered samplers with personalized cost and incentive models are capable of very high

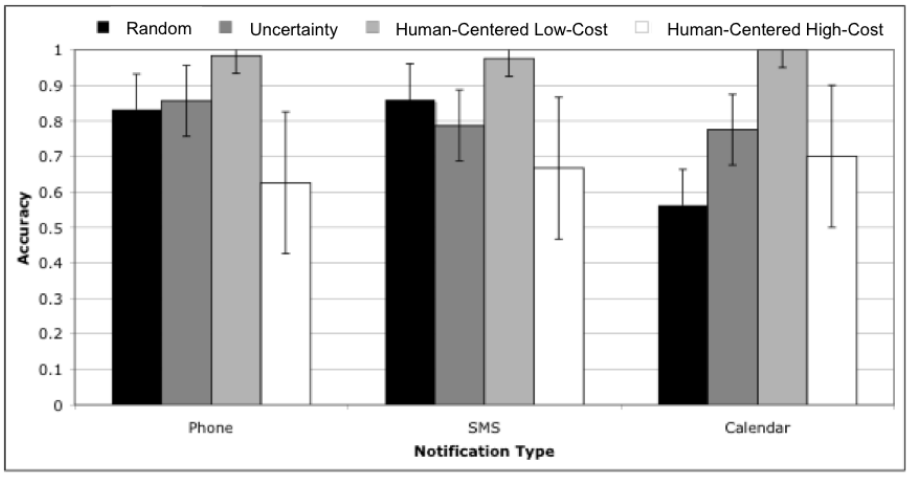


Figure 2.7: Participants with low costs of asking in Human-Centered conditions had the highest accuracy classifiers for each notification type (mean 0.99, 0.97, 1.00 respectively). Three participants in the Human-Centered conditions had high costs of asking because they were not asked enough questions to create accurate classifiers.

accuracy when users are willing to answer questions. The experience samplers with high costs could not identify enough situations to ask but maintain usability, and the lack of labeled training data resulted in low accuracy for these classifiers.

In an effort to create more accurate classifiers for three participants with high costs of asking, we examined the participants' survey responses to understand if their predictions were accurate. One participant's schedule and corresponding volume preferences changed after providing survey responses and the training period. Because the participant did not anticipate these changes, a classifier trained on these survey responses could not have been accurate. For the two other participants, however, the survey responses would have increased the classifier accuracy. For example, one participant's classifier turned the volume off in the evenings when he was relaxing causing him to miss many phone calls and text messages. The human-centered sampler never asked for his preferences in this situation in order to preserve usability. If the classifier had used his single response to the survey that he did want his phone to ring and beep - his accuracy would have increased from 75% to over 92%. We conclude that we can use participants' survey responses as additional training data for inaccurate classifiers.

In summary, participants in human-centered conditions reported that they were very satisfied with the timeliness of the questions they were asked compared to the participants who received random and uncertainty sampling. The resulting models were more accurate for most of the participants in these conditions as well. However, some human-centered sampling participants received fewer questions than others due to their high cost models and this affected the accuracy of their classifiers. We find that in most cases we can use participants' survey responses to increase the accuracy of the classifiers when they have high interruption costs.

2.4 Discussion

In our human-centered conditional planning algorithms, we contributed two techniques for evaluating the cost and incentive for devices to take asking actions - on the robot by finding a low cost plan and on the phone by finding times when the user has more benefit than cost. On the phone in particular, we then deployed the algorithm for several weeks and compared the accuracy and usability of three different experience sampling algorithms. We found that our human-centered sampling with personalized cost and incentive to help models was most accurate and asked questions at the most appropriate times. Next we address some of the phone participants' difficulties and suggestions that they made after using our application for four weeks.

Modeling Costs and Incentives: Our main assumption in using experience sampling was that participants have difficulty predicting their volume preferences in advance, but that we could use prior predictions to approximate their costs and incentives for helping in different situations. We found that overall, this approach was very successful in maintaining very high accuracy while limiting the interruptions at inappropriate times. We conclude that we were able to use the cost and incentive survey responses to build an accurate model of the phone users. However, our surveys were very long and repetitive. Chapter 7 details the approach to learning the phone models from the survey data as well as a novel approach to learning from shorter surveys.

Asking for Training Examples: Thirteen participants out of twenty preferred answering questions over time and thought their *in situ* responses were more accurate than their survey predictions. Three more thought a combination of surveys and experience sampling would be most accurate. Participants who preferred the questions reported that they liked that "it prompted me because it made me think of what I'm doing now" and that is hard to do before using it. This finding mirrors other experience sampling findings that participants answer more accurately in the moment, but contradict other HCI arguments that users should not be interrupted to train classifiers (Faulring et al. (2010)).

Participants who received few questions and therefore poorly trained classifiers said that they would have been willing to answer more questions if they were told that their costs affected the classifier accuracy. A visualization, for example, showing the costs of interruption and the average resulting accuracy could allow participants to see the results of their tradeoffs concretely before using the application in the future.

Volume Preferences Change over Time: We also found that participants' volume preferences changed throughout the study. Participants started new routines in the middle of the study - either starting classes or their kids started new activities. Because they had already started or even completed the training of their classifier, they could not reverse or change the previous responses and their classification accuracy suffered. Participants reported at the end of the study that they wanted to change or start the training over because they had such different preferences. As a result, we argue that applications should continue asking and employ lifelong learning techniques such as forgetting (Kapoor and Horvitz (2007)) or at least allow users to change their preferences to maintain accuracy as they drift or schedules change over time. We discuss this more in future work.

Need for Intelligibility: In addition to helping users understand the questions they were being asked (described further in Chapter 6), intelligibility of the learned volume classifier itself became a big issue for our participants as their phone applications transitioned to testing mode. Uncertain of what their classifiers had learned, many participants emailed the authors asking how to find out what they should do if their classifiers learned the wrong thing. We argue that offering a "what if" interface (in which participants could have set different features to see the resulting prediction, Lim, Dey, and Avrahami (2009)) could have reduced some of the uncertainty and lack of control that users felt during testing mode in our study. Users could check that their classifiers make accurate predictions and provide extra examples for those situations in which it does not.

Participants also requested an interface in which they could see and change the rules that were generated for their classifier, especially if it was consistently wrong about a set of situations. We found that the classifiers were most overconfident in the uncertainty sampling condition and if users could adjust the classifiers during both training and testing phases, it could have reduced the potential errors and helped identify opportunities for the sampler to request more preference data. One student participant, for example, said that his classifier learned to turn his ringer off too early in the evening and this could have been easily resolved if he could have set the time feature. However, it is often difficult to show the rules of a classifier in a simplified way.

2.5 Chapter Summary

In this chapter, we have contributed human-centered conditional planning algorithms that model costs and incentives of users to help devices overcome limitations. We first enumerated robot and human's states and the robot's actions to complete its task of escorting users such as building visitors to a schedule of meetings during the day. We then contribute a human-centered conditional plan to determine when to ask for help to overcome limitations during the task. While the robot plans what actions to take, it also computes the cost of help of the many possible satisfying plans to determine which is least costly for the user.

Second, we contribute a human-centered conditional plan to determine when to ask phone users for their volume preferences. We introduced a human-centered experience sampling technique that asks users to predict their costs and incentives to help train their volume classifier. We deployed our volume application to learn users' preferences over 2 weeks and test the resulting classifier for 2 weeks, comparing the usability and accuracy of our experience sampling technique against other traditional techniques. We demonstrated that our human-centered conditional planning algorithm learns accurate classifiers while interrupting users for their volume preferences few times.

Chapter 3

Deliberative Planning with Environment Occupants

We have contributed conditional planners to determine whether devices should ask from help from device users as they execute with reasoning uncertainty and actuation limitations. However, CoBot in particular will not always be near device users when it requires help. Instead, we realize that as a robot navigates down an office corridor, it may pass by people who could provide it additional observations to help it perform its tasks - environment occupants. This chapter contributes a probabilistic deliberative planner that determines which navigational path to take, given that it has localization reasoning uncertainty and environment occupants with varying characteristics that are located in different areas of the building.

In particular, when CoBot must travel autonomously from one location to another, it plans its path. CoBot may receive different observations about its location along different paths and as a result may have localization uncertainty along each. In order to overcome this uncertainty, as in the previous chapter, CoBot can either continue navigating autonomously or can stop and ask for help from environment occupants. The help serves as an *observation* and is an indication of the (x, y) location of the robot. Given the new observation, CoBot becomes more certain of its location and continue navigating.

However, just as different paths may have different likelihoods of localization uncertainty, we find that environment occupants in different locations may not always be **available** to help or provide **accurate** help (details in Chapters 5 and 6). They each may also have a **cost of helping** in terms of the annoyance of interrupting the human and the time it takes for them to respond. A robot that executes optimal actions without taking into account availability, accuracy, and cost of human help may fail to receive observations when it needs them and may fail to perform tasks and navigate successfully.

Towards the goal of planning when needing help, we introduce Human Observation Provider POMDPs (HOP-POMDPs) as a framework for reasoning about the locations and limitations of environment occupants as well as device limitations to determine the optimal navigation policy. This framework hinges on the idea that humans are like sensors and actuators that can be queried at a cost and with varying accuracy and availability. By modeling humans as observation providers like sensors and actuators, we reduce the complexity of modeling humans as full agents without assuming that they are oracles and always available.

3.1 Humans as Observation Providers

We first formalize the state of environment occupants who may provide observations to improve reasoning uncertainty (Rosenthal and Veloso (2011); Rosenthal, Veloso, and Dey (2011)). In particular, we will model the probability of a robot receiving an observation from an occupant in terms of the human's availability, accuracy, location, and cost of help.

Location: We assume that environment occupants are located in a particular known location in the environment (*e.g.,* an office), and can only help the robot from that location. When the robot is in state s it can only ask for help from the human h_s in the same state. While h_s in an office location represents a single person, h_s in a kitchen or public area represents the type of people who may be at that location at any time.

Availability: Because occupants are not directly benefitting from the robot at the time of execution, they may be busy with meetings or other activities and they may not be in their office or public area at all. We define availability of an environment occupant as their response rate. This availability represents both how often they provide an observation how often they perform an action that results in a different observation (*e.g.,* pressing buttons for the elevator or putting objects on the robot to deliver). This probability actually combines whether occupant is in the office and whether they were busy in their office or not, because the robot only needs to know how often they respond. For the remainder of this chapter, we assume that humans are helping with localization help, although this can be easily extended by adding additional observations for action limitation help.

Concretely, as a result of taking the ask action a_{ask} from state s, the robot receives an observation o from the occupant human h_s. We define availability α_s as the probability that a human provides a non-null observation o in a particular state s: $0 \leq \alpha_s \leq 1$. Receiving the observation o_{null} is equivalent to receiving no observation or timing out waiting for an answer from that occupant (*i.e.,* the occupant was unavailable). If there is no occupant ever available in particular state, then $\alpha_s = 0$.

When the occupant is available, it provides an observation $o \neq o_{null}$:

$$\sum_{o \neq o_{null}} p(o|s, a_{ask}) = \alpha_s \tag{3.1}$$

When the occupant is not available, we say it results in observation o_{null}:

$$p(o_{null}|s, a_{ask}) = 1 - \alpha_s \tag{3.2}$$

The probability of providing any observation $\sum_o p(o|s, ask) = 1$. However, the observation o that an available occupant provides may or may not be accurate.

Accuracy: When human occupant h_s responds to the action a_{ask} with observation $o \neq o_{null}$, the probability that he correctly responds with an observation representing his state s - o_s - depends on his accuracy η_s. If the occupant is not accurate, he responds with a different state observation $o_{s'}$. The more accurate the human h_s, the more likely they are to provide a true observation o_s.

Formally, we define the accuracy η_s of occupant h_s as the probability of providing o_s given that they provide any observation at all $o \neq o_{null}$ (their availability α_s).

$$\eta_s = \frac{p(o_s|s, a_{ask})}{\sum_{o \neq o_{null}} p(o|s, a_{ask})} = \frac{p(o_s|s, a_{ask})}{\alpha_s} \tag{3.3}$$

Cost of Help: While robot researchers and supervisors are generally willing to answer an unlimited number of questions, we find that environment occupants have a cost of helping. The cost of help may be in terms of the time it takes to answer the question and the cost of interruption, limiting the number of questions that should be asked Armstrong-Crews and Veloso (2007).

Let λ_s denote the cost of help from h_s. These costs vary for each person, but are assumed to be known at planning time. The cost for querying the human if they answer with a non-null observation $o \neq o_{null}$ is

$$R(s, a_{ask}, s, o) = -\lambda_s \tag{3.4}$$

In our experiments, our robot receives no reward if the person does not respond, $R(s, a_{ask}, s, o_{null}) = 0$, because we assume that they were not bothered by a request that the do not answer. We will show that because there is no negative reward for not responding, the policy will favor asking less available but less costly occupants rather than asking more available but more costly ones.

Next, we use these definitions to introduce Human Observation Provider POMDPs (HOP-POMDPs) to plan device actions based on both reasoning uncertainty and the environment occupant models. We then introduce algorithms to solve and find a policy for the HOP-POMDP and to execute that policy.

3.2 HOP-POMDP: Human as Observation Providers POMDP

Goal-directed action policies in uncertain environments have primarily been modeled using Partially Observable Markov Decision Processes (POMDPs) (Schmidt-Rohr et al. (2008); Karami, Jeanpierre, and Mouaddib (2009); Armstrong-Crews and Veloso (2007)). To briefly review, POMDPs (Kaelbling, Littman, and Cassandra (1998a)) are represented as the tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{O}, \Omega, T, R\}$ of states $\mathcal{S}$, actions $\mathcal{A}$, observations $\mathcal{O}$ and the functions:

- $\Omega(o, s, a) : \mathcal{O} \times \mathcal{S} \times \mathcal{A}$ - observation function, likelihood of observation o in state s after taking action a
- $T(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ - transition function, likelihood of transition from state s with action a to new state s'
- $R(s, a, s', o) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{O}$ - reward function, reward received for transitioning from s to s' with action a

In particular, we are interested in modeling environment occupant requests for help within the POMDP framework. Prior work on modeling humans has focused on Multi-Agent POMDPs that combine the possible states of the robot R, human H, and the environment E to form a new POMDP representing the task for both the human and robot (*e.g.,* Schmidt-Rohr et al. (2008); Karami, Jeanpierre, and Mouaddib (2009)). These models represent the human as an agent in the robot's environment that it can interact with. However, multi-agent POMDPs have increased complexity in terms of their exponentially larger state spaces which are less tractable to solve.

Additionally, information provider "oracles" who are always available and accurate have also been considered to reduce uncertainty in POMDPs. Oracular POMDPs (OPOMDPs) plan for needing help to reduce uncertainty, modeling the oracle states separately from the robot's states (Armstrong-Crews and Veloso (2007)). OPOMDPs assume that there is an always-available oracle that can be queried for observations from any of the robot's states at a constant cost of helping, λ. The robot executes the best non-asking for help policy (the Q^{MDP} policy, Littman, Cassandra, and Kaelbling (1995)) unless the cost of helping is lower than the cost of executing under uncertainty. However, we will show that taking into account occupant's accuracy, availability, and cost of help is important in determining the optimal action policy.

3.2.1 HOP-POMDP Formalization

We define the Human as Observation Provider POMDP (HOP-POMDP) to model environment occupants' availability, accuracy, and cost of help within the robot's own state, increasing the tractability of solving optimal policies for them while still taking them into account at planning time.

Let HOP-POMDP be $\{\Lambda, \mathcal{S}, \alpha, \eta, \mathcal{A}, \mathcal{O}, \Omega, T, R\}$. where

- Λ - the cost of helping defined for each environment occupant
- α - the availability defined for each environment occupant
- η - the accuracy defined for each environment occupant
- $\mathcal{A} = A \cup \{a_{ask}\}$ - autonomous actions and an ask action
- $\mathcal{O} = O \cup \{\forall s, o_s\} \cup o_{null}$ - autonomous observations, a human observation per state, and a null observation
- $T(s, a_{ask}, s) = 1$ - self-transition for asking actions

Specifically, let h_s be the human in state s with availability α_s, accuracy η_s, and cost of being asked λ_s. Our observation function Ω and reward function R reflect the occupants. Remaining rewards, observations, and transitions are defined as in any POMDP. We rely on traditional POMDP solvers for generating policies for HOP-POMDPs.

Unlike Multi-Agent POMDPs, our humans are not modeled in the states of the HOP-POMDP, significantly reducing the number of states and increases the feasibility of solving the HOP-POMDP policies. Additionally, unlike OPOMDPs, the humans in the environment do have varying availability and cost of helping. Because our model includes these limitations, unlike other approaches that query for help during execution, a robot using our model can plan its policy to take these limitations into account and determine how to navigate and who to ask for observations.

3.2.2 Assumptions: Humans vs. Noisy Sensors

While we rely on traditional POMDP solvers, the execution of HOP-POMDP policies is non-standard. In particular, POMDP execution assumes that actions and observations are probabilistic due to noisy sensors and actuators. However, when a robot arrives at their location, the occupant is either available to answer or is not. While we define the availability as the probability of providing observations, this is not completely true. When humans are available, we assume they *always* provide observations. Querying the human multiple times (as is common in POMDPs to overcome sensor noise) will not result in an occupant *becoming* available.

When planning, the robot should take into account the likelihood of receiving a response based on availability. However, when executing, the robot should sense the availability of the human while executing so that it does not query the occupant repeatedly waiting for him to become available. We contribute an algorithm for avoiding multiple queries during the execution of our HOP-POMDP model. In particular, we use the Q^{MDP} action which chooses the best non-asking action (Kaelbling, Littman, and Cassandra (1998a)) (like OPOMDPs, Armstrong-Crews and Veloso (2007)) when the ask action fails.

3.2.3 HOP-POMDP Policies

There have been many proposed algorithms to solve the state-action policy for the POMDP (Aberdeen (2003)), and HOP-POMDPs can be solved with any general POMDP policy solver that allows for pure information gathering actions (actions with no state change, only observations). Those that do not include pure information gathering actions such as heuristic MDP solvers (*e.g.,* Q^{MDP}) will not include a_{ask} actions in their policies, because they assume complete observability and thus that the robot should never need to ask for help. Intuitively, we can roughly divide policy solutions into two categories:

1. A robot could assume that the best navigational path will contain available humans (as OPOMDPs do), or

2. A robot could also take actions to move towards areas where help is more likely and less costly while navigating to the goal.

Optimal POMDP solvers can be used to solve our HOP-POMDP policies and will take into account occupant availability, accuracy, and the cost of helping using our observation function. However, it has been shown that solving POMDPs optimally is PSPACE-HARD (Papadimitriou and Tsisiklis (1987); Madani (2000)).

A heuristic MDP solver for a HOP-POMDP such as JIV for OPOMDPs (Armstrong-Crews and Veloso (2007)), does not take into account the human observation provider availability and accuracy when planning paths. The JIV algorithm uses the Q^{MDP} policy to determine how to act without asking, and during execution tests whether there is an information gain in getting an observation to reduce localization uncertainty (if the value of asking v_h the occupant for the true state is greater than the value of acting autonomously - Q^{MDP} value). The algorithm can be solved more tractably, finding the best path and asking along the way when it has high localization uncertainty.

We are interested in understanding the availability and cost of help parameters that would cause the two different solvers to create different policies, in order to understand when it is possible to use the more tractable solution. We will compare adapted OPOMDP JIV policies with optimal HOP-POMDP policies on a benchmark task and in our real-world environment.

3.2.4 Policy Execution

Before comparing the policies, we first must address the difference that humans do not answer probabilistically as typical sensor observations do. When executing heuristic or optimal HOP-POMDP policies, the robot must sense the availability of the humans. We assume that this can be done at the time of the ask query. The occupant is available if the observation $o_{s'}$ is provided and is not available if o_{null} is given. In real world experiments, the robot received o_{null} if the human did not respond with $o_{s'}$ within 30 seconds (discussed further in Chapter 5). If a human is not available, it is not feasible to execute a_{ask} repeatedly until o_s is received because availability will not change immediately.

Therefore, when the policy specifies a_{ask} but o_{null} is received on first query, policy execution should specify a different action. In our implementation, our policy executes the Q^{MDP} policy action that chooses the best non-asking action to avoid querying a human that is unavailable. This change in policy when humans are not available can also be implemented in the JIV algorithm for OPOMDPs (Algorithm 1). We adapt the JIV algorithm - $\mathrm{EXECUTE\_JIV}$ - to include occupant availability modeled in HOP-POMDPs and

Algorithm 1 EXECUTE_JIV(HOP-POMDP)

```
// Solve the underlying MDP
$(Q^{MDP}, V^{MDP}) \leftarrow$ SOLVE_MDP$(\mathcal{S}, \mathcal{A}, T, R)$
// Initialize the belief
$b \leftarrow b_0$
loop
  // Find the best MDP action given
  // the current belief
  $(v_s, a_s) \leftarrow$ BEST_QMDP_ACTION$(b, \mathcal{A}, V^{MDP})$
  // Determine the Information Value
  // of asking a human
  $v_h \leftarrow \rho(b, ask) + \gamma * V^{JIV}$
  // Ask a human if available and
  // value of asking is greater than acting
  if $\alpha_s$ and $v_h > v_s$ then
    $b \leftarrow$ true state
  else
    $b \leftarrow \tau(b, a_s)$
  end if
end loop
```

determine the cost of help using the belief reward function ρ.

$$\rho(b, a_{ask}) = \sum_s -b(s)\lambda_s \tag{3.5}$$

and tries only once to ask before executing another action

$$\text{ask if } \alpha_s \text{ and } v_h > v_s \tag{3.6}$$

While it is possible that the robot could leave state s after asking once and return to the same state soon after, we believe this is valid compared to asking continually without trying a different path as it is possible for the occupant to become available since the robot left the state.

We next compare these algorithms in terms of their acting and querying policies and their final reward during execution.

3.3 Experiments

In order to understand the differences between the oracle OPOMDP and optimal HOP-POMDP policies when tested on humans with limited availability and varying costs of helping, we compare the EXECUTE_JIV Algorithm to the execution of a policy solved using the Witness algorithm (Kaelbling, Littman, and Cassandra (1998a)) implemented by Cassandra and distributed online (Cassandra (2011)). For the purposes of our example, the JIV heuristic OPOMDP solver plans identical policies compared to optimal OPOMDP solvers.

We created a benchmark HOP-POMDP with two routes to two goal states and systematically varied the cost of helping each of two environment occupants, the cost of traveling to each occupant, and their availabilities and executed the policies to understand how the reward differs.

Our benchmark HOP-POMDP contains 5 states and 2 actions with two humans h_2 and h_3 (in states 2 and 3) and two final states (4 and 5) (Figure 3.1).The robot starts at state 1 and chooses to take action B or C, where

$$\begin{array}{ll} T(1,B,2) = 0.75 & T(1,B,3) = 0.25 \\ T(1,C,2) = 0.25 & T(1,C,3) = 0.75 \end{array}$$

The robot can then take action B or C from states 2 and 3 to states 4 or 5, where

$$\begin{array}{ll} T(2,B,4) = 1.0 & T(2,C,5) = 1.0 \\ T(3,B,5) = 1.0 & T(3,C,4) = 1.0 \end{array}$$

However, the reward for state 4 is -10 and the reward for state 5 is +10. To be consistent with previous OPOMDP work, our benchmark requires the robot to ask for help to receive an observation and it receives no observations (o_{null}) except when it does ask for help. The robot has the opportunity to ask for help in states 2 and 3 to determine its state and ensure it receives +10 by choosing the correct action (action C from state 2 and action B from state 3).

In our experiments, we varied the availability of each human $0 \leq \alpha_2, \alpha_3 \leq 1$ in increments of 0.1, the cost of help λ_2 and the cost of traveling to state 2 $R(1, B, 2, *)$ each with the values in Table 3.1 while keeping $\lambda_3 = 1$ and $R(1, C, 3, *) = 1$. Note that we did not also vary the occupant accuracy as the OPOMDP assumes that occupants are always accurate. We do show how to learn both availability and accuracy for HOP-POMDPs in Chapter 7.

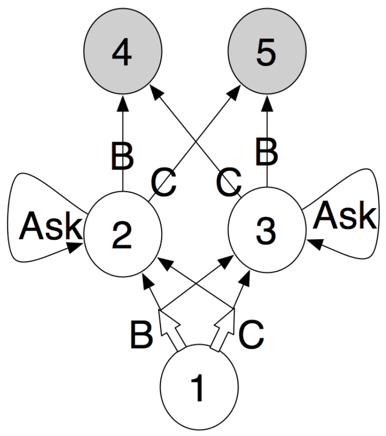


Figure 3.1: We varied the availability and cost of helping humans at states 2 and 3 along with the cost of traveling from state 1 to each of 2 and 3.

Cost of help and cost of traveling to state 2						
0.125	0.25	0.5	1.0	2.0	4.0	8.0

Table 3.1: We varied the cost of help h_2 and the cost of traveling to state 2 with the same values.

We, then, created a HOP-POMDP for our building based on observed availabilities collected previously to show how the two algorithms perform in practice on larger state spaces.

3.4 Experiment Results

3.4.1 Benchmark Policy

In total, we tested a simulated robot navigating using 5929 combinations of cost of traveling, availability, and cost of helping to understand the differences between the OPOMDP and optimal HOP-POMDP policies. We compare the policy in state 1 (taking action B or C), whether the occupant is queried for an observation, and the average collected reward over 1000 executions of each policy. As discussed in the previous section, we limited the robot to only a single attempt to ask a question per execution - the robot could not continually query the same human until they provided an observation.

State 1 - Optimal Policy for $\alpha_2 = 1.0$ and $\alpha_3 = 0.0$

cost of	cost of traveling to state 2						
help fromh_2	0.125	0.25	0.5	1.0	2.0	4.0	8.0
0.125	B	B	B	C	C	C	C
0.25	B	B	B	C	C	C	C
0.5	B	B	B	C	C	C	C
1.0	B	B	C	C	C	C	C
2.0	C	C	C	C	C	C	C
4.0	C	C	C	C	C	C	C
8.0	C	C	C	C	C	C	C

Table 3.2: Although $\alpha_2 = 1.0$ and $\alpha_3 = 0.0$, the optimal HOP-POMDP policy chooses action C when the costs of helping for h_2 and the cost of traveling to h_2 are high.

Policy in State 1 As expected, the OPOMDP policy always chooses the shortest path, taking action C in state 1 when $R(1, B, 2, *) < R(1, C, 3, *)$ irrespective of the availability of the occupants. It expects to always be able to find an available occupant along this path. The optimal HOP-POMDP policy is different from the OPOMDP policy in 39.67% of the tests because it takes into account who is available and their costs of help.

Interestingly, as occupant h_2 becomes more available, the optimal HOP-POMDP policy chooses action B **less** often than the OPOMDP policy because the cost of helping is taken into account. For example, at the most extreme when $\alpha_2 = 1.0$ and $\alpha_3 = 0.0$, the policy chooses B in only the cases when the cost of help and traveling to state 2 are less than those to state 3 (Table 3.2). The HOP-POMDP policy tries to ask the less expensive human even if they are less likely to be available because there is no cost for failing to ask a human but the cost is high for asking someone who does not want to be asked. If there was a cost of help when the occupant did not answer, the policy would likely change.

Deciding Whether to Ask: We find that the OPOMDP policy queries the occupant in state 2 or 3 after executing the action B at all times except when the cost of help h_2 is 8.0 and the cost of traveling to state 2 is $\leq$ the cost of traveling to state 3 (Table 3.3). In other words, the OPOMDP policy does not ask when it has taken action B and the cost of helping for occupant h_2 is very high.

The optimal HOP-POMDP policy indicates that the robot should ask for help when one or both of the occupants has availability $\alpha > 0.1$. This, again, is because there is no penalty for asking if a human does not respond. It is worth trying to ask for an observation when there is any chance someone is available.

States 2 and 3 - OPOMDP Policy for Asking $\forall \alpha_2, \alpha_3$

cost of	cost of traveling to state 2						
help for h_2	0.125	0.25	0.5	1.0	2.0	4.0	8.0
0.125	Y	Y	Y	Y	Y	Y	Y
0.25	Y	Y	Y	Y	Y	Y	Y
0.5	Y	Y	Y	Y	Y	Y	Y
1.0	Y	Y	Y	Y	Y	Y	Y
2.0	Y	Y	Y	Y	Y	Y	Y
4.0	Y	Y	Y	Y	Y	Y	Y
8.0	**N**	**N**	**N**	**N**	Y	Y	Y

Table 3.3: Based on the cost of help and the cost of traveling to the human, the policy determines that it should not ask (N) when the cost of helping is 8 and the cost of traveling is ≤ 1.

Average Reward: Finally, we compare the average reward received using the OPOMDP policy with the reward from the optimal HOP-POMDP policy. Over all costs of traveling and help, and availabilities, the average reward for an optimal policy was 6.43 and the reward for a OPOMDP policy was 6.01. We find that the rewards are the same when the availability of the occupants are the same ($\alpha_2 = \alpha_3$) and when the cost of help for h_2 is greater than the cost of help h_3. The optimal HOP-POMDP policy reward is almost double the OPOMDP policy reward in the worst case.

For example, when h_2 has availability $\alpha_2 = 1.0$ and h_3 has availability $\alpha_3 = 0.0$, the optimal HOP-POMDP policy reward is on average 8.55 (min 6.92, max 9.78) (Figure 3.2 light gray). The OPOMDP policy reward (average 5.73) is the same as the HOP-POMDP reward when the cost of help for h_2 is 2 or higher. However, the reward drops significantly to an average of 3.54 for lower costs of helping for h_2 (Figure 3.2 dark gray).

Next, we model our own building to demonstrate the HOP-POMDP policies in a practical, larger-scale state space.

3.4.2 Real-World Building

We model the indoor robot navigation problem as a HOP-POMDP in which the human observation providers are the occupants of the offices around the building. Their availability differs depending on their schedules and their cost of asking depends on their willingness to help the robot. We gathered this data through a study of the 78 offices over 9 test times collected over three days (described further in Chapter 5). The availability of our office

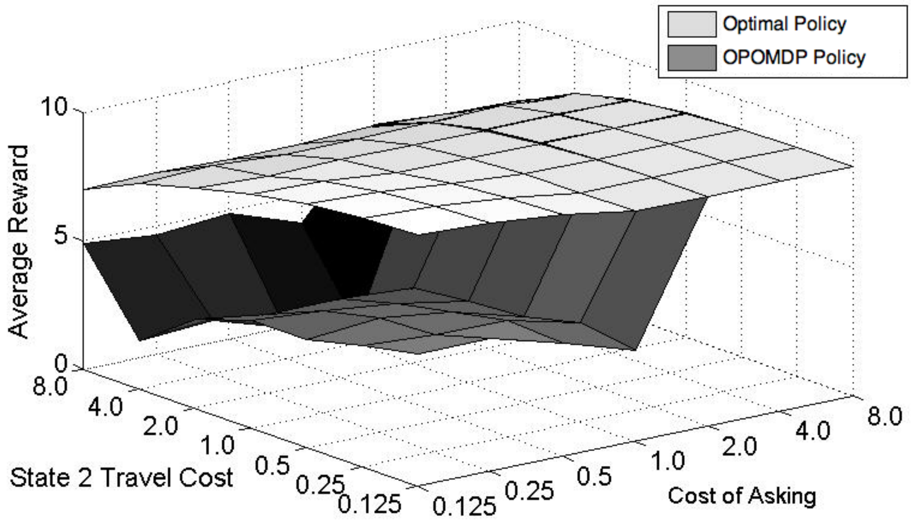


Figure 3.2: The optimal HOP-POMDP policy reward for $\alpha_2 = 1.0, \alpha_3 = 0.0$ is 8.54 (light gray). The OPOMDP policy reward is equal for high cost of help for h_2 but drops significantly to an average of 3.54 (dark gray).

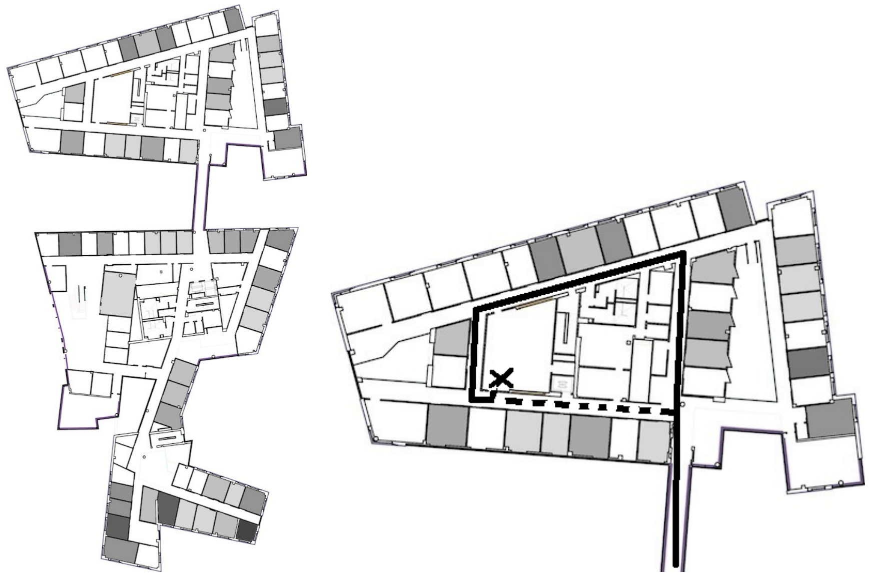
occupants is shown in Figure 3.3(a) where darker gray represents higher availability.

We tested the top portion of the building from the hallway to the lab marked with an X, with a graph consisting of 60 nodes including 37 offices (Figure 3.3(b)). Taking an action between a current node s and a connected node s' on the graph had the following transition probabilities:

$$T(s,a,s) = 0.1 \qquad T(s,a,s') = 0.9$$

We assigned a constant cost $\lambda = -1$ as the cost of asking each occupant and a reward $R(final, a) = 100.0$ for reaching its laboratory space. We were able to find an optimal solution using the Witness algorithm for this environment (unsurprising for the size of the environment).

The OPOMDP policy takes the shortest path (dashed line) to the lab while the optimal HOP-POMDP policy takes a longer route (solid line) that has more available building occupants (Figure 3.3(b)). Because the costs of help were all the same for all occupants, the difference in paths indicates that the likelihood of finding an occupant to ask is higher on the longer path and results in a higher expected reward than the shorter path. Interestingly, this same policy can be used for many offices around the lab and remains constant



(a) Availability (b) Policies

Figure 3.3: (a) We measured the availability of occupants in each of 78 offices in our building - darker gray represents higher availability. (b) The OPOMDP policy takes the shortest path to the lab (dashed line) while the optimal HOP-POMDP policy takes a longer route with more available people (solid).

throughout the deployment of the robot. While the optimal policy takes much longer to solve, the precomputation time may be worth it to increase the expected reward and reduce the cost of help from the occupants along the suboptimal path or the cost of replanning if a human is not available.

To summarize, we found that the optimal HOP-POMDP policy is better in nearly 40% of our tests - surprisingly small given that the OPOMDP does not take into account humans. The optimal HOP-POMDP policy attempts to minimize the cost of asking while maximizing expected reward, while the OPOMDP policy only maximizes reward. As a result, the optimal HOP-POMDP policy chooses to travel to the less available but less costly human to reduce costs. Finally, we found that the optimal HOP-POMDP policy does in fact differ from the OPOMDP in practical environments with more dispersed and less available humans, and therefore it is reasonable to compute the HOP-POMDP policy to reduce the expected cost of asking sub-optimal humans.

3.5 Feasibility Discussion

While we were able to implement the HOP-POMDP for one part of our building, it took over one day to solve the optimal policy for a single destination in the building. While we could have used a faster or more approximate algorithm, optimal POMDP solutions are in general intractable to compute for large state and action spaces such as building environments. In this section, we discuss the feasibility of implementing HOP-POMDPs for larger environments.

Environment Occupant Models: The HOP-POMDP model requires availability, accuracy, and cost of help estimates for each environment in order to solve for the optimal policy. This means collecting information from occupants prior to deploying our robot. If occupants move or even as their calendar appointments change, the availability at their location will change and may change differently at different times of day. We performed an extensive experiment to measure availability in our building prior to deploying the robot (see Chapter 5). However, this is difficult in practice - even after gaining access to occupancy sensors in the building. In Chapter 7, we contribute an algorithm to learn these values online while the robot is deployed.

Number of Policies: POMDPs are typically solved to accomplish a single goal. However, our robot will have to navigate to many different offices in the environment. The HOP-POMDP would need a policy for many start-end location pairs throughout the building. For example, the policy to go from location A to B cannot be reused to go from B to A. Additionally, a new HOP-POMDP policy must be created for each possible availability, accuracy, and cost at each location in the building depending on the time of day.

Even though there are relatively few actions to take between states, even precomputation of all of these policies (by destination as well as availability) would take a long time, possibly weeks or more. By the time that the models were computed, the environment might have changed. As a result, we suggest using approximate POMDP solvers that do include asking actions but take much less time to compute rather than the exact solver or heuristic solver that we used in these experiments.

Because of these limitations, we were unable to implement the HOP-POMDP policy on CoBot for deployment.

3.6 Chapter Summary

We introduced Human Observation Provider POMDPs (HOP-POMDP) to take into account the availability, accuracy, and costs of asking for help from environment occupants in addition to the robot's own reasoning uncertainty. Solved HOP-POMDP policies trade off the uncertainty when navigating autonomously with the cost and uncertainty of asking environment occupants for help to overcome the reasoning uncertainty.

In order to solve the HOP-POMDP policies, we rely on POMDP solvers. However, optimal HOP-POMDP policies can differ from OPOMDP policies adapted for HOP-POMDPs and we realize that the execution of HOP-POMDP policies is not standard because occupants do not provide probabilistic observations due to noise like other sensors do.

We have shown that the approximate policy that does not take into account humans when determining a navigational path is suboptimal nearly 40% of the time when the shortest distance to the goal is not the one with the best occupant to ask for observations. We conclude that, because the optimal HOP-POMDP policy only has to be computed once to be used throughout a robot deployment, it should be used to ensure higher reward and better expected usability for environment occupants.

Chapter 4

Replanning with Environment Occupants

We have presented conditional and probabilistic deliberative plans for determining whether devices should ask for help to overcome reasoning uncertainty and actuation limitations. All of our presented requests for help thus far assume that the help takes place where the robot is currently located. For example, if the robot has localization uncertainty, a person helps the robot by indicating its current (x, y) position. Most other devices that ask questions while performing tasks also have planning algorithms that assume that help is given at the current location (*e.g.,* for learning (Asoh et al. (1997); Chernova and Veloso (2008a); Grollman and Jenkins (2007); Katagami and Yamada (2001)) as well as without learning (Fong, Thorpe, and Baur (2003); Weiss et al. (2010))).

However, many of CoBot's actions that it requires help with are spatially-situated - they must be performed in a particular location or set of locations in the environment (*e.g.,* at the elevator or in the kitchen). People may visit these help locations at different frequencies, but the potential cost of them helping the robot is low because they are already performing the task the robot requires help with. Because the help is required at the particular location, the robot may have to wait a long period of time for help to arrive which delays its ability to complete its task.

In order to reduce wait time, this thesis argues that because the robot is mobile, it could travel to offices in our building to find immediate help at the higher cost of interrupting environment occupants and having them travel to the help location. The problem of identifying an optimal help policy (*i.e.,* waiting at the help location versus proactively navigating to find an environment occupant in an office) hinges on evaluating this tradeoff between interruption costs to the environment occupants and task completion time. In this chapter, we contribute our human-centered replanning algorithm that takes into account where the robot will need help and who is available in the environment to determine who to ask to help and where.

We first present results from the deployment of our conditional planning algorithm to request help using the elevator. We show that while CoBot could receive help immediately at times, it also could wait up to 10 minutes. We then illustrate the process of seeking spatially-situated help from environment occupants and highlight the challenges and tradeoffs of planning to request such help. We next contribute a decision-theoretic human-centered replanning algorithm that takes into account both the robot's and occupants' costs and likelihood to help to determine who to ask for help (details of the evaluation of the occupant model in Chapter 5). Finally, we demonstrate that our algorithm limits the number of occupants that are interrupted in their offices while also limiting the expected waiting time to complete tasks to just a few minutes.

4.1 Conditional Plan Deployment

Using the conditional task plans, we deployed CoBot to perform tasks for people on the upper four floors of an office building for a two week period. CoBot was deployed for two hours every weekday and made available to the building occupants. Occupants were alerted of CoBot's availability through email and physical signs posted on bulletin boards and on the robot itself. The deployment times varied each day, and were announced beforehand on CoBot's website.

Over one hundred building occupants registered to use CoBot on the website, requesting 140 tasks in the first two weeks. Users found creative ways to exploit the robot's capabilities, including, but not limited to:

- Sending messages to friends.
- Escorting visitors between offices.
- Delivering printouts and inter-office mail.

Particularly in the first couple days of deployment, we found building occupants followed the robot around to see where it was going and how it worked. We discuss later how this behavior affected our modeling results.

In fulfillment of the user requested tasks, CoBot travelled a total of 8.7 km, which covered most of the building. In particular, we found that occupants often scheduled the

Task Type	**Total Requests**	**# Multi-Floor Requests**
Escort	3	2
GoToRoom	52	22
DeliverMessage	56	20
Transport	29	22

Table 4.1: CoBot performed 66 multi-floor tasks out of the total 140 indicating the importance of the conditional plan to ask for help using the elevator.

robot to perform tasks on multiple floors of our building, saving the task solicitors time because they did not have to travel between floors themselves. They scheduled the robot to transport objects between multiple floors of the building more often than they used the multi-floor functionality for other tasks (Table 4.1). However, even the other scheduled tasks utilized the elevator 40% of the time.

This finding about multi-floor use is significant as CoBot could not have otherwise performed these tasks without our conditional plan to ask for help using the elevator. We conclude that our elevator help planning algorithm is valid for successfully completing tasks. Furthermore, it supports our model of symbiotic autonomy that humans are willing to help a robot complete its tasks if they see potential benefit for them in the future. Building occupants (even those that had never scheduled a task) were willing and able to help the robot in and out of the elevator.

As we break down these tasks further, we find that CoBot spent

- 6 hours and 17 minutes navigating,
- 36 minutes with a blocked path waiting for a person to move out of its way,
- 1 hour and 2 minutes waiting for help with the elevator,
- 1 hour and 18 minutes waiting for task solicitor help to complete its tasks.

Figure 4.1 shows how much time CoBot took to execute each transport task, and how that time was apportioned. Based on these times, we find that task solicitors did respond to the robot's request for help at the start and end of tasks as well as using the elevator. However, the robot spent a lot of time waiting for the help. This waiting time is only expected to grow, as novelty of the robot meant that people were willing to help relatively quickly in the first few weeks.

While CoBot cannot algorithmically reduce its time waiting for a device user to confirm task completion (only the user knows if it is completed), it can possibly reduce its

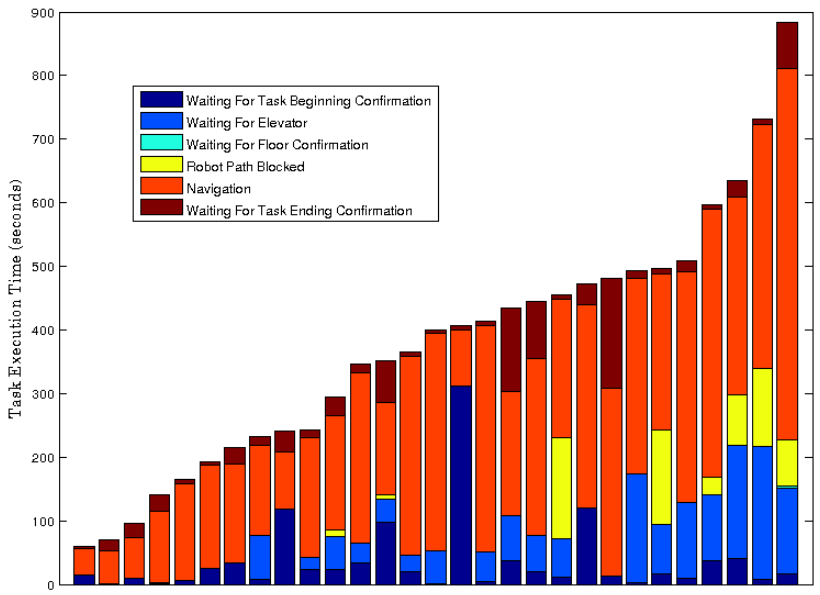
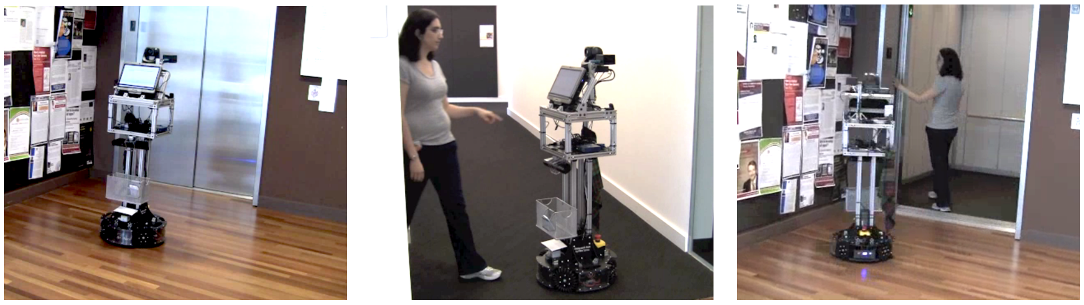


Figure 4.1: Transport tasks execution time breakdown includes 1) waiting for help to start the task, 2) waiting for the elevator 3) riding the elevator, 4) navigating (not including time blocked by obstacles), 5) waiting blocked by an obstacle, and 6) waiting for help to end the task.

elevator wait time by proactively finding help away from the elevator. In the next section, we describe a scenario in which CoBot could proactively navigate to find help instead of waiting for it.

4.2 Example of Spatially-Situated Help

Rather than only waiting at the elevator for help, we argue that robots can proactively navigate to request help to perform spatially-situated actions that require a helper to physically go to a location to perform the action, such as using the elevator and retrieving coffee from the kitchen. In order to illustrate the many decisions and challenges that must be considered, we describe one possible scenario that CoBot could itself in when it must use the elevator and tradeoffs the robot must make in determining where to find help (Figure 4.2,



(a) (b) (c)

Figure 4.2: (a) CoBot autonomously navigates to the elevator that it must use to reach a destination on a different floor. It waits at the elevator for someone to arrive and help. (b) If no one arrives at the elevator, CoBot replans to find someone in an office rather than waiting longer, determines the best office to ask, and proactively navigates there. At the office, CoBot asks a person in the office if they are willing to help. If not, CoBot replans to find someone else. (c) When CoBot and a helper arrive at the elevator, it can ask for the help it needs. CoBot asks them to press the elevator buttons as well as to hold the doors open.

subfigures referenced below).

(a) Waiting at the Help Location. People arrive in help locations at varying frequencies. If CoBot navigates to the elevator, it may or may not find a person who is also trying to use the elevator and who could help it get to the correct floor. The elevators are less frequently used during class time, for example, so the robot could be waiting a long time, delaying its task completion. The benefit of asking the person already at the help location is that they are already performing the action themselves and should have little cost to helping the robot. If CoBot waits at the help location for a long time, it may instead be beneficial to proactively navigate to find a person in an office who can help immediately.

(b) Deciding Where to Proactively Travel. If CoBot travels to find a person in an office who could help, they must travel together back to the elevator. Determining who to ask for help is important in maintaining robot usability over long term deployments. In particular, we identify several possible factors of the decision of who to ask. First, the distance between CoBot's current location and the location of the office helpers may be a factor. Once CoBot arrives at an office, other factors may include the potential helper's availability, meeting schedule, or unwillingness to respond due to too many questions from the robot. If the helpers are willing to help, another factor is the travel distance to the help location with CoBot. When the robot and helper arrive at the help location, there may be a new possible helper at the help location which may factor in to the decision about whether to help again later. Finally, if the helper does not know how to help (*e.g.,* use the coffee maker), it may impact their willingness to help and success of actually helping.

(c) Requesting Spatially-Situated Help. Once a person is at the help location, CoBot also must plan its task questions. To use the elevator, CoBot asks the helper to press the up/down button and notify it when he/she has done so. It also tells the person to hold the doors open so that it does not get stuck. Then, CoBot waits for the elevator doors to open and navigates autonomously inside. After stopping inside the elevator, it asks the person to press the appropriate floor number and to hold the doors open when they get to that door. to again assist in keeping the doors open when the elevator gets to the correct floor. Upon reaching the correct floor, CoBot navigates out of the elevator and then continues navigating to its destination.

We surveyed environment occupants to understand what factors are important in determining whether environment occupants are wiling to help and what the cost of help is (details in Chapter 5). We use the results to derive a decision-theoretic human-centered replanning algorithm that plans to ask for spatially-situated help both near the elevator and away from it.

4.3 Spatially-Situated Help Algorithm

We present our spatially-situated action help algorithm for the robot to execute when it requires actuation help (Rosenthal and Veloso (2012)). Based on our survey findings presented in Chapter 5, our algorithm always navigates to the help location first to wait for someone there to request help from. Then, only after waiting without someone agreeing to help, the algorithm employs our proactive travel tradeoff to determine who to seek help from in offices also based on our survey findings. We demonstrate that our robots can receive help faster by proactively navigating to find help compared an algorithm that only wait at the help location. Additionally, we show that our algorithm is more usable than an algorithm that always proactively navigates because it asks at the help location first which our survey shows is preferable to our participants.

4.3.1 Spatially-Situated Action Help (SSAH)

We contribute our Spatially-Situated Action Help Algorithm to plan (and replan) to proactively seek help to overcome its actuation limitations while completing tasks in the environment. When CoBot reaches an action in its plan that it cannot complete autonomously, it calls SSAH(help,l_{help}) with the type of help it needs and the location l_{help} where it should be receiving help (Algorithm 2).

The algorithm first initializes the taskSuccess indicator variable, list of offices, sets a waitThreshold to indicate how long to wait for a person to answer in offices, always chooses to travel to the help location l_{help} first (line 1). In order to ensure that the robot does not wait too long at the help location, we set the taskThreshold for the max time that the robot should wait before proactively navigating to offices (line 2) (See Setting Wait Threshold for details).

After setting these values, the robot navigates to find a helper at the help location (line 3). Then, it asks for help (line 5) and waits for a response depending on where it is (lines 6-10). If someone is willing to help, it tries to execute with them and updates its information about the helper at location lot (line 11-14). If it was not successful or could not find a person, it picks a new lowest cost location using our Proactive Travel Tradeoff (PTT) (line 15-18), subtracts the office from its list so that it doesn't revisit it (line 16). The PTT tradeoff uses the updated location information to accurately determine the cost of asking each office for help. The algorithm returns when it either has been helped successfully or there are no more offices to visit (line 20).

4.3.2 Proactive Travel Tradeoff (PTT)

In order to determine who to ask for help, our Proactive Travel Tradeoff (PTT) computes the expected cost to complete the help with proactive navigation to each possible office and chooses the minimum cost office. The costs are computed based on our survey findings. This tradeoff is decision-theoretic Lehmann (1950); Schoemaker (1982) in that it computes the best action with the lowest expected cost by taking into account helpers':

- availability α or probability the person is in their office,
- interruptibility ι or the probability the person is not busy
- expertise e or the probability of successful help,
- location l of offices,
- recency of help r or the time since person's last help,
- frequency of help f per week, and

Algorithm 2 SSAH(help,l_{help})

```
1: taskSuccess ← false, waitThreshold ← waitTime, offices ← getAllOffices(), travelLoc ← $l_{help}$
2: (taskThreshold,loc) ← PTT($l_{help}$,$l_{help}$,offices)
3: while ¬taskSuccess AND |offices| $> 0$ do
4:   Navigate(loc)
5:   Ask(helptype)
6:   if loc = $l_{help}$ then
7:     willing ← WaitForResponse(taskThreshold)
8:   else
9:     willing ← WaitForResponse(waitThreshold)
10:  end if
11:  if willing then
12:    taskSuccess ← ExecuteWithHuman(help)
13:    updateHelper(loc)
14:  end if
15:  if ¬taskSuccess then
16:    (time,travelLoc) ← PTT(travelLoc,$l_{help}$,offices)
17:    offices ← offices - travelLoc
18:  end if
19: end while
20: return taskSuccess
```

- willingness to answer $w(\iota, r, f)$ based on past experiences and current interruptibility.

Using this model of helpers, the PTT tradeoff computes the cost of asking at each office o including:

- COT cost traveling to the office o,

- the probability of a person being available α_o and the cost of asking them to help COA,

- the willingness of them to help $w(\iota_o, r_o, f_o)$ and the cost of the helper traveling back to the help location $COTH$,

- the probability of being unnecessary α_{help} and the corresponding cost COU,

- the probability of failing due to expertise e_o and the cost of failure COF,
- the cost of replanning from the current location with PTT in case a person is not available or willing to help.

Formally, $PTT(l_{start}, l_{help}, \mathit{offices}) = \min_{o \in \mathit{offices}}$

$$\begin{aligned}
&COT(l_{start}, l_o) \\
&\quad + \alpha_o \Big[COA(\iota_o, r_o, f_o) \\
&\qquad + w(\iota_o, r_o, f_o) \big[COTH(l_o, l_{help}) \\
&\qquad\quad + \alpha_{help} * COU(l_{help}) \\
&\qquad\quad + (1 - e_o)[COF(l_o, l_{help}) \\
&\qquad\qquad + PTT(l_{help}, l_{help}, \mathit{offices} - o)]\big] \\
&\qquad + (1 - w(\iota_o, r_o, f_o)) * PTT(l_o, l_{help}, \mathit{offices} - o) \Big] \\
&\quad + (1 - \alpha_o) * PTT(l_o, l_{help}, \mathit{offices} - o)
\end{aligned}$$

While this tradeoff finds the optimal office, it is intractable to compute for any large number of offices given the recursion for failure and the branching factor equal to the number of offices in the building. For our implementation, we compute the greedy best office that does not recurse on PTT and instead uses a constant FAIL_COST that is greater than the cost of successfully asking any office (approximately 1000 seconds).

4.3.3 Setting Task Thresholds

The SSAH algorithm also required a task threshold to wait at the help location before starting to proactively ask for help. We explored two methods for setting the threshold. First, we looked to queueing theory to model the arrival of a person at the help location using a Poisson Process Arlitt and Williamson (1997). If the robot were to model the likelihood of finding help using this distribution, however, we found that the probability that a person will arrive increases over time so the robot never proactively navigates away from the help location. Second, we used the Buy or Rent problem (Ski Rental problem) as an analogy for setting this threshold, in which there is a small cost to continuing to wait and a large cost to finding a person elsewhere Karlin et al. (1994). The solution to this problem is not optimal in hindsight but ensures that the robot will take no longer than twice the expected time to proactively find a person to complete the task. We choose this solution because not only does threshold task completion time but also has the property that it will wait at the help location.

4.4 Experiments

Our SSAH algorithm combines waiting at the location of help with proactive navigation in order to speed task completion while limiting the number of questions asked at offices. In order to characterize the performance of our greedy PTT algorithm, we performed real-world and simulated experiments. These results are meant to be examples of how the algorithm behaves and not a complete analysis of behavior over all possible environments.

4.4.1 Simulated Experiments

We simulated four hallways of our building, the elevator nearby, and the occupants in their offices. The hallways contain 28 offices in an elongated rectangle shape with the elevator in the middle of one of the long hallways. Our real building contains occupancy sensors in each office, so we define $\alpha_o \in \{0, 1\}$ randomly. Additionally, we generate random probabilities for willingness to respond $w \in [0, 1]$. Finally, we use real world data to compute the frequency that a person appears at the elevator (α_{help}). Through an observational study, we found that on one floor $\alpha_{help} = 5$min and on another floor $\alpha_{help} = 10$min.

We test our SSAH algorithm against two other algorithms. In the Wait Only algorithm, the robot travels to the help location and waits indefinitely until someone helps there. This algorithm is guaranteed to find a low cost helper at the help location but may result in long task completion times. In the Proactive Only algorithm, the robot uses the PTT tradeoff to immediately find a helper in an office. This algorithm is guaranteed to find help quickly but at the cost of always interrupting an office worker.

Time to Find Help

Our SSAH algorithm finds help faster than Wait Only but slower than Proactive Only (Figure 4.4). The Wait Only algorithm that only waits at the elevator has high variance and takes on average 5 minutes and 10 minutes respectively on the two different floors of the building. The Proactive Only algorithm which goes directly to find help in an office rather than waiting at the elevator almost always finds help in under three minutes. The SSAH algorithm is in between the two algorithms as it waits first and then navigates away.

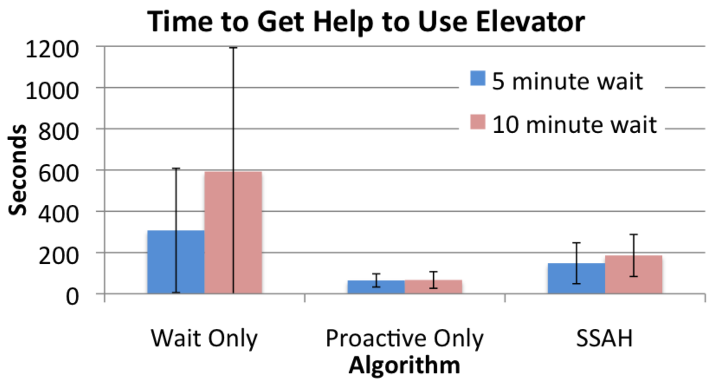


Figure 4.3: Our SSAH algorithm finds help use the elevator faster than the Wait Only, but not as fast as Proactive Only which always goes directly to find someone in an office.

Number of Offices Asked

While the Proactive Only algorithm found a helper faster, it also interrupted people in offices every time it needed help. This is costly according to our survey results. Instead, our SSAH algorithm was able to cut the number of offices visited in half for the 5 minute floor and by 20% for the 10 minute floor compared to the Proactive Only algorithm. While we would like the robot to complete tasks quickly, we also want to make sure that people are willing to help the robot months and even years. The less frequently the robot actually interrupts people in offices and can instead ask people who are already using the elevator, the more usable and deployable we expect the robot and algorithm to be for our building in the long run.

We conclude that while SSAH takes longer to find help than Proactive Only, the reduction in office help requests is significant for the future usability of our robot.

4.4.2 Real Robot Deployment

In order to understand how our algorithm performs in practice, we conducted an experiment in which the robot performed multi-floor tasks which required help using the elevator. In the first phase of the experiment, we deployed CoBot with the Wait Only algorithm. With this algorithm, the robot was able to accomplish 66 multi-floor tasks which it could not have completed without help, waiting on average 1 minute to find help. Interestingly, this result conflicts with our previous finding that people use the elevator once every 5-10 minutes depending on the floor. We found that people were often following the robot or

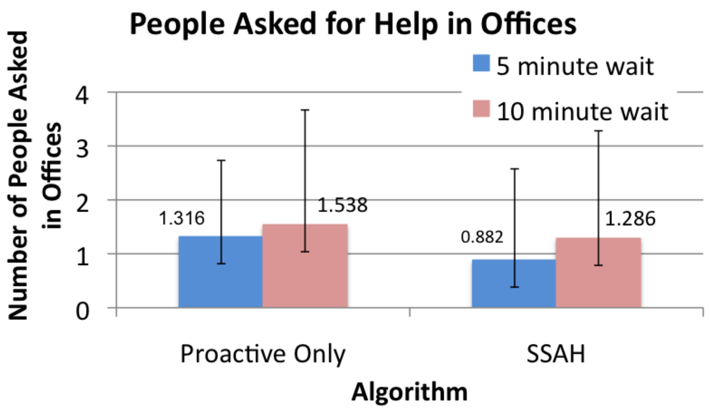


Figure 4.4: Our SSAH algorithm requires less help from offices, making it more usable for our occupants long-term.

interested in its deployment and therefore helped more often and faster than what one can expect on a more normal day.

In the second phase, we conducted a preliminary experiment of CoBot using our SSAH algorithm to find help. After waiting for help at the elevator, CoBot contacted a server to access the occupancy sensors in each office of the building and then estimated the interruptibility of those who were available. It then chose the best office using our PTT algorithm, navigated there, and asked for help. If the person refused to help or ignored the robot's question, CoBot would replan using the PTT tradeoff to find another office.

Because the novelty had worn off by the time we deployed this phase, we found the robot waiting times were much more of what we would have expected. CoBot waited an average of 190 seconds for a person to arrive at the elevator before proactively navigating. This result is much shorter than the 5-10 minutes for Wait Only and results in more satisfaction for the people who request the tasks as well as those located around the elevator.

From these results, we conclude that our SSAH algorithm is expected to find help faster than Wait Only without asking as many people in offices as the Proactive Only algorithm.

4.5 Chapter Summary

Our CoBot robots are capable of autonomous localization and navigation, but have actuation limitations that prevent them from performing some actions such as pushing buttons to use the elevator and making coffee in the kitchen. Interestingly, these limitations require humans to be spatially-situated in the help location in order to help the robots perform these actions. We take advantage of the fact that our robots are mobile and have them proactively seek humans in offices to travel to the help location. We contribute a human-centered decision-theoretic replanning algorithm to take into account potential office helpers' preferences about where to navigate and who to ask in the environment based on helper interruptibility, and how recently and frequently the robots might ask them for help. We demonstrated in simulation and in a real-world implementation to that our algorithm balances the time waiting at the elevator with the expected interruption of proactively finding helpers.

Chapter 5

Modeling Human State

Our human-centered planning algorithms use models of humans in the environment to determine device actions and whether and who to ask for help. In particular, unlike prior work that assumes humans are always available and willing to help (*e.g.,* device users Lee et al. (2010), supervisors (Fong, Thorpe, and Baur (2003); Shiomi et al. (2008)), teachers (Argall et al. (2009); Hayes and Demiris (1994); Lockerd and Breazeal (2004); Price (2003)), and passers-by in the environment (Asoh et al. (1997); Hüttenrauch and Eklundh (2006); Michalowski et al. (2007); Weiss et al. (2010))), our algorithms used models of human state attributes:

- availability - whether the human is in the environment,
- interruptibility - whether the human is performing other tasks,
- distance to the help location - how far the human will have to travel to help the device,
- accuracy/expertise - whether the human understand how to help and can do so well,
- cost of help - how much time and effort the help takes, and
- incentive to help - how much benefit the human receives from helping.

A person may not be available or be located near the device in order to provide it help. Similarly she may be around but busy in a meeting or interacting with someone else (not interruptible Fogarty et al. (2005)). Even if she was available and interruptible, she may not understand the question to answer accurately. She may also perceive a high cost of helping and not see the benefits toe incentivize her to help.

Additionally, our algorithms use other factors that may affect willingness to help:

- help type - which question the device is asking,
- frequency of questions - how often the device asks the human for help, and
- recency of the last question - how long ago the last question occured.

A person may not want to help with particular questions, or after they are asked frequently or recently.

In this chapter, we contribute studies of both device users and occupants in the environment to demonstrate our human-centered planning algorithms should include models of each of these attributes. First, contribute two studies of device users to demonstrate that they have their own costs and incentives to help their devices. We survey phone users to measure their personalized costs of help and incentive to help their phone learn their volume preferences, and then describe a short study of the cost and incentive of 5 visitors to help answer CoBot's localization questions as it escorts them to meetings around our building.

Second, we contribute two studies of environment occupants to demonstrate the need for all nine attributes identified above. We survey building occupants to demonstrate that all the factors affect their willingness to travel down the hall to help CoBot use the elevator, and then describe a wizard-of-oz (Green and Wei-Haas (1985)) remote-controlled CoBot experiment in which the robot actually asks occupants to leave their offices to further demonstrate the need to model distance to location, interruption, incentive to help, help type, and frequency of questions.

5.1 Modeling Phone Users: A Survey

Phones have reasoning uncertainty in that they do not know whether to turn the volume on or off given users' state. To train a classifier to reduce this reasoning uncertainty, we previously contributed a human-centered algorithm to ask for users' volume preferences when the incentive to answer questions was higher than users' cost of helping. In this section, we present surveys of users of smart phones who receive several phone calls, SMS messages, and calendar alarms daily to understand phone users interruption costs of help and incentives to help prevent phone volume misclassification across a variety of situations Rosenthal, Dey, and Veloso (2011).

GPS: Longitude, Latitude, Speed	Accelerometer X, Y, Z axes	Time until Next Meeting
User in Meeting	Noise (in dB)	Hour of Day
Day of Week	User on Phone	Count of Times On-Phone Caller has Contacted User
User on Phone with Someone in Contact List	User on Phone with Someone in Favorite List	Next Meeting is a Repeated Meeting
Contactor is in Contact List	Contactor is in Favorites List	Count of Times Contactor Has Contacted User

Table 5.1: Smart phones have a variety of sensors that we can use to describe the user's state and predict their cost and incentive to help the phone.

In the survey, participants were asked to rate their preferences for receiving audible notifications in a variety of hypothetical, but real world, situations and their expected costs and incentives to train their volume classifier. We analyzed the differences in preferences and cost ratings between participants in the same situation as well as differences that a single participant provided across multiple situations to determine if a single model of all users (as found in Khalil and Connelly (2005)) is sufficient or if personalized models of users are also needed. We show that each user has different costs and incentives to help, warranting the need for personalized models of the cost and incentive of help for each user.

5.1.1 Method

Before the survey began, participants were first asked a series of questions about their work schedule and common modes of transportation, which might affect their survey responses about situations in which they want audible notifications. Participants were then given 20 hypothetical situations when their phone might display a notification for each notification type. These situations were drawn from the sensor features in Table 5.1 and described participants environments (*e.g.,* work or movie theater) or activities at the time of the interruption (*e.g.,* driving a car or relaxing at home).

Participants were given a short description of each of the situations and notification reason for the interruption, and were asked 1) if they would want audible notifications in that situation (interruption preference). Then they were asked to rate 2) their expected annoyance if the phone has the wrong volume setting (cost of misclassification/incentive to help) and 3) their expected annoyance if the phone asked which volume it should use (cost of asking for help). The questions were as follows:

1. In this situation, would you want your phone to ring out loud? *Answer*: Yes/No

2. How upset would you be if the phone did the opposite (rang when it should have been silent or vice-versa)? *Answer*: Likert scale 1 (no problem) to 7 (I would be very upset).

3. In this situation, how upset would you be if your phone asked what it should do if it didnt know? *Answer*: Likert scale 1 (no problem) to 7 (I would be very upset).

An example of the questions for a situation where a user is in a meeting at work is found in Table 5.2, Additionally, participants were able to list exceptions to their interruption preferences for each situation.

All combinations of situations, notification reasons and notification types (phone call, SMS message, or calendar alarm) were presented to participants. Because of the number of situations that would be necessary to train a classifier, we split the survey into twelve parts. Each participant was given the option of answering all questions through all 12 surveys, but was not required to complete them all. Before each survey, participants confirmed that they did receive each notification type the survey focused on (*e.g.,* only those who received calendar alarms filled out the calendar surveys).

Participants Participants were recruited through a Carnegie Mellon participant recruiting website to complete the online surveys. We are interested in both within-subject differences across notification types, as well as between-subject differences for each situation. In total 44 participants took all 12 surveys and 50 more participants took subsets of the surveys for an average of 69.25 participants per survey. Sixty-five out of 94 participants reported that they were students. The rest reported jobs such as cashier, machine shop manager, photographer, and administrative assistant. The average age of the participants was 25.27 with standard deviation 6.3.

5.1.2 Results

We received a total of 9219 responses to our surveyed situations questions and analyzed the proportion of participants who wanted audible notifications for each notification type (calls, SMS messages, or calendar alarms), situation, and notification reason to understand interruption preferences.

Notification Type	Notification Context	Question
Phone	Favorite List, Contact List, Frequently Calls	If you were at work in a meeting and someone in your **favorites list** called, would you want your phone to ring aloud?
Phone	Favorite List, Contact List, Occasionally Calls	If you were at work in a meeting and someone in your **contact list** called, would you want your phone to ring aloud?
Phone	Favorite List, Contact List, Few (if any) Calls	If you were at work in a meeting and someone in **not in your contact list** called, would you want your phone to ring aloud?
SMS	Favorite List, Contact List, Frequently Texts	If you were at work in a meeting and someone in your **favorites list** texted you, would you want your phone to ring aloud?
SMS	Favorite List, Contact List, Occasionally Texts	If you were at work in a meeting and someone in your **contact list** texted you, would you want your phone to ring aloud?
SMS	Favorite List, Contact List, Few (if any) Texts	If you were at work in a meeting and someone in **not in your contact list** texted you, would you want your phone to ring aloud?
Calendar	Repeating Meeting	If you were at work in a meeting and a **repeating meeting** was about to start, would you want your phone to beep aloud to remind you?
Calendar	Non-Repeating Meeting	If you were at work in a meeting and a **non-repeating meeting** was about to start, would you want your phone to beep aloud to remind you?

Table 5.2: Eight questions were asked about whether the users phone should ring in a meeting at work. Prior to taking the survey, participants were given definitions of the notification contexts to help them answer the questions.

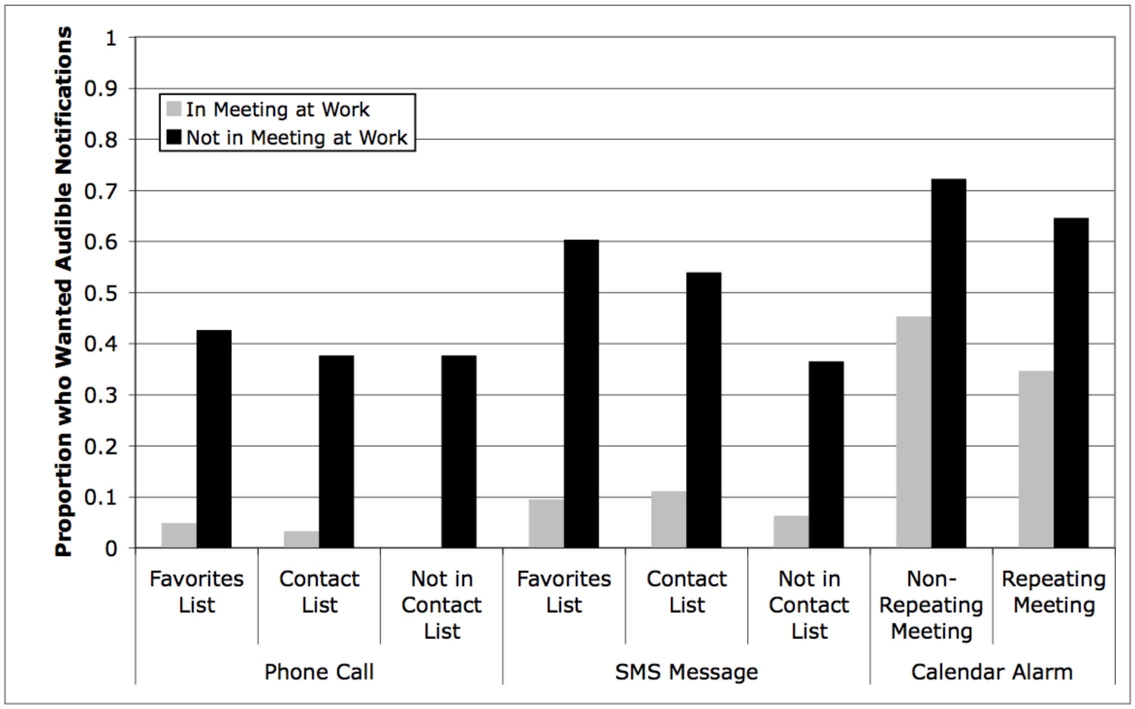


Figure 5.1: Participants varied greatly in their preferences for audible notifications at work when they were not in meetings, but mostly agreed that they should not receive calls or text messages during meetings.

Volume Preferences: We found that participants had very different interruption preferences for each type of notification, which is contrary to current phone settings that only allow a single phone volume for all notification types. For example, at work, 45% of participants wanted calendar notifications during meetings compared to 7% on average who wanted phone calls or text messages in the same situation (Figure 5.1). Only 35% of participants wanted to receive phone calls at work, but more wanted text messages, especially from those on their favorites list. This finding supports our need for an autonomous phone volume changing application and the fact that any out-of-the-box application would have reasoning uncertainty about how to set the phone volume.

Predefining Volume Preferences: Participants noted that, currently, they often kept their phone on vibrate rather than silent or loud volume because of these situational and notification type differences. One participant said that they prefer to err on the side of caution when it comes to phone volume and I can find the time to check the onscreen message if I'm not too busy rather than listening for an audible notification. When they had to decide on a loud or silent volume setting, participants often responded that they would not want their phone to ring unless it was a family emergency or unless Im getting a ride from that person. These exceptions are hard to enumerate and predefine and indicate a need to ask for help and request preferences *in situ*.

Cost of Help and Incentive to Answer: In order to be able to collect these *in situ* responses in a human-centered way, we use their surveyed costs of misclassification and asking to prevent unneeded interruptions and increase the likelihood of a usable application. Participants reported varying costs of misclassification (incentive to answer) responses on the Likert scale from 1-7 (mean 4.3, s.d. 2.1). Participants responded nearly half of the time (4436/9219 responses) that they would have "No Problem" if their phone asked them for their preference (mean 2.6, s.d. 1.95). There was no particular situation where a majority of participants indicated that they would not be willing to answer. In fact, some participants indicated that they would always be willing to answer questions while others indicated there were situations when they never wanted to answer questions. These results show that a single cost model for all situations and/or all participants (from Kapoor and Horvitz (2008)) would likely interrupt many participants who indicated they did not want questions. Additionally, we find that we do need to model both cost and incentive to help.

Based on these findings and analysis, we conclude that both cost of help and incentive to help (cost of misclassification) are important human state attributes to model when determining whether to ask for help with reasoning uncertainty. Additionally, we need to learn a personalized model of these two attributes for each device user.

5.2 Modeling Robot Users: A Study

Our CoBot robot has reasoning uncertainty in its ability to localize while navigating. If CoBot is navigating with a visitor in the building and does not ask for help with localization, it may wind up backtracking along its path and frustrating the visitor. However, if it does ask for help, the visitor may also be frustrated unless they recognize the incentives that the robot has to offer as it navigates with them. Next, we present the results of five visitors using our CoBot to navigate to meetings, and show that most visitors would use the robot again despite the costs of providing help with the incentives that the robot provides. However, because each visitor has different expectations of the robot's abilities and differing incentives to answer the robot's questions, we again show that personalized models of incentives are necessary for human-centered planning algorithms to use to determine whether or not to ask for help.

5.2.1 Method

In order to test CoBot's capabilities in assisting visitors through their meeting schedules, we invited five participants to participate in a four-meeting schedule over a single floor of our building but spread out in four hallways (Rosenthal, Biswas, and Veloso (2010)). The participants were true visitors and had never been in the building before. The robot could assist them in three ways:

- navigating quickly to meetings,
- inform about offices and labs along the way,
- bring drinks to meetings,
- providing additional information about meeting hosts (by displaying the host's website)

CoBot autonomously navigated and localized in the building and responded to verbal commands for drinks and additional information about meeting hosts. If visitors asked for drinks, CoBot navigated to a predetermined location to find a person to place the drink on it.

When participants arrived for the study, they were told about their meeting schedule, and CoBot's abilities to help them navigate and provide information about different rooms and labs as they walk between meetings. They were also told that they should request the CoBot bring drinks and provide information at least once in their schedule, but it was up to them to choose when to make the requests. Because participants could choose the order and time of each, it more accurately reflects a typical day. Additionally, they were told that sometimes the robot got lost and would ask for localization help.

After completing the meeting schedule, participants were given surveys about their experiences and were asked to rate each feature of the robot on a scale from -2 (not useful) to 2 (very useful). Additionally, we asked each participant to rate the number of questions CoBot asked from -2 (too many) to 2 (too few).

Ability	P1	P2	P3	P4	P5
Host Info	2	0	2	1	2
Drink	2	-1	2	1	1
Labs	1	1	1	1	1
Help Requests	-1	-1	-1	-2	0
Use it Again?	Yes	Yes	Yes	No	Yes

Table 5.3: Participants ratings from -2 (not useful) to 2 (very useful) of CoBot's abilities and questions. The results indicate costs as well as incentive to help the robot.

5.2.2 Results

All participants were able to follow the CoBot to their meetings and answer the robot's questions. CoBot successfully retrieved drinks and provided participants with information about three labs they were passing and the meeting hosts as the participants were guided to the meetings. While participants typically would have had to search through the hallways to find the rooms, CoBot led them directly there.

CoBot gave the same information and asked nearly the same number of questions (about 10) to each participant. While participants mostly felt the robot could have asked fewer questions, they had different opinions about how many were too many - reflecting the need for personalized costs of requesting help. Table 7.2 shows the ratings each participant (P1-5) gave for the robot's abilities. We found that each participant rated the usefulness of the incentives differently, showing that each participant had different expectations for each ability and subsequent state.

When we combine the robot's incentives along with the cost of help, we found that four out of five participants said they benefitted from the navigation guidance and other assistance and would use CoBot again, even thought they felt the robot asked them for help too many times. The one participant who would not use it again placed high cost on asking for help and said he would use it again if it asked fewer questions.

Our findings in this case study again necessitate modeling the cost of help and incentive to help in our human-centered planners. The visitors could either state their interests in the robot upfront before using it or the robot can learn the visitor's costs as they travel together through the day.

5.3 Modeling Environment Occupants: A Survey

CoBot has limitations in its ability to manipulate objects and press buttons. However, we would still like it to traverse our multi-floor building to perform tasks. Previously, we contributed human-centered planning algorithms to wait at the elevator for help and then use models of humans in the environment to replan where to navigate and who to ask for help. In this section, we present a survey of humans in the environment who may potentially be asked for help in order to understand what human state attributes should be modeled in our planning algorithms.

We made five hypotheses based on our intuitions about what human state attributes matter in determining where and who to ask for help. The first two hypotheses represent the spatial considerations that CoBot robot should take into account.

Cost of Help: Asking someone for help who is already in the location is preferred over finding someone in an office. A benefit of asking the in-location person is that they are already performing the action themselves and should have little cost to helping the robot.

Distance to Help Location: If someone in an office must be asked because it is unlikely that anyone will be at the help location, there should be a preference for asking someone close to the location to avoid making someone travel too far. Although the robots are mobile and are capable of traveling to find help, an in-office helper would have to travel back to the help location.

The second three hypotheses represent the considerations the robot should make to increase the likelihood that people are willing to comply and help the robot, because the robot need help performing these actions over a long period of time.

Availability: If a robot travels with a person to the help location and there is someone already at the location of help, the traveling person may feel that they were asked unnecessarily.

Interruption: The robot should avoid requesting help from people in offices that are likely to be busy.

Recency of Last Question and Frequency of Questions: The robot should take into account how recently it asked different helpers to avoid asking too often.

We test these hypotheses using the results from our survey.

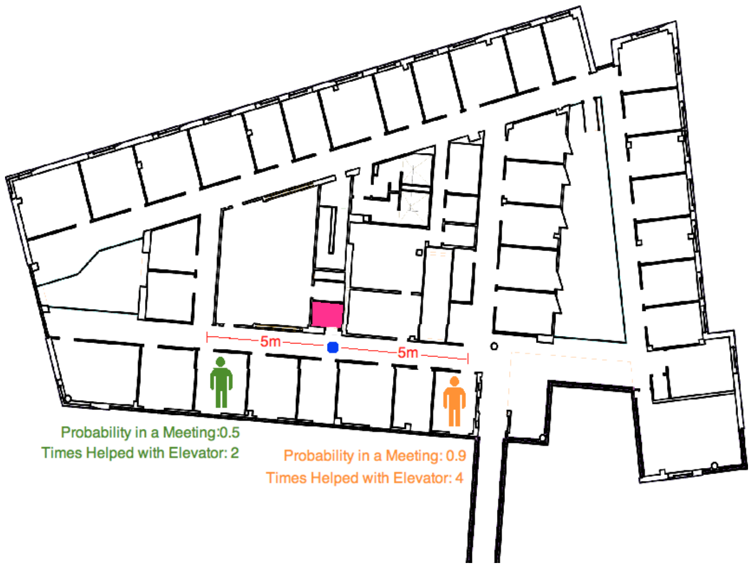

5m
5m
Probability in a Meeting:0.5
Times Helped with Elevator: 2
Probability in a Meeting: 0.9
Times Helped with Elevator: 4

Figure 5.2: Participants were shown an image with two humans (orange and green) in different offices or locations and different state attributes and asked which they thought the robot (blue dot) should navigate to to ask for help using the elevator (pink square) or coffee maker.

5.3.1 Method

In order to test our hypotheses, we conducted a web survey about participants' preferences for when, under what conditions, and how frequently they would be willing to help the robot (Rosenthal and Veloso (2012)). In the first half of the survey, subjects were shown a partial map of our building with different configurations of people in offices who could be available, different locations of the robot, and different locations for receiving help - the elevator or the kitchen to make coffee. They were asked which person the robot should choose to ask for help. In the second half of the survey, participants were told to suppose that they were the one being asked for help and answered questions about their willingness to help the robot under different conditions of interruptibility, recency of the last time the robot could have asked for help, and frequency of the number of questions the robot could ask per week. Figure 5.2 shows an example image of what participants saw as they compared and chose which human the robot should travel to.

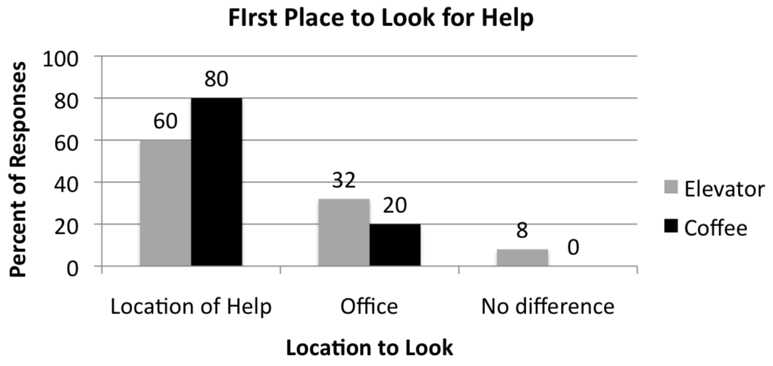


Figure 5.3: While a majority of participants specified that they thought the robot should look in the location of help first, some thought that it should ask in an office to avoid wasting time waiting for someone to arrive.

5.3.2 Results

Thirty participants were recruited through a Carnegie Mellon University website that hosts advertisements for human-subject studies. The survey contained 50 questions and took about 45 minutes for participants to complete.

Cost of Help: We found that in those office configurations where there was one person in an office near the help location and there was a chance of another person at the help location, participants indicated that the robot should check if there was a person at the help location 60% of the time when asking for help to use the elevator and 80% of the time for making coffee (Figure 5.3). As expected, participants noted that they chose the help location because the helper would already be performing the actions and it would not be much more costly to help the robot.

However, we were surprised that a large number of people chose to ask a person in an office first. Most said that if the robot were closer to an available person, it would be better not to waste time navigating past them to the help location to check first since “it is not guaranteed that someone will be standing in front of the elevator.” While it is true that the robot may “waste” time waiting, 80% of participants said that they would be more willing to help the robot if they knew it had tried first to wait by the help location, confirming our hypothesis that there is a different cost of help if the person is already at the help location.

Availability: Because we were specifically concerned about the possibility that office helpers would feel unnecessary if they found another person already at the help location, we told participants to suppose this situation happened. We then asked when they would be willing to help the robot again. If participants felt unnecessary, we would expect a

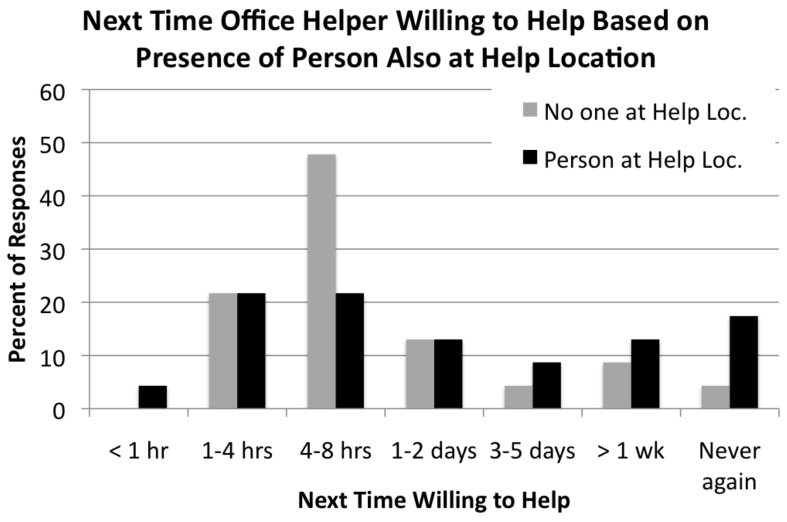


Figure 5.4: Participants would be less willing to help the robot and would want to wait longer before they helped next if they traveled to the help location when another person could have helped.

larger number to be less willing to help in the near future and potentially never want to help again.

Our results show that participants did want the robot to wait longer after helping in this scenario compared to when there was no one else present at the help location (Figure 5.4). Despite feeling unnecessary, 83.4% participants said they would be willing to use the robot again. Additionally, 80% of participants responded that if the robot explained that it had already checked the help location and no one was there, they would be more likely to accept its request for help. As a result, we conclude that the availability of helpers in the help location as well as in offices impacts the willingness of people to help the robot.

Distance to Help Location: Next, we tested whether the robot should ask the closer person to the elevator (everything else being equal) and how far people were willing to travel to help the robot. When shown different configurations of the robot and available people in offices, surprisingly only 75% participants responded that the robot should choose the closer person, irrespective of whether the person would be helping with the elevator or coffee. Participants said that the robot could ask someone further away if it would pass the further person first.

When asked the furthest distance they were willing to travel, 43.5% of participants said they were willing to travel up to 15 meters to help the robot. We were surprised to

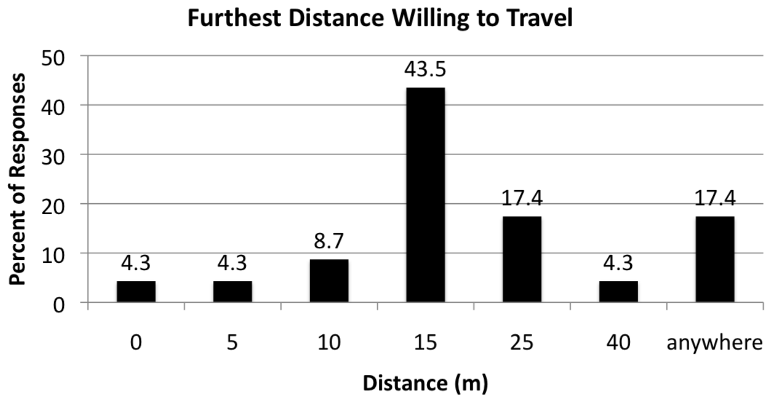


Figure 5.5: Most participants were willing to travel up to 9 meters to help the robot, but 17.4% responded that they were willing to help from anywhere in the building.

learn that 39.1% were willing to travel more than 15 meters. Half of the participants who were willing to travel more than 15 meters (17.4%) said they were willing to travel from anywhere in the building. This indicates that the distance to the help location may not be a significant factor for all helpers in determining which person to navigate to to ask for help.

Interruption, Frequency of Questions, and Recency of Last Question:

Finally, we hypothesized that the frequency and recency of questions would significantly affect a person's likelihood to want to help the robot in addition to interruptibility. We asked participants to predict whether they would be likely to help the robot depending on whether they were in a meeting, how many times they had been asked in the last week, and the last time they had been asked. While 69.5% of participants were willing to help the robot within 8 hours (within the same day) if they felt needed, 47.7% of participants responded the same way when there was someone else present. In comparison, when participants were told to assume that the robot had interrupted them in a meeting to ask for help, 69.5% were still willing to help again in the same day and 56.5% were willing to help again within 4 hours.

In order to test whether recency and frequency of help had a statistically significant effect on the willingness to help, we performed a Nominal Logistic Regression (measured with the χ^2 statistic) testing for differences in the binary response variable *willingness to respond* based on independent variables *participantID* (nominal), *in a meeting* (nominal), *frequency of questions per week* (continuous), *days since last helped* (continuous), and all pairs and the triple of *in a meeting, frequency of questions, and days since last helped.*

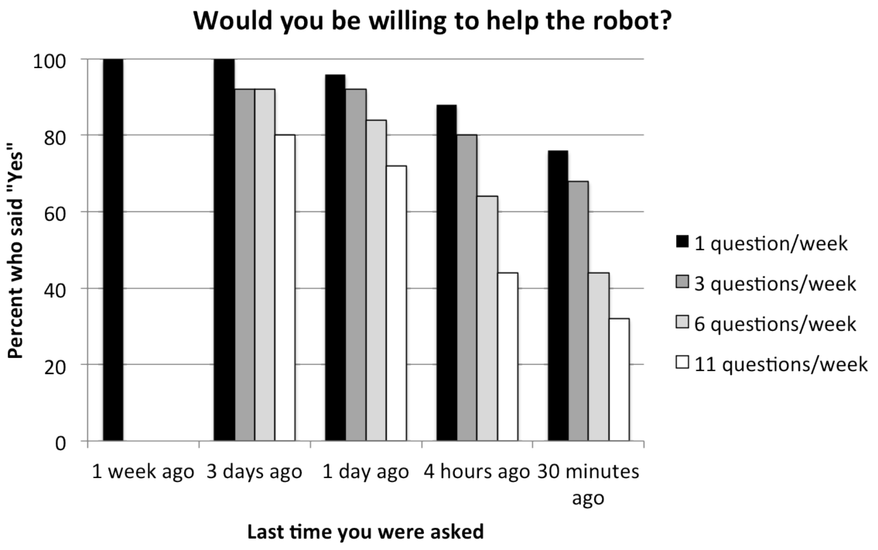


Figure 5.6: There is a significant effect of both interaction history variables on participants' willingness to help the robot.

We found a statistically significant main effect of all four independent variables: *participantID* ($\chi^2[29, 29] = 337.71, p < 0.0001$), *in a meeting* ($\chi^2[1, 1] = 462.28, p < 0.0001$), *frequency of help per week* ($\chi^2[1, 1] = 87.40, p < 0.0001$), and *days since last helped* ($\chi^2[1, 1] = 143.62, p < 0.0001$). Figure 5.6 shows the combined effect of both interaction history variables (frequency of help and days since last help). As expected, there is a negative effect of being in a meeting and being asked more frequently in a week, while there is a positive effect of the helper being asked for help more days ago. We also found that the pairs *in a meeting* $\times$ *frequency of help* ($\chi^2[1, 1] = 12.89, p < 0.0003$) and *in a meeting* $\times$ *days since last helped* ($\chi^2[1, 1] = 20.82, p < 0.0001$) were statistically significant. The pair *frequency of help* $\times$ *days since last helped* was not statistically significant ($\chi^2[1, 1] = 2.32, p = 0.1279$). Finally, we found that the triple *in a meeting* $\times$ *frequency of help* $\times$ *days since last helped* had a statistically significant effect on *willingness to help* ($\chi^2[1, 1] = 19.17, p < 0.0001$). These results confirm our hypothesis that interruptibility, recency and frequency of questions do play a significant role in helpers' willingness to help the robot.

To summarize, we confirmed all five hypotheses. Robots (and other devices) should consider the cost of help, distance to help location, availability, interruptibility, and frequency and recency of questions. However, some participants were willing to help irrespective of the distance to the help location. We use these human state attributes in our human-centered replanning algorithm to determine who to ask for help and where to navigate.

5.4 Modeling Environment Occupants: A Study

CoBot has both localization and actuation limitations that it will need to overcome simultaneously while deployed. In order to understand the feasibility of asking environment occupants for different types of help during tasks, we designed a study in which CoBot visited every occupant in offices on one floor of our building to ask each type of question at different times of day (Rosenthal, Veloso, and Dey (2012a)). Additionally, we test how the distance to help location and incentives to help impact the willingness to help.

We measure the number of times each occupant is available in their office and willing to help for each question type as well as the amount of time they spend helping the robot. Because we were exploring the feasibility of asking different types of questions in this study and not testing the autonomy of the robot, CoBot was wizard-of-oz'd (Green and Wei-Haas (1985)) or remote-controlled.

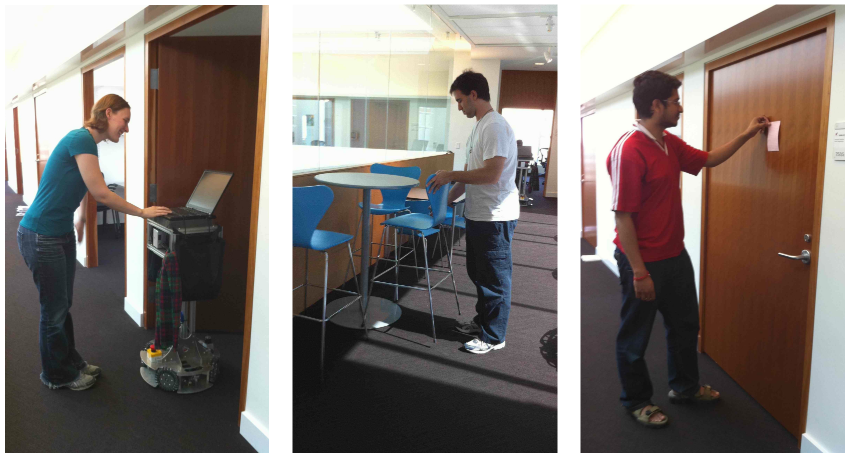
5.4.1 Help Types

We test environment occupants' willingness to help the robot with three help types that take different amounts of time to answer and are different distances from the occupants' offices. The questions are all spoken out loud for the occupants to hear and displayed on the robot's laptop screen, but we require that the occupants answer the questions using the visual user interface on the laptop as it cannot understand speech.

Localization Reasoning Uncertainty Help When CoBot requires localization help, it can find an open door to ask the occupants to share their room number in the following way:

> *I cannot determine my location. What is the room number of this office?* (multiple choice)

After speaking the question, CoBot lists three predictions of the possible office numbers and an additional textbox (in case the three office numbers were all incorrect) for the occupant to respond with. Because building occupants should know their own office number,



(a) Localization (b) Moving Chairs (c) Writing Notes

Figure 5.7: Occupants were asked to answer a multiple choice localization question, move chairs out of the way, and write a note on another occupant's door.

CoBot's localization questions should be fast for them to respond to and do not require the occupants to leave their offices except for accessing the robot's backwards facing screen.

Moving Chairs Actuation Limitation Help Our building contains many seating areas with moveable chairs that are hard for the robot to detect. Even if CoBot could detect the chairs, it has no way of physically moving chairs that are blocking its path. CoBot navigated to occupants' offices and asked them to check and move chairs in the closest common area so that it could pass:

> *My laser range finder cannot determine the location of chair legs in the common area. Can you please move chairs in the common area to clear a path for me?*

While occupants can easily identify and move chairs out of CoBot's way, this task requires that participants leave their offices to help the robot (Fig. 5.7(b)). Help with this limitation ensures that CoBot can safely navigate through the environment, assuming that the occupants actually move the chairs as requested. The occupants are asked to confirm their action on the laptop user interface so that the robot knows it is safe to continue.

Writing Notes about Mail Delivery Actuation Help In this work, we assume that CoBot will eventually perform a mail delivery task as an incentive for occupants to help the robot. However, CoBot does not have the manipulation abilities to select mail for an occupant or leave a message that a package is available. This limitation affects its ability to perform its task and therefore requires occupants to perform some of the robot's task for it. The robot requested that occupants write a note to notify a nearby occupant that they have a package waiting in the mailroom located one floor below our test floor (Fig. 5.7(c)):

> *I am trying to deliver a package to room 7505, but the door is closed. Can you please use the paper and pen in the bag on my left to write a note that a package is available downstairs and place it on their door?*

We assume that the robot can find an office close-by to write the note. In this study, the office number of the nearby office (here 7505) changed depending on the occupants' location to ensure the occupants did not have to walk too far out of their office to deliver this message - the room was on average 5 offices away from a helper (closer than most participants indicated they were willing to travel in the previously-described survey). When CoBot navigated to offices to request help writing a note, it carried paper and pens in a tote bag for the occupants to use.

5.4.2 Method

Prior to the study, occupants on one floor of our academic building were told that the robot would soon be deployed in our environment to perform services for them, such as mail delivery. Additionally, they were told that it sometimes requires help to overcome its limitations, and that we were currently testing the robot's ability to ask for and receive help. Occupants were given the choice to help the robot if they were **available**, but did not have to help if they did not want to and could close their office doors to indicate that the robot should not ask them for help. Only one graduate student office emailed the authors to ask not to participate. As a result, CoBot sought help from each of 78 offices, nine times over three days (three times per day).

In order to compare occupants' availability to help with each request, CoBot attempted to ask each occupant for each **help type** each day for a within-subjects study design. In order to simulate an actual deployment of the robot, we:

- randomly assigned the order of the three requests each day such that each question was asked once per day and at different times on different days,
- randomly chose two of the nine requests to offer a gift of candy to represent the benefit provided when a robot performs services for them (*i.e.,* brings mail).

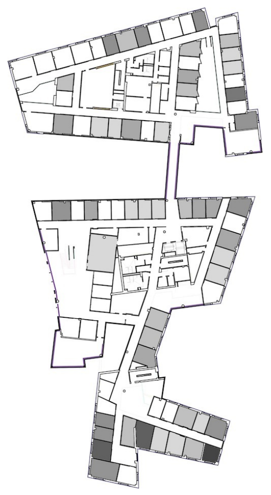
The occupants each received at most two gifts total during Cobot's nine potential visits to reflect the fact that the robot will likely need help from an occupant even when they are not receiving some benefit (*e.g.*, his/her mail). The assignment of gifts for each occupant was randomly chosen before the study started, and it was not guaranteed that the occupants would be available at the gift times. However, occupants were told about the gifts ahead of time so these gifts served as the **incentives** for occupants to help the robot.

The robot traversed the floor at 9:30am, 12:00pm, and 2:30pm for three days along the same predefined path. The occupants were not able to see the wizard drive the robot or trigger the question from their offices. When the robot arrived at the door to each office, it first spoke "Hello" to get the occupant's attention and then spoke the the question and printed it on the laptop screen. The robot required participants to click on the laptop to respond after performing the requested help at a **help location**. Upon pressing "Done," the robot would speak "Thank you" as an indication to the wizard to move the robot to the next office. Some occupants ignored the robot and did not click "No, I cannot help." After 10 seconds without a response from an occupant, the wizard timed out the question, moved the robot to the next office in the sequence, and this was logged as a refusal to help the robot. The wizard skipped offices that had closed doors.

After the study, the authors conducted interviews with occupants to understand their perceptions of the robot, their feelings about answering questions through the study, and to follow up on any observations about the occupants' interactions with the robot.

Robot Apparatus: In order to ensure that the robot stopped at the correct sequence of doorways, the wizard controlled the robot's motion and triggered the robot to speak the questions and display them on the screen. The screen interface on the robot's laptop contained one large text area with the question, and two buttons - "Yes, I am willing to help" and "No, I cannot help". For all questions, if the occupant clicked yes, the robot automatically provided instructions to click an additional "Done" button when the task was complete. Multiple choice locations were also displayed for the localization question. Whether the task was completed or not, the robot thanked the occupant and the wizard navigated the robot to the next office. As occupants clicked on the interface, it logged the office number along with the question type, responses to the question and the time stamp to use in the analysis. The occupants were required to use the screen interface - the robot did not respond to speech.

Measures: In order to evaluate the willingness of occupants to answer the robot's questions we use four main measures: number of open doors, number of times occupants


(a) Total Open Doors

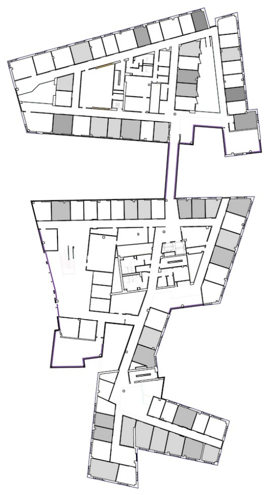

(b) Refused or Ignored

Figure 5.8: (a) 130 doors were open out of 702 in nine trials (darker means the door was open more frequently). (b) Out of 130 open doors, 53 occupants in those offices refused to help the robot.

helped, locations of the occupants, and time spent responding. The number of open doors is an upper bound on the number of occupants who will help the robot. The number of times each occupant helped the robot allow us to understand the availability of humans to help the robot throughout the building. We determine whether there is a difference in response frequency and the amount of time it took occupants to respond to the different question types over time. Due to the small sample size (78 rooms tested 9 times each), we only test for trends in our data and not statistical significance.

5.4.3 Quantitative Results

Our results show that some, but not all, occupants were willing to help the robot at any given trial and that they were largely distributed through the environment. Interestingly, their willingness changes with availability at different times of day but not depending on the type of question. Participants were equally willing to help with all types of questions although some took much longer than others and were at a further distance from their office.

Availability: In total, 130 doors were open out of a combined 702 in nine trials (Figure 5.8(a) - darker means the door was open more frequently). Occupants helped the robot 78 times out of the 130 possible open doors. 46 offices were open at least once and 31 of those offices contributed responses. Each office was open on average 1.8 (s.d. 1.9) times out of 9 possible and occupants answered on average 1.1 questions (s.d. 1.7). The high standard deviation for the offices indicates that there were a few occupants available almost all of the time, while many occupants were unavailable. Seven out of the 78 offices contributed 36 of the 78 responses to the robot. This indicates that there is a group of people that would likely be available for the robot to ask for help, although at any particular time there are likely to be many more occupants that the robot could ask to further distribute the help.

Help Type: In terms of the question types, we found that each type of question took a very different amount of time to complete but occupants helped equally with them all. Occupants took on average 30.1 sec (s.d. 18.1) to complete localization questions, 55.6 sec (s.d. 24.6) for chair questions, and 88.3 sec (s.d. 45.3) for the note writing questions. Despite these differences, when occupants were available to help, they were willing to answer any type of question. We found little difference in the response rate for each question type - 57.5% of the localization questions, 62.5% of the chair questions, and 50% of the notes questions. This finding indicates that a robot would not need to reduce the asking frequency of questions that take longer to answer.

Incentive to Help: We found no statistical difference in answering frequency when occupants were offered gifts to when they were not, but some occupants indicated in interviews that the gift did affect their decision to help. In particular, we observed some occupants deliberately opening their doors when they heard the robot down the hall so that they could help and possibly receive a gift. While gifts were only offered at random times, some participants stated in the interview that they would be more willing to help the robot if it offered candy more often.

Distance to Help Location: Figures 5.10(a), 5.10(b), and 5.10(c) show the frequency of help from each office by time of day, with darker colors representing more frequent help. Interestingly, the few frequently available occupants were largely distributed around the building - especially in the areas where the robot has the most uncertainty (Fig. 1.2(b)). A random selection of seven offices would not necessarily result in such an even distribution across the building. Every help location in the building was at most 10 offices from an occupant that helped the robot during many of the trials, except for the north side of the building at 2:30. Because the robot is least uncertain in the north side of the building, the

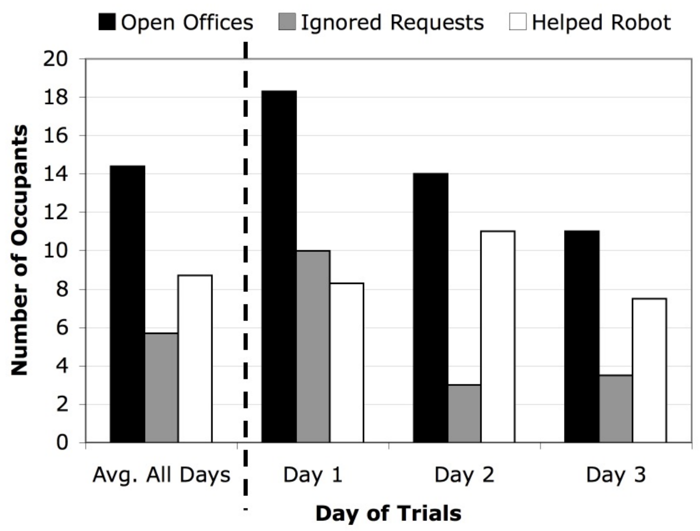
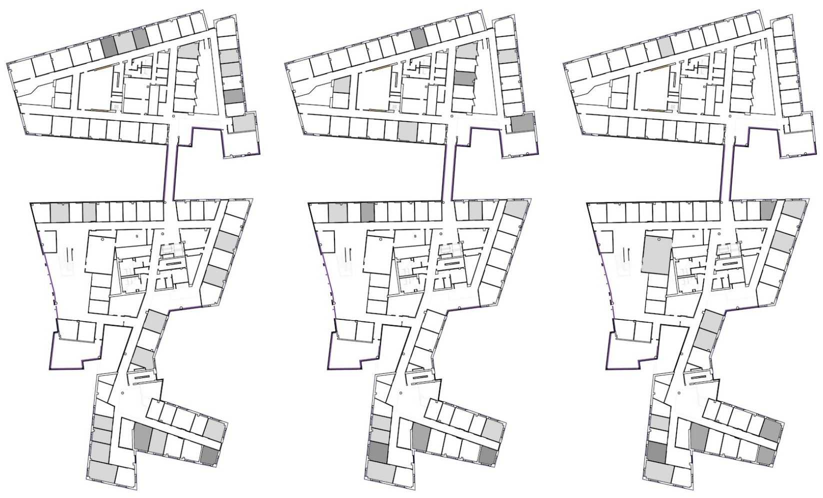


Figure 5.9: On average, 14.4 (s.d 7.5) offices were open for each trial and 8.7 (s.d. 3.6) occupants helped the robot. While there were fewer open doors over time, more occupants with open doors were willing to help the robot.

robot may be able to navigate despite the low availability. However, the distance between available occupants indicates that any particular helper will not need to travel far to help the robot.

Novelty Effect: On an average trial, 14.4 (s.d. 7.5) doors were open, and occupants in those offices responded to the robot's requests for help 8.7 times and refused to help or ignored the robot the remaining 5.7 times (Fig. 5.9). However, there is a strong effect of day on the proportion of open offices and available helpers. The number of open office doors dropped each day likely due to occupants' prior knowledge about when the robot would be visiting. The number of occupants that did help the robot remained constant over the three days indicating less of a novelty effect for those 7 occupants who helped the robot the most.

To summarize, we found that availability, incentive to help, and the novelty effect had a significant impact on the willingness of occupants to answer questions. Because our robot asked participants to travel only a short distance to help locations, we did not see an effect of distance on willingness to help. While help type did not have significant effects on overall willingness, we did find that it had a significant effect on usability based on our qualitative interviews after the study.



(a) Helped at 9:30am (b) Helped at 12:00pm (c) Helped at 2:30pm

Figure 5.10: (a), (b), (c) While occupants in each part of the building answered the robot's questions for most times of the day, we find almost no available occupants at 2:30 on the north side of the building. The darker the office color, the more often the occupant responded to questions.

5.4.4 Qualitative Results

We interviewed participants after the study to further understand their actions during the study. We found that these actions depended on more of our human state attributes.

Help Types and Question Repetition: While most participants did respond to the repeated questions (each of the 3 questions were repeated each day), during the interviews, occupants reported that they were confused as to why the robot asked them to perform the same help multiple days in a row. One occupant reported that he wrote the incomplete notes out of frustration when he was asked to do the same task multiple times. This finding mirrors previous work that showed that people who are asked for help too frequently tend to stop responding to help requests in the future Scollon, Kim-Prieto, and Diener (2003).

While this repetition of questions is an artifact of our study, it indicates the need for the robot to keep track of which occupants it has asked for help to purposefully plan to avoid those offices unless there is no other help available. This would require that the robot also model the history of questions to more heavily weigh the occupants who have not been asked for help recently. The robot would then need to model not only who is available to help but the cost of asking someone too frequently, and the additional constraints for navigational planning and determining who to ask for help.

During actual deployments of a robot, however, reducing question repetition could be difficult. If only a single occupant is frequently available in areas of frequent limitations, the robot would have no choice but to travel in that area sometimes. In order to reduce the likelihood of this happening, it could request help from an occupant who is further away and is not asked for help as often. Additionally, the robot could include time in planning to vary the time of day it would complete the task (possibly delaying its task) if another occupant is available at another time.

Interruptibility: In designing the robot's initial interaction, we used an assumption that an open office door indicates that the occupant is interruptible. However, we found that often doors are left open even when occupants are in meetings or on the phone. Occupants who were in meetings and did not want to help the robot either verbally tried to send the robot away or ignored the robot until it left their doorway. Surprisingly, however, many occupants did interrupt their meetings or put their phone call on hold to help the robot and some reported that the interruptions were "well-needed breaks in their day."

Models of interruption have been used for supervisors to warn them about the robot needing help soon Shiomi et al. (2008). While it might seem obvious that a robot in human environments should also have a model of interruptibility through real-time sensing, a naïve interruption model may predict that occupants are not available to help when they are in meetings or on the phone Fogarty et al. (2005). However, if CoBot used this model, it would have received fewer responses compared to asking for help at every open door. A robot must learn, through its long-term interactions, which occupants are willing to be interrupted and under which conditions (*e.g.*, who they are speaking with, whether they are working on the computer) to take full advantage of the occupant help.

Accuracy and Deception: We also assumed that participants would answer the robot's questions accurately and completely when they agreed to help. While occupants did answer 100% of the localization questions accurately, we found that several participants deceived the robot, responding that they moved the chairs or that they had written the note when they had not. In particular, two occupants submitted blank notes and two wrote notes with incomplete information about the package location.

It is unclear whether participants deceived the robot because they were told it was a study and not deployed for real. However, these results indicate that a robot must maintain some uncertainty about whether a task was actually performed for it. The robot could ask another occupant to confirm the task was completed since it may be difficult for the robot to detect deception itself, or otherwise use extra sensors to detect that the task was completed (*e.g.*, a sensor near the paper and pens to detect if an occupant picked them up to write a note), or mitigate failures by apologizing and requesting help from someone else Lee et al. (2010). If the robot can determine and learn through its interactions over time which occupants are deceptive Donmez and Carbonell (2008); Donmez, Carbonell, and Schneider (2009), it should avoid asking for help from them in the future by either lowering the availability of the occupant or adding additional costs related to the trustworthiness of the occupants.

5.5 Chapter Summary

This chapter has presented a series of surveys and studies of device users and environment occupants to understand what state impacts their willingness to provide help. In the survey of phone users and study of robot users, we showed that the cost of help and incentive to help impacted willingness and that these costs were highly personalized to each individual user. For phone users, the cost of help depended on the situation the user was in (*e.g.,* in the car or in a meeting). For robot users, the cost and incentive depended on the expectations that users had about what robots should be able to do.

In our survey and study of environment occupants around our robot, we showed that willingness to help depended on many more human state attributes:

- availability
- interruptbility
- cost of help
- incentive to help
- distance to help location
- accuracy/expertise
- help type
- frequency of questions
- recency of questions

Notably, because environment occupants are not near the robot, they must travel to the location of help and they may receive many questions throughout deployment. Additionally, although the occupants are not receiving direct benefit from the robot's current task, there is an overall incentive for them to help in order for the robot to perform tasks for them in the future.

As a result of these studies, our human-centered algorithms include models of these attributes are used in determining whether and who to ask for help.

Chapter 6

Increasing Human Response Accuracy

Prior approaches to requesting help with reasoning uncertainty from humans that assume humans are supervisors and always knowledgeable about their devices' state inferences when providing help, such as active learning (Cohn, Atlas, and Ladner (1994); Mitchell (1997)), learning by demonstration (Argall et al. (2009)) and mixed-initiative and semi-autonomous robots (Asoh et al. (1997, 1996); Shiomi et al. (2008)). However, we do not necessarily assume that humans in the environment such as device users and occupants are supervisors and will be knowledgeable and always answer the device's reasoning uncertainty questions correctly (Fong, Thorpe, and Baur (2003); Donmez and Carbonell (2008)). This chapter focuses on how intelligent devices can increase the likelihood of receiving correct responses to reasoning uncertainty questions from human non-supervisors by changing the questions they ask.

Human-human interactions are often grounded in the common references and experiences we have with others (Clark and Wilkes-Gibbs (1986)). When we ask for help from other humans, these common experiences help clarify the question being asked and help us answer as accurately as possible. However, because humans may not share knowledge or references with robots, it has been suggested that robots should explicitly share their state information with humans as they act in the world (Clark (2008)). We are interested in the types of information that could and should be shared with humans.

We performed an extensive human-robot interaction (HRI) and human-computer interaction (HCI) literature review to understand the types of information that other robots and devices have used to ask for help from humans. We found that asking for help is common when there is a lot of uncertainty in inference (*e.g.,* recognizing or labeling objects in images (von Ahn and Dabbish (2004)) or localizing a robot (Asoh et al. (1997))). Additionally, we found several sets of guidelines for the types of information that intelligent devices should provide (*e.g.,* Bellotti and Edwards (2001); Erickson and Kellogg (2001); Horvitz (1999); Shadbolt and Burton (1989)). In analyzing prior work and the guidelines, we identified four common types of information that researchers in HRI and HCI have commonly used:

- Context: The current sensor information related to the task (*e.g.,* features detected through vision or LIDAR);
- Prediction: The current inferred state (*e.g.,* the object detected with vision or the location of the robot),
- Uncertainty: The probability the inference is incorrect, and
- Feature feedback: The critical features from context used in inference (*e.g.,* the number of sides of the object or the carpet pattern near the location).

While different combinations of these four types of information are often used when requesting help from humans, little work has been performed to identify which combinations result in more accurate responses.

In order to understand how each of these types of information impacts human response accuracy to devices' requests for help with reasoning uncertainty, we performed five studies on different intelligent devices and systems. On a robot, we tested how to ask for help with localization uncertainty and also shape recognition. On other intelligent systems, we tested how to ask for help with activity recognition on a smart phone (similar recognition of volume preferences in terms of device users giving their preferences), email filtering online, and interruptibility recognition in offices. We briefly describe each task below:

- *Localization* Many mobile robots perform localization tasks to determine where they are and how to navigate to a goal location. If a robot cannot determine its location, it may miss turns and have to backtrack down the hallway. We have demonstrated that a robot can ask a human to identify its location on a map when it is uncertain (*i.e.,* "Can you point to where we are on this map?" (shows map)) to navigate more quickly and accurately to its goal (Rosenthal, Biswas, and Veloso (2010)).
- *Shape/Object Recognition* Shape recognition (*i.e.,* identifying shapes of building blocks as a cube, cylinder, *etc.*) is similar to a camera-based object recognition task a robot might have to perform. In such recognition tasks, if a robot cannot determine the shape of an object that it is supposed to pick up, for example, it may fail to complete its task. It could instead ask a human to identify the shape or object

when it is uncertain (*i.e.,* asking "What shape is the red building block?") in order to overcome the recognition failure and complete more tasks.

- *Activity Recognition* The sensors on mobile devices are often hidden and their data is hard to explain, but they capture activities that their users are aware of, such as exercise patterns (Consolvo et al. (2008)). For example, a physical activity coach performs activity recognition using sensors to identify exercises the user performs. An application like this may record users' activities for doctors to analyze physical activity levels, and thus users have an interest in answering its questions such as "What activity are you doing?" to ensure it correctly identifies their activities.
- *Email Filtering* While most users do not think of their desktop computers as learning from their actions, word processors learn to spell-check new words and email applications learn which emails are spam and which are not. Because these labels are subjective, the user carries the burden of having to label their own data and may make mistakes. An email filter could try to sort emails in the inbox into a folder and request help by asking "Where does this email belong?" if it is uncertain.
- *Interruptibility* The problem of recognizing when someone is interruptible (on a scale 1-5) has been widely studied in the literature (*e.g.,* Fogarty et al. (2005); Horvitz, Koch, and Apacible (2004)). We asked non-supervisors to predict interruptibility by answering the question "How interruptible is this person?" on Amazon.com's Mechanical Turk - an actual large-scale system that is often used to have non-supervisors help answer questions cheaply.

We test other applications in addition to our two main device tasks in order to understand how our findings generalize.

Next, we describe each type of information that devices could include. Then, we describe our general study design that we followed for each of the 5 experiments. In the sections following, we describe our experiments and results. Finally, we discuss our findings, generalizations, and considerations for implementing them on actual devices.

6.1 Types of Information to Provide Helpers

We analyzed different systems that request help in different domains and categorized the types of information they provide to contextualize or ground those requests (Clark and Wilkes-Gibbs (1986)). We found four popular categories of information also proposed by other researchers (*e.g.,* Bellotti and Edwards (2001); Erickson and Kellogg (2001);

Work	Amt.	Lvl.	Unc.	Pred.	FF	Act.	Int.
Horvitz (1999)	X		X	X	X		X
Bellotti and Edwards (2001)	X			X	X		
Erickson and Kellogg (2001)	X	X	X		X	X	

Table 6.1: Types of information most commonly provided when asking non-supervisors for feedback include amount of context (Amt.), level of context (Lvl.), uncertainty (Unc.), prediction(Pred.), feature feedback (FF), executed action (Act.), and social interaction (Int.).

Horvitz (1999)): context, prediction, uncertainty, and feature feedback. While these other researchers have also proposed including other information such as the current action that is being executed (Bellotti and Edwards (2001)), acknowledgment of accountability to the humans in the environment (Bellotti and Edwards (2001); Erickson and Kellogg (2001)), and the costs and benefits of different user responses (Horvitz (1999)), we did not find these other types to be as prevalent in implemented systems. Table 6.1 demonstrates the information that each researcher suggests.

Next, we provide operational definitions for each kind of information and provide examples of how to implement them in our two task domains (summarized in Table 6.3). We compare the accuracy of non-supervisor responses to robots' questions that include different combinations of the information.

Context Many robots and other applications provide humans with some contextual information about their sensor data before asking a question. However, some provide more contextual information than others and some provide it at different levels.

Local vs Global Context A robot user interface to monitor speech recognition errors provides the audio that could not be recognized and a transcript of the conversational context (Shiomi et al. (2008)). However, another robot provides no context at all about its current sensor readings when asking for help with localization and navigation in an office hallway (*e.g.,* Asoh et al. (1997)). Similarly, other interfaces provide more or less context. For example, when BusyBody asks users to estimate their own interruptibility, it does not explain what it thinks the user is doing (Horvitz, Koch, and Apacible (2004)). Hoffman et al. request help from Wikipedia users to fill in missing summary data as the users are reading an article (Hoffmann et al. (2009)). When users are asked if the text they are reading in the article belongs in the summary, important keywords are not provided in the text. When reading the summary, users are provided with excerpts that could be added to make the summary more complete. In studies of interruptibility, it has been shown that people make judgments with relatively small amounts of context (15 seconds) and extra context (30 seconds) does not improve accuracy (Fogarty et al. (2005)).

We define two kinds of contextual information: local context and local+global context. Local context are the features immediately in the area that the robot is trying to classify or infer. For example, in the speech recognition user interface described above, the local context is the audio recording of the sentence that is not recognized. The local+global context additionally contextualizes the local context in the entire state space (*e.g.,* the unrecognized sentence within the current conversation). In our shape recognition task, the local context is the color feature of the object in question and the local+global context is the location of the object in the image area (*e.g.,* top, left, bottom, right). In our interruptibility study, we use 15 seconds of video as local context, while 30 seconds is local+global.

High vs Low Level Context We also vary the context in terms of the feature level information that is provided (not possible in the robot experiments because the data is too dense and complicated to provide low level context). Recently, researchers demonstrated that labelers' accuracy can depend on the level of contextual information they are provided. When users understand and use their own rules for classification, they are better at making those classifications compared to classifying based on the computers' rules (Stumpf et al. (2005, 2007a)). This finding is supported by work in feedback in information retrieval applications (Rui et al. (1998a); Salton and Buckley (1990a)) which mask the low-level sensor-level features that computers use and collect (i.e., individual keywords in documents or accelerometer data) and allow users to search for information using high-level meaning attributed to the low-level data (*i.e.,* summaries of documents or physical motion inferred from accelerometers). However, because it is often difficult to generate the high-level explanation of context, many applications provide only the low-level raw data, like pictures, to labelers instead of a summary with the assumption that they can find their own meaning (von Ahn and Dabbish (2004); von Ahn et al. (2008)).

Subjects in the low-level context condition receive information about sensor readings on the activity recognizer, keywords in an email, and raw interruptibility video footage, to help them make their classifications. For instance if someone was jumping, the sensor might read "shaking" - we do not expect users to interpret exact numerical sensor readings or graphs. With high-level context, participants received explanations such as email summaries or body motions like "your feet are leaving the floor" that correspond to sensor readings.

Prediction The prediction is the most likely state based on the inference. For example, in our shape recognition task, the prediction is the most likely shape (*i.e.,* cube or cylinder). In speech recognition, the prediction is the most likely sentence that was spoken (Shiomi et al. (2008)). An interface may automatically fill in fields in an online form or provide a prediction for which folder to sort a piece of email into (*e.g.,* Culotta et al. (2006); Faulring et al. (2010)). Providing a prediction may reduce a user's work to respond because they only have to confirm an answer rather than generate it (Eagle and Leiter (1964)). In this work, we test the accuracy of participant responses when the intelligent system provides a correct prediction. Testing accurate predictions allows us to understand how people trust the intelligent system and how much they are paying attention to what it says.

Uncertainty Many classification and inference algorithms give a measure of uncertainty - the probability that a prediction is inaccurate - in addition to the prediction itself. Studies of context-aware and recommender systems show that providing users with the level of uncertainty in predictions improves its overall usability (*e.g.,* Banbury et al. (1998); Mcnee et al. (2003)), even if the system does not provide the exact uncertainty value (Antifakos, Schwaninger, and Schiele (2004)). For example, in the shape recognition task, the robot indicates that it is uncertain with the phrase "Cannot determine the shape."

Feature feedback We define feature feedback as a set of contextual features that are most important for the inference. For example, in the shape recognition domain, feedback might include the number of edges or sides a shape has. It has been shown in both the active learning and HCI literature that people are capable of providing useful feature feedback to a system. For example, in text classification domains, people were able to indicate not only the type of news article (sports, current events, *etc.*) but also keywords in the article that determine the type (*e.g.,* team or score for sports (Raghavan, Madani, and Jones (2006a))). People have also been able to successfully provide corrective feedback for handwriting recognition, email classification, and other domains (*e.g.,* Mankoff, Abowd, and Hudson (2000); Scaffidi (2009); Shilman, Tan, and Simard (2006)). We test whether providing this additional feedback influences the accuracy of non-supervisor responses to inference questions.

Combining the Information

Despite the common use of our four types of information, we found that different combinations of them have been used on different robots and other intelligent systems. For example, search and rescue robot interfaces for supervisors almost always include the robot's local context and inference predictions (Yanco, Drury, and Scholtz (2004)). However, sensor uncertainty and feature feedback did not appear in interfaces, because supervisors implicitly also knew about the robot's uncertainty and were able to give feedback about the features without being asked.

In total, there are three ways to provide amount of context: no context, local context,

State Info.	Operational Definition	Examples
Local Context	The features around the area that the robot is trying to classify.	*Shape:* "You are working with the red and green blocks." *Loc.:* "I am near the kitchen." *Act Rec:* "Your feet are leaving the ground" *Email:* "The email has keywords A and B." *Int:* (15 seconds of video)
Local+Global Context	The location of the local context in the state space.	*Shape:* "You are working with the red and green blocks on the top left" (of tower) *Loc.:* "I am near the kitchen by the 7100 corridor." *Act Rec:* "Your feet are leaving the ground together and repeatedly." *Email:* "The email has keywords A and B and title C and D." *Int:* (30 seconds of video)
High Level Context	The high level activity data	*Shape and Loc:* Untested *Act Rec:* "Your feet are leaving the ground." *Email:* "The email is summarized by F & G." *Int:* "The door is open and there are people inside."
Low Level Context	The sensor level data	*Shape and Loc:* Untested *Act Rec:* "Shaking motion detected." *Email:* "The email has keywords A and B." *Int:* (raw video)
Prediction	The most probable answer	*Shape:* "Prediction is a rectangular prism." *Loc.:* "I think I'm at the red dot." (on map) *Act Rec:* "Prediction: Jumping." *Email:* "Prediction: Session Changes." *Int:* "Prediction: 4 (on scale 1-5)"

Table 6.2: Operational definitions for each of the types of information we focus on in this work, and examples of that information in our five task domains: Shape Recognition (*Shape*), Localization (*Loc*), Activity Recognition (*Act Rec*), Email filtering (*Email*), and Interruptibility (*Int*).

State Info.	Operational Definition	Examples
Uncertainty	Probability the inference is incorrect	*Shape, Loc, Act Rec:* "Cannot determine the (shape, location, activity)." *Email:* "Cannot confidently make a prediction." *Int:* "Cannot determine if the person is interruptible."
Feature Feedback	Ask the human for a set of contextual features that are indicative of the answer	*Shape:* "What features describe the block?" *Loc.:* "Please describe the location." *Act Rec:* "How can this action be detected?" *Email:* "Why is this folder correct?" *Int:* "How did you make that determination?"

Table 6.3: Continued from Table 6.2: Operational definitions for each of the types of information we focus on in this work, and examples of that information in our five task domains: Shape Recognition (*Shape*), Localization (*Loc*), Activity Recognition (*Act Rec*), Email filtering (*Email*), and Interruptibility (*Int*).

or local+global context and each at high or low level for non-robots. For each of those choices it could provide an inference prediction or not. For each of those six choices, it could provide uncertainty information or not. And finally, for each of those choices, it could request feature feedback or not. As a result, there are $3 \times 2 \times 2 \times 2 = 24$ different combinations of this information that could be provided on robots and $3 \times 2 \times 2 \times 2 \times 2 = 48$ different combinations for the other intelligent systems. We combine the types of information in the following order: 1) uncertainty, 2) context, 3) the question the robot wants answered, 4) prediction, 5) feature feedback. For example, when the robot in the shape recognition task asks about a block with all four kinds of information, it would say:

> **Robot:** "Cannot determine the shape. You are working with the red and green blocks in the top left. What shape is the red block? Prediction is Rectangular Prism."
> **Human:** Answers
> **Robot Follow Up:** "What features describe this block?"

However, if the robot only asks with uncertainty and prediction (no context or feature feedback), it would say:

> **Robot:** "Cannot determine the shape. What shape is the red block? Prediction is Rectangular Prism."

Human: Answers

In our studies, we explore the impact of these different combinations on the accuracy of non-supervisor responses. In the shape recognition, activity recognition, and email filtering tasks, we will test all combinations to find the most accurate. In the localization and interruptibility tasks, due to time constraints, we test only some of the combinations. While the exact statements are domain specific, they illustrate how we use the operational definitions and can be easily generalized to other similar applications. Next, we contribute our study design that we used to test the combinations of information.

6.2 Study Design

We define non-supervisors as humans who have a task to attend to and do not monitor an intelligent system's progress (Rosenthal, Veloso, and Dey (2012b)). Because they are busy with their own task, they may not hear the information the intelligent system might provide when asking for help, they may be rushed to answer, and as a result, their answers may be incorrect or they may not answer at all. However, despite the interruptions, some non-supervisors, such as system users, have incentive to accurately answer questions in order for the intelligent system to be able to complete its tasks for them. For example, visitors, who are escorted to meetings by a robot, may have incentive to answer questions about localization so that they can continue following the robot to their meetings (Rosenthal, Biswas, and Veloso (2010)). Recent studies on email systems confirm that people are willing to be interrupted if there is a perceived benefit for them later (Stumpf et al. (2007a,b)). We are interested in combinations of information that improve response accuracy under these conditions.

We contribute a two-phase study design, namely an initial exploration phase to test many combinations of information and a validation phase to explicitly compare our best found combination with a baseline combination. In the initial exploration phase when the intelligent system asks for help, we vary the combination of information that participants receive to understand how it affects the accuracy of their responses to the questions. Given the number of combinations, we cannot test each combination separately. Instead, we test the impact of each of these different types of information. In our initial experiment, we show which types of information have a positive effect on the accuracy of the participants, either alone or in conjunction with other information in a between-subjects design. Using the between-subject design, we can understand how user accuracy varied through the experiment without confounding user opinion by presenting multiple combinations in a short period of time. We measure the response rate and accuracy to the systems' questions.

Although we do not test each combination, we perform a validation comparing the combination which includes all positive information from the initial phase against a predicted best combination from a set of HRI and HCI experts - researchers who work in these areas - to show that our combination statistically more accurate.

The design is intended to mirror real-world conditions of asking non-supervisors while controlling for variations in the timing of the questions.

Cost of Help In both the initial exploration and validation phases, participants are given a task to complete and limited time to complete it, preventing them from supervising the system. They are told that they will only be judged on their task performance but that they can help the system if they have time to complete the task.

Incentive to Answer Non-supervisors have incentive to answer questions despite the interruption, because they want the intelligent system to perform tasks for them. In our study design, participants are told that the system will interrupt their task to ask them for help if it is uncertain of predictions it is making. They are told that helping the system during their current task will improve their performance on a second task (which they are never actually given), but answering is optional as it may slow down their performance on their current task. During the study, participants must determine if they have time to answer the question without affecting their task performance. We measure the response rate to understand how participants evaluated the tradeoff.

Control of Question Timing The timing of a question may significantly affect response accuracy if the question is referring to something that the participant is doing at the time. In order to control the timing, the experimenter triggered the same questions during the same events in the task for all participants in all conditions (the systems were wizard-of-oz'd (Green and Wei-Haas (1985))). Controlling the timing ensures that the only difference between study conditions is the information the participants receive when being asked for help. The ground truth of what the systems are asking about are based on the experimenter triggers. We compare the participants' responses to the experimenter's ground truth to measure response accuracy.

Two-Phase Validation The initial exploration phase is a between-subjects experiment. Participants are assigned to one condition of the study and receive a single combination of information for all questions asked. By comparing participants' responses in the different conditions, we can determine the best combination of information. After an initial study, our study design includes a second within-subject validation study to explicitly test our best found combination of information against a combination that HRI or HCI experts predict will result in the most accurate responses. We chose the HRI or HCI expert combination to serve as our baseline instead of a baseline with no information, because we expect that a system that asks for help would be implemented with their predicted combination. The validation serves to show that our results are an improvement over this baseline.

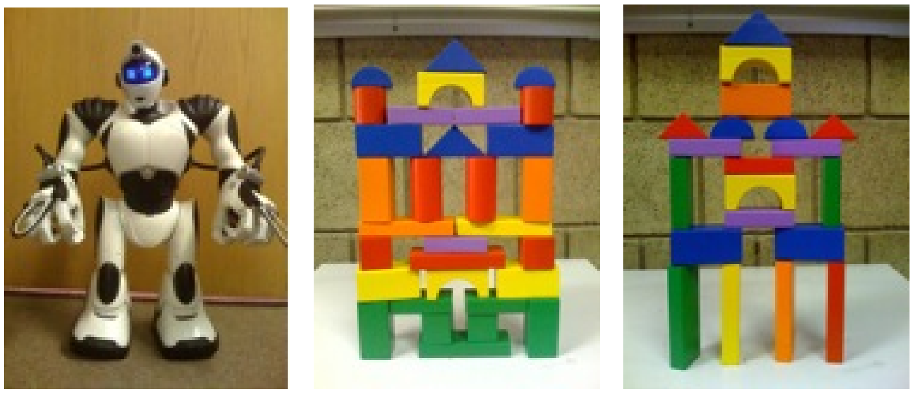
Next, we describe two experiments conducted using our study design.

6.3 Experiments

To investigate the impact of an intelligent system providing different combinations of information when asking for help, we compared the accuracy of non-supervisor responses during each of our five tasks described in detail below. We first conducted the initial phase of the shape recognition, activity recognition, and email filtering tasks, testing all combinations of information. At the same time, HRI and HCI researchers were brought together to come to a consensus on which combination of information they thought would result in the highest accuracy - which we call the HRI or HCI expert input combination. After finishing our initial tasks, we ran validations of teach task to directly compare our best combination to the HRI or HCI expert input. Finally, we conducted the localization and interruptibility task experiments to test our best validated combinations against the HRI and HCI expert input once again. We show that our single best robot combination outperforms all others in both robot domains, and best intelligent system combination outperforms the other combinations in the our other three domains.

6.3.1 Task Procedures

Questions and Information Combinations Before each study began, we generated the questions and information the robot provided based on the expected state at the time the questions would be asked. We, first, determined which sensors that would be used in the task (*e.g.,* camera for shape recognition and a WiFi sensor readings for localization). Then, we chose the blocks, locations, emails, activities, and videos that the intelligent systems would ask for help about and used our operational definitions to generate the information the systems would provide about them. We combined the information and questions in the following order: 1) uncertainty, 2) context, 3) the question the robot wants answered, 4) prediction, 5) feature feedback. For example, for the shape recognition task, when the robot provided all information it would say:



(a) Robosapien V2 robot (b) A block structure (c) A block structure

Figure 6.1: (a) The robot asked participants to indicate the shapes of blocks they were holding as they built the structures. (b) and (c) Examples of structures participants were asked to build out of multi-colored blocks.

Robot: "Cannot determine the shape. You are working with the red and green blocks in the top left. What shape is the red block? Prediction is Rectangular Prism."
Human: Answers
Robot Follow Up: "What features describe this block?"

Table 6.3 outlines examples of each type of state information for each task and outlines the different combinations of information.

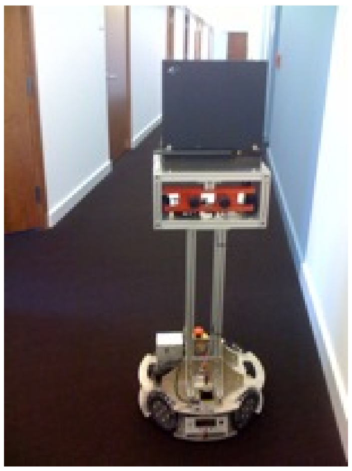
Shape Recognition Initial Task For the shape recognition task, we asked participants to build structures out of blocks while the robot tried to recognize the block shapes (Rosenthal, Dey, and Veloso (2009)). Our robot in this study, the RoboSapien V2 robot (Figure 6.1(a)), contains a camera to track primary colors and LEDs that rotate towards the motion so that it appeared to be watching the participants build the structures. Upon arrival for the study, participants were randomly but evenly assigned to one of the 24 combinations of information, given an explanation of the study and signed a consent form. Before starting the task, participants were told that while they were building the structures, the robot might ask them for help. The building task prevented the participants from supervising the robot too closely. They could choose not to respond to the questions if they were too busy with building the structures, but they were told that answering questions would benefit them in a second related task (which we did not actually have them perform).

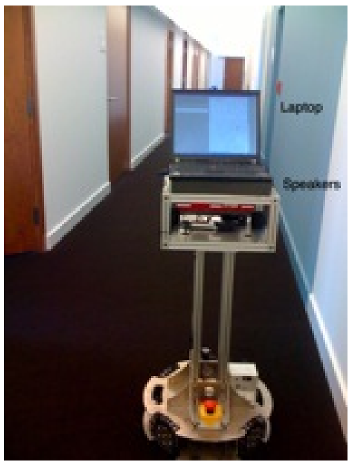
Participants were, then, given 50 colored blocks and four pictures of structures each containing 20-30 blocks to build in 12 minutes (Figure 6.1(b) and 6.1(c)). When each of 8 pre-designated blocks were picked up by the participant, the experimenter pressed a button to make the robot ask participants to identify the shape of a block they were holding in their hand. The participants were then given a chance to answer the questions verbally if they chose to. After completing the task, participants were given a survey about their experiences with the questions. Then, participants were told there was not enough time to conduct the second task and were dismissed after being paid.

Shape Recognition Validation While the initial phase was run, we sought advice from three members of the HRI community about which information they believe the robot should use when asking for help. The community members understood both the technical data that could be collected and the usability requirements necessary for effective communication to non-supervisors. We explained each type of information and how the information could be combined together. They suggested that the robot should provide *uncertainty, local context, no prediction, and feature feedback*, which we call the HRI expert input combination, to achieve maximum accuracy. They believed that longer sentences in the global context condition would make participants have to listen longer, interrupting them more. Additionally, they thought that participants would not believe the predictions if a robot was asking questions. We test the HRI expert input against our best found combination from the initial shape recognition task to validate that our best combination is better than what would commonly be implemented on robots that ask for help.

The shape recognition validation was conducted as a within-subject design with participants receiving questions both with our best combination of state information and with the HRI expert input combination. Participants were randomly but evenly assigned to the combination of information they would receive first. Subjects were given the same shape recognition task instructions in the initial phase. When they finished building their first four structures in 12 minutes, they were given a questionnaire. Then, they were given a second set of four structures (the order of the groups of structures were randomly and evenly assigned between the two conditions) to complete in 12 minutes while the robot asked questions using the second combination of information. In total, participants performed two 12 minute tasks, filled out a survey after each task, and then filled out a final survey to compare the two question conditions. When they completed the third survey, they were paid and dismissed.

Localization Task Our robot CoBot (Figure 6.2), a real custom-built mobile robot, is capable of autonomous localization and navigation and provides services such as tours of


(a) CoBot - front



(b) CoBot - back

Figure 6.2: CoBot from the front (a) and back (b). Participants walked behind CoBot so that they could see the messages and questions. CoBot spoke the questions through speakers below the laptop.

our computer science building. However, it can be uncertain of its location when using WiFi localization (Biswas and Veloso (2010)). We have shown that if CoBot could ask for localization help from people in the environment as it navigates, it can avoid localization errors and speed navigation time (Rosenthal, Biswas, and Veloso (2010)).

In our localization task, participants were asked to walk around with CoBot while it gave a 15-minute tour of one floor of the building. These participants had never been on this floor of the building and thus could benefit from the tour. Upon arrival, they were randomly assigned to one of five conditions: 1) no information, 2) uncertainty and local+global context, 3) uncertainty and prediction, 4) uncertainty, local+global context, and prediction, and 5) uncertainty, local+global context, prediction, and feature feedback (our best found combination from the shape recognition task). In pretests, we found that local context was not enough for people who had never seen our building before. Additionally, we include uncertainty in four conditions because it has previously (Antifakos, Schwaninger, and Schiele (2004)) been found that users tend to trust agents more when they admit they are uncertain. The other conditions all included context and predictions. Our 5th condition tests our best combination which also includes feature feedback.

The experimenter remote-controlled the robot to each location in the building, triggering information about seven different laboratories, art installations, and views from the windows as it navigated. Participants were told that the robot would not be able to continue the tour if they did not help it. Because the experimenter was standing behind the participant while he/she was following the robot, the participants could not see the experimenter trigger the questions or control the robot and they believed the robot was moving autonomously. During the tour, the experimenter stopped the robot in 13 pre-defined locations to ask participants to indicate the robot's location on a map (Figure 6.3). We used CoBot's uncertainty and predictions from its autonomous navigation to guide our decisions in where we triggered questions during the study. After the participant clicked on the map to indicate their location, the robot would continue navigating. After participants completed the 15-minute tour containing 7 places of interest and 13 questions, they were given a survey about their experiences with the robot. Upon completing the survey, participants were paid and dismissed.

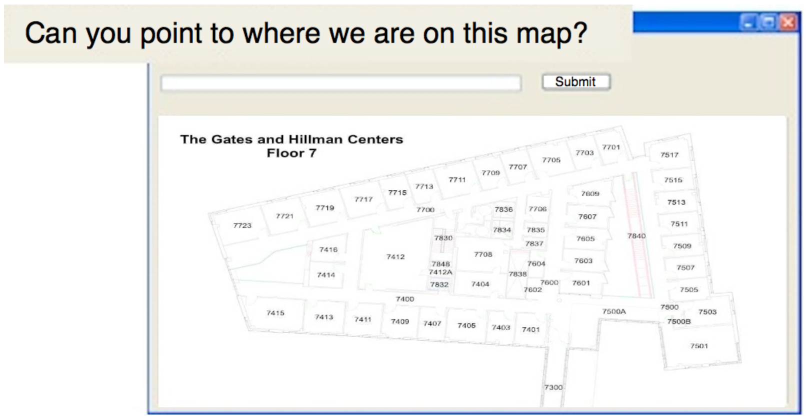

Can you point to where we are on this map?
Submit
The Gates and Hillman Centers
Floor 7

Figure 6.3: CoBot stopped in 13 locations to ask participants to indicate their current location by clicking on the map on the user interface.

6.3.2 Activity Recognition Initial Task

Subjects were told they were testing a new physical activity coach on a handheld device that could detect the different activities they performed (Figure 6.4). The subjects' primary task was to perform each of the 12 physical activities from a list provided (Table 6.4). Subjects were given all equipment required to complete the activities, including a soccer ball, tennis balls, rackets, step stools, and golf clubs.

They were required to carry a Nokia 770 Internet Tablet that would recognize their activities and beep when it had questions. They were to respond to questions on the tablet using a stylus on a virtual keyboard. We randomly pre-selected 8 out of the 12 activities to ask participants about. Questions were sent from the experimenter's computer, 10-20 seconds after each activity was initiated. Subjects were told that they should answer the questions if they felt they had time as their responses would help them in a second task (which was never given). Subjects had 12 minutes to complete as many activities as possible, while answering the agent's questions when they had time.

As with the other initial studies, when the participants arrived they were randomly but evenly assigned to one of the study conditions in a between-subject design. After completing the experiment, they were asked to fill out a survey about their experiences with the activity coach. Upon completion, participants were paid and dismissed.

Activity	Description
Walk	Walk around the room once
Soccer	Dribble a soccer ball around the room once
Steps	Step up and down off a stool 10 times
Tennis	Bounce a tennis ball on a racket 10 times
Golf	Putt golf balls on a mini course 5 times
Hula Hoop	Use a hula hoop 10 times
Read	Sit and read 2 pages of a travel book
Toss Ball	Throw a ball in the air 10 times
Bounce Ball	Bounce a ball on the ground 10 times
Jump	Jump up and down 20 times
Jumping Jacks	Do 10 jumping jacks
Push Objects	Push 5 chairs from table to the wall

Table 6.4: Participants were told that their activity coach could detect 12 tasks.

6.3.3 Activity Recognition Validation

As with our shape recognition task, while the initial phase was run, we sought advice from three members of the HCI ubiquitous computing community about which information they believe the activity recognizer should use when asking for help. The community members understood both the technical data that could be collected and the usability requirements necessary for effective communication to users. We explained each type of information and how the information could be combined together. To achieve maximum accuracy, they suggested that the device should provide *no uncertainty, high-level and local+global context, predictions, and feature feedback*, which we call the HCI expert input combination.

(a) (b)

Figure 6.4: Activity recognition task participants answered questions while performing activities such as tennis.

The activity recognition validation was conducted as a within-subject design with participants receiving questions both with our best combination of state information and with the HCI expert input combination. Participants were randomly but evenly assigned to the combination of information they would receive first. Subjects were given two different sets of activities to perform with the two combinations of information and were asked to fill out a survey after each. In total, participants performed two 12 minute tasks, filled out a survey after each task, and then filled out a final survey to compare the two question conditions. When they completed the third survey, they were paid and dismissed.

6.3.4 Email Filtering Initial Task

In this task, participants were asked to read provided emails about an upcoming academic conference and consolidate all the changes that need to be made to the conference schedule and website (Steinfeld et al. (2006)). They were given a spreadsheet with information about conference speakers, sessions, and talks, and asked to make changes to it based on change requests in the email, in 12 minutes. The emails and task were modified from the RADAR dataset (Steinfeld et al. (2006)). The emails in the data set were labeled with a folder name, which was removed to test the participants. Additionally, we added high-level summaries of the emails and low-level keywords for the agent to use to ask for help.

When subjects arrived, they were randomly but evenly assigned to their combination of information condition (between-subjects design). Subjects were given an email application with the emails and were told that the classifier had sorted most emails into folders based on the type of changes that needed to be made (schedule or website). The email interface was built with Adobe Flex and presented on a 15" Apple MacBook Pro. The participants were asked to try to sort the "Unsorted" emails and answer the questions that popped up automatically when the participant read an email while they were busy updating the spreadsheet with the relevant information. After 12 minutes were up, participants were given a survey about their experiences with the email tool. They were then paid and dismissed.

6.3.5 Email Filtering Validation

We again sought HCI community expert advice while running our initial Email Filtering task. Community members worked on the usability of email systems and understood the technical data and usability requirements of such an email filtering application. Like the activity recognition validation, they suggested that to improve accuracy of responses, an email filtering application should provide *provide uncertainty, low-level extra context, predictions, and do not request user feedback*, which we call the HCI expert input combination.

Like the other validation experiments, participants were randomly but evenly assigned to which of the two conditions (expert input or our guideline) they would receive first. Subjects were given two different sets of activities to perform with the two combinations of information and filled out surveys after completing each set. In total, participants performed two 12 minute tasks, filled out a survey after each task, and then filled out a final survey to compare the two question conditions. When they completed the third survey, they were paid and dismissed.

6.3.6 Interruptibility Validation

The problem of recognizing when someone is interruptible has been widely studied (*e.g.,* Fogarty et al. (2005); Horvitz, Koch, and Apacible (2004)). Specifically it has been shown that strangers are fairly accurate at rating someone elses interruptibility. With crowdsourcing technologies widely available today, we conducted an additional task on Amazon.com's Mechanical Turk, an actual system that is often used to pair label requestors with people willing to label data. The labelers on this website have never seen the applications that collect the data (they are non-supervisors); they only fill out forms online for a small fee.

We recruited subjects from Amazon.com's Mechanical Turk to estimate the interruptibility of office workers from video data previously collected. When the interruptibility

Figure 6.5: Participants were asked to judge whether people were interruptable in their offices.

video data was collected, office workers made the ratings without specifying who the interrupter was. Our dataset included 586 45-second videos from 5 offices at a university that had been labeled with an interruptibility value from 1 (Highly Interruptible) to 5 (Highly Non-Interruptible) by the five office workers themselves. Twelve videos were selected from the data set and put on the Mechanical Turk website, two randomly chosen from each interruptibility level plus two more from randomly chosen levels. Participants on Mechanical Turk were asked to rate the person in each of the 12 videos on the same 1-5 scale (Figure 6.5) using questions with our guideline or HCI expert input for a between-subjects design. Participants were randomly but evenly assigned to which condition they received and were paid upon completion of their survey.

6.3.7 Participants

Pittsburgh residents with ages ranging from 18-61 (mean 27.6, s.d 2.4) with a variety of occupations including students, bartenders, teachers, and salesmen performed the shape recognition, activity recognition, and email filtering tasks, Thirty-seven subjects in the initial phase performed all three tasks and 33 more performed the validations - 11 per task. Forty-two participants were included in the localization study all of whom were graduate or undergraduate students at Carnegie Mellon University who had not spent time in our new computer science building. 180 participants in the Interruptibility task were recruited anonymously on Mechanical Turk, but were only allowed to complete the task once by comparing usernames. Only a few participants (15%) had experience with machine learning technology, and all spoke fluent English.

6.3.8 Measures

Because a robot agent would benefit more from correct answers to questions rather than incorrect ones, we assessed the non-supervisor responses to the questions primarily based on correctness. The responses in the shape recognition task were classified as a binary value: correct or incorrect. The responses in the localization task were measured as the real distance from the true robot location to the location the participants clicked on the map. We also gave surveys to all subjects about their opinions of the robots, asking questions including whether they found the applications to be annoying.

Shape Recognition, Activity Recognition, and Email Filtering: Participants' responses were classified as correct answers if their last answer (some users changed their minds) was correct and incorrect otherwise. For example, if a subject disagreed with the prediction, but gave an equally correct answer, it was classified as correct. Synonyms were determined to be correct as long as they were not too vague. For example, "rectangle" was considered a synonym to "rectangular prism" but "square" and "cylinder" were not.

We also analyzed the amount of feature feedback that was given for those conditions. If non-supervisors can provide accurate labels for their data, their ability to give quality, or helpful, feedback is of particular interest to possibly speed learning tasks (Raghavan, Madani, and Jones (2006b)). If participants received a request for feedback, their response was coded based on how many features about the data were provided. A value of 0 was given to a response that provided no additional information (e.g., "I don't know"). For every piece of valid information, the value increased by 1. For example, "I'm doing jumping jacks if my arms move up and down and my legs go in and out", would be given a value of 2.

Localization Task: Participants clicked on a map displayed on the robot's screen to indicate their current location, and the (x, y) locations of their clicks were logged in order to determine the Euclidean distance to the actual robot location. These mouse clicks could be used directly by the robot by translating the pixel coordinates into (x, y) coordinates in the building, making it an ideal way to ask for help. Each pixel is equal to about 4 inches and a hallway in the building is 15 pixels across. The mouse clicks were deliberate as participants often considered which pixel to press within a 1-2 pixel granularity. We recognize Euclidean distance does not distinguish incorrect hallways or inside offices as worse than a click in the appropriate hallways. However, our data indicates that when these errors occur, they are at large distances from the true location anyway.

Interruptibility Validation: Participants in this study gave a numerical value from 1 to 5 for their interruptibility predictions. We analyzed the mean squared error (MSE) of the given response to the true label for that data.

$$MSE = (\text{true} - \text{predicted})^2 \tag{6.1}$$

Although we do not expect participants to necessarily give accurate predictions, we expect them to be close to the true interruptibility. This schema more heavily penalizes predictions that were further from the true interruptibility.

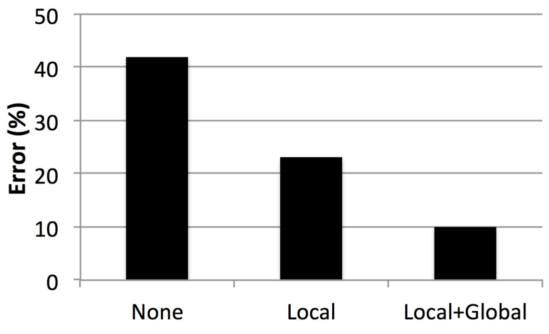
Surveys: After completing any of the tasks, participants were given questionnaires on their subjective experiences with each technology. They were asked about whether they thought the robot's questions were annoying and whether they found each dimension particularly useful. Responses were coded as either "Yes" or "No" because participants were not exposed to their combination of information many times and we did not believe that using a Likert scale to understand information preferences would produce valid results. Participants were also asked whether it was easy or hard to answer the questions on a Likert scale from 1 (very easy) to 5 (very hard).

6.4 Experiment Results

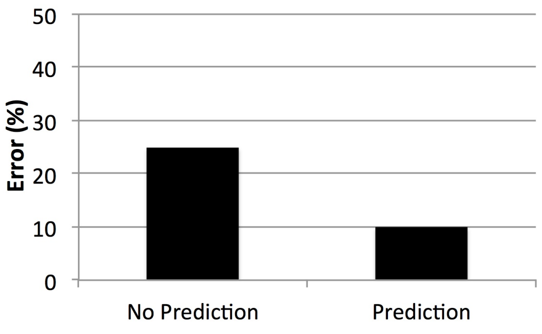
We analyze the results of our studies to determine the combination of the four types of information that results in the most accurate responses. We find the same combination results in the highest accuracy in both robot domains while a different best combination results in the highest accuracy in the other 3 domains. We will compare our results with the HRI and HCI expert input to show that our combinations improves accuracy a statistically significant amount.

6.4.1 Shape Recognition Initial Task

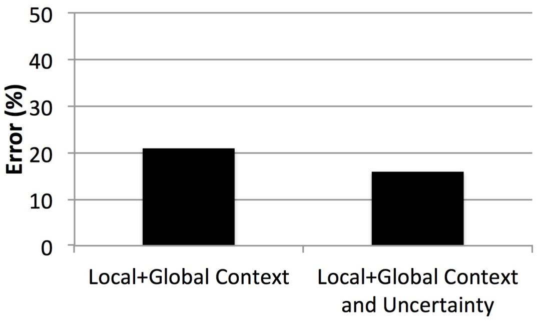
The robot asked all subjects at least 5 out of the 8 possible questions, due to some subjects running out of time. There was no significant difference in the number of questions answered for any particular combination of state information. Six percent of the questions that were asked were ignored due to the primary task. Seven participants skipped at least one question with two participants accounting for nearly half of the skipped questions. Of the answered questions, participants had an average error rate of 16.4% (s.d. 25%). This



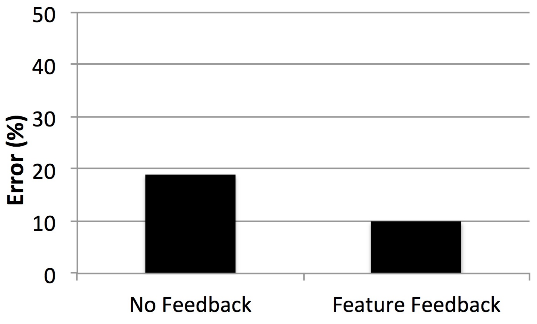
(a) Context



(b) Prediction



(c) Uncertainty



(d) Feature Feedback

Figure 6.6: Shape recognition initial task results. (a) The more context the robot provides, the higher the accuracy of the participants' responses. (b) When the robot includes a prediction, the participants answer more accurately. (c) When the robot provides at least local context, the accuracy increases when the participant also receives uncertainty information. (d) When the robot asks for feature feedback, the participants answer more accurately.

high standard deviation indicates that many (15) participants answered all questions correctly while several had very high error. We performed an ANOVA with the F statistic to test for ordering effects of whether the question number affected the participant accuracy. We found that there was no order effect and the accuracy did not change over the eight questions ($F[7, 170] = 1.70, p > 0.05$). McNemar tests with the χ^2 statistic were used to analyze the significance of the categorical response (correctness) against the categorical independent variables (our four types of information).

We analyzed the effects of each individual type of information on the proportion of correct answers the robot received. Figure 6.6(a), 6.6(b), 6.6(c), 6.6(d) show the percentage of questions that were incorrectly answered for context and predictions, uncertainty, and feature feedback, respectively. Subjects made statistically significantly fewer errors as they were given more context, dropping from 42% (none) to 23% (local) to 10% (local+global) ($\chi^2[2,2] = 8.61, p < 0.02$). Subjects made significantly fewer errors when they received predictions (10%) compared to when they did not (25%) ($\chi^2[1,1] = 3.59, p < 0.05$) and made fewer errors when asked about feature feedback (10%) compared to when they were not (19%) ($\chi^2[1,1] = 4.05, p < 0.05$). There were no significant effects of uncertainty alone, but we found a significant paired effect of uncertainty and context reducing the error from 21% to 16% with local+global context and no significant difference in error without context ($\chi^2[2,2] = 5.98, p < 0.05$). There were no other significant effects. Overall, we find that providing all four types of state information - local+global context, prediction, uncertainty, and feature feedback - increases the accuracy of non-supervisor responses. We will refer to this combination of information as our guideline for robots to ask non-supervisors for help and compare it to the HRI expert input next.

Subjects did not find any combination of dimensions more annoying than the others. Of the participants who received feature feedback, predictions, uncertainty or contextual information (local and local+global), 35%, 64%, 37% and 71%, respectively, found them to be useful.

6.4.2 Shape Recognition Validation

We compared the responses of participants in a within-subject design when the robots asked questions with the HRI expert input (local context, uncertainty, and feature feedback) to our guideline (local+global context, prediction, uncertainty, and feature feedback). T-tests were used to analyze the significance of the categorical response (correctness) against the two combinations of information (expert input and our guideline). There was no significant effect in the ordering of the conditions ($t[186] = 0.00, p > 0.05$). Figure 6.7 shows the percent of questions subjects answered incorrectly for each condition. There are significant effects of the combination on the proportion of correct answers subjects gave. Subjects provide significantly more correct answers (2.22% error) to the robot's questions when using our guideline compared to the expert input (15.63%) ($t[186] = 10.05, p < .01$).

Participants were asked whether they thought each kind of information was useful in helping them to answer the robot's questions. Subjects only scored the two systems differently for the contextual information dimension. While six participants gave our guideline combination (with local+global context) a score of 5 (very useful) for contextual information, only two participants gave the HRI expert combination (with local context) the

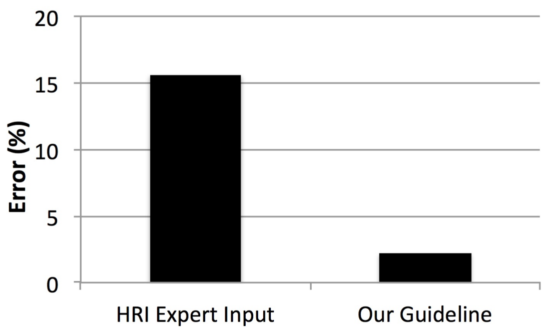


Figure 6.7: Shape recognition validation participants made significantly fewer errors when the robot provided our guideline combination compared to the combination determined by HRI expert input.

same score. However, a t-test shows no statistical difference between local context (3.46 average score) and local+global context (4.15) ($t[13] = 1.39, p > 0.05$). Participants rated our prediction on average 3.69 which is more positive than neutral, but we could not compare this to the expert condition which did not receive predictions. Subjects rated our uncertainty and the expert uncertainty (which were the same), 2.67 and 2.77 respectively ($t[13] = 0.18, p > 0.05$). Similarly, participants rated the feature feedback (which were the same) identically at 2.66 ($t[13] = 0.0, p > 0.05$).

Subjects were given another survey at the end of the experiment asking which system they preferred, which they thought was smarter, and which learned more. On all three survey questions, our guideline scored higher. Twelve out of fourteen respondents preferred our guideline over the expert input, eleven thought ours was smarter, and ten reported they thought ours learned more.

6.4.3 Localization Task

We collected the clicks on the map for each participant and calculated the Euclidean distance from the clicks to the actual robot location. Because the distribution of these distances was skewed, we performed a log transformation to normalize the data. We, then, analyzed the results of the localization test of log distances with a mixed model with participant ID as a random effect and the question condition as a fixed effect analyzed using the F statistic. Our results show there are statistically significant differences between the five conditions ($F[4, 38.53] = 3.93, p < 0.001$). We used contrasts to analyze whether there were statistically significant differences between our guideline (condition 5) from

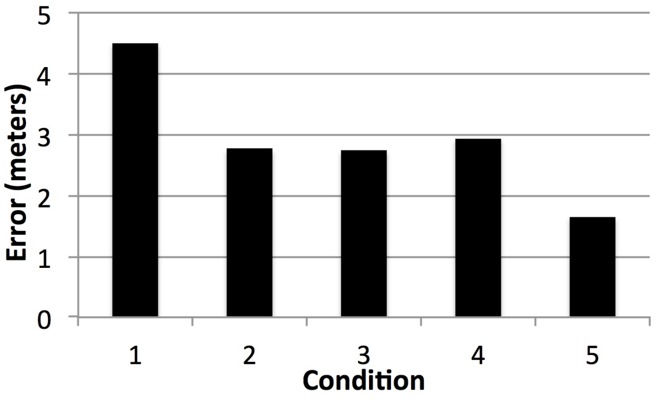


Figure 6.8: The localization task had 5 conditions: 1) no state information, 2) uncertainty and local+global context, 3) uncertainty and prediction, 4) uncertainty, local+global context, and prediction, and 5) (our guideline) uncertainty, local+global context, prediction, and feature feedback. Participants who received our guideline combination of information responded with the least error.

the shape recognition study and the other four conditions tested. Running 4 contrasts means that statistical significance is determined at the level of $p < .05/4$.

Although we analyzed the log distances, we report the true distances in meters for clarity (Figure 6.8). Participants who received no state information clicked further away from the robot's true location (4.5 meters) compared to those who received our guideline (1.65 meters) ($F[1, 38.45] = 22.17, p < 0.001$). Participants who received only uncertainty and local+global context or uncertainty and predictions clicked 2.76 and 2.74 meters respectively from the true location, a marginally significant difference ($F[1, 38.9] = 3.18, p = 0.082$) ($F[1, 38.7] = 3.78, p = 0.059$). While our guideline shows a 1 meter improvement to these two conditions, there was a larger range of click distances for these conditions leading to only marginal significance. Finally, participants who received local+global context, uncertainty, and prediction clicked significantly further from the true location (2.94 meters) than those with our guideline ($F[1, 37.6] = 8.17, p < 0.001$).

6.4.4 Activity Recognition Initial Task

We collected 119 responses from participants, including 8 for which participants (6 of them) said they were too busy to respond. When we analyzed the remaining 111 responses for the effects of the individual dimensions on the proportion of errors participants made, we found that subjects were correct nearly 100% of the time and there was no effect of any of the dimensions or their combinations. However, we found that when an intelligent system requests feature feedback, subjects were able to provide on average of .81 pieces of quality feedback compared to almost 0 pieces without being asked (some subjects provided feedback without prompting). We include feature feedback in our best combination, as this is a statistically significant difference ($F[6, 112] = 8.87, p < 0.001$).

We then used the McNemar test on the amount of feedback with all five dimensions as independent variables to analyze the significance. We find that subjects who received local context provide a significantly larger amount of quality feedback (.77 pieces) compared to those provided either no context (.30 pieces) or local+global context (.31 pieces) ($F[2, 2] = 5.38, p < .002$). Additionally, subjects who received low-level context provided statistically significantly greater amount of feedback (.58 pieces) compared to high-level context (.34 pieces) ($F[1, 1] = 3.33p < 0.05$). There were no significant effects and no combined effects for providing predictions or uncertainty so we use qualitative results to understand the impact of those dimensions.

We find that 25% of subjects who did not receive predictions reported it hard or very hard to answer the questions. Additionally, 0% of subjects with predictions reported the task difficult and 83% thought the questions were useful. There were no effects of uncertainty on the qualitative data so we do not include it in our best combination. Based on these results, we determine that the best combination for a non-supervisor labeling their own data is the following: no uncertainty, do provide sufficient low-level context, predictions, and request feature feedback.

6.4.5 Activity Recognition Validation

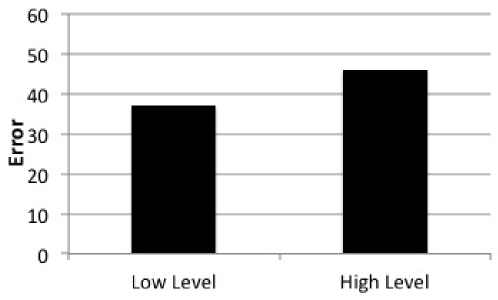
We validate our best combination against the HCI community advice: do not explain uncertainty, but provide high-level and extra context, predictions, and request feature feedback. We collected 113 responses from participants including 11 non-responses. Four participants were too busy to respond at least once. We found that for both conditions, subjects gave correct responses 100% of the time and there were no statistically significant effects on feedback quality, so we use the qualitative results to differentiate the conditions. Subjects found that our dimensions were useful but only 30% realized they were receiving contextual information. Subjects did not prefer either system and could not identify which one learned more, but 70% of participants thought the system using our guidelines was smarter. Participants that believe a computer is smarter will respond with more sophistication than to one they think is not as smart (Pearson et al. (2006)). So, we conclude that our combination is at least as good as, if not better than, other combinations of the information.

6.4.6 Email Filtering Initial Task

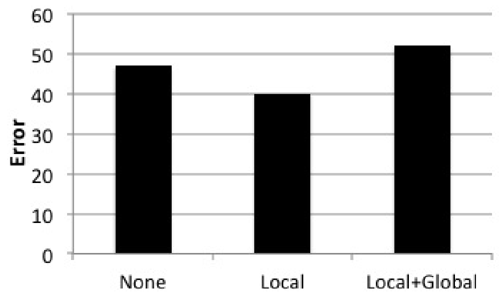
We collected 153 responses from participants including 13 non-responses. Four participants answered that they were busy at least once. We first analyzed the effects of each individual type of information on the proportion of errors. Subjects answered a statistically significant larger proportion of questions incorrectly when given high-level context (46%) versus low-level context (37%) ($\chi^2[2,2] = 10.57, p < .01$) (Figure 6.9(a)). Subjects had significantly higher error rate when they received no context (47%) or extra context (52%) compared to subjects who received sufficient context (40%) and this effect is heightened when combined with the level of context ($\chi^2[4,2] = 11.04, p < .01$) (Figure 6.9(b)). No other single type of information was significant.

To understand how the other three dimensions affected participant performance, we analyzed the effects of pairing them with the significant dimension and each other. Subjects provided statistically significantly more errors when they did not receive a prediction with local context (50%) compared to when they did (22%) ($\chi^2[2,2] = 7.72p < .01$) (Figure 6.9(c)). We found that if we provide local context, providing uncertainty decreases error from 54% to 30% ($\chi^2[4,4] = 11.56p < .01$). There is a significant paired effect of prediction with uncertainty ($\chi^2[2,2] = 8.70p < .01$). Finally, we found that requesting feature feedback resulted in a decrease from 70% to 10% in errors when paired with uncertainty but an increase in error from 13% to 55% when no uncertainty information is provided ($\chi^2[2,2] = 12.21p < .02$).

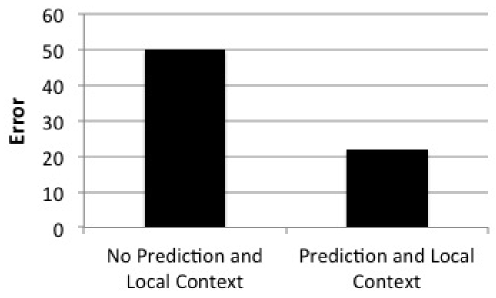
We analyzed the survey responses to understand how useful subjects felt each dimension was. We found that 50% of subjects thought the questions were useful to them during their task while 41% found answering them annoying. A majority of subjects who saw each dimension thought they were useful. 90% of subjects found context useful when they received at least sufficient context, and 100% of subjects who received predictions found them useful. 78% and 71% of subjects who were asked for feedback and who received uncertainty respectively, found it useful. We conclude that the intelligent system should use the following combination when asking non-supervisors to label other data: provide uncertainty, sufficient low-level context, predictions, and request feature feedback.



(a) Level of Context



(b) Amount of Context



(c) Prediction

Figure 6.9: Email filtering initial task results. (a) Low level context improved accuracy more than high level context. (b) Participants answered most accurately when given local context. (c) When paired with local context, participants answered more accurately when given a prediction.

6.4.7 Email Filtering Validation

HCI researchers that work in the email domain came to the following consensus on our dimensions: provide uncertainty, low-level extra context, predictions, and do not request feature feedback. We collected 301 responses including 4 non-responses. Three participants refused to respond at least once. We found a significant effect of the combination on the proportion of correct responses ($t[2, 250] = 2.48, p < .01$). Subjects who received our combination were 100% correct, while those who received the community advice were 94% correct. A majority (8/11) people preferred the community advice but (7/11) people thought our intelligent system was learning more. When we analyze the dimensions that differed between combinations, more people preferred our context (58% vs. 40%) and predictions (63% vs. 40%).

6.4.8 Interruptibility Validation

Participants in this study were required to answer all 12 questions. Half of our 180 participants estimated the interruptibility for 12 videos with the best combination from the email task and half received the email communitys advice. We analyzed the average mean squared error (MSE) of each participants estimation compared to the true interruptibility across the videos and performed a between-subjects ANOVA analysis to compare the error between the combinations. We removed 16 of the 180 subjects that had MSE results that were more than 3 times the median of the entire data set (average MSE=1.37, outliers ¿ 4.11). Subjects who received our combination had a statistically significant lower average MSE (mean 1.17, std. dev. 0.62) than those who received the community advice (mean 1.42, std. dev. 0.92) ($F[1, 164] = 6.02, p < 0.01$). Subjects who received our combination were correct or off by one level of interruptibility 85% of the time, while subjects that received the community advice were correct 80% of the time. Both of these are better than the previously published interruptibility result, reporting a 65% off-by-one accuracy with only sufficient context.

6.5 Discussion

Our results show that we were able to find two combinations of information for intelligent systems to provide non-supervisors to increase the accuracy of their responses - one for robots to provide and one (differing in only one dimension) for other systems such as mobile phones or email applications. Additionally, we were able to validate these combinations of information against HCI and HRI expert input. Next, we discuss the impact of each kind of information on the human as well as the impact to intelligent systems of providing this information.

6.5.1 Benefits of Robot State Information

Interestingly, the combination of information that resulted in the most accurate responses for the non-supervisors is the same as the combination found useful for supervisors even though all the information was not found on supervisors' interfaces (Yanco, Drury, and Scholtz (2004)). We found that participants in the shape recognition study rated the context and the prediction as useful to helping them respond, while supervisors similarly include the two types of information in their interfaces. Additionally, while neither our participants nor the supervisors rated the uncertainty and feature feedback as not useful, both types of information were found to increase the accuracy of responses. Contrary to HRI expert intuition about which information increases the accuracy of responses, our result shows that all types of information have an impact on the non-supervisor. However, we believe that a robot will not need to provide all information to supervisors, shortening the length of each required question, because they implicitly know the uncertainty and feature feedback information.

Context and Prediction: The supervisors and non-supervisors use contextual information and prediction to focus their attention on what the robot is asking about. In the shape recognition task especially, where the participants were not shown the camera view of the robot, the full (local+global) description of where the robot was looking was useful to help participants find the block in question. Although the prediction was always correct, participants often did not trust it. The contextual information was used by the non-supervisors to check that the robot's prediction was consistent with the context it was providing. Additionally, the subjects' high rating of the predictions indicates that they listened to the predictions despite being busy with their primary task. A robot with less accurate predictions would need to focus more on providing contextual information to help people determine the accuracy of the prediction.

Uncertainty: Although non-supervisors were frequently interrupted with questions in their structure building task and in the tour, they almost always answered the questions when it was prefaced with uncertainty information. Although these interruptions slowed them down, when the robot said explicitly that it was uncertain, the participants felt they should answer the question. We found a significant interaction of uncertainty and context in our analysis marked by improvement in accuracy with high levels of context, which confirms previous findings that users tend to trust or rely on systems more when the system displays uncertainty information (Antifakos, Schwaninger, and Schiele (2004)). However, when participants were asked whether they valued uncertainty, they did not remember if they had received the uncertainty information and did not report it as useful. We believe that participants underestimated how much they were using the uncertainty in their predictions.

Feature feedback: When participants were asked to provide feature feedback about their response, they sometimes changed their labels to the correct answer when they thought about why they chose the particular label. While it may be difficult for a system to incorporate such freeform feedback as we allowed, it has been shown that feature feedback can improve classifier accuracy (Raghavan, Madani, and Jones (2006a)). Additionally, and perhaps more importantly, we have shown that the robot will benefit from increased response accuracy just by asking the question and irrespective of using the response.

6.5.2 Benefits of State Information for Other Systems

We found that each of the five dimensions amounts of context, level of context, uncertainty, prediction, and requests for user feedback had a positive effect on the non-supervisors of other intelligent systems as well. In particular, the best combinations of information for the activity recognition and email filtering tasks were nearly the same only differing by uncertainty.

Context and Prediction First, participants used the context and prediction to match the intelligent systems focus. For example, many participants used the key words and summaries of the emails when deciding on a label instead of reading the entire email. As a result, the questions did not take as long for participants to answer compared when they had to pick out the important context themselves. Additionally, participants checked to confirm their label was consistent with the given context.

We had assumed that because the email labelers did not know the context the data was drawn from, the intelligent email filter would need to provide extra context to maximize accuracy compared to users of activity recognizers that knew their own context as they were acting. However, because participants had some domain knowledge for sorting emails and determining if someone is interruptible, they did not need as much context to be accurate. We also found that just as participants did not require high-level context about their own activities, they did not require high-level context in the email or interruptibility tasks because the raw data (email keywords and video clips) were already human-understandable. This is significant because it reduces computation time for constructing questions, and eliminates the need to translate low-level sensor data into high-level context, allowing more time for processing data.

Uncertainty The only difference between the email filtering and activity recognition task combinations is uncertainty. Uncertainty offers no help to the labeler but indicates that the classification is hard. Participants were aware of the difficulty of activity recognition without the acknowledgement from the system, reporting that they were impressed that a mobile device was able to recognize their activities. Receiving uncertainty did not change their opinion about the recognizer and there were no significant changes in accuracy as a result. However, participants in the email filtering and intelligibility tasks saw human-understandable data and assumed the classification was, in fact, easy. When these participants received uncertainty, we believe they recognized the difficulty of the task and tried harder, resulting in higher accuracy responses. In general, non-supervisors that realize the classification is hard do not require uncertainty information.

Furthermore, although participants were frequently interrupted with questions in their 12-minute tasks, they almost always answered when it was prefaced with uncertainty. For example, in the activity recognizer task, one participant who was interrupted only seconds after starting a task said, "Its interrupting me again! Oh, well, I guess it must be hard to distinguish between these [activities]." This shows that participants excused the interruption when they felt they could help the intelligent system. However, when we asked participants whether they valued uncertainty, they did not remember if they had received that information and therefore reported it as being not useful. We believe participants underestimated the usefulness of uncertainty for the usability of the questions.

Feature Feedback Finally, when participants were asked to provide feature feedback about the label, they sometimes changed their labels to the correct answer when they thought about a reason for the label. While it may be difficult for a system to incorporate such freeform feature feedback, we find that the intelligent system will benefit from increased response accuracy just by asking the question and irrespective of using the response. To make it easier to use such feedback, the intelligent system could ask a multiple-choice question. Overall, we found each piece of information was useful for participants.

6.5.3 Errors

We were initially surprised by the number of errors that participants made in the domains, particularly the robot ones. However, upon further examination of the data, we found that there were two main causes of errors among our participants in each task. In the shape recognition, the robot asked each question about the block the participants were currently holding in their hands. The participants often picked up multiple blocks at a time, causing mix-ups in shape when they were not paying attention. Additionally, participants continued building while the robot asked for help. If they put down their block and picked up another one of the same color, sometimes they would respond with the shape of the latter block although the robot started asking earlier. Similarly, in the activity recognition task, participants continued performing their activities even after the device asked them for help, causing errors if they completed the task and moved to the next one before answering the question. While these two problems could have been solved by requiring the participant to stop what they were doing to listen to the intelligent system, it is unlikely that non-supervisors would stop what they were doing in real world situations. We believe, therefore, that our shape and activity recognition results reflect real world scenarios at the cost of increased numbers of errors in many conditions. Our shape recognition validation results show that using our guidelines, the error rate drops to 2% even when participants do continue working during the question.

In the localization study, the robot stopped moving to allow the participants to click on its attached laptop. Realistically, an error of 3 meters or more, as we received in all conditions except our best found combination, would not resolve the robot location uncertainty around an intersection to know whether to turn now or continue straight for another meter before turning. The two main causes of error in the localization task were due to 1) lack of knowledge of the building and 2) misunderstanding the robot's question. When participants did not know the building, they often found it hard to read the map even with every room labeled. Participants would often click on the correct corridor of rooms but did not focus their clicks close to specific room they were nearest to, resulting in clicks further down the hall away from the true robot location. Additionally, participants who found the room sometimes would click on the room itself rather than their location in the hallway. While this is an interesting response, it would result in large localization errors on a real robot.

Both of these errors were greatly reduced using our combinations of information. The participants in our guideline condition in the localization task had an average error of 1.65 meters, roughly the width of the hallway. While the robot can account for such error in its sensors, responses with larger errors would be difficult to use because the questions often occurred near hallway intersections when the robot is uncertain of whether to turn yet. The predictions on the map indicated to the participant to click in the hall instead of in a room. Most importantly, the feature feedback question resulted in participants looking around at room numbers more than in other conditions. This heightened awareness likely impacted the responses the most in our guideline condition.

6.5.4 Computation Required to Calculate State Information

As we are motivated by task-driven robots that interact with people in the environment, we acknowledge that computation is limited on these robots and generating these questions may sometimes not be possible. We aimed to use information that was largely already calculated or known by the robot in order to reduce the computational requirements of asking for help. However, with limited resources, we found that a robot can increase the accuracy of its responses most with the least computation by providing at least local context, and also providing uncertainty and asking for feature feedback. We have found the most significant increases in accuracy when adding additional context, and suggest maintaining at least local context when asking for help. When CoBot provided context, participants' errors dropped from 4.5 meters to 3 meters to the robot's true location. Providing uncertainty information and asking for feature feedback both increase accuracy without having to generate any new information. The feature feedback, in particular, requires that the non-supervisor be more alert of the robot and the environment and results in more accurate responses, dropping localization error significantly from 3 meters to less than 2 meters.

6.6 Chapter Summary

People in the environment can help intelligent devices reduce their need for supervision. However, they may not understand the device's questions and may not answer accurately. We have demonstrated in 5 different domains, including several validation experiments, that devices can improve accuracy by changing the way they ask for help.

We first contribute a novel two-part study design to evaluate which combination of the four types of information results in the most accurate non-supervisor responses. In the design, participants are asked to perform a task to limit their ability to supervise their intelligent systems. As an intelligent system requests help on its own task, participants determine whether it is worth interrupting their task to respond. The intelligent system is wizard-of-oz'd to ensure that the only difference between study conditions is the information provided; each participant receives the same questions at the same time during their task. After an initial test of many different combinations of information, our design includes a validation study to explicitly test our best found combination against a combination that HRI and HCI experts believed would result in the most accurate responses.

Our second contribution is the results of 5 experiments based on this study design. Two experiments are on robots – a shape recognition task and a localization task. In the shape recognition task, a wizard-of-oz'd toy robot asks participants to identify the shapes of blocks they are manipulating. In the localization task, a remote-controlled (wizard-of-oz'd (Green and Wei-Haas (1985))) real robot asks participants to identify their current location while giving a tour of the building. We find that providing all four kinds of information together - *context, prediction, uncertainty, and feature feedback* - most improved the accuracy of participants' responses in both studies (Rosenthal, Dey, and Veloso (2009); Rosenthal, Veloso, and Dey (2012a)). The last three experiments are on smart phone activity recognition, email filtering, and interruptibility. Interestingly, we find that a different set of information from the robot studies results in the highest accuracy for all three of these experiments (Rosenthal and Dey (2010)).

While this result may seem obvious, our group of HRI and HCI researchers predicted that different combinations would result in more accurate responses compared to our findings. In a direct comparison with their selected combination, we validated that our best found combinations provides a statistically significant improvement in accuracy. We thirdly contribute our combinations as guidelines, validated in five domains and against HRI and HCI expert input, that can be used in new domains to increase non-supervisor accuracy. Because these combinations of information have been validated many times, we argue that they can be used directly in new applications using our operational definitions of each type of information.

Chapter 7

Learning Models of Humans

We have shown that there are many important human state attributes to model in human-centered algorithms when determining whether, how, and who to ask for help. However, we have not yet contributed algorithms for learning these attributes and the corresponding reward functions that the human-centered algorithms utilize.

In this chapter, we contribute three algorithms to learn these attributes and rewards. Our first algorithm uses linear regression over Likert scale survey ratings from phone users to determine their cost of help and incentive to help (details in Chapter 5). The resulting predictive models were used in the human-centered volume learning algorithm (details Chapter 2). While users indicated that the algorithm made accurate decisions about whether to ask for help, the survey was very long and could not be deployed to many people. The longer that the surveys become, the more likely the participants will abandon the survey leaving no usable results (Heberlein and Baumgartner (1978)).

Second, we contribute an algorithm to learn reward functions in which the reward is a linear function of several subrewards such as human state attributes. We call these reward functions multi-attribute additive reward functions. Our goal for the algorithm is to elicit rewards and preferences from device users and environment occupants in a "usable" way that asks easy questions for people to answer and does not require too many responses. In particular, our algorithm asks comparison questions between concrete scenarios, which is easier and more accurate for humans to do than provide arbitrary Likert ratings.

Finally, because it can be difficult to measure human state attributes such as availability and accuracy prior to deploying the device, we contribute an algorithm that learns availability and accuracy while executing the human-centered HOP-POMDP navigation policy presented in Chapter 3. We demonstrate that the algorithm converges to the true availability and accuracy of environment occupants while only recomputing the optimal POMDP policy when the approximated values change significantly from the current policy.

7.1 Learning Models of Phone Users

In Chapter 5, we showed that phone users have costs and incentives to help their phones learn their volume preferences. The phone needs to learn the costs and incentives for different situations that the user might encounter in order for the volume learning application to determine whether or not to ask. To collect costs and incentives, we asked participants to fill out our survey. We contribute our method for using their responses to learn the personalized cost and incentive models which were inputted into our human-centered phone volume learning application.

7.1.1 Algorithm

As phone users filled out surveys, we created artificial but plausible sensor values for each of the features on the phone for each situation in the surveys (*e.g.,* driving, in meetings). In order to do this, we captured sensor values as we ourselves were in each situation. For example, while driving, we measured example speeds measured from the GPS, accelerometer vibrations, and microphone noise levels.

Then for each user, we used our artificial sensor values to train a linear regression (easily computable on a phone) with the users' surveyed Likert ratings for their cost of help and incentive to help. For example, we used the Likert value for the cost of asking while driving as the label for the artificial sensor values of driving. The linear regression model was then loaded onto the user's phone for use by the human-centered learning algorithm.

For any new sensor values collected while the user is using the phone, the linear regression will predict the user's cost and incentive to help. By comparing these two values, our algorithm can determine if it should ask for the user's phone volume preferences. It is important to note that because we used only a linear combination of the features in Table **??** and did not use complex features, our cost approximations are easy to calculate on phones but may not always be predictive for all users. We aimed to overestimate the cost of help for each user in order to minimize the number of questions asked.

7.1.2 Results

Our linear regression models varied in their ability to capture each participants predicted asking costs, as measured by the R^2 test, but overall was successful for such a simplistic model. Some of our linear regressions had R^2 values near 1, indicating that the linear regression almost completely modeled the features that determine user's cost or incentive. Others were only about 0.3 (mean 0.65, s.d. 0.15), indicating that there were other features that could be added to the linear regression to better model the cost and incentive.

Our results of the human-centered learning algorithm that used these linear regressions, however, show that the asking is usable and our learned models were successful in trading off the costs and incentives. However, our surveys were too long to be deployed to many users.

7.2 Active Learning of Additive Reward Functions

We have shown that surveys can be effective in approximating human cost and incentive to help. However, research on how to write surveys shows that the order the questions are asked influences the numerical value they are given (Schwarz and Hippler (1995)) and if the questions are multiple choice the scale of the choices matters for the answers they will provide (Schwarz et al. (1991)). Additionally, the length of our phone survey is prohibitive to deployment, as people typically are not willing to fill out long surveys (Heberlein and Baumgartner (1978)).

In this section, we are interested in using active learning techniques to reduce the length of surveys. Now, rather than maintaining one cost and one incentive function, we combine the two values into a single multi-attribute reward function. We learn these multi-attribute additive reward functions as a linear combination of the the human state attributes while minimizing the number of survey questions that humans are asked.

7.2.1 Problem Definition

Let a state of a human s be factored into features $\langle s_1, s_2, \ldots, s_i, \ldots s_n \rangle$ that reflect the tradeoffs (*e.g.,* time, distance, interruption, etc) that humans are making to determine their action a (*e.g.,* to help a device or not). The additive reward function of a state,action pair

$$R(s,a) = \sum_i \lambda_i r_i(s_i, a) \tag{7.1}$$

is the sum of scaled (by λ_i) subrewards $r_i(s_i, a)$ computed over the state's features. Without loss of generality, we require the values of each subreward to be normalized $r_i(s_i, a) \in [0, 1]$, where values towards 1 represent higher reward. Additionally, let a concrete state,action pair *scenario* with specific feature values be denoted σ where $\sigma.s$ and $\sigma.a$ refer to the scenario's state and action respectively. The goal is to learn the personalized reward scale values of λ_i subject to the constraints that $\forall i, 0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$ from humans.[1]

While there have been several very successful algorithms for learning these reward functions from humans, they are in general not feasible to deploy to real users. Inverse Reinforcement Learning learns a reward function based on demonstrations of optimal behavior from a human (Ng and Russell (2000)). The algorithm hinges on the fact that because the human's actions are optimal, the reward for taking the action a from the observed state s must be greater than any other possible action $a' \in A$ from the same state:

$$\forall a' \in A, R(s, a) > R(s, a') \tag{7.2}$$

The set of inequalities resulting from the observed behavior can be solved using linear programming to find a solution to the reward function that is optimal for the given set of demonstrations. This algorithm was extended to problems in which each state is factored into features (multi-attribute) and the reward function is learned over features as our problem statement requires (Abbeel and Ng (2004)). However, while experts can easily demonstrate optimal states and actions, it may be difficult for real users to 1) perform actions optimally (Argall, Browning, and Veloso (2008)) and 2) know which demonstrations will best teach the learner leading to the need for many demonstrations (Eagle and Leiter (1964)).

In order to overcome the errors in demonstration, other work has focused on the learner (algorithm) proactively asking the human which of two probabilistic scenarios is preferred in an active learning style (Regan and Boutilier (2011)). The algorithm asks questions of the form "Would you prefer action a in 1) a concrete state or 2) with probability in the best or worst state. This algorithm has been shown to ask relatively few questions while learning λs (Regan and Boutilier (2011)) which is important for real people as survey response rate has been shown to decline as the number of questions increase (*e.g.,*Heberlein and Baumgartner (1978)). However, it has been shown that people often overestimate the probability that good things will happen (Weiten (2011)) and as a result, the values of λ may be skewed.

[1]Because of the subreward normalization, $\forall \lambda_i \geq 0$. Proof by contradiction. If there was a $\lambda_i < 0$, the higher the subreward the lower the product $\lambda_i * r_i(s_i, a)$. This contradicts the requirement that feature values near 1 are better. The subreward function could instead be negated (and renormalized) so that λ_i is positive. Additionally, without the constraint $\sum_i \lambda_i = 1$, there are an infinite number of valid scaling factors (*e.g.,* for numFeatures = 2, $\lambda_1 = \lambda_2 = 0.5$ is equivalent to $\lambda_1 = \lambda_2 = 1.0$).

Next, we describe our algorithm to learn additive reward functions that asks few questions but also asks questions that are less susceptible to human error by asking about two concrete states rather than probabilistic ones.

7.2.2 LearnImportance Algorithm

We contribute the LearnImportance algorithm that asks people for their preferences between two scenarios σ_1 and σ_2 with concrete values for each feature s_i

$$\sum_i \lambda_i * r_i(\sigma_1.s, a) \overset{?}{>} \sum_i \lambda_i * r_i(\sigma_2.s, a) \tag{7.3}$$

For example, it could ask about human preferences of helping or not helping in different scenarios. We use a Monte Carlo method to determine the pairs of scenarios to ask about by generating a set of hypothesis λs and choosing a pair that best divides the hypothesis space in half. By dividing the space in half, our algorithm ensures it asks relatively few questions (log in the number of hypotheses). The last hypothesis best approximates the person's preferred λ scaling factors with the error rate decreasing as the number of hypotheses increases. Additionally, because preferences form linear inequalities, it is possible to solve the linear program to approximate λ like IRL does.

Our LearnImportance algorithm is outlined in Algorithm 3. We instantiate the algorithm with a number of features *numFeatures* and a set of concrete state,action *scenarios* that would make sense to explain to people. Given the number of features, the algorithm generates a set of hypotheses h for the possible values of λ subject to the constraints defined above (Line 1, Figure 1a). Our algorithm also instantiates the list of user *preferences* to the constraint that $\sum_i \lambda_i = 1$.

While the number of valid hypotheses h is still greater than 1, the algorithm iterates over all pairs of *scenarios* to find the pair which divide the hypothesis space most evenly (Line 4, Figure 1b). We use a Monte Carlo method to find the best pairs of scenario rather than computing the true area of the hypothesis space. When the user responds with their preference of either σ_1 or σ_2 (Line 5), the algorithm generates the correct preference inequality (Lines 6-10), adds that preference to the list of preferences (Line 11), and then iterates through h to remove the hypotheses that are invalidated by the new preference (Line 12) (Figure 1c). It repeats the process until it narrows down the hypothesis space to a single hypothesis (Figure 1d). Then, it finds a solution λ (Line 14).

We next discuss the details of each function in turn.

Algorithm 3 LearnImportance(numFeatures, scenarios)

```
1:  $h \leftarrow$ GenerateMonteCarloHypotheses(numFeatures)
2:  preferences $\leftarrow \sum_i \lambda_i = 1$
3:  **while** |h| $> 1$ **do**
4:      ($\sigma_1$,$\sigma_2$) $\leftarrow$ FindBestPair($h$, scenarios)
5:      betterScenario $\leftarrow$ Ask($\sigma_1$,$\sigma_2$)
6:      **if** betterScenario = $\sigma_1$ **then**
7:          pref $\leftarrow \sum_i \lambda_i(r_i(\sigma_1.s, \sigma_1.a) - r_i(\sigma_2.s, \sigma_2.a)) > 0$
8:      **else**
9:          pref $\leftarrow \sum_i \lambda_i(r_i(\sigma_2.s, \sigma_2.a) - r_i(\sigma_1.s, \sigma_1.a)) > 0$
10:     **end if**
11:     preferences $\leftarrow$ preferences $\cup$ pref
12:     $h \leftarrow$ RemoveHypotheses($h$, pref)
13: **end while**
14: **return** FindSolution(preferences)
```

Generating scenarios: Because our algorithm asks about concrete state-action scenarios in a survey form, those scenarios must be explainable and understandable to people. We recommend making a list of scenarios by hand that are easy to describe to users in order to ensure that they meet this requirement. In practice, we have not found the particular scale or values of these scenarios to have an impact on the ability to accurately learn λ, except values spanning each feature be used.

For example, we can use two human state attributes that we previously found to determine the reward for helping a device - a user interruption level computed on a scale $[1, 10]$ and the number of hours ago that the last query was asked $[0, 24]$. While a valid scenario is any combination of these two features, it may not be easy for a user to imagine a situation where their interruptibility was 5.346 and they were asked 8.82 hours ago. Instead, we could create concrete scenarios that require the interruptibility be given in whole numbers and last question asked hours ago could be [0.5, 1, 2, 4, 8, 24] (hour values that are relatively easy to think about).

In general, the generated scenarios should be easy for a user to understand without much explanation.

GenerateMonteCarloHypotheses: We generate a uniform random set of hypotheses for λ subject to our constraints that $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$. Each hypothesis is in $\mathbb{R}^{|\lambda|-1}$, as λ_0 can be completely explained by the remaining features. As with all Monte

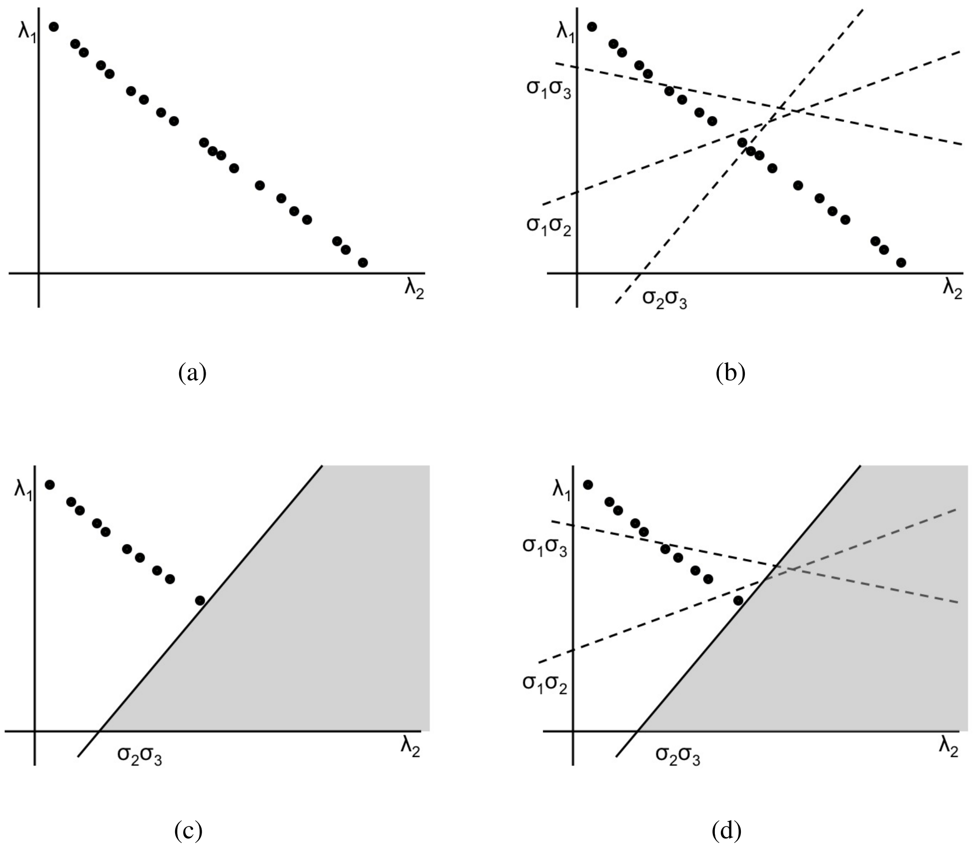


Figure 7.1: a) The LearnImportance algorithm first GeneratesMonteCarloHypotheses for λ where $\sum_i \lambda_i = 1$ (here $|\lambda| = 2$). b) The algorithm searches the through all pairs of scenarios to FindBestPair that cuts the hypothesis space in half. c) When it finds the best pair, it asks the person for their preference and removes the invalidated hypotheses. d) Then, it repeats the process of finding a new pair of scenarios, asking, and removing the invalid hypotheses.

Carlo methods, increasing the number (and therefore the density) of hypotheses increases the accuracy of the learned λ. In our experiments section, we vary the number of generated hypotheses to show how the accuracy of the learned λ increases.

FindBestPair: The additive reward for each scenario is $R(s,a) = \sum_i \lambda_i r_i(s_i, a)$. If a user preferred scenario σ_1 over σ_2, we can generate an inequality

$$\begin{aligned} \sum_i \lambda_i r_i(\sigma_1.s, \sigma_1.a) &> \sum_i \lambda_i r_i(\sigma_2.s, \sigma_2.a) \\ \text{or } \sum_i \lambda_i (r_i(\sigma_1.s, \sigma_1.a) - r_i(\sigma_2.s, \sigma_2.a)) &> 0 \end{aligned} \tag{7.4}$$

which forms a plane through the hypothesis space h and invalidates hypotheses that do not satisfy the new preference.

Let $invalid(h,$pref) be the number of hypotheses in h rendered invalid by the preference. FindBestPair iterates through all pairs and finds the pair of scenarios σ_1 and σ_2 that minimizes

$$abs(invalid(h, \text{pref}) - |h|/2) \tag{7.5}$$

where pref is an equality like Equation 7.4. In other words, it finds the pair of scenarios that best splits the hypotheses in half. Dividing h in half each time through the loop (Line 3) implies that the number of user preference questions needed to learn λ is $log_2(|h|)$ questions.

RemoveHypotheses: Once the best pair is found, the user is asked for their preference by listing the features of each scenario. Then, the algorithm generates the inequality preference *pref* based on their response. The RemoveHypotheses function iterates through all hypotheses in h, determining whether each is valid by setting in the hypothesized values of λ into the inequality, and removing those from the list that are invalid. The hypotheses remaining in h after RemoveHypotheses satisfy all preference inequalities in *preferences*.

Rather than removing hypotheses, it is possible to redistribute the invalid hypotheses in the valid area keeping the number of hypotheses the same over time. This process will increase the accuracy of the algorithm because there are more hypotheses to divide in half at each step, but at the cost of possibly asking the person more questions.

FindSolution: The FindSolution algorithm could use the remaining hypothesis as its approximation of λ. As the number and density of hypotheses increases, the error of the remaining hypothesis decreases because the valid hypothesis area is more constrained (Metropolis and Ulam (1949)).

It is also possible to evaluate the linear program of the preference inequalities that were generated through the algorithm to constrain the hypothesis space. An algorithm such as the simplex algorithm (Dantzig (1963)) find the valid area within the inequalities and and find the minimum or maximum point subject to another constraint (*e.g.,* minimize or maximize the distance from (0,0)) For the purposes of understanding how the selected scenarios impacted the preference inequalities, we will evaluate the linear program in our following experiments.

Next, we present experiments towards demonstrating the use of our algorithm on an example additive reward problem. Because the scenarios are not probabilistic and we are not asking for numerical values as in our previous surveys, we show that people should find this exercise to be understandable and that their preferences are consistent over the length of the survey.

7.2.3 Experiment

In order to understand how our algorithm performs on real-world tasks, we present our experiment to evaluate its ability to learn different reward functions for asking different environment occupants to travel to help CoBot use the elevator. We first describe the occupants' reward functions that we would like to learn. Then, we briefly describe our surveys again to show the use of scenario comparison questions. Finally, we present our simulated results to learn random scaling factors λ with different numbers of generated hypotheses and different types of generated scenarios for $|\lambda| = 2$ and 3.

CoBot maintains state information about itself (*i.e.,* its own location) as well as information about offices and people in the environment. In order to determine where to navigate in order to ask for help, we presented our replanning algorithm that determines the offices that are closest to its current location and the location where it will need help (*e.g.,* the elevator, kitchen, etc). Additionally, the algorithm included the cost of asking each human in terms of human state attributes such as how interruptible they are, how frequently they help, and the last time they were asked. As a result, our robot minimizes the sum of subrewards computed over the robot and human state to determine where to navigate and who to ask for help. For example:

$$\arg\min_{l_o} \texttt{COT}(l_{curr}, l_o) + \texttt{COA}(l_o, \iota_o, f_o, t_o) + \texttt{COTH}(l_o, l_{help})$$

where

- $\texttt{COT}(l_{curr},l_o)$: distance to travel from current location l_{curr} to office l_o,
- $\texttt{COA}(l_o,\iota_o,f_o,t_o)$: cost of asking at l_o for help, based on their interruptibility ι_o, frequency of help f_o, and the last time t_o they helped,

- COTH(l_o,l_{help}): distance to travel from the office l_o to the location of help l_{help} with the human.

In our problem, our three subrewards: cost of asking (COA) and the cost of traveling with and without the human (COT and COTH) are independent and additive. While we could manually tweak subreward functions so that the values are roughly in the range to get to our desired behavior, this takes time and is subject to error. Because each person has different preferences, determining the COA and COTH for each office is infeasible without asking them for their preferences but is also infeasible to require them to demonstrate their preferences. We would like the environment occupants to fill out a survey of their own preferences for when they get asked for help in order to increase their satisfaction with the robot and likelihood that they will continue helping it over time. The survey should not be too long in order to increase the likelihood that it gets filled out (Heberlein and Baumgartner (1978)), and should include questions that are easy to answer and not susceptible to error.

Using our LearnImportance algorithm, we will train 3 scaling factors for our three subrewards:

$$\begin{aligned}\lambda_1 * r_1(COT, ask(l_o)) + \lambda_2 * r_2(COA, ask(l_o))\\ +\lambda_3 * r_3(COTH, ask(l_o))\end{aligned} \tag{7.6}$$

Interestingly, in this problem, we also have a secondary additive reward learning problem in relationship between our 3 COA features - ι_o interruptibility,f_o frequency of asking, and t_o time of last question - to compute the COA for the larger reward function:

$$\lambda_4 * r_4(\iota_o, ask) + \lambda_5 * r_5(f_o, ask) + \lambda_6 * r_6(t_o, ask) \tag{7.7}$$

We took a two part approach in order to learn these scaling factors. First, we deployed a survey (summarized in Chapter 5) with 50 of these comparison questions in order to evaluate the consistency of responses and the ease of understanding from participants. These questions were not dynamically generated using our algorithm; instead they were static and the same for each person. Then, we performed simulated experiments to understand how our algorithm performed with different preferences, scenarios, and numbers of hypotheses.

7.2.4 Survey Results

We conducted a web survey about participants preferences for what conditions and how frequently they would be willing to help our robot (summarized in Chapter 5). In the

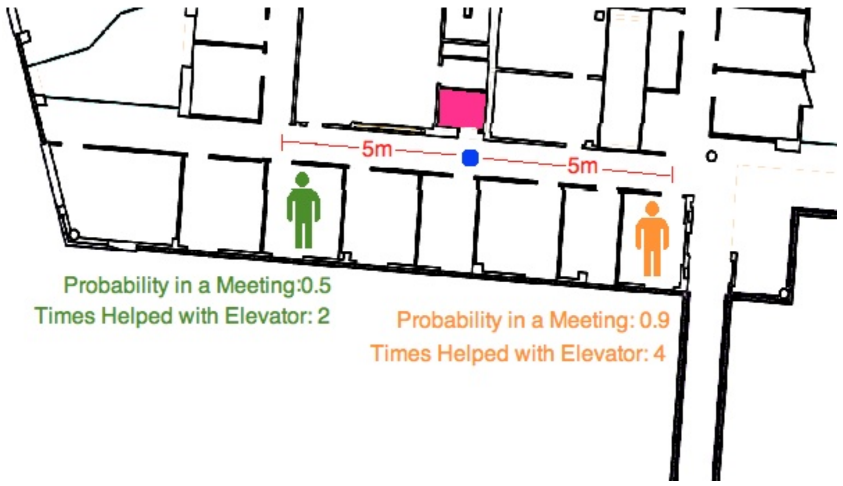

5m
5m
Probability in a Meeting:0.5
Times Helped with Elevator: 2
Probability in a Meeting: 0.9
Times Helped with Elevator: 4

Figure 7.2: In the survey, participants were asked which of two people (orange and green) should help a robot (center in blue) use the elevator (pink box). For example, the people here are equidistant from the elevator but have different availability and frequency of helping.

survey, participants were given two concrete scenarios and asked which person should be asked for help (Figure 7.2). Participants were able to successfully answer all of our concrete scenario comparison questions and did not ask us to clarify the scenarios or the instructions. As expected, they did respond that the 50 questions was extreme for an online survey format.

We analyzed the results to understand if and when participants' responses began conflicting with previous answers. Importantly, while participants did tire of the questions and became inconsistent in their preferences over time, the first 10-20 questions depending on the person were consistent. This indicates that our new algorithm should not need to ask more than this many questions to learn the additive reward functions. We were unable to use the actual responses to learn additive reward functions because of the error by the end of the 50 questions. Instead, we next simulate the comparison questions.

7.2.5 Simulated Results

In order to understand how our algorithm performed with different scenarios and different numbers of generated hypotheses for random scaling factor preferences, we performed a series of simulated experiments. In the experiments, we randomly generated sets of true λ such that $\sum_i \lambda_i = 1$, and then ran our algorithm with auto-generated responses to the Ask

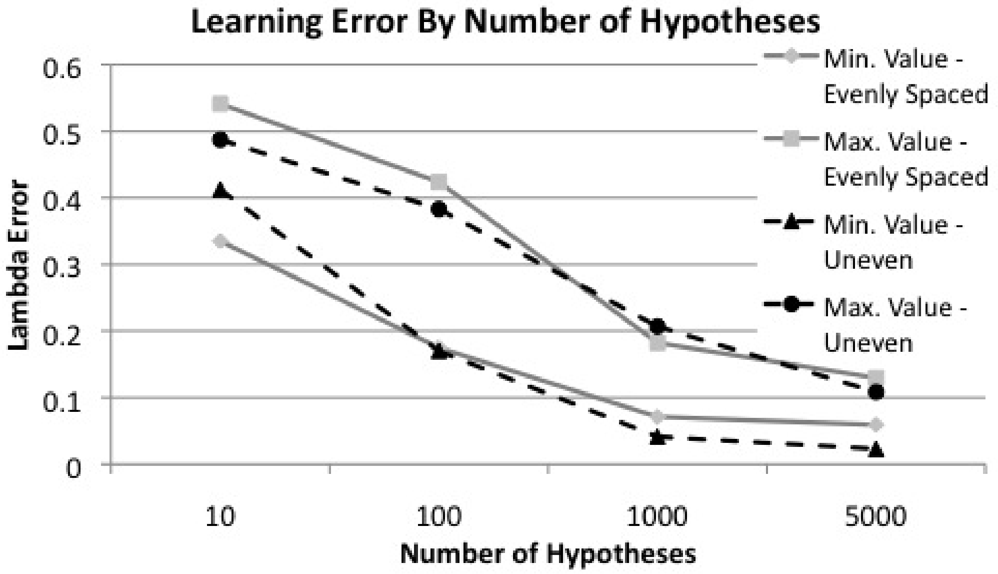


Figure 7.3: As the number of hypotheses increases for $|\lambda| = 3$, the error rate of the solutions decreases regardless of the evenness of the scenarios chosen.

questions based on the true λ. In total, we tested 30 random λs for each variable of $|\lambda| = 2$ or 3, the number of hypotheses (number of queries), preference solution analysis, and types of scenarios generated.

We analyze the *minimum* and *maximum* intersection point of the simplex formed by the constraints to understand the size of the simplex for the number of questions asked. However, these solutions may not satisfy the $\sum_i \lambda_i = 1$ requirement. For implementation, one should instead use the last remaining hypothesis left after asking questions.

Figure 7.3 shows the average minimum and maximum distance error of the solution λ from the randomly generated true λ for $|\lambda| = 3$. The error rates of both solutions are parallel and decreasing with the number of hypotheses indicating that the preference constraints are narrowing down λ from the high and low values equally. In our scenarios, the minimum intersection point had less error than the maximum distance solution with average difference 0.14.

Next, we evaluate our choices in scenarios and the resulting accuracy. We generated both evenly spaced scenarios for our robot navigation example (whole number evenly spaced distances, values of a potential cost of asking, and cost of traveling with the human) as well as unevenly spaced scenarios (interruptibility in tenths, hours ago of last question [0.5, 1, 2, 4, 8, 24, 48, 168], and frequency of questions per day [0.05, 0.1, 0.5, 1.0, 2.0, 5.0]). Our subrewards map both even and uneven scale values to $[0, 1]$. Because these scenarios dictate the dividing hyperplanes through the hypothesis space, it is possible that they could affect the error of the learned λ. The grey lines in Figure 7.3 represent the evenly spaced conditions and the black dashed lines represent the uneven conditions. The differences between the lines is negligible compared to the number of generated hypotheses, indicating that the algorithm is accurate irrespective of the scenarios.

Finally, we compare the number of hypotheses or questions generated compared to the accuracy of the solution based on the size of λ. As Figure 7.3 shows, as the number of hypotheses and number of questions = log(hypotheses) increases, the error decreases. With 5000 hypotheses or 12 questions, the algorithm is able to learn the solution with 0.02 (s.d. 0.1) error on average. The high standard deviation implies that most of the solutions had 0.0 error. For $|\lambda| = 2$, the algorithm required only 5 questions on average to reach error 0.015(s.d. 0.05) although increasing the number of hypotheses to 5000 reduced error to 0.0(s.d.0.001). This indicates that our algorithm can learn 2 and 3 attribute reward functions with a small number of questions, making it feasible to deploy.

Additionally, it is worth noting that rather than adding 10 times as many hypotheses for each $|\lambda|$ increase, it is possible to redistribute the hypotheses in the valid area rather than removing them as we mentioned above. This would reduce the computation of iterating through all hypotheses to find the pair that best pair of scenarios that divides them without sacrificing accuracy.

In summary, we contribute a novel Monte Carlo algorithm for eliciting scenario preferences from people and learn additive reward functions in log of the number of generated Monte Carlo hypotheses. In particular, our algorithm asks people for their preferences between two concrete state,action scenarios and refines its list of valid hypotheses accordingly. We demonstrate using a survey that real people can understand the questions and answer accurately. Then, we provided simulated results to show that the error decreases as the number of hypotheses increases and that the specific concrete scenarios that the algorithm asks about do not affect the accuracy. We conclude that our algorithm is less susceptible to human error while learning user preferences proportionately to the number of hypotheses.

7.3 Learning Human Attributes During Deployment

Finally, our human-centered planning algorithms require models of human attributes such as availability and accuracy not just for reward functions but also in observation functions. Our HOP-POMDP presented in Chapter 3, for example, is defined as $\{\Lambda, \mathcal{S}, \alpha, \eta, \mathcal{A}, \mathcal{O}, \Omega, T, R\}$. where

- Λ - cost of asking each human
- α - availability for each human
- η - accuracy for each human
- $\mathcal{A} = A \cup \{a_{ask}\}$ - autonomous actions and a query action
- $\mathcal{O} = O \cup \{\forall s, o_s\} \cup o_{null}$ - autonomous observations, a human observation per state, and a null observation
- $T(s, a_{ask}, s) = 1$ - self-transition for asking actions

It requires predefined availability and accuracy for each human in the environment. However, it may be difficult to acquire accurate approximations prior to robot deployment. Additionally, these values may change over time. In this section, we introduce our LM-HOP algorithm to learn the availability and accuracy of human observation providers while the robot executes its current policy using an explore/exploit strategy. Because the availability and accuracy are used in the observation function, our algorithm learns this observation function.

Recent work has also focused on using oracles to learn the transition and observation probabilities of POMDPs when it is difficult to model a robot before it is deployed in the environment (Kearns and Singh (2002); Jaulmes, Pineau, and Precup (2005); Doshi, Pineau, and Roy (2008); Cai, Liao, and Carin (2009)). In these algorithms, a robot instantiates hypothesis POMDPs that could possibly represent its transition and observation functions. The robot executions the action consensus from all of the hypotheses until there is disagreement between them of which action to take. The robot then asks an oracle to reveal the current state, the hypotheses which do not include the current state in the belief are removed, and new hypotheses are instantiated to replace them. In this way, the robot converges to choosing hypothesis POMDPs with the correct observation and transition functions. However, the robot must solve hypothesis POMDP policies $10^3 - 10^6$ times to learn the observation and transition functions for small problems like the Tiger Problem, which is intractable for robots in real time (Kearns and Singh (2002); Jaulmes, Pineau, and Precup (2005)). Additionally, we cannot depend on humans to be available to help the robot.

7.3.1 Algorithm

We introduce an online algorithm to learn the availability and accuracy of humans in the environment (the HOP-POMDP observation function) while the robot executes the current optimal policy using an explore/exploit strategy. While prior work on learning observation functions would also learn our HOP-POMDP observation function, the work is not tractable to solve in real environments due to their instantiation of multiple hypothesis POMDPs and the requirement that an oracle provide accurate observations about the robot's state (Jaulmes, Pineau, and Precup (2005); Doshi, Pineau, and Roy (2008)). Instead, our algorithm for Learning the Model of Humans as Observation Providers (LM-HOP):

1. requires only one HOP-POMDP to be executed at a time,
2. selectively recalculates the policy only when the observation probabilities have changed significantly, and
3. does not require an always-accurate and available oracle to provide accurate observations.

In particular, the algorithm keeps track of the most up-to-date availability and accuracy for each human while balancing exploration of the environment with exploitation of the current policy. When the up-to-date availability is significantly different from the current policy availability, the robot recalculates the optimal policy using the new values. We detail the LM-HOP Algorithm (Algorithm 4) in terms of these three contributions.

Single HOP-POMDP Instantiation We only instantiate a single HOP-POMDP for learning rather than several hypothesis POMDPs. We maintain counts ($\#o_{s',s}$) for each observation $o_{s'}$ in each state s and for each null observation in each state, as well as the robot's belief $b(s)$, and the availability and accuracy of each human (Lines 1-5). Before each action, the robot chooses a random number ρ to determine if it should explore or exploit the current best policy π (Lines 8-12). Then, as usual, the belief is updated according to the current policy and observation (Line 14), rather than taking the consensus action of many hypothesis POMDPs.

Learning Human Accuracy and Availability In HOP-POMDPs, the robot only needs to learn after a_{ask} actions. Prior work assumes that a human will always answer accurately (o_s). However, in our algorithm, if o_{null} is received after asking, the robot does not know its current state. As a result, it must update the approximate availability of all possible states it could be in, weighted by the probability of being in each state $b(s)$ (Lines 17-18). If an observation o_s is received, its availability of each state is incremented by the belief $b(s)$, because we still do not know if the human answered accurately (Lines 19-22). In order to learn accuracy from observations o_s, each η_s is incremented by the belief $b(s)$ of the robot (Lines 23-24). The accuracy and availability are calculated as averages (over time t) of observations over all questions asked.

It should be noted that due to the limitations of humans in the environment, our algorithm may attribute some unavailability to some available humans. In particular, the robot attributes the unavailability o_{null} to all states weighted by $b(s)$. If one human is always available and another is never available, some unavailability will still be attributed to the always-available human because the robot is uncertain and does not know it is not asking the always-available human.

Selective Learning In order to reduce the number of times a HOP-POMDP policy must be recomputed, we selectively update the policy when the estimated availability or accuracy of a human has changed significantly from the current HOP-POMDP estimate. We determine if any availability or accuracy has significantly changed using Pearson χ^2 test which tests the difference between an observed set of data and the expected distribution of responses. For example, with availability $\alpha_s = 0.7$ we would expect that only about 30% of the received observations are o_{null}. To compute the statistic, we define the number of observations from state s as n_s : $n_s = \sum_{s'} \#o_{s',s} + \#o_{null,s}$

Then we define $\chi^2(s) =$

$$\frac{(\sum_s' \#o_{s',s} - n_s\alpha_s)^2}{n_s\alpha_s} + \frac{(\#o_{null} - n_s(1-\alpha_s))^2}{n_s(1-\alpha_s)} \tag{7.8}$$

to test whether the observed availability $\hat{\alpha_s}$ is different than the initialized $\alpha_{init,s}$ or the accuracy $\hat{\eta_s}$ is different than $\eta_{init,s}$ with 95% confidence ($\chi^2(s) > 3.84$, Lines 27-31). If so, then it is unlikely that our current HOP-POMDP represents the humans in the environment. The approximations of all accuracies and availabilities must be updated, and the HOP-POMDP policy must be recomputed. The confidence parameter can be adjusted to further reduce the number of policy recalculations (*e.g.,* 99% confidence would require $\chi^2(s) > 6.64$). We expect our algorithm to recompute few times in contrast to the prior algorithms which recalculate each time the hypothesized POMDP actions conflict.

Algorithm 4 LM-HOP$(\pi, \tau, \alpha_{init}, \eta_{init}, b_{init})$

```
1: // Initialize availability, accuracy and counts
2: $\hat{\alpha_s} \leftarrow \alpha_{init,s}, \forall s, s', \#o_{s',s} = 0, \#o_{null,s} = 0$
3: // Execution Loop
4: for $t = 1$ to $\infty$ do
5:   $b \leftarrow b_{init}$
6:   loop
7:     // Choose explore or exploit action
8:     if $\rho > \frac{1}{t}$ then
9:       $a \leftarrow$ random action
10:    else
11:      $a \leftarrow \pi(b)$
12:    end if
13:    // update belief using transitions $\tau$, receive observation $o$
14:    $b \leftarrow \tau(b, a), o \leftarrow \Omega(b, a)$
15:    // if a = ask, update availability based on $o_{null}$ and
       accuracy based on $o_{s'}$
16:    if $a = a_{ask}$ then
17:      if $o = o_{null}$ then
18:        $\forall s, \hat{\alpha_s} \leftarrow (1 - \frac{1}{t}b(s))\hat{\alpha_s}, \#o_{null,s} \leftarrow b(s)$
19:      else
20:        #observed $o_{s'}$
21:        $\forall s, \hat{\alpha_s} \leftarrow (1 - \frac{1}{t}b(s))\hat{\alpha_s} + \frac{1}{t}b(s)$
22:        $\forall s, \#o_{s',s} \leftarrow b(s)$
23:        for $s \neq s'$, $\hat{\eta_s} \leftarrow (1 - \frac{1}{t}b(s))\hat{\eta_s}$
24:        $\hat{\eta_s} \leftarrow (1 - \frac{1}{t}b(s'))\hat{\eta_{s'}} + \frac{1}{t}b(s')$
25:      end if
26:    end if
27:    // is $\hat{\alpha}$ different than $\alpha_{init}$?
28:    if for any $s$, $\chi^2(s) > 3.84$ for $\alpha_s$ or $\eta_s$ then
29:      $\alpha_{init,s} \leftarrow \hat{\alpha_s}, \eta_{init,s} \leftarrow \hat{\eta_s}$
30:      $\pi \leftarrow$ SOLVE_POLICY$(\alpha_{init})$
31:    end if
32:  end loop
33: end for
```

7.3.2 Experiment

In order to test our learning algorithm, we use the same benchmark HOP-POMDP with two available humans as we used previously in Chapter 3. Our benchmark HOP-POMDP is a finite-horizon POMDP that contains 5 states and 2 actions with two humans (states 2 and 3) and two final states (4 and 5) (Figure 7.4). The robot starts at state 1 and chooses to take action B or C, where

$$T(1,B,2) = 0.75 \qquad T(1,B,3) = 0.25$$
$$T(1,C,2) = 0.25 \qquad T(1,C,3) = 0.75$$

The robot can then execute B or C from 2 and 3 to 4 or 5:

$$T(2,B,4) = 1.0 \qquad T(2,C,5) = 1.0$$
$$T(3,B,5) = 1.0 \qquad T(3,C,4) = 1.0$$

However, the reward for state 4 is -10 and the reward for state 5 is +10. The robot has the opportunity to ask for help in states 2 and 3 to ensure it receives +10. The costs of asking when the humans respond are $R(2, a_{ask}, 2, o) = R(3, a_{ask}, 3, o) = -1$ and when they do not respond $R(2, a_{ask}, 2, o_{null}) = R(3, a_{ask}, 3, o_{null}) = 0$. We vary the true availability and accuracy of the humans in states 2 and 3, and learn them while executing the current best policy.

Depending on the availability and accuracy of the humans h_2 and h_3, the optimal policy will determine whether the robot should take action B or C from state 1 and whether it should ask at the next location. This POMDP is similar in size and complexity to other POMDP benchmarks such as the Tiger Problem, and we will compare our results to those of other results on similar problem sizes.

We show that our algorithm significantly reduces the number of HOP-POMDP policies that must be computed compared to prior work. Additionally, compared to these approaches and other explore/exploit learning algorithms for POMDPs (*e.g.,* Kearns and Singh (2002); Cai, Liao, and Carin (2009)), we show that our algorithm converges towards the true accuracy and availability of human observation providers without requiring an additional oracle to provide true state observations. As a result, the algorithm is more tractable to execute in a real environment.

7.3.3 Results

We first tested the LM-HOP algorithm, assuming that the humans were 100% accurate. We initialized the availability $\alpha_{init,2} = \alpha_{init,3} = 0$ to understand how fast the algorithm

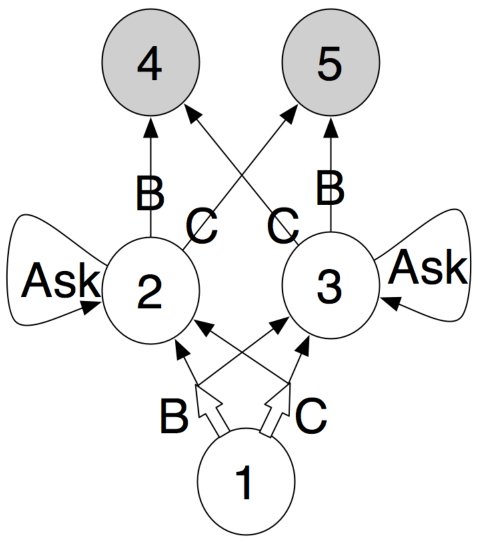


Figure 7.4: The robot starts at state 1 and can take actions to travel to states 4 (reward -10) or 5 (with reward 10). There are humans in states 2 and 3 that the robot can decide to ask so that it travels to state 5 to maximize its reward.

would converge to true availabilities. We compare to an explore-only algorithm that does not take into account the current best policy to determine how to act and an exploit-only algorithm that only acts based on the current best policy. As with the LM-HOP algorithm, if the estimated availability is significantly different than the current policy estimates, we recompute the current policy. As an example, Figure 7.5 shows the estimated availability of h_2 (light grey) and h_3 (black) over 5000 executions of the HOP-POMDP with true availability $\alpha_2 = 0.7$ and $\alpha_3 = 0.4$.

We then tested the simultaneous learning of both the availability and accuracy of humans in the environment. We initialized the availability of both humans to $\alpha_{init,2} = \alpha_{init,3} = 0.0$ and the accuracy of both humans to $\eta_{init,2} = \eta_{init,3} = 1.0$. We then varied the true accuracy and availability of our humans to understand the learning curves. Figure 7.6 shows an example of the learning rates for the availability and accuracy of h_2 and h_3 when true accuracy of each humans is 0.5 and the true availabilities are $\alpha_2 = 0.7$ and $\alpha_3 = 0.4$.

Explore/Exploit Strategy We find that our LM-HOP algorithm and the explore-only algorithm closely approximate the true availability and accuracy. The approximate availabilities in Figure 7.5 are 67% (compared to 70%) and 41% (compared to 40%). Compared to the explore-only algorithm (Figure 7.5 dot-dash lines), our LM-HOP algorithm is

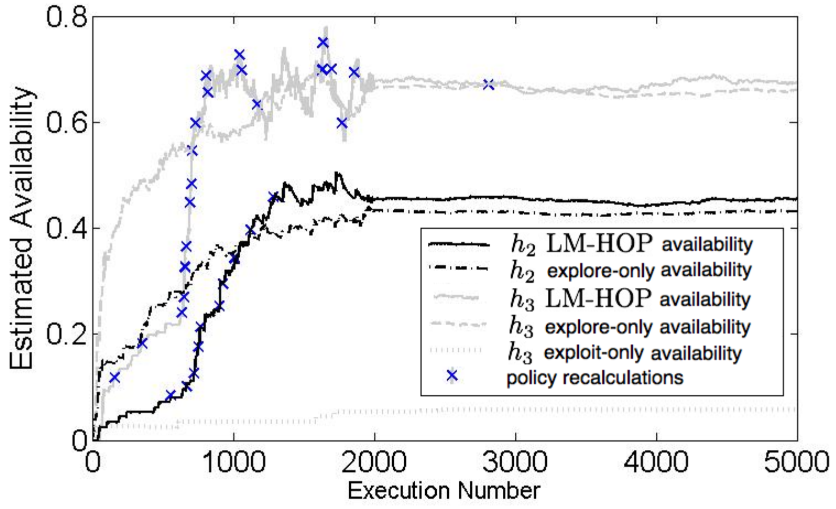


Figure 7.5: The estimated availability of h_2 (light grey) and h_3 (black) over 5000 executions of the HOP-POMDP with true availability $\alpha_2 = 0.7$ and $\alpha_3 = 0.4$.

slower to start converging because it tries to maintain high expected reward by exploiting the current best policy of not asking for help. If we modified the explore-exploit learning parameter ρ, our LM-HOP algorithm would spend more time exploring at first and would converge faster. Our algorithm does converge faster in the end because, after recalculating the policy, the policy includes asking. The exploit-only algorithm learns very slowly in our example because the initial optimal policy does not include ask actions.

We also compare the average reward (collected over 10000 executions) between the learning algorithms and the optimal policy reward if true accuracy and availability were known. Although the explore-only algorithm performs similarly to our LM-HOP algorithm in terms of number recalculations and convergence, it earn only an average -.215 reward compared to our algorithm which earns 3.021. The exploit-only algorithm earns 4.742 reward, and the optimal policy initialized to the true availability and accuracy earns 5.577. While the exploit-only algorithm earns more on average than our LM-HOP algorithm, we find that it earns very little reward when it chooses the path with lower availability first and high reward otherwise. Our algorithm does not have this dichotomy, and therefore we believe it performs better. We found no statistical difference between the average reward received when only learning availability and when learning availability and accuracy.

Comparison to Prior POMDP Learners The X's on Figure 7.5 show the number of policy recalculations while learning availability only. On average, our LM-HOP algorithm

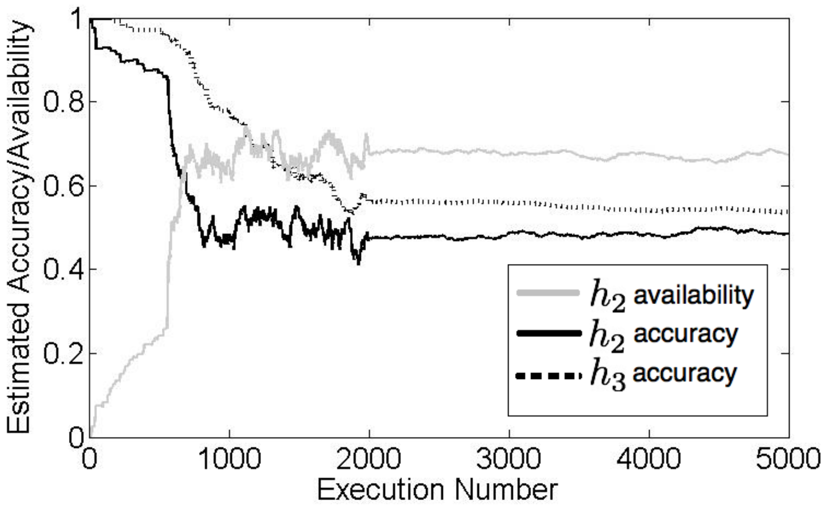


Figure 7.6: The estimated availability (light grey) is learned at the same time as the accuracy of the humans (black). h_2 is visited more often and his accuracy (0.5) is learned faster.

recalculated the policy 20-30 times when learning only availability. When learning both accuracy and availability, our algorithm recalculated the policy an average of 10 more times for a total of 30-40. Additionally, our algorithm converges to the true availability and accuracy within 1000-2000 executions of a single HOP-POMDP.

By varying the statistical significance confidence interval, it is possible to speed up how many executions it takes for our algorithm to converge. By requiring less confidence in the new availability and accuracy values, our algorithm does not have to wait so many executions before recomputing the policy. However, the tradeoff is that there will be more POMDP recomputations. Overall, the LM-HOP algorithm recalculates the policy significantly fewer times than the prior work's $10^3 - 10^6$ recalculations of 20 POMDPs of the similar-sized Tiger Problem Jaulmes, Pineau, and Precup (2005); Doshi, Pineau, and Roy (2008).

7.4 Chapter Summary

In this chapter, we contribute algorithms for learning human state attributes and reward functions in a usable way. We first introduced our linear regression algorithm that used survey situations and Likert value responses to predict phone users' cost of and incentive to help. While the algorithm did not always accurately model these two attributes, the human-centered algorithm that used these predicted values had high usability.

Second, in order to reduce the number of questions that people must answer and also avoid using Likert values that people find hard to estimate, we introduces an active learning algorithm that determined scenarios to ask people about and learned multi-attribute reward functions. We demonstrated using our survey results that participants did find these questions easier to answer, and using simulated results that our algorithm learns more accurate reward functions as the number of questions increases.

Finally, we introduced our LM-HOP explore/exploit algorithm to learn the availability and accuracy of the humans while executing the current best policy. We demonstrated that in terms of the explore/exploit strategy, our algorithm converges faster and with consistently higher reward than the explore-only and exploit-only algorithms. Compared to prior learning algorithms, we demonstrated that our algorithm, with a single hypothesized HOP-POMDP, recomputes POMDP policies at least 2 orders of magnitude fewer times. Our LM-HOP algorithm is effective in approximating the true availability and accuracy of humans without depending on oracles to learn.

Chapter 8

Related Work

This thesis merges ideas from Human-Computer Interaction (HCI), Human-Robot Interaction (HRI), and Artificial Intelligence (AI) about devices eliciting help from humans to perform tasks more efficiently. We review the research efforts in HCI, HRI, and AI for all aspects of asking for help that are covered in this thesis: determining when an agent requires help, determining who to ask, how to ask a person specifically, when it is appropriate to ask a person, and finally how agents can use those responses. We use the term "agents" in this chapter to refer to both software planners and robots.

8.1 Kinds of Help

The literature defines a wide spectrum of ways that people and agents can offer advice and help to another agent. Parasuraman *et al.* suggest there are four types of help that a human can give: information acquisition, information analysis, decision and action selection, and action performance (Parasuraman, Sheridan, and Wickens (2000)). When acquiring or analyzing information, agents that use sensors may ask for help with reducing reasoning uncertainty (*e.g.,* locations on a map (Asoh et al. (1996); Biswas and Veloso (2010))) using experience sampling (Barret and Barrett (2001)) or observation (Hayes et al. (2004)) or online during execution (*e.g.,* Cohn, Atlas, and Ladner (1994); Lewis and Gale (1994)). Because it can be easier for people to confirm a label rather than generate a new one (Eagle and Leiter (1964)), some agents only require a person to confirm a help response (*e,g,,* Fong, Thorpe, and Baur (2003)) or provide corrective feedback (*e.g.,* Shilman, Tan, and Simard (2006); Argall, Browning, and Veloso (2007); Faulring et al. (2010)). Raghavan *et al.* found that in some domains like text classification, people are good at determining the important features of differentiating between labels in addition to providing the labels themselves (Raghavan, Madani, and Jones (2006a)).

Deciding which action to take depends on the mapping from states to actions (the policy) that should be taken. In robot learning by demonstration, a demonstrator creates that mapping for the robot. Methods for learning include telling the agent which action to take if the agent already knows how to perform it (Chernova and Veloso (2008b)), physically moving the robot through teleoperation (*e.g.,* Katagami and Yamada (2001); Nicolescu (2003); Browning, Xu, and Veloso (2004); Saunders, Nehaniv, and Dautenhahn (2006)), or physically performing the task for the agent to imitate (*e.g.,* Hayes and Demiris (1994); Atkeson and Schaal (1997); Price (2003); Lockerd and Breazeal (2004)). In reinforcement learning (*e.g.,* Kaelbling, Littman, and Moore (1996); Sutton and Barto (1998)), people can provide positive or negative rewards directly to encourage or discourage actions (Thomaz and Breazeal (2006)) or demonstrations can be used as reinforcement of a control policy (*e.g.,* Atkeson and Schaal (1997); Abbeel and Ng (2004)). Finally, if it is not possible for an agent to learn to perform a task or to execute the selected actions, someone can take control of the agent to teleoperate it (*e.g.,* Sellner et al. (2006); Dorais et al. (1999)) or wizard of oz the actions (Green and Wei-Haas (1985)). Multiple people and agents can also perform a task together with mixed-initiative interactions (*e.g.,* Ferguson, Allen, and Miller (1996); Veloso (1996); Horvitz (1999)). Hearst outlines a spectrum of human assistance from directed control (teleoperation) through dynamic autonomy (teamwork) for robotics to plot the level of autonomy of different robots and the help they require (Hearst (1999)).

In this thesis, similar to the information acquisition and analysis related work, our smart phone and robot ask for additional observations to reduce uncertainty for both learning and non-learning purposes to improve task performance. In contrast to the action policy help of the related work, this thesis largely assumes the action policy is known if the state is known (the exception is learning the action policy for phone volume). Additionally, unlike the related work which takes control of the robot or teaches the correct action, our work focuses on the action limitations that the robot could never perform. If the device cannot perform the action in the action policy, it asks for help to overcome the actuation limitation.

8.2 How to Use Help

We first outline different ways in which human help has been used in prior work. This thesis focuses planning for help that will be used during execution and for learning.

8.2.1 Learning

Traditional active learning and learning by demonstration techniques have taken place offline before a task has started with a dedicated oracle or perfect teacher (*e.g.,* Cohn, Atlas, and Ladner (1994); Atkeson and Schaal (1997); Price (2003)). Online active learning (*e.g.,* Horvitz (1999); Kapoor and Horvitz (2008)) and learning by demonstration while performing a task (*e.g.,* Lockerd and Breazeal (2004); Chernova and Veloso (2007)) allow labelers and demonstrators to focus their attention on data as it occurs rather than on instances that occurs infrequently but are classified incorrectly. As discussed previously, Kapoor and Horvitz incorporate interruptibility measures into their online active learner to ask fewer times and with a more balanced number of "busy" and "not busy" labels compared to random sampling. The users of the decision-theoretic dynamic model was additionally more usable and preferred by users. Proactive Learning is an online learning process that determines which of multiple possibly imperfect or fallible supervisors that provide labels for data to trust and request labels from (Donmez and Carbonell (2008)). Donmez and Carbonell offer algorithms to determine who to ask for help if some of the labelers fail to respond, if some answer incorrectly but offer a confidence in their answer, and if some are more expensive to query than others (Donmez and Carbonell (2008)). Additionally, Donmez et al provide an online algorithm for determining which of many labelers on Mechanical Turk to query based on their expected accuracy while still minimizing the number of label queries (Donmez, Carbonell, and Schneider (2009)).

Corrective feedback is used in many predictive applications to learn to overcome overconfidence which can arise when the learner believes that predicted labels are more accurate than they actually are. Rather than limiting learners to only request help from demonstrators in learning by demonstration, Chernova and Veloso also allow the demonstrator to provide corrective feedback when possible (Chernova (2009)). Argall *et al.* allow humans to select multiple data points to correct at once when controlling the robot (Argall, Browning, and Veloso (2007)). These techniques begin to bridge the gap between the assumptions of perfect labelers and true human labelers while improving the accuracy of the resulting predictions.

8.2.2 Planning

While planners model the states of the world and the actions of the agents in the world, incorporating communication into these models has proven difficult. The OPOMDP model does not include an answering agent in the model, but does include the asking action for the single agent to perform (Armstrong-Crews and Veloso (2007)). By asking for help, the agent receives perfect information to know its state with certainty and can continue taking optimal actions. The Decentralized decision processes model the necessity for communication between agents (*e.g.,* Bernstein, Zilberstein, and Immerman (2000); Goldman and Zilberstein (2004)). Often, it is assumed that all agents can broadcast their messages to all agents in the team and do not need to determine exactly which agents would know the most accurate answer to the question. Dec-MDP-Com and Dec-POMDP-Com are models which include an extra parameter about the cost of communicating each message to determine the most cost effective times to communicate (Goldman and Zilberstein (2003)). Goldman *et al.* offer three methods of communication 1) one way, 2) two way, and 3) confirmation for more one-one questions. While subsequent approaches have been offered other methods for communicating, including execution-time communication, they continue to broadcast messages (Scerri et al. (2004); Spaan, Gordon, and Vlassis (2006); Roth, Simmons, and Veloso (2007)). Like the thesis, these techniques aim to plan how to communicate between agents to resolve inherent uncertainty in sensor observations and plans. However, they do not focus on modeling the capabilities of each agent to determine who is best to communicate with.

8.2.3 Semi-Supervised Robot Control

Dorais *et al.* introduced sliding or adjustable autonomy as a technique to transition between autonomy and teleoperation during a task (Dorais et al. (1999)). Because a human cannot always be available to provide fine-grained teleoperation remotely to a robot on Mars and because a robot may have varying capabilities to complete tasks, they present four robot modes: 1) entirely ground-controlled, 2) ground-controlled with autonomous recoveries, 3) autonomous with ground-commands executed simultaneously with the plan and 4) fully autonomous. A robot can transition between these modes based on the commands that are given (*e.g.,* "go forward 5 meters" or "explore the area ahead"). Sellner *et al.* extend this idea for multi-robot systems where a human cannot be in control of all robots at once (Sellner et al. (2006)). They define a "mixed-initiative" control model in which the human can assume control and the robot can request the human's control and a "system-initiative" control model in which the robot must always request the human's time. While participants who tried the models had different preferences for the models, the models necessitate the robots having shared knowledge about who needs the human's attention in order to not overwhelm the human. Shiomi *et al.* developed an interface for reducing the cognitive load of humans controlling multiple robots by 1) alerting the human ahead of time that a robot will need help soon and extending time that it can be autonomous if the human is busy with another robot and 2) providing context about why the help is needed so that the teleoperator can take control faster (Shiomi et al. (2008)). Although these robots require some teleoperation from an expert, they can perform autonomously and transition between modes of autonomy by proactively requesting help from humans rather than relying on the complete attention of the human teleoperator during tasks.

Collaborative control models the human teleoperators to reduce the requirement for an expert controller (Fong et al. (2001)). Before a human begins controlling the robot, models about both the human's capabilities and knowledge and the robot's safety capabilities are developed by hand. Using these model, the robot takes commands from the human and then can create a dialogue with the teleoperator to ensure its own safety, determining when to ask the human to confirm a possibly dangerous action and with what language. Fong *et al.* found that different users have different expectations about how urgent these questions would be - some did not answer the questions because of the interruption (Fong, Thorpe, and Baur (2003)). Additionally, robotics experts found that questions about the safety of the robot were not necessary for them, but did appreciate the robot maintenance questions that allowed them to keep track of the battery power and other sensors through a task. Bruemmer *et al.* extend collaborative control to allow the robot to perform other sensor tasks semi-autonomously while the human is teleoperating it (*e.g.,* looking for people in search and rescue scenarios) (Bruemmer et al. (2005)).

This thesis instead focuses on autonomous task execution in which the device is expected to perform on its own and proactively request help rather than depending on a supervisor to help take control of it or teach it.

8.3 When to Ask for Help

Determining the timing of when to ask for help is important both for receiving relevant labels and for ensuring that there is a person available who can help. Prior work in asking for human help has focused on receiving relevant labels. In contrast, this thesis adds models of humans into the decision for determining whether and when to ask for help.

8.3.1 During Data Collection

To limit the amount of data to label, Experience Sampling (ESM) relies on people in the environment to label their own data (*e.g.,* Barret and Barrett (2001)). While people in the environment can be very accurate at labeling both qualitative and quantitative data (Consolvo and Walker (2003)), they require a lot of prompting by the systems to remind them to provide labels. While some ESM label collections depend on people to remember to label their data at the end of the day (*e.g.,* diaries at the end of the day (Carter and Mankoff (2005))), others require users to enter data at throughout the day (*e.g.,* at random intervals (Horvitz, Koch, and Apacible (2004)) or when users recognize an event occurring (Consolvo and Walker (2003))). These techniques for data collection do not take into account the agent's needs for data. It assumes that by collecting data randomly, the data sample reflects what an agent will encounter when it is trained and deployed.

Including a model of classifier uncertainty into the decision of when to ask for help, initial work in the active learning community focused on generating new data points to label that would help define a classifier (*e.g.,* Angluin (1988); Plutowski and White (1993)). These techniques have been shown to reduce the number of labels compared to labeling all of the data while maintaining the accuracy of the classifiers. However, they could produce queries that did not make sense for people to answer (*e.g.,* "Is a pregnant man more prone to heart disease?") (Lewis and Catlett (1994)). To generate more appropriate data requests, many recent techniques have focused on querying data from the given set. Cohn *et al.* find the optimal method of choosing data to label by selecting the data point which, once labeled, will result in the highest accurate classifier for all future data (Cohn, Atlas, and Ladner (1994)). However, this technique is often difficult to compute in practice. Lewis and Catlett use uncertainty sampling choose which data point the current classifier is most uncertain about (Lewis and Catlett (1994)). Roy and McCallum use sampling estimation calculate the expected error rate if each point is labeled and choose the one with the highest reduction in error (Roy and McCallum (2001)). Nguyen proposed applying clustering algorithms to the data and choosing a single point to label from each cluster (Nguyen H (2004)).

Horvitz argues that while there are techniques to reduce the number of requests that an application can make to a user, applications would be more usable if they take into account the social implications of requesting information from users (Horvitz (1999)) - particularly based on human interruption level.

8.3.2 While Planning

Other agents plan their actions (including when to communicate) based on expected uncertainty in their state. Here, the help is about the agent's current state. Markov Decision Problems (MDPs) are used to model the states and actions of an agent that has complete knowledge of its environment (Howard (1960)). Partially Observable MDPs (POMDPs) are used if an agent cannot directly observe its states and instead uses noisy observations such as sensor readings (Kaelbling, Littman, and Cassandra (1998b)). Because there is only a single agent, there is no need to communicate. However, an extension of POMDPs, Oracular POMDPs (OPOMDPs), assume that there is a ground truth source of information such as a human that can be obtained at a cost and an agent can take an additional action to query that source in order to reveal the fully observable state (Armstrong-Crews and Veloso (2007)). By solving OPOMDP models, agents determine when to ask for the ground truth information and when to act independently. However, this work still assumes that a human is always available and accurate.

For multi-agent systems that do not include humans, Multi-Agent MDPs (MMDP) can be used if each agent can fully observe all agents' states and no communication is needed between the agents to clarify that state information (*e.g.,* Pynadath and Tambe (2002)). Decentralized MDPs (DEC-MDP) can be used when each agent cannot observe the whole state, but by sharing information between them, all agents can fully observe the state of the world and similarly DEC-POMDPs can be used when the combination of all agents' information is not enough to complete observe the world (Bernstein, Zilberstein, and Immerman (2000)). Dec-MDP-Com and Dec-POMDP-Com are models which include an extra parameter about the cost of communicating each message to determine the most cost effective times to communicate (Goldman and Zilberstein (2003)). While it the DEC-POMDP models can be simplified to their respective MDP and POMDP models when agents can broadcast communication for free to all agents all the time, determining when to communicate at a cost in DEC-MDPs and DEC-POMDPs has been shown to make the problem of solving these models NEXP-complete (Bernstein, Zilberstein, and Immerman (2000)). In addition to the complexity of solving the plans, these models include full state and action representation for multiple agents. It would be infeasible to model humans this way, especially if they are not necessarily always helping with the task.

Heuristic approaches have been proposed to reduce the overhead in communication or coordination between multiple non-human agents. For example, Nair *et al.* propose that agents should communicate synchronously every K timesteps, which allows the agents to also synchronize their states at that time (Nair, Roth, and Yohoo (2004)). Roth *et al.* instead define a factored DEC-MDP in order to allow individual agents to act alone without requiring communication until the agents must coordinate actions to avoid failure (Roth, Simmons, and Veloso (2007)). Melo and Veloso predefine the states that could require coordination between agents and learn the communication requirements during task execution (Melo and Veloso (2009)). However, these still do not model the availability of the agents to respond to communication and much literature including (Roth, Simmons, and Veloso (2005)) argues that it is unclear when communication is needed until the time of task execution.

8.3.3 During Execution

Often the need for communication comes from the inherent uncertainty in executing in an unknown environment and collecting noisy sensor data. During the execution of a plan, agents can stop and choose to communicate their state with each other when 1) one agent becomes too uncertain or 2) one agent receives an observation that would affect task actions. When learning by demonstration, Chernova and Veloso, for example, request a new demonstration of an action when it is unsure which discrete action to take during execution of the task (Chernova and Veloso (2007, 2008a)). Even during teleoperation, Fong *et al.* request the robot to confirm whether an action should be taken in potentially dangerous conditions for the robot (Fong, Thorpe, and Baur (2003)).

For some tasks, the execution is not discrete and a human must provide corrective feedback only after the complete set of actions has been taken. Argall *et al.* request corrective feedback on successful actions taken only after the robot fails to complete a task (Argall, Browning, and Veloso (2007)). An interface may automatically fill in fields in a form or provide a prediction for which folder to sort a piece of email into and leave it up to a person to determine when a correction must be made (*e.g.,* Culotta et al. (2006); Faulring et al. (2010)). In CueTIP, users see their handwriting and the word prediction and can make corrections (Shilman, Tan, and Simard (2006)). However it can be difficult in some cases for a user to notice when the execution is wrong (*e.g.,* when a spam filter puts an important email in the spam folder). The OOPS toolkit helps users "discover" if the learners prediction is incorrect and then provides a set of interaction techniques for the user to correct it (Mankoff, Abowd, and Hudson (2000)).

This thesis instead included models of humans at different locations in order to plan for the need for help. Then, during execution, we use replanning in case there is no one available in the expected locations.

8.4 Who to Ask for Help

Scholtz defined five different types of people that are available in robots' environments: supervisor who ensures the robot software is operating properly, operator who is using the robot, mechanic who fixes the physical system, teammate who works with the robot to perform a task, and bystander who does not directly interact with the robot (Scholtz (2002)). Goodrich and Schultz extend this definition to include mentors who help teach robots and information consumers who benefit but do not work directly with robots (Goodrich and Schultz (2007)). This thesis explores proactively requesting help from device users (most closely related to operators) and environment occupants (most closely related to information consumers) rather than supervisors and mentors who are assumed to always be available, accurate, and willing to help.

8.4.1 Users, Mentors, and Supervisors

We distinguish a user as a person the device/agent is performing a task for while a supervisor ensures it is working appropriately. In general, the user knows less about how the application works, but both the user and a supervisor can help an application. Because they both use the devices long-term, it is possible to model them. Fong *et al.* model the participants (users) helping to control their robot and change the types of questions that are asked based on expertise (Fong, Thorpe, and Baur (2003)). Experts tend to provide extra help by injecting extra knowledge and creating more common references compared to novices which can be useful for future questions (Isaacs and Clark (1987)). However, other than asking the human explicitly, most techniques to determine expertise are limited to a specific domain like software engineering (Mockus and Herbsleb (2002)) and require additional information about the user such as their social networks (Zhang, Ackerman, and Adamic (2007)), emails (Campbell et al. (2003); Balog and de Rijke (2006)) or bulletin board posts (Bouguessa, Dumoulin, and Wang (2008)).

While there is often only a single user or supervisor available, there may be instances, especially in learning environments, when multiple supervisors or mentors are available to help. Proactive Learning is an online learning process that determines which of multiple possibly imperfect or fallible supervisors that provide labels for data to trust and request labels from (Donmez and Carbonell (2008)). Donmez and Carbonell offer algorithms to determine who to ask for help if some of the supervisors fail to respond, if some answer incorrectly but offer a confidence in their answer, and if some supervisors are more expensive to query than others (Donmez and Carbonell (2008)). This thesis focuses on modeling users rather than supervisors who may be less accurate and available to help.

8.4.2 Teammates

In multi-agent domains, teammates, include both computer agents and humans, act autonomously but can perform actions or tasks together to achieve goals. Multi-Agent MDPs (MMDPs, DEC-MDPs, and DEC-POMDPs) have been used to model these teams determine when to communicate their state (*e.g.,* Pynadath and Tambe (2002); Bernstein, Zilberstein, and Immerman (2000)). In many cases, this communication is in the form of a broadcasted message to all agents (*e.g.,*Bernstein, Zilberstein, and Immerman (2000);

Roth, Vail, and Veloso (2003); Nair, Roth, and Yohoo (2004); Roth, Simmons, and Veloso (2005)) rather than directing a question to a particular agent to save communication costs. In contrast, we assume that devices are performing for humans, not performing tasks with them. In these cases, we assume that direct communication is needed.

8.4.3 Bystanders and Environment Occupants in the Environment

When users, supervisors, or teammates are not available or not accurate enough to help an agent, this thesis argues that an agent may be able to ask other bystanders in the environment. However, the agent has no information about these bystanders before the first response. One example of a domain where there are many bystanders in the environment is recommender systems. While some recommender systems have used weighted majority of bystanders to determine how to help a user (Nakamura and Abe (1998)), many systems now use collaborative filtering techniques which cluster similar bystander users together without requiring a user's preferences (Herlocker et al. (1999)). More directly related is Weiss *et al.*'s robot asked for help navigating to a location from bystanders on the street (Weiss et al. (2010)).

This thesis instead seeks to understand how to seek help from environment occupants who are in the environment for longer periods of time than bystanders and can receive benefit from device tasks in the future.

8.4.4 Crowd-Sourcing and Online Help

Recently, human computation and crowd-sourcing on websites like Amazon.com's Mechanical Turk (M.Turk (2010)) have been used to gather multiple humans' labels cheaply (e.g., von Ahn and Dabbish (2004), von Ahn et al. (2008)). The ESP Game labels images on the internet by showing them to two people at once and asking them to guess the same word (von Ahn and Dabbish (2004)). The game takes advantage of the idea that if two people can generate the same response, it is more likely to be correct, making these labels more accurate than ones generated using captions and other heuristics on the original websites the pictures are embedded in. Human Computation has sparked recent interest in generating games to help solve problems including recommender systems (Hacker and von Ahn (2009)) and search (Ma et al. (2009); Law, Mityagin, and Chickering (2009)). Recently, Mechanical Turk has been used to cheaply translate audio (Callison-Burch (2009)), create automated directory assistance by identifying semantically equivalent speech (Paek, Ju, and Meek (2007)), and transcribe audio from experiments (Marge, Banerjee, and Rudnicky (2010)).

Shahaf and Horvitz explore the idea of modeling online users to plan when to post questions online to be answered by the crowd (Shahaf and Horvitz (2010)). Additionally, while human computation makes it possible to request and get data cheaply, it is unclear how many people must perform a task to be confident that the response is accurate. In the ESP Game, Von Ahn and Dabbish require that pairs of participants type the same label, but once the label is generated it is not confirmed (von Ahn and Dabbish (2004)). In ReCAPTCHA, Von Ahn *et al.* use one known word and one unknown word to test whether a user is authenticated (von Ahn et al. (2008)). However, when determining the accuracy of the user's response to the unknown word, they require at least two users to agree with an optical character recognition (OCR) prediction or three users without OCR. The authors say that relaxing or strengthening these requirements does not affect the overall accuracy of the unknown word prediction as long as the human users are relied upon more than OCR when determining the prediction. Donmez *et al.* determine which people on Mechanical Turk to request labels from (possibly a different number for each label) based on the expected reward each participant has received thus far answering questions and show this technique approaches the accuracy of a single perfect labeler (Donmez, Carbonell, and Schneider (2009)). This thesis does not focus on online users because of the asynchrony of receiving responses, however, our methods for modeling online users and determining who and whether to ask for help can be applied to crowd-sourcing.

8.5 How to Ask for Help

When humans ask for assistance from other humans, it has been shown that the language used in a question can affect how people answer it, both in terms of the language used in the answer and the correctness of the answer (Clark et al. (1992); Presser (2004)). Additionally, while humans share common experiences and can take each others perspective easily to answer questions, they also provide contextual information to *ground* their responses when necessary to supplement their questions (Clark and Wilkes-Gibbs (1986)). Based on these phenomena, the social psychology and HCI communities have developed guidelines on how to write survey questions and for techniques like focus groups, interviews, cognitive walkthrough, and pretests to help researchers iterate on and improve their questions, to reduce any possible ambiguities that people may find (Jeffries et al. (1991)).

While several different sets of guidelines have been proposed for agents that require help from humans (*e.g.,* Bellotti and Edwards (2001); Erickson and Kellogg (2001); Horvitz (1999); Shadbolt and Burton (1989)), there are seven main types of proposed information:

different amounts of context, human-understandable context, uncertainty, predictions, user control and feedback, action disclosure, and social interaction. We find that the first five types of information, operationalized in Chapter 6, are the most commonly used in current applications. This thesis contributes an understanding of how to ask for help in order to increase the accuracy of human responses.

Human-Understandable Contextual Information: Many of the guidelines suggest that agents should provide people with some contextual information about the environment and sensors that the agents have difficulty with. Work in creating common ground with robots have shown that humans and robots can communicate more effectively when provided context about the robot's sensors and environment (Torrance (1994); Steels (2003); Topp et al. (2006)). However, we found that different researchers interpret this principle differently when implementing their agents.

When the BusyBody application asks users to estimate their own interruptibility to use for training an interruption predictor, it does explain the reason for asking at that moment (Horvitz, Koch, and Apacible (2004)). Hoffman *et al.* request help from Wikipedia users to fill in missing summary data as the users are reading an article (Hoffmann et al. (2009)). When users are asked if the text they are reading in the article belongs in the summary, important keywords are not highlighted in the text. When reading the summary, users are provided with excerpts that could be added to make the summary more complete. Additionally, in studies of interruptibility, it has been shown that people make judgments with relatively small amounts of context (15 seconds) and extra context (30 seconds) does not improve accuracy (Fogarty et al. (2005)). Asoh *et al.* use Jijo the robot to ask about locations without context to build a map of the building (Asoh et al. (1997)) while Shiomi *et al.* provide teleoperators with complete video and audio from their robot Robovie (Shiomi et al. (2008)).

Recently, researchers demonstrated that labelers accuracy can depend on the level of contextual information (low level sensor readings vs high level summaries of the sensor data) they are provided. When users understand and use their own high level rules for classification, they are better at making those classifications compared to classifying based on the computers rules (Stumpf et al. (2005, 2007a)). This finding is supported by work in feedback in information retrieval applications (Rui et al. (1998b); Salton and Buckley (1990b)) which mask the low-level sensor-level features that computers use and collect (*i.e.,* individual keywords in documents or accelerometer data) and allow users to search for information using high-level meaning attributed to the low-level data (*i.e.,* summaries of documents or physical motion inferred from accelerometers). However, because it is often difficult to generate the high-level explanation of context, many applications provide only the low-level raw data, like pictures, to labelers instead of a summary with the assumption that they can find their own meaning (von Ahn and Dabbish (2004); von Ahn et al. (2008)). Shiomi *et al.* provide both the predicted text transcription and the low-level audio to let teleoperator supervisors distinguish the voice themselves if the text does not make sense (Shiomi et al. (2008)).

Predictions: Like Shiomi *et al.* providing text transcriptions, other work has focused on making it easier for people to help an agent by providing a predicted response rather than require a person to recall the answer Eagle and Leiter (1964) possibly simplifying their work (*e.g.,* Stumpf et al. (2005)). An interface may automatically fill in fields in a form or provide a prediction for which folder to sort a piece of email into (*e.g.,* Culotta et al. (2006); Faulring et al. (2010)). When asking teleoperator supervisors to confirm their possibly dangerous actions, Fong et al gives users a suggested threshold for defining dangerous conditions in the future and found that users either followed this threshold or set the threshold extremely high to prevent future questions (Fong (2001)).

Prediction Feedback: Users could also provide corrective feedback for incorrect predictions to improve later classification (Culotta et al. (2006)). In the active learning community, Raghavan asked people to label text documents as news articles, sports, etc., and also asked them to pick words (features of the classifier) that should have high weight for each class (Raghavan, Madani, and Jones (2006a)). Participants knew the article words they were looking for and could identify them easily. The classifier could correctly weight the important features faster than asking for article labels alone, because people had narrowed down the important features. This same method can easily be used for email classification and the other domains. For example, in CueTIP, users see their handwriting and the word prediction and can make corrections (Shilman, Tan, and Simard (2006)). Scaffidi allows users to provide feedback by creating new rules to make better predictions of phone numbers and other personal information found in emails (Scaffidi (2009)). The OOPS toolkit helps users "discover" if the learners prediction is incorrect and then provides a set of interaction techniques for the user to correct it (Mankoff, Abowd, and Hudson (2000)). By asking for feedback and providing predictions, an agent can correct errors and use feedback to improve its learning.

Prediction Uncertainty: Many agents calculate a prediction probability in order to decide whether it should ask for help. Studies of context-aware, expert, and recommender systems all show that providing users with the level of uncertainty in a system's predictions improves its overall usability (*e.g.,* Banbury et al. (1998); Mcnee et al. (2003)), even if the learner does not provide the exact uncertainty probability of correctness. Antifakos *et al.* studied the effect of displaying uncertainty information in conjunction with a system's predictions about previously seen lists of numbers (Antifakos, Schwaninger, and Schiele (2004)). By displaying this uncertainty information, the authors found that participants depended less on the predictions and remembered the lists of numbers better themselves even when the uncertainty values were incorrect. This result was replicated in both desktop and ubicomp domains.

8.6 Considerations when Asking for Human Help

Along with the considerations for the type of information that could be included when asking for help in order to improve grounding, people also consider the types of questions people are good at answering and how people decide when to ask questions. There are some questions that may help an agent but may not be natural for people. Additionally, people not only incorporate ideas of interruption into whether they should ask for help, but also in the length of help required and the history of how many questions have been asked.

This thesis asks questions that are easy for people to answer. They are multiple choice questions or lend to perception tasks such as identifying locations and pressing buttons that anyone can perform.

8.6.1 Asking Appropriate Questions

While grounding can help resolve some misunderstanding when asking for help, Thomaz *et al.* found that sometimes people have different models for how agents could or should use their help compared to how they actually are using the help (Thomaz, Hoffman, and Breazeal (2006)). They asked participants in a study to provide reinforcement to teach an online character to cook. While participants were eventually capable of using the reinforcement technique, participants wanted to use the reinforcement to encourage future behavior and spent a long time figuring out that they were reinforcing previous behavior. For example, participants continually clicked on a bowl, actually reinforcing that the bowl was not picked up, while they expected that their clicking would suggest picking up the bowl. Participants were also more prone to positive reinforcement rather than giving a negative cost for performing a particular action. As a result, Thomaz *et al.* recommend and implement a second type of reinforcement to better align with human understanding of the help they are providing (Thomaz and Breazeal (2006)). Expertise of the participant can also be used to determine the kinds of questions an agent should ask (Fong, Thorpe, and Baur (2003)).

8.6.2 Cost of Asking

One of the biggest problems of requesting help from users is the required time investment (Scollon, Kim-Prieto, and Diener (2003)). Data collection techniques such as ESM are susceptible to inappropriate interruptions, which may become is a significant distraction (Carter, Mankoff, and Heer (2007)). The distractions and interruptions may lead to inaccuracy of reporting or failures to report data (Kahneman et al. (2004)). Individuals asked to recall labels after an activity of interest has occurred rather than during the activity can also be inaccurate in their labeling if they forget the context or situation to label (Csikszentmihalyi and Larson (1987)). Jameson et al find that giving users a lot of information and then expecting them to recall it all can be difficult for users to do and model this cost of giving too much information in order to determine when it is appropriate to give one instruction at a time versus all together (Jameson et al. (2000)). In general, asking participants too many times can lead to a lack of compliance, reducing the number of labels that can be collected (Scollon, Kim-Prieto, and Diener (2003)).

People take into account this cost of interruption and annoyance when deciding whether to ask for help. They internally calculate the cost of failing to complete a task with the cost of the immediate interruption and of the continuing cost of how many questions they have asked and how many more questions they might need to ask (DePaulo and Fisher (1980)). Additionally, people tend to ask fewer questions when the help required takes a long time. Many solutions have been proposed to reduce the burden of ESM and other data collection techniques on participants by making it easier to understand the data (e.g., visualizations Hsieh et al. (2008)) and reducing the number of questions that should be asked per day. Horvitz argues that it is important to balance both the costs of asking and interrupting people with the benefits of collecting the data in order to create more usable interfaces that require help or data collection from users (Horvitz (1999)). While interruptibility is important in determining whether to ask for help, models about the time for help and the history of questions may also reduce the expected cost of asking for help.

8.6.3 Interruptibility

The problem of recognizing when someone is interruptible has been widely studied in the literature (e.g., Horvitz, Jacobs, and Hovel (1999); McFarlane (1999); Czerwinski, Cutrell, and Horvitz (2000); Horvitz et al. (2002); McFarlane (2002); Horvitz and Apacible (2003);

Czerwinski, Horvitz, and Wilhite (2004); Horvitz, Koch, and Apacible (2004); Fogarty et al. (2005)). Czerwinski *et al.* showed that even when users that ignore instant messages, these messages are still interruptions which affect task performance (Czerwinski, Cutrell, and Horvitz (2000); Czerwinski, Horvitz, and Wilhite (2004)). While Horvitz showed that modeling interruptibility is possible with expert labelers after the fact, labeling the data after the events does not capture the real-time application of interruptibility models to avoid unnecessary or unimportant distractions (Horvitz, Jacobs, and Hovel (1999)). Building personalized predictive models of interruptibility have been shown to be accurate in predicting users' busyness (Horvitz, Koch, and Apacible (2004); Fogarty et al. (2005)). However, McFarlane compared four different methods for when to interrupt people (assuming it was possible to detect their interruptibility level) and found that none of them overall improved users' performance compared to the others (McFarlane (1999, 2002)).

Recently, Kapoor and Horvitz take an active learning approach to determine when participants are interruptible and model both the cost of misclassification and the cost of asking for help when deciding which data points to ask about (Kapoor and Horvitz (2008)). They compare random sampling methods with uncertainty-based sampling (ask when the classifier is uncertain), decision-theoretic sampling (ask when benefit of information is greater than the cost of interrupting), decision-theoretic dynamic (same as decision-theoretic except that the classifier can also ask about previously seen data). They find that the decision-theoretic dynamic model not only results in more accurate interruption classifiers with fewer label requests than random sampling but that users of the dynamic model is also find it significantly less annoying than the random sampling. This shows that agents can increase the usability of agents that request help by taking into account their interruptibility.

8.7 Chapter Summary

Prior work has focused on when an agent should ask for help based on uncertainty, and although prior work has also shown that people have limited availability and accuracy, the asking literature has not adopted that model of humans. In this thesis, we contributed planning algorithms to reason about human helpers in addition to uncertainty to determine which actions to take and who to ask for help.

Chapter 9

Conclusion and Future Work

9.1 Contributions

This thesis makes the following contributions:

- **A symbiotic approach for devices to ask for human help to overcome limitations during tasks.**

  Rather than limiting device tasks to those in which it can perform autonomously, we include actions for devices to plan to ask for help from humans in the environment to overcome actuation limitations and reasoning uncertainty. We define symbiotic relationships as those in which the device asks for help from humans while performing tasks for them. Because the devices are performing tasks for humans, the humans have incentive to help the device overcome its limitations to complete its tasks effectively.

- **A human-centered planning approach in which we study how humans interact with devices, and then use those models of humans in planning algorithms**.

  In our human-centered planning approach, a device reasons about humans in the environment in addition to its own state and goals to determine which actions to take. In particular, we incorporate models of human state, such as availability and accuracy, and asking actions into devices' own state and actions. Using this new model, our planning algorithms can proactively weigh the costs and benefits of different actions in different states to determine an action plan. Because the actions are based on both the human state and the device's state, the planner optimizes not only the task goals but also usability during deployment. This thesis contributes a formal approach to human-centered planning that focuses on building human models that accurately represent the tradeoffs that humans make when interacting with their intelligent devices in the environment.

- **A classification of two types of help that devices can request help for and two types of humans that devices can request help from.**

  We classify device limitations into two types: action limitations, reasoning uncertainty (for state and policy). In order to overcome action limitations, humans must perform those actions for the device. In order to overcome state or policy reasoning uncertainty, humans indicate which state or action the device is in, increasing the device's capability of then continuing to perform autonomously. While a robot could reduce its reasoning uncertainty and learn to perform tasks it requests help for, we do not expect any robot to be able to overcome actuation limitations autonomously. For example, a robot without arms cannot ever lift a cup of coffee. When the device does not have limitations, it performs autonomously.

  This thesis examines two different types of humans in the environment that could help devices overcome these limitations - device users who are near the robot and receive task benefit and environment occupants of buildings who have predefined work spaces and conduct work which requires that they be present in the environment over a period of time but cannot monitor the device's progress. We contribute algorithms for overcoming different limitations with different types of human helpers.

- **Human-centered planning algorithms (conditional, deliberative, and replanning) that model human helpers to plan device actions.**

  We contributed three conditional plans for devices to use to perform tasks that include help requests from different types of humans. First, we contributed conditional plans for completing each of CoBot's user-requested tasks in our multi-floor building. The plans included reducing state localization uncertainty and helped the robot overcome manipulation limitations to use the elevator. Second, we contributed a human-centered conditional planner for CoBot that requests help to reduce uncertainty while navigating. The planner explicitly models state costs to determine when to ask for help and when to act autonomously. Finally, we contributed a human-centered conditional planner for a smart phone to learn to reduce uncertainty about when it should ring and when it should be on silent mode. This algorithm trades off the cost of asking humans for help with the incentives for them to answer.

  We then contributed a human-centered probabilistic deliberative planner for CoBot to use to determine which navigational path to take to minimize uncertainty and maximize the likelihood of finding available help. The HOP-POMDP planner models the availability, accuracy and human cost of help along with the robot uncertainty and capabilities. Rather than taking the shortest path and waiting for help, the HOP-POMDP plan chooses the highest reward path that may be longer with less uncertainty or more humans available to help in order to reduce the cost of backtracking and errors. We also extend this planner to include policies for determining which help location to navigate to to ask for actuation help.

Finally, we recognized that many of CoBot's actuation limitations are spatially-situated in particular help locations. Based on an analysis of the deployment results from the conditional planners, we contributed our SSAH replanning algorithm to proactively bring environment occupant helpers from offices to help locations such as elevators rather than wait indefinitely at the help locations. The algorithm takes into account models of human availability, interruptibility, and cost of help as well as interaction history of questions over time to determine who to ask for help. We show that the human-centered replanning algorithm, which first waits at the help location and then navigates away, completes tasks faster than only waiting at the help location while limiting the number of people in offices that it asks.

- **Validation of the human centered planning algorithms in simulation and deployment in the environment against both naïve uses of human state as well as state-of-the-art algorithms that do not model humans.**

  The human-centered planning algorithms were all tested in realistic yet simulated environments to understand how they behaved with a variety of humans. In order to do so, we randomly generated availabilities and accuracies and other parameters of humans in different locations in building graphs. We demonstrate that our algorithms behave appropriately in each environment. Then, we demonstrated its use in our building environment as well to show that they do scale to many humans and how it works in real-world spaces. We, also, deployed those algorithms on the real robot to understand how help works in practice.

- **An increased understanding of two types of human helpers and how their human state affects their willingness to respond to device questions through multiple survey, in-lab, and *in situ* studies.**

  Our human-centered planning algorithms use models of humans in the environment to determine device actions and whether and who to ask for help. We contributed studies of both device users and occupants in the environment to demonstrate our human-centered planning algorithms should include models of each of these attributes. First, we contribute two studies of device users to demonstrate that they have their own costs and incentives to help their devices. We survey phone users to measure their personalized costs of help and incentive to help their phone learn their volume preferences, and then describe a short study of the cost and incentive of 5 visitors to help answer CoBot's localization questions as it escorts them to meetings around our building. Second, we contribute two studies of environment occupants. We survey building occupants to demonstrate that all the factors affect their willingness to travel down the hall to help CoBot use the elevator, and then describe a wizard-of-oz CoBot experiment in which the robot actually asks occupants to leave their offices.

- **Learning algorithms for capturing personalized models of human states during deployment for use in the planning algorithms**

  One major challenge in our human-center planners is generating the human model parameters. We contributed three algorithms to learn the human models. The first algorithm used linear regression techniques to generalize responses to survey questions. The second algorithm proactively and dynamically changed the survey questions to more quickly learn the models. Finally, the third algorithm uses the deliberative planning algorithm and learns the model online as the robot performs tasks. We demonstrate that all three learn accurate models of humans and the latter two also do so in a human-centered way - learning quickly and with minimal human effort.

- **Use of human-centered planning algorithms to determine whether, how, and who to ask for help and where to navigate to get the help.**

  We have applied our algorithms to several different aspects of asking for help. Our conditional algorithms determine whether to ask for help to both users and environment occupants. Our deliberative and replanning algorithms trade off uncertainty and task performance with cost of help to determine where to navigate and who to ask for help. Finally, our user studies guided our questions to determine how to ask for help. We showed that providing humans with device context, classification prediction and uncertainty, and additional feedback all increase the accuracy of human responses to device questions.

9.2 Future Work

There are several ways to extend human-centered planning to further understand humans as well as relax the assumptions we make about them.

- **Personalized Dialog with Humans**

  Because this thesis assumes that devices are deployed long term for users, we argue that the usability of the devices is important. Devices, such as CoBot often use dialog systems as their main interaction with users. However, dialog systems, like CoBot's, have traditionally assumed short term interactions and have employed static algorithms. In practice, this is not usable for long-tern deployments. For example, throughout our deployment of CoBot, we found that users became confused when it asked the same questions multiple times in the same day. Additionally, they became bored with interaction when it greeted them identically for each task. Future work is needed to understand how this dialog affects long term usability.

  Humans, instead, change their dialog over time to express a change in relationship and familiarity with a person. Because our algorithms take into account interaction history, it would be possible for future work to develop algorithms that take into account previous interactions and vary the dialog over time. The Carnegie Mellon robot commentators changed their dialog probabilistically based on previously-said expressions (Veloso et al. (2006)). Future work would also be needed to understand how people expect to interact long term with a device. For example, it would be interesting to know how often dialog should change what personal touches should belong in dialog. Additionally, future work is needed to understand how to identify people and overcome uncertainty in identity predictions as well as how new and changing dialog affects the device usability over time.

- **Learning Model Parameters**

  The algorithms contributed in this thesis assume that we have prior probabilities and costs collected from surveys to use in planning algorithms. These surveys are extensive in order to collect enough data to create accurate prior probabilities. We have also contributed algorithms that learn online rather than using priors; however, these online learning techniques trade off learning speed with computability and tractability of optimal policies. Future work is needed to understand how the learning occurs in real world deployments, given few interactions with humans each day. It is important for these algorithms to learn quickly and with few observations in order to be usable in large environments. Additionally, our current estimates err on the side of caution and predict less availability and higher costs in order to increase the usability even for inaccurate models. Future work is needed to understand whether this is a reasonable assumption as well as understand how learning affects deployability.

- **Prediction of Model Parameters**

This thesis assumes that models of humans are stationary with respect to within task time as well as throughout deployment. While it does assume that models could change through the day, it does not make predictions about how models will change while planning each task. In order to be able to predict the parameters over time, future work would be needed to collect hundreds of very fine-grained data points for each person at each time of day about parameters such as availability, interruption, accuracy, etc. Currently, we are only able to collect occupancy data on a 10-minute time scale, far longer than the average task takes to complete. With this additional information, we could predict future values given current ones.

Given this prediction algorithm, future work would then need to contribute planning algorithms that reason over possibly-infinite future prediction times to plan *when* to take actions. Additionally, these algorithms should run in real time and include replanning in order to take into account additional information collected during task execution. Possible ways to reduce the complexity of the problem include including time bins to discretize predictions and including a finite prediction horizon if replanning is used.

- **Incentives for Humans to Help**

  This thesis assumes that humans are in a symbiotic relationship with their devices and have incentive to help because they request tasks and receive benefit from them. However, some environment occupants in particular may have more incentive to help despite requesting few tasks if they are given other incentives (*e.g.,* thank you's, candy or coffee, or performing tasks without requesting the robot). In order to be able to reason about incentives, CoBot would need to model how much benefit each incentive contributes to each human's reward function. Different people may have different reward functions for different incentives and we would need to model them in order to personalize interaction. Additionally, it would need to create and add its own tasks to plans to complete these incentives. This requires trading off benefits of completing requested tasks with benefits of providing incentives to improve future interactions. Future work is also required to understand this tradeoff in long term deployment.

Additionally, there are many future directions for applying human-centered planning to other domains.

- **Learning By Demonstration**

  Learning by Demonstration is a technique that should be easily adoptable by even novice users. Most work in this area, particularly on robots, has focused on improving the learning algorithms themselves and assumes that demonstrators are experts that are available whenever the system needs a new demonstration. In order to be able to deploy such demonstration systems in real world settings, the action planning algorithms need to be able to reason about which possibly-novice people might be available to demonstrate, and must be able deal with possible delays or imperfect help demonstrations. While my thesis work modeled some of these human constraints, it assumed there was a symbiotic relationship in which the humans knew how to and wanted to help the systems. Using models of humans, learning by demonstration algorithms may be able to be deployed to real??world users as a tool to personalize and improve the performance of their intelligent systems.

- **Crowdsourcing**

  The internet has become a valuable resource to recruit crowds of volunteers to perform short tasks cheaply. Most people who use crowdsourcing depend on it to train data for offline classification rather than for tasks with real??time components. My thesis work focused on asking people in the intelligent systems environment for help rather than using crowdsourcing because the devices could use the idea of the symbiotic relationship to increase the likelihood of receiving fast and accurate responses. Additionally, there is some concern about how quickly responses can be received on crowd sourcing websites. However, by modeling the volunteers on websites like Mechanical Turk and their patterns of use on the site, algorithms could proactively post tasks or questions without having to ask people in the systems environment. The questions could be posted at times when there are more accurate or more specialized people available to reduce the number of responses, monetary cost, or time to complete tasks. These algorithms would need to reason over hundreds of thousands or millions of people online to receive help or feedback performing real-?time tasks.

- **Accessibility**

  People with disabilities and the elderly are limited in how they interact with computers. I believe there are many opportunities to automatically adapt interfaces to their users in order to increase usability and task performance. For example, graphical user interfaces could use knowledge about their users to determine the size of text and buttons, the background colors, and even the layout automatically rather than depending on add-?on products to modify the existing layout. Additionally, touch interfaces could incorporate more sophisticated prediction algorithms to understand users intended actions rather than depending on precise button clicks. There are also opportunities for intelligent systems to help people with disabilities or the elderly interact with and use other products more easily. With a model of low-?vision users, for example, intelligent devices could magnify or read printed text aloud, determine and describe the elements of the environment that the user might miss, and assist them in navigating through those environments. Accurate, usable, deployable algorithms could help people with disabilities use technology more effectively.

Appendix A

Surveys

You can find the appendix of surveys online at www.cs.cmu.edu/ srosenth/research/appendix.pdf.

Bibliography

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*. 7.2.1, 8.1

Aberdeen, D. 2003. A (revised) survey of approximate methods for solving pomdps. *National ICT Australia, Technical Report*. 3.2.3

Angluin, D. 1988. Queries and concept learning. *Machine Learning* 2(4):319–342. 8.3.1

Antifakos, S.; Schwaninger, A.; and Schiele, B. 2004. Evaluating the e?ects of displaying uncertainty in context-aware applications. In *UbiComp 2004*, 54–69. 6.1, 6.3.1, 6.5.1, 8.5

Argall, B.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483. 5, 6

Argall, B.; Browning, B.; and Veloso, M. 2007. Learning by demonstration with critique from a human teacher. In *HRI '07: ACM/IEEE International Conference on Human Robot Interaction*, 57–64. 8.1, 8.2.1, 8.3.3

Argall, B.; Browning, B.; and Veloso, M. 2008. Learning robot motion control with demonstration and advice-operators. In *IROS 08*. 7.2.1

Arlitt, M. F., and Williamson, C. L. 1997. Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking* 5(5):631. 4.3.3

Armstrong-Crews, N., and Veloso, M. 2007. Oracular pomdps: A very special case. In *ICRA '07*, 2477–2482. 3.1, 3.2, 3.2.2, 3.2.3, 8.2.2, 8.3.2

Asoh, H.; Motomura, Y.; Hara, I.; Akaho, S.; Hayamizu, S.; and Matsui, T. 1996. Acquiring a probabilistic map with dialogue-based learning. In *Proceedings of ROBOLEARN-96*, 11–18. 6, 8.1

Asoh, H.; Hayamizu, S.; Hara, I.; Motomura, Y.; Akaho, S.; and Matsui, T. 1997. Socially embedded learning of the office-conversant mobile robot jijo-2. In *IJCAI-97*, 880–885. 2.2.1, 4, 5, 6, 6.1, 8.5

Atkeson, C. G., and Schaal, S. 1997. Robot learning from demonstration. In *International Conference on Machine Learning*, 12–20. 8.1, 8.2.1

Balog, K., and de Rijke, M. 2006. Finding experts and their eetails in e-mail corpora. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, 1035–1036. 8.4.1

Banbury, S.; Seldcon, S.; Endsley, M.; Gordon, T.; and Tatlock, K. 1998. Being certain about uncertainty: How the representation of system reliability affects pilot decision making. In *Human Factors and Ergonomics Society 42nd Annual Meeting*. 6.1, 8.5

Barret, L., and Barrett, D. 2001. An introduction to computerized experience sampling in psychology. *Social Science Computer Review* 19(2):175–185. 8.1, 8.3.1

Bellotti, V., and Edwards, K. 2001. Intelligibility and accountability: human considerations in context-aware systems. *Human.-Computer Interaction* 16(2):193–212. 6, 6.1, 8.5

Bernstein, D. S.; Zilberstein, S.; and Immerman, N. 2000. The complexity of decentralized control of markov decision processes. In *UAI '00: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 32–37. 8.2.2, 8.3.2, 8.4.2

Biswas, J., and Veloso, M. 2010. Wifi localization and navigation for autonomous indoor mobile robots. In *ICRA*, 4379–4384. 1.2.2, 1.2, 2.1, 2.1, 6.3.1, 8.1

Biswas, J.; Coltin, B.; and Veloso, M. 2011. Corrective gradient renement for mobile robot localization. In *IROS*, 73 – 78. 1.2.2, 1.2, 2.1, 2.1

Bouguessa, M.; Dumoulin, B.; and Wang, S. 2008. Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 866–874. 8.4.1

Browning, B.; Xu, L.; and Veloso, M. 2004. Skill acquisition and use for a dynamically-balancing soccer robot. In *Nineteenth National Conference on Artificial Intelligence (AAAI04)*, 599–604. 8.1

Bruemmer, D. J.; Few, D. A.; Boring, R. L.; Marble, J. L.; Walton, M. C.; and Nielsen, C. W. 2005. Shared understanding for collaborative control. *IEEE Transactions on Systems, Man, and Cybernetics* 35(4):494–504. 8.2.3

Cai, C.; Liao, X.; and Carin, L. 2009. Learning to explore and exploit in pomdps. In *NIPS*, 198–206. 7.3, 7.3.2

Callison-Burch, C. 2009. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, 286–295. 8.4.4

Campbell, C. S.; Maglio, P. P.; Cozzi, A.; and Dom, B. 2003. Expertise identification using email communications. In *CIKM '03: Proceedings of the twelfth international conference on Information and knowledge management*, 528–531. 8.4.1

Carter, S., and Mankoff, J. 2005. When participants do the capturing: the role of media in diary studies. In *CHI 2005*, 899–908. 8.3.1

Carter, S.; Mankoff, J.; and Heer, J. 2007. Momento: support for situated ubicomp experimentation. In *CHI '07*, 125–134. 8.6.2

Cassandra, A. 2011. pomdp-solve software, version 5.3. In *http://www.pomdp.org/pomdp/code/index.shtml*. 3.3

Chernova, S., and Veloso, M. 2007. Confidence-based policy learning from demonstration using gaussian mixture models. In *AAMAS '07: 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, 1–8. 8.2.1, 8.3.3

Chernova, S., and Veloso, M. 2008a. Multi-thresholded approach to demonstration selection for interactive robot learning. In *HRI '08*, 225–232. 4, 8.3.3

Chernova, S., and Veloso, M. 2008b. Teaching multi-robot coordination using demonstration of communication and state sharing. In *AAMAS '08: 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, 1183–1186. 8.1

Chernova, S. 2009. *Confidence-Based Robot Policy Learning from Demonstration*. Ph.D. Dissertation, Carnegie Mellon University. 8.2.1

Clark, H., and Wilkes-Gibbs, D. 1986. Referring as a collaborative process. *Cognition*. 6, 6.1, 8.5

Clark, H. H.; Schober, M. F.; Tanur, J. M.; and Tanur, J. M. 1992. *Asking questions and influencing answers.* 15–48. people interpret others' questions in different ways, so we need to be careful when forming questions that we aren't influencing the answers and that our questions are understandable to a wide audience (including adding grounding and references). 8.5

Clark, H. H. 2008. Talking as if. In *HRI*, 393–394. 6

Cohn, D.; Atlas, L.; and Ladner, R. 1994. Improving generalization with active learning. *Machine Learning* 15(2):201–221. 2.3.1, 6, 8.1, 8.2.1, 8.3.1

Coltin, B.; Biswas, J.; Pomerleau, D.; and Veloso, M. 2011. Effective semi-autonomous telepresence. In *RoboCup Symposium*. 2.1

Consolvo, S., and Walker, M. 2003. Using the experience sampling method to evaluate ubicomp applications. *IEEE Pervasive Computing* 2(2):24–31. 8.3.1

Consolvo, S.; McDonald, D.; Toscos, T.; Chen, M.; Froehlich, J.; Harrison, B.; Klasnja, P.; LaMarca, A.; LeGrand, L.; Libby, R.; Smith, I.; and Landay, J. 2008. Activity sensing in the wild: a field trial of ubifit garden. In *CHI 2008*, 1797–1806. 6

Csikszentmihalyi, M., and Larson, R. 1987. Validity and reliability of the experience sampling method. *The Journal of Nervous and Mental Disease* 175(9):526–536. 2.3, 2.3.1, 8.6.2

Culotta, A.; Kristjansson, T.; McCallum, A.; and Viola, P. 2006. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence* 170(14-15):1101–1122. 6.1, 8.3.3, 8.5, 8.5

Czerwinski, M.; Cutrell, E.; and Horvitz, E. 2000. Instant messaging and interruption: Influence of task type on performance. In *OZCHI 2000*, 356–361. 8.6.3

Czerwinski, M.; Horvitz, E.; and Wilhite, S. 2004. A diary study of task switching and interruptions. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, 175–182. 8.6.3

Dantzig, G. B. 1963. *Linear Programming and Extensions*. Princeton University Press. 7.2.2

DePaulo, B. M., and Fisher, J. D. 1980. The costs of asking for help. *Basic and Applied Social Psychology* 1:23 – 35. 8.6.2

Donmez, P., and Carbonell, J. G. 2008. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *CIKM 2008*, 619–628. 5.4.4, 6, 8.2.1, 8.4.1

Donmez, P.; Carbonell, J. G.; and Schneider, J. 2009. Efficiently learning the accuracy of labeling sources for selective sampling. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 259–268. 5.4.4, 8.2.1, 8.4.4

Dorais, G. A.; Bonasso, R. P.; Kortenkamp, D.; Pell, B.; and Schreckenghost, D. 1999. Adjustable autonomy for human-centered autonomous systems. In *IJCAI Workshop on Adjustable Autonomy Systems*, 16–35. 8.1, 8.2.3

Doshi, F.; Pineau, J.; and Roy, N. 2008. Reinforcement learning with limited reinforcement: using bayes risk for active learning in pomdps. In *ICML '08*, 256–263. 7.3, 7.3.1, 7.3.3

Eagle, M., and Leiter, E. 1964. Recall and recognition in intentional and incidental learning. *Journal of experimental psychology* 68:58–63. 6.1, 7.2.1, 8.1, 8.5

Erickson, T., and Kellogg, W. A. 2001. Social translucence: an approach to designing systems that support social processes. *ACM ToCHI* 7(1):59–83. 6, 6.1, 8.5

Faulring, A.; Myers, B.; Mohnkern, K.; Schmerl, B.; Steinfeld, A.; Zimmerman, J.; Smailagic, A.; Hansen, J.; and Siewiorek, D. 2010. Agent-assisted task management that reduces email overload. In *IUI '10: Proceeding of the 14th international conference on Intelligent user interfaces*, 61–70. 2.3, 2.4, 6.1, 8.1, 8.3.3, 8.5

Ferguson, G.; Allen, J. F.; and Miller, B. 1996. Trains-95: Towards a mixed-initiative planning assistant. In *3rd International Conference on Artificial Intelligence Planning Systems*, 70–77. 8.1

Fogarty, J.; Hudson, S. E.; Atkeson, C. G.; Avrahami, D.; Forlizzi, J.; Kiesler, S.; Lee, J. C.; and Yang, J. 2005. Predicting human interruptibility with sensors. *ACM ToCHI* 12(1):119–146. 2.3, 5, 5.4.4, 6, 6.1, 6.3.6, 8.5, 8.6.3

Fong, T. W.; Thorpe, C.; ; and Baur, C. 2001. Collaboration, dialogue, and human-robot interaction. In *10th International Symposium of Robotics Research*. 8.2.3

Fong, T. W.; Thorpe, C.; and Baur, C. 2003. Robot, asker of questions. In *Robotics and Autonomous Systems*, volume 42, No. 3-4, 235–243. 4, 5, 6, 8.1, 8.2.3, 8.3.3, 8.4.1, 8.6.1

Fong, T. W. 2001. *Collaborative Control: A Robot-Centric Model for Vehicle Teleoperation*. Ph.D. Dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. 8.5

Ghallab, M.; Howe, A.; Knoblock, C.; McDermott, D.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL: the planning domain definition language. *AIPS-98 planning committee*. 2.2

Goldman, C. V., and Zilberstein, S. 2003. Optimizing information exchange in cooperative multi-agent systems. In *AAMAS '03: Second International Conference on Autonomous Agents and Multiagent Systems*, 137–140. 8.2.2, 8.3.2

Goldman, C. V., and Zilberstein, S. 2004. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research* 22:143174. 8.2.2

Goodrich, M. A., and Schultz, A. C. 2007. Human-robot interaction: a survey. *Found. Trends Human-Computer Interaction* 1(3):203–275. 8.4

Green, P., and Wei-Haas, L. 1985. The rapid development of user interfaces: Experience with the wizard of oz method. *Human Factors and Ergonomics Society Annual Meeting* 29(5):470–474. 5, 5.4, 6.2, 6.6, 8.1

Grollman, D. H., and Jenkins, O. C. 2007. Learning robot soccer skills from demonstration. In *International Conference on Development and Learning*, 276–281. 4

Hacker, S., and von Ahn, L. 2009. Matchin: eliciting user preferences with an online game. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, 1207–1216. 8.4.4

Hayes, G., and Demiris, J. 1994. A robot controller using learning by imitation. In *2nd International Symposium on Intelligent Robotic Systems*, 198–204. 5, 8.1

Hayes, G.; Kientz, J.; Truong, K.; White, D.; Abowd, G.; and Pering, T. 2004. Designing capture applications to support the education of children with autism. In *UbiComp 04*, 161–178. 8.1

Hearst, M. A. 1999. Mixed-initiative interaction: Trends and controversies. *IEEE Intelligent Systems* 14–23. 8.1

Heberlein, T. A., and Baumgartner, R. 1978. Factors affecting response rates to mailed surveys: A quantitative analysis of the published literature. *American Sociological Review* 43:447–462. 7, 7.2, 7.2.1, 7.2.3

Herlocker, J. L.; Konstan, J. A.; Borchers, A.; and Riedl, J. 1999. An algorithmic framework for performing collaborative filtering. In *SIGIR*, 230–237. 8.4.3

Ho, J., and Intille, S. 2005. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *CHI*, 909–918. 2.3

Hoffmann, R.; Amershi, S.; Patel, K.; Wu, F.; Fogarty, J.; and Weld, D. S. 2009. Amplifying community content creation with mixed initiative information extraction. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, 1849–1858. 6.1, 8.5

Horvitz, E., and Apacible, J. 2003. Learning and reasoning about interruption. In *International Conference on Multimodal Interfaces (ICMI)*, 20–27. 2.3, 8.6.3

Horvitz, E.; Koch, P.; Kadie, C.; and Jacobs, A. 2002. Coordinate: Probabilistic forecasting of presence and availability. In *Conference on Uncertainty and Artificial Intelligence (UAI 2002)*, 224–233. 8.6.3

Horvitz, E.; Koch, P.; Sarin, R.; Apacible, J.; and Subramani, M. 2005. Bayesphone: Precomputation of context-sensitive policies for inquiry and action in mobile devices. In *User Modeling*, 251–260. 2.3

Horvitz, E.; Jacobs, A.; and Hovel, D. 1999. Attention-sensitive alerting. In *Conference on Uncertainty and Artificial Intelligence (UAI 1999)*, 305–313. 8.6.3

Horvitz, E.; Koch, P.; and Apacible, J. 2004. Busybody: creating and fielding personalized models of the cost of interruption. In *CSCW '04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, 507–510. 6, 6.1, 6.3.6, 8.3.1, 8.5, 8.6.3

Horvitz, E. 1999. Principles of mixed-initiative user interfaces. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, 159–166. 6, 6.1, 8.1, 8.2.1, 8.3.1, 8.5, 8.6.2

Howard, R. A. 1960. *Dynamic Programming and Markov Processes*. MIT Press. 8.3.2

Hsieh, G.; Li, I.; Dey, A.; Forlizzi, J.; and Hudson, S. 2008. Using visualizations to increase compliance in experience sampling. In *UbiComp '08*, 164–167. 8.6.2

Hüttenrauch, H., and Eklundh, S. 2006. To help or not to help a service robot: Bystander intervention as a resource in human-robot collaboration. *Interaction Studies* 7(3):455–477. 5

Isaacs, E. A., and Clark, H. H. 1987. References in conversation between experts and novices. *Journal of Experimental Psychology: General* 116:26–37. 8.4.1

Jameson, A.; Gromann-hutter, B.; March, L.; Rummer, R.; Bohnenberger, T.; and Wittig, F. 2000. When actions have consequences: Empirically based decision making for intelligent user interfaces. *Knowledge-Based Systems* 14:75–92. 8.6.2

Jaulmes, R.; Pineau, J.; and Precup, D. 2005. Active learning in partially observable markov decision processes. In *ECML 2005*, volume 3720, 601–608. 7.3, 7.3.1, 7.3.3

Jeffries, R.; Miller, J. R.; Wharton, C.; and Uyeda, K. 1991. User interface evaluation in the real world: a comparison of four techniques. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, 119–124. 8.5

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998a. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2). 3.2, 3.2.2, 3.3

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998b. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134. 8.3.2

Kaelbling, L. P.; Littman, M.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285. 8.1

Kahneman, D.; Krueger, A.; Schkade, D.; Schwarz, N.; and Stone, A. 2004. A survey method for characterizing daily life experience: The day reconstruction method. *Science* 306(5702):1776–1780. 8.6.2

Kapoor, A., and Horvitz, E. 2007. On discarding, caching, and recalling samples in active learning. In *Uncertainty in Artificial Intelligence (UAI)*, 209–216. 2.4

Kapoor, A., and Horvitz, E. 2008. Experience sampling for building predictive user models: a comparative study. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, 657–666. 2.3.1, 5.1.2, 8.2.1, 8.6.3

Karami, A.-B.; Jeanpierre, L.; and Mouaddib, A.-I. 2009. Partially observable markov decision process for managing robot collaboration with human. *Tools with Artificial Intelligence* 0:518–521. 3.2

Karlin, A. R.; Manasse, M. S.; McGeoch, L. A.; and Owicki, S. 1994. Competitive randomized algorithms for non-uniform problems. *Algorithmica* 11(6):542–571. 4.3.3

Katagami, D., and Yamada, S. 2001. Real robot learning with human teaching. In *4th Japan-Australia Joint Workshop on Intelligent and Evolutionary Systems*, 263–270. 4, 8.1

Kearns, M., and Singh, S. 2002. Near-optimal reinforcement learning in polynomial time. *Machine Learning* 49:209–232. 7.3, 7.3.2

Kern, N., and Schiele, B. 2006. Towards personalized mobile interruptibility estimation. In *International Workshop on Location- and Context-Awareness*, 134–150. 2.3

Khalil, A., and Connelly, K. 2005. Improving cell phone awareness by using calendar information. In *International Conference on Human-Computer Interaction (INTERACT)*, 588–600. 2.3, 5.1

Krishnan, M. 2008. Availability and mobile phone interruptions: Examining the role of technology in coordinating mobile calls. Master's thesis, Blekinge Institute of Technology. 2.3

Law, E.; Mityagin, A.; and Chickering, M. 2009. Intentions: a game for classifying search query intent. In *CHI '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, 3805–3810. 8.4.4

Lee, M. K.; Kielser, S.; Forlizzi, J.; Srinivasa, S.; and Rybski, P. 2010. Gracefully mitigating breakdowns in robotic services. In *HRI '10: 5th ACM/IEEE International Conference on Human Robot Interaction*, 203–210. 5, 5.4.4

Lehmann, E. L. 1950. Some principles of the theory of testing hypotheses. *Annals of Mathematical Statistics* 21(1):1–26. 4.3.2

Lewis, D. D., and Catlett, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In *11th International Conference on Machine Learning*, 148–156. Morgan Kaufmann. 2.3.1, 8.3.1

Lewis, D. D., and Gale, W. A. 1994. A sequential algorithm for training text classifiers. In *SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, 3–12. 8.1

Lim, B. Y.; Dey, A. K.; and Avrahami, D. 2009. Why and why not explanations improve the intelligibility of context-aware intelligent systems. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, 2119–2128. 2.4

Littman, M. L.; Cassandra, A. R.; and Kaelbling, L. P. 1995. Learning policies for partially observable environments: Scaling up. In *ICML*, 362–370. 3.2

Lockerd, A., and Breazeal, C. 2004. Tutelage and socially guided robot learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3475 – 3480. 5, 8.1, 8.2.1

Ma, H.; Chandrasekar, R.; Quirk, C.; and Gupta, A. 2009. Improving search engines using human computation games. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, 275–284. 8.4.4

Madani, O. 2000. Complexity results for infinite-horizon markov decision processes. *Ph.D. dissertation, University of Washington*. 3.2.3

Mankoff, J.; Abowd, G.; and Hudson, S. 2000. Oops: a toolkit supporting mediation techniques for resolving ambiguity in recognition-based interfaces. *Computers and Graphics* 24(6):819–834. 6.1, 8.3.3, 8.5

Marge, M.; Banerjee, S.; and Rudnicky, A. I. 2010. Using the amazon mechanical turk for transcription of spoken language. In *ICASSP-2010*. 8.4.4

McFarlane, D. C. 1999. Coordinating the interruption of people in human-computer interaction. In *Human-Computer Interaction-INTERACT '99*, 295–303. 2.3.1, 8.6.3

McFarlane, D. 2002. Comparison of four primary methods for coordinating the interruption of people in human-computer interaction. *Human-Computer Interaction* 17(1):63–139. 8.6.3

Mcnee, S.; Lam, S. K.; Guetzlaff, C.; Konstan, J. A.; and Riedl, J. 2003. Confidence displays and training in recommender systems. In *Proceedings of the 9th IFIP TC13 International Conference on HumanComputer Interaction (INTERACT)*, 176–183. IOS Press. 6.1, 8.5

Melo, F. S., and Veloso, M. M. 2009. Learning of coordination: exploiting sparse interactions in multiagent systems. In *AAMAS*, 773–780. 8.3.2

Metropolis, N., and Ulam, S. 1949. The monte carlo method. *Journal of the American Statistical Association* 44:335. 7.2.2

Michalowski, M.; Sabanovic, S.; DiSalvo, C.; Busquets, D.; Hiatt, L.; Melchior, N.; and Simmons, R. 2007. Socially distributed perception: Grace plays social tag at aaai 2005. *Autonomous Robots* 22(4):385–397. 5

Milewski, A., and Smith, T. 2000. Providing presence cues to telephone users. In *Conference on Computer Supported Cooperative Work (CSCW)*, 89–96. 1.2.1, 2.3

Mitchell, T. 1997. *Machine Learning*. McGraw Hill. 6

Mockus, A., and Herbsleb, J. D. 2002. Expertise browser: a quantitative approach to identifying expertise. In *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, 503–512. 8.4.1

M.Turk. 2010. Amazon.com mechanical turk – artificial, artificial intelligence. 8.4.4

Nair, R.; Roth, M.; and Yohoo, M. 2004. Communication for improving policy computation in distributed pomdps. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 1098–1105. 8.3.2, 8.4.2

Nakamura, A., and Abe, N. 1998. Collaborative filtering using weighted majority prediction algorithms. In *International Conference on Machine Learning*, 395–403. 8.4.3

Ng, A. Y., and Russell, S. 2000. Algorithms for inverse reinforcement learning. In *ICML*, 663–670. 7.2.1

Nguyen H, S. A. 2004. Active learning using pre-clustering. In *International Conference on Machine Learning (ICML)*, 623–630. 8.3.1

Nicolescu, M. N. 2003. *A framework for learning from demonstration, generalization and practice in human-robot domains*. Ph.D. Dissertation, University of Southern California. 8.1

Paek, T.; Ju, Y.-C.; and Meek, C. 2007. People watcher: A game for eliciting human-transcribed data for automated directory assistance. In *INTERSPEECH '07*, 1322–1325. 8.4.4

Papadimitriou, C., and Tsisiklis, J. 1987. The complexity of markov decision processes. *Mathematics of Operations Research* 12(3):441450. 3.2.3

Parasuraman, R.; Sheridan, T.; and Wickens, C. 2000. A model for types and levels of human interaction with automation. *IEEE Transactions on Systems, Man and Cybernetics* 30(3):286–297. 8.1

Pearson, J.; Hu, J.; Branigan, H. P.; Pickering, M. J.; and Nass, C. I. 2006. Adaptive language behavior in hci: how expectations and beliefs about a system affect users' word choice. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems*, 1177–1180. 6.4.5

Plutowski, M., and White, H. 1993. Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks* 4(2):305318. 8.3.1

Presser, S. 2004. Methods for testing and evaluating survey questions. *Public Opinion Quarterly* 68(1):109–130. 8.5

Price, R. R. 2003. *Accelerating reinforcement learning through imitation*. Ph.D. Dissertation, The University of British Columbia (Canada). Adviser-Boutilier, Craig. 5, 8.1, 8.2.1

Pynadath, D. V., and Tambe, M. 2002. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research* 16:389–423. 8.3.2, 8.4.2

Raghavan, H.; Madani, O.; and Jones, R. 2006a. Active learning with feedback on features and instances. *Journal of Machine Learning Research* 7:1655–1686. 6.1, 6.5.1, 8.1, 8.5

Raghavan, H.; Madani, O.; and Jones, R. 2006b. Active learning with feedback on features and instances. *Journal of Machine Learning Research* 7:1655–1686. 6.3.8

Regan, K., and Boutilier, C. 2011. Eliciting additive reward functions for markov decision processes. *IJCAI*. 7.2.1

Rosenthal, S., and Dey, A. K. 2010. Towards maximizing the accuracy of human-labeled sensor data. In *Intelligent User Interfaces (IUI)*, 259–268. 6.6

Rosenthal, S., and Veloso, M. 2011. Modeling humans as observation providers using pomdps. In *Ro-Man*. 3.1

Rosenthal, S., and Veloso, M. 2012. Mobile robot planning to seek help with spatially-situated tasks. In *AAAI*. 4.3, 5.3.1

Rosenthal, S.; Biswas, J.; and Veloso, M. 2010. An effective personal mobile robot agent through a symbiotic human-robot interaction. In *AAMAS '10*, 915–922. 5.2.1, 6, 6.2, 6.3.1

Rosenthal, S.; Dey, A. K.; and Veloso, M. 2009. How robots' questions affect the accuracy of the human responses. In *Ro-Man*, 1137–1142. 6.3.1, 6.6

Rosenthal, S.; Dey, A. K.; and Veloso, M. 2011. Using decision-theoretic experience sampling to build personalized mobile phone interruption models. In *International Conference on Pervasive Computing (Pervasive 2011)*, 170–187. 2.3.2, 5.1

Rosenthal, S.; Veloso, M.; and Dey, A. K. 2011. Learning accuracy and availability of humans who help mobile robots. In *AAAI*, 1501–1506. 3.1

Rosenthal, S.; Veloso, M.; and Dey, A. K. 2012a. Is someone in this office available to help me? proactively seeking help from spatially-situated humans. *Journal of Intelligent and Robotic Systems, Special Issue on Domestic Service Robots in the Real World* 66(1-2):205–221. 5.4, 6.6

Rosenthal, S.; Veloso, M.; and Dey, A. K. 2012b. Is someone in this office available to help? proactively seeking help from spatially-situated humans. *Journal of Intelligent and Robotic Systems* 66(1-2):205–221. 6.2

Roth, M.; Simmons, R.; and Veloso, M. 2005. Reasoning about joint beliefs for execution-time communication decisions. In *AAMAS*, 1098–1105. 8.3.2, 8.4.2

Roth, M.; Simmons, R.; and Veloso, M. 2007. Exploiting factored representations for decentralized execution in multiagent teams. In *AAMAS '07: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 1–7. 8.2.2, 8.3.2

Roth, M.; Vail, D.; and Veloso, M. M. 2003. A real-time world model for multi-robot teams with high-latency communication. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, 2494 – 2499. 8.4.2

Roy, N., and McCallum, A. 2001. Toward optimal active learning through sampling estimation of error reduction. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, 441–448. 8.3.1

Rui, Y.; Huang, T.; Ortega, M.; and Mehrotra, S. 1998a. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Trans. on Circuits/Systems for Video Technology* 8(5):644–655. 6.1

Rui, Y.; Huang, T.; Ortega, M.; and Mehrotra, S. 1998b. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Trans. on Circuits/Systems for Video Technology* 8(5):644–655. 8.5

Salton, G., and Buckley, C. 1990a. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science* 41:288–297. 6.1

Salton, G., and Buckley, C. 1990b. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science* 41:288–297. 8.5

Saunders, J.; Nehaniv, C. L.; and Dautenhahn, K. 2006. Teaching robots by moulding behavior and scaffolding the environment. In *HRI*, 118–125. 8.1

Scaffidi, C. 2009. *Topes: Enabling End-User Programmers to Validate and Reformat Data*. Carnegie Mellon Technical Report CMU-ISR-09-105. 6.1, 8.5

Scerri, P.; Xu, Y.; Liao, E.; Lai, J.; and Sycara, K. 2004. Scaling teamwork to very large teams. In *AAMAS '04: Third International Joint Conference on Autonomous Agents and Multiagent Systems*. 8.2.2

Schmidt-Rohr, S. R.; Knoop, S.; Lösch, M.; and Dillmann, R. 2008. Reasoning for a multi-modal service robot considering uncertainty in human-robot interaction. In *HRI '08*, 249–254. 3.2

Schoemaker, P. J. H. 1982. The expected utility model: Its variants, purposes, evidence and limitations. *Journal of Economic Literature* 20:529–563. 4.3.2

Scholtz, J. 2002. Evaluation methods for human-system performance of intelligent systems. Technical Report 990, NIST Special Publication. 8.4

Schwarz, N., and Hippler, H. J. 1995. Subsequent questions may influence answers to preceding questions in mail surveys. *Public Opinion Quarterly* 59(1):93–97. 7.2

Schwarz, N.; Knauper, B.; Hipler, H. J.; Noelle-Neumann, E.; and Clark, L. 1991. Numeric values may change the meaning of scale labels. *Public Opinion Quarterly* 55(4). 7.2

Scollon, C.; Kim-Prieto, C.; and Diener, E. 2003. Experience sampling: Promise and pitfalls, strengths and weaknesses. *Journal of Happiness Studies* 4:5–34. 2.3.1, 5.4.4, 8.6.2

Sellner, B. P.; Heger, F.; Hiatt, L.; Simmons, R.; ; and Singh, S. 2006. Coordinated multi-agent teams and sliding autonomy for large-scale assembly. *IEEE - Special Issue on Multi-Robot Systems* 94(1):1425 – 1444. 8.1, 8.2.3

Shadbolt, N., and Burton, A. M. 1989. The empirical study of knowledge elicitation techniques. *SIGART Bulletin* 108:15–18. 2.3, 6, 8.5

Shahaf, D., and Horvitz, E. 2010. Generalized task markets for human and machine computation. In *AAAI*. 8.4.4

Shilman, M.; Tan, D. S.; and Simard, P. 2006. Cuetip: a mixed-initiative interface for correcting handwriting errors. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, 323–332. 6.1, 8.1, 8.3.3, 8.5

Shiomi, M.; Sakamoto, D.; Takayuki, K.; Ishi, C. T.; Ishiguro, H.; and Hagita, N. 2008. A semi-autonomous communication robot: a field trial at a train station. In *HRI '08*, 303–310. 5, 5.4.4, 6, 6.1, 6.1, 8.2.3, 8.5

Spaan, M. T. J.; Gordon, G. J.; and Vlassis, N. 2006. Decentralized planning under uncertainty for teams of communicating agents. In *AAMAS '06: Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 249–256. 8.2.2

Steels, L. 2003. Evolving grounded communication for robots. *Trends in Cognitive Sciences* 7(7):308–312. 8.5

Steinfeld, A.; Bennett, R.; Cunningham, K.; Lahut, M.; Quinones, P.-A.; Wexler, D.; Siewiorek, D.; Cohen, P.; Fitzgerald, J.; Hansson, O.; Hayes, J.; Pool, M.; and Drummond, M. 2006. The radar test methodology: Evaluating a multi-task machine learning system with humans in the loop. Technical Report CMU-CS-06-125, Computer Science Department, Pittsburgh, PA. 6.3.4

Stumpf, S.; Bao, X.; Dragunov, A.; Dietterich, T. G.; Herlocker, J.; Li, L.; and Shen, J. 2005. Predicting user tasks: I know what youre doing. In *AAAI Workshop on Human Comprehensible Machine Learning*. Press. 6.1, 8.5, 8.5

Stumpf, S.; Rajaram, V.; Li, L.; Burnett, M.; Dietterich, T.; Sullivan, E.; Drummond, R.; and Herlocker, J. 2007a. Toward harnessing user feedback for machine learning. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, 82–91. 6.1, 6.2, 8.5

Stumpf, S.; Rajaram, V.; Li, L.; Burnett, M.; Dietterich, T.; Sullivan, E.; Drummond, R.; and Herlocker, J. 2007b. Toward harnessing user feedback for machine learning. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, 82–91. 6.2

Sutton, R., and Barto, A. 1998. *Reinforcement Learning: An Introduction*. MIT Press. 8.1

Thomaz, A. L., and Breazeal, C. 2006. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI*. 8.1, 8.6.1

Thomaz, A. L.; Hoffman, G.; and Breazeal, C. 2006. Reinforcement learning with human teachers: Understanding how people want to teach robots. In *Robot and Human Interactive Communication, 2006. ROMAN 2006*, 352–357. 8.6.1

Toninelli, A.; Khushraj, D.; Lassila, O.; and Montanari, R. 2008. Towards socially aware mobile phones. In *Social Data on the Web Workshop*. 1.2.1, 2.3

Topp, E.; Huttenrauch, H.; Christensen, H.; and Eklundh, K. 2006. Bringing together human and robotic environment representations–a pilot study. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 4946–4952. 8.5

Torrance, M. C. 1994. *Natural Communication with Robots*. Ph.D. Dissertation, MIT. 8.5

Veloso, M.; Armstrong-crews, N.; Chernova, S.; Crawford, E.; Mcmillen, C.; Roth, M.; and Vail, D. 2006. A team of humanoid game commentators. In *International Conference on Humanoid Robotics*, 228–233. 9.2

Veloso, M. M. 1996. Towards mixed-initiative rationale-supported planning. In *Advanced Planning Technology*, 277–282. 8.1

von Ahn, L., and Dabbish, L. 2004. Labeling images with a computer game. acm conference on human factors in computing systems. In *CHI 2004*, 319–326. 6, 6.1, 8.4.4, 8.5

von Ahn, L.; Maurer, B.; McMillen, C.; Abraham, D.; and Blum, M. 2008. recaptcha: Human-based character recognition via web security measures. *Science* 1465–1468. 6.1, 8.4.4, 8.5

Weiss, A.; Igelsböck, J.; Tscheligi, M.; Bauer, A.; Kühnlenz, K.; Wollherr, D.; and Buss, M. 2010. Robots asking for directions: the willingness of passers-by to support robots. In *HRI '10*, 23–30. 4, 5, 8.4.3

Weiten, W. 2011. *Psychology: Themes and Variations*. Wadsworth Publishing. 7.2.1

Yanco, H.; Drury, J. L.; and Scholtz, J. 2004. Beyond usability evaluation: analysis of human-robot interaction at a major robotics competition. *Human-Computer Interaction* 19(1):117–149. 6.1, 6.5.1

Zhang, J.; Ackerman, M. S.; and Adamic, L. 2007. Expertise networks in online communities: structure and algorithms. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, 221–230. 8.4.1