An Effective Personal Mobile Robot Agent Through Symbiotic Human-Robot Interaction

Stephanie Rosenthal
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
srosenth@cs.cmu.edu

Joydeep Biswas
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
joydeepb@cs.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
veloso@cs.cmu.edu

ABSTRACT

Several researchers, present authors included, envision personal mobile robot agents that can assist humans in their daily tasks. Despite many advances in robotics, such mobile robot agents still face many limitations in their perception, cognition, and action capabilities. In this work, we propose a symbiotic interaction between robot agents and humans to overcome the robot limitations while allowing robots to also help humans. We introduce a visitor's companion robot agent, as a natural task for such symbiotic interaction. The visitor lacks knowledge of the environment but can easily open a door or read a door label, while the mobile robot with no arms cannot open a door and may be confused about its exact location, but can plan paths well through the building and can provide useful relevant information to the visitor. We present this visitor companion task in detail with an enumeration and formalization of the actions of the robot agent in its interaction with the human. We briefly describe the wifi-based robot localization algorithm and show results of the different levels of human help to the robot during its navigation. We then test the value of robot help to the visitor during the task to understand the relationship tradeoffs. Our work has been fully implemented in a mobile robot agent, CoBot, which has successfully navigated for several hours and continues to navigate in our indoor environment.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Experimentation

Keywords

Human-robot/agent interaction

1. INTRODUCTION

Robotic technology has had many advances, but mobile robot agents are still not universally present in our daily environments (*e.g.,* robots in our offices [2][10] or malls [11]). While the ultimate goal is for robots to perform tasks autonomously, we realize that robots still have many limitations, at the perception, cognition, and execution levels. Interestingly, many of the limitations may not be limitations for humans who may coexist with robots in the environment. We investigate methods for robots to overcome their limitations by asking humans for help while also helping humans along other dimensions. We hence define a *symbiotic relationship* as one in which the robot performs tasks for humans and the humans in return help the robot agents. This relationship is in contrast to those in which the human or the robot is responsible for helping the other without benefit in return (*e.g.,* social learning [2] collaborative control [5][7], sliding autonomy [9]).

We introduce a visitor's companion robot agent as a natural task for such symbiotic interaction. In this task, a robot autonomously navigates in the building, accompanying a visitor through a schedule of meetings and satisfies other needs of the visitor (*e.g.,* [13][15][16]). In the visitor companion task, the visitor could lack knowledge of the good paths in the building, of specific locations or people. The limitations of a robot (without manipulation capabilities) include occasional confusion about its localization, and ability to open a door or get a cup of coffee. Robot and humans could help each other to overcome each others' limitations to mutually benefit in satisfying the goals of the task. We have developed a fully operational visitor companion robot agent, CoBot, which continues to successfully navigate in our indoor environment.

In this paper, we formalize the symbiotic relationship, which can be viewed as an asynchronous team coordination between the human and the robot. We introduce the state and actions of the robot in order to capture the symbiotic interaction with the human. We identify different types of equivalent robot actions, including physical execution and information actions. Each of the physical actions has a probability function of success or failure, where failure triggers an action to ask for help from the human rather than a failure to complete the task. We briefly describe the wifi-based robot localization and navigation algorithm and show how different amounts of human help for localization affect the robot's navigation. We then test the symbiotic relationship with real visitors to understand the tradeoffs between the value of robot help and and the costs of the robot's questions.

The paper is organized as follows. Section 2 defines the symbiotic relationship with respect to related work in human-robot relationships. Section 3 formalizes the Visitor-Companion

Cite as: An Effective Personal Mobile Robot Agent Through a Symbiotic Human-Robot Interaction, Stephanie Rosenthal, Joydeep Biswas, Manuela Veloso, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 915-922

Task as an example of a symbiotic relationship in which a robot agent is benefits from human knowledge as it accompanies and helps the human. Section 4 presents the human help to the robot, while Section 5 analyzes the robot help to the human.

2. SYMBIOTIC RELATIONSHIPS

We are interested in robots that can overcome their limitations by asking humans for help. While many other robots achieve their goals with human assistance, we differentiate symbiotic relationships in several ways.

In symbiotic relationships, the agents in the team are performing separate *asynchronous* actions and the results affect all team members involved. Unlike sliding autonomy or collaborative control systems (*e.g.,* [5][7][9]) in which the robot can seek assistance or confirmation from humans, symbiotic agents are autonomous and do not control or direct each others' actions in any way. All agents can take these actions to achieve the goals of the team, and coordinate through *synchronous* communication actions to request and provide help to team members.

The agents in symbiotic relationships benefit each other by requesting and receiving help on actions they could not have performed alone due to lack of *capabilities*, coordinating their actions only when they need help. The help can come in two forms:

- an agent performs an action for another (*e.g.,* socially embedded learning [2] in which the human escorts the robot to the desired location)
- an agent increases another's capability to complete the action (*e.g.,* learning by demonstration [1] in which a human tells the robot which state they are in or which action to take)

While the robot could learn to perform tasks it requests help for, we do not expect any robot to be able to complete all actions. For example, a robot without arms cannot ever lift a cup of coffee.

Finally, because there may be many possible plans that achieve the same goals, the agents assign costs to their state (*expectations*) which all the agents can use when evaluating the best actions. When the agents take actions that affect each other, they take actions to minimize cost of each others' state while achieving the goal, further benefitting the group. This relationship is in contrast to those in which the human or the robot is responsible for helping the other without benefit in return.

3. VISITOR COMPANION TASK

To illustrate the symbiotic relationship, we contribute a formal definition of states (Table 1) and actions (Table 2) from the robot's perspective in the Visitor-Companion Task. The Task requires the robot to accompany a visitor to each meeting throughout the day. Additionally, it could complete other tasks for the human like getting coffee. Although both the robot and human have the same goals, they are not performing the actions together. When possible, the robot acts autonomously and performs actions to satisfy both the visitor's and its own goals (*e.g.,* the robot navigates to the meeting and the visitor follows without negotiation).

However, the robot may have limitations, either due to state uncertainty or lack of capabilities which may cause some of its actions to fail. Because the companion problem requires that a human be present near the robot for a majority of the time, it offers the flexibility of the robot proactively requesting assistance from the visitor or other humans when needed. The visitor can answer questions (*e.g.,* tell the current location) or physically help the robot (*e.g.,* lift a coffee cup). The visitors actions satisfy the robots' subgoals which in turn satisfy the shared goals of both the human and robot. The help mutually benefits the robot, which can now complete the task, and the human when the request is accomplished or expectation is satisfied.

This formalism is written in the PDDL planning language [8] with extensions[1].

3.1 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 *expectations* when deciding on its actions. 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 another communication action (Table 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 it can inform the visitor:

```
(:action inform-loc
  :parameters (?loc)
  :precondition (and (known-loc ?loc)
      (still-interesting location ?loc)
      (robot-at-loc ?loc))
  :effect (and (visitor-informed location ?loc)
      (not (still-interesting location ?loc)))
)
```

The robot can `move` 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 `open-door` and `put-coffee` include a *capability* or probability of success based on the robot's uncertainty (discussed later), which can result in either a success or failure:

```
(:action move
  :parameters (?loc)
  :precondition (nav-target ?loc)
  :effect (prob-or (and (success move ?loc)
            (robot-at-loc ?loc))
      (failed move whereAmI))
)
```

Based on the failure (*e.g.,* localization error from `move`), the robot can ask a human nearby for help:

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

Table 1: The Visitor-Companion Robot's state predicates and the predicates it holds for the visitor's state.

Action Type	Actor (→ Actee)	Action(Parameters)	Description
Execute	robot	nextMeeting(?loc)	robot determines next meeting loc given the current loc
		move(?loc)	robot moves from current location to ?loc
		open-door	robot opens the door in front of it
		put-coffee	robot picks up coffee (to bring somewhere else)
Inform	robot → visitor	inform-loc(?loc)	robot informs visitor about current location
		inform-meeting(?type ?info)	robot informs visitor about meeting info
Ask	robot → human	ask(?ques)	robot asks human whereAmI, openDoors, or getCoffee
Respond	**human → robot**	respond(?ques)	visitor responds to the robot
Process-Response	robot	process-loc(?ans)	robot processes visitor's loc answer
		process-action(?ques ?ans)	robot processes visitor's action (door open, put coffee)
Request	**visitor → robot**	request-info(?type)	visitor requests info about next meeting or current loc.
		request-act(?act)	visitor requests the robot take an action
Process-Request	robot	proc-req-info(?type)	processes request for information about meeting host
		proc-req-act(?act)	robot processes request for action
Notify	robot → visitor	notify-IAmThere(?loc)	robot notifies visitor they have arrived at ?loc
		notify-done(?act)	notifies visitor that it completed the requested action

Table 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).

```
(:action ask
  :parameters (?ques)
  :precondition (and (failed ?action ?ques)
      (next-to-robot ?human))
  :effect (asked ?human ?ques)
)
```

When the visitor responds to a location question, the robot processes the response and updates its location information to continue moving:

```
(:action process-loc
  :parameters (?ques ?ans)
  :precondition (and (asked ?human whereAmI)
      (h-responded ?human whereAmI ?ans)
      (robot-at-loc ?loc))
  :effect (and (not (asked ?human location))
      (not (h-responded ?human whereAmI ?ans))
      (not (robot-at-loc ?loc))
      (not (failed move whereAmI))
      (robot-at-loc ?ans))
)
```

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:

```
(:action notify-IAmThere
  :parameters (?loc)
  :precondition (and (success move)
      (next-to-robot visitor)
      (robot-at-loc ?loc))
  :effect (and (not (success move))
      (visited-loc ?loc))
)
```

3.2 Capabilities

Unlike socially embedded learning [2] 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 hap-

Communication Type	Action	Effect	Description
Speak	ask	asked	human is asked
	proc-req-info	visitor-informed	visitor is informed of meeting information they requested
	informed-loc	visitor-informed	visitor is informed of the location
	informed-meeting	visitor-informed	visitor is informed of meeting information
	notify-IAmThere	visited-loc	the visitor's location is updated when notified of arrival
	notify-done(getCoffee)	req-satisfied	the visitor knows their request has been satisfied
Email	proc-req-act(emailLate)	req-satisfied	the meeting host is emailed the visitor is late

Table 3: Visitor-Companion Robot's communication

pen according to capabilities - the probability of success p. If there is no chance of completing an action, $p = 0$. For example, if the robot does not have arms, there is no change it could perform `open-door` or `put-coffee` 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 giveCoffee)`. When $p > 0$, the robot may not complete an action successfully due to the uncertainty in the robot models. For example, in the `move` action, the robot may be uncertain of its location which contributes its successful completion.

3.3 Expectations

While the robot only requires state and actions in order to complete a task, there may be many plans that equally satisfy the constraints. In order to find the "best" plan or determine when it is best to take an action, we define expectations that an agent may have on their state. Formally, we define an expectation 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 expectation 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:

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

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 team members incur a cost of c. Using these expectations, 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.

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 getting coffee. In cases when the robot finds it has limited capabilities, the visitor offers to help the robot with the expectation that the robot completes the plan to his satisfaction with a minimal cost. Next, we present our real robot, CoBot, which performs the Visitor-Companion Task and results when the it asks for localization help from the visitor without any expectations. Then, we analyze the usability of the robot based on the costs of asking for help and the value of the robot's actions.

4. HUMAN HELP TO THE ROBOT

The Visitor-Companion Task best illustrates the symbiotic relationship between humans and robots when the robot lacks some capabilities for which it can ask the visitor for help. We have implemented this task on our robot, CoBot, a custom built mobile robot (Figure 1) capable of autonomous navigation under localization uncertainty. However, CoBot does not have arms and therefore cannot open doors or pick up coffee. Additionally, CoBot represents its location under uncertainty so CoBot's capability to uneventfully move between locations changes with time. Thus, the `move` action has the probabilistic effect of either success or failure.

The capability probability for the robot to complete action a, p_a, could be used in many different ways to determine when to ask, including as a learned threshold (*e.g.,* [6]) or along with time constraints (*e.g.,* [12]). CoBot decides to seek help by thresholding the capability:

$$\text{if } p < \text{threshold, ask}$$

As CoBot is incapable ($p = 0$) of `open-door` and `put-coffee`, any threshold > 0 will require to appropriately ask for help. For any action, like `move`, where $0 < p < 1$, the threshold is action- and state-dependent and hence can either have the effect of autonomous execution or ask the human for assistance, as was captured by the `prob-or` conjunction in the description of the `move` action in Section 3.1. We illustrate

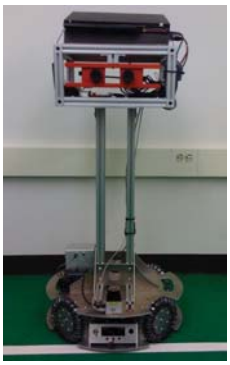

Figure 1: The CoBot Visitor-Companion Robot. Thanks to Mike Licitra, who designed and built it.

CoBot's requests for assistance with localizing itself based on its capability to navigate under uncertainty.

4.1 Determining When to Ask

CoBot uses a WiFi localization algorithm to localize itself for the purpose of navigation [3]. The location of the robot is tracked using a particle filter with a set of particles $\mathbb{P}$. The inferred location of the robot l^{Robot} is computed by clustering the particles [4] and taking the weighted mean of the best cluster $\mathbb{P}^*$, $\mathbb{P}^* \subseteq \mathbb{P}$. The destination location l_d and the graph map of the building are used to generate a topological policy π over the map to be used by the robot to navigate to l_d. Using this topological policy π and the inferred location l^{Robot} of the robot, a PATH is computed, which is a list of navigation primitives n from set of navigation primitives N:

$$\begin{aligned} N \;=\; & \{\textsc{MoveDownCorridor}(\mathit{distance}),\\ & \textsc{InPlaceTurn}(\phi),\\ & \textsc{TakeNextTurn}(\mathit{direction})\} \end{aligned}$$

While the algorithm allows CoBot to autonomously navigate, localization errors and environmental factors may affect successful navigation over time. We quantify the localization uncertainty by two measures:

- Deviation Uncertainty: the weighted standard deviation of the best cluster of particles $\mathbb{P}^*$:

$$D(\mathbb{P}^*) > t_{unc}$$

- Policy Uncertainty: the existence of a policy conflict within the best cluster:

$$\exists \rho_1, \rho_2 \in \mathbb{P}^* : \pi(\rho_1) \neq \pi(\rho_2)$$

When there is Deviation Uncertainty, the location of the robot cannot be estimated accurately. When there is Policy Uncertainty, the location of the robot is not known with enough certainty to unambiguously select a navigation primitive to execute. CoBot's capability to move successfully to a goal location is determined by localization uncertainty, $p_{move} = \langle p_1, p_2 \rangle$, where :

$$p_1 = \begin{cases} 1 - \frac{D(\mathbb{P}^*)}{2t_{unc}}, \text{ if } D(\mathbb{P}^*) \leq t_{unc} \\ 0, \text{ if } D(\mathbb{P}^*) > t_{unc} \end{cases}$$

$$p_2 = \max{}_{n_i} \left[\frac{\|\mathbb{P}^{n_i}\|}{\|\mathbb{P}^*\|} : \mathbb{P}^{n_i} = \{\rho_j \in \mathbb{P}^* : \pi(\rho_j) = n_i\} \right]$$

We have constructed p_1 such that $p < 0.5$ *iff* $D(\mathbb{P}^*) > t_{unc}$ (it has Deviation Uncertainty). CoBot could ask when $p_1 <$ threshold1 $= 0.5$. Similarly, when $p_2 <$ threshold2 $= 1$, it has Policy Uncertainty. Because there are two parameters which each contribute to p_{move}, we can define four rules for determining when to ask (*i.e.,* when $p <$threshold, ask)

- Ask-D-Only : $p_1 <$ threshold1, ask
- Ask-π-Only : $p_2 <$ threshold2, ask
- Ask-D-or-π : $(p_1 <$ threshold1$) \vee (p_2 <$ threshold2$)$, ask
- Ask-D-and-π : $(p_1 <$ threshold1$) \wedge (p_2 <$ threshold2$)$, ask

Before performing each navigation primitive n_i, CoBot assesses its capability to move using one of the four methods above. If it finds itself uncertain and unable to perform the primitive, CoBot asks the visitor for help. Thus, the `move` action can have the effect of success (and autonomous navigation) or of failure (and asking the human for assistance).

4.2 Acting based on Capabilities

When CoBot is navigating autonomously (will not ask a human for help) and detects these uncertainties, CoBot performs a Replan algorithm:

1. If Deviation Uncertainty and not Policy Uncertainty: then $\forall \rho_1, \rho_2 \in \mathbb{P}^* : \pi(\rho_1) = \pi(\rho_2) = n_i$ and CoBot should execute n_i with updated parameters
2. If Policy Uncertainty and not Deviation Uncertainty: then the robot is likely at l^{Robot} and CoBot should execute primitive $\pi(l^{Robot})$ with updated parameters
3. If Deviation Uncertainty and Policy Uncertainty:
   (a) Stop and wait until enough WiFi signal strength readings are collected that $D(\mathbb{P}^*) < t_{unc}$
   (b) Recalculate the location of the robot l^{Robot}
   (c) Regenerate PATH using l^{Robot}, and continue

If only one of uncertainty is true, CoBot knows which action to take and can continue navigating without stopping. It only relocalizes and generates a new plan when uncertain in both ways.

When, instead, CoBot can ask for help to determine its location, it first uses one of the four functions above to determine whether to ask for help and otherwise uses the appropriate condition in the Replan algorithm to continue navigating without stopping. Concretely, when CoBot asks the visitor for help, the visitor clicks on the robot's current location on a map of the building, the most specific question it could ask [14]. Given the visitor's response coordinate pair (x, y), CoBot reinitializes all particles $\rho_i \in \mathbb{P}$ with weights:

$$w_i = \frac{1}{|\mathbb{P}|}$$

at locations around (x, y):

$$(x + \mathcal{N}(0, \sigma), y + \mathcal{N}(0, \sigma))$$

The Gaussian normal $\mathcal{N}(0, \sigma)$ is an additive term which accounts for human error when indicating the robot's location on the map. When $t_{unc} > \sigma$, the visitor's response immediately reduces CoBot's Deviation Uncertainty. Because the reinitialized particles have a smaller deviation, it is also more likely to reduce the Policy Uncertainty.

4.3 Experimental Results

In order to understand the impact of asking questions on CoBot's uncertainty, we tested CoBot's ability to navigate through a twelve-meeting schedule. We varied when to ask questions based on the four asking functions presented above, and compared it against fully autonomous navigation. The schedule was contained on a single floor of our building and measured 818 meters long (the mean over all the runs, as calculated from the localization of the robot). Fig. 2 shows the reconstructed path of the robot from one of the runs, with photographs of some events during the run.

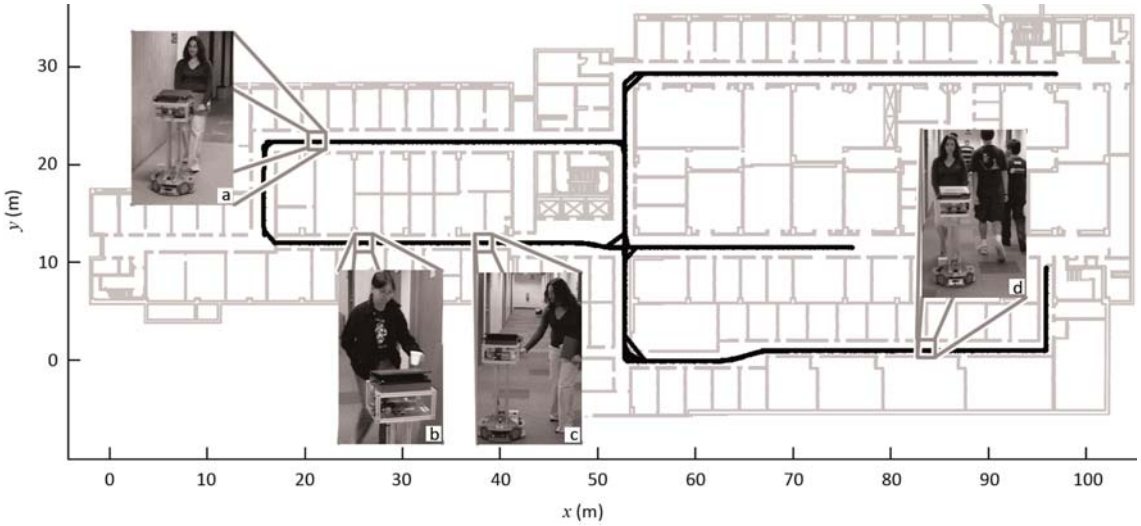

Figure 2: Trace of the path traversed by the robot while following the schedule. Snapshots: a) CoBots leads the visitor to her first meeting b) CoBot requests assistance with getting a cup of coffee c) The visitor graciously accepts her cup of coffee d) CoBot navigates around other humans in the corridor while leading the visitor to her last meeting

We report uncertainty in terms of $D(\mathbb{P}^*)$ in cm, although Policy Uncertainty was also used to calculate when to ask. We set t_{unc} to 200cm for the decision algorithms that require it, and note that higher thresholds would result in fewer questions and lower thresholds results in more frequent questions. While the threshold would change the frequency of questions, the proportion of questions between conditions would be impacted uniformly.

4.4 Autonomy vs. Asking

While CoBot is capable of autonomous localization and navigation in the environment, it can suffer from high uncertainty. Table 4 shows the median $D(\mathbb{P}^*)$, navigation time, and total number of replanning steps taken during autonomous navigation and the four question-asking conditions. On average, CoBot has 12cm more uncertainty while navigating autonomously (80.77cm) compared to any of the asking conditions (average 68.88cm). CoBot benefits from the 12cm difference, as it is able to navigate doorways around people with more precision. Additionally, we find that CoBot must run the replan algorithm 3349 times when navigating autonomously compared to 1601 times using the Ask-D-and-π decision algorithm and only 38 times using Ask-π-Only. Note that most of the replans were performed while navigating - the robot did not have to stop.

CoBot spent 1666 seconds navigating autonomously, compared to 1519 seconds on average when asking for help. The 147 second difference can be attributed to higher average uncertainty which resulted in more backtracking when CoBot drove past meeting room locations or hallway intersections. While asking for help requires time to stop and wait for a response, it results in at least a 50% reduction in the replan steps, and a 9% reduction in navigation time.

4.5 When to Ask

While CoBot benefits from asking for help in terms of navigation and planning time, there are significant differences between the asking functions. Table 5 shows the number of requests for help, time spent stopping to wait for help, and time between help requests for each condition. With Ask-D-and-π, CoBot requests help 8 times, two-thirds fewer questions than with the other asking functions. Over the entire twelve-meeting schedule, the number of questions asked averages to one question every 214 seconds for Ask-D-and-π, compared to 32-68 seconds between questions in the other conditions. CoBot was able to navigate to 33% of the meetings completely autonomously in the Ask-D-or-π and asked one question per meeting otherwise compared to more than 5 questions between each meeting for Ask-D-or-π.

Because CoBot asked few questions with Ask-D-and-π, it spent only 64 seconds waiting for help, compared to 200 seconds with Ask-D-Only and 496 seconds times with Ask-D-or-π. Figure 3 shows the total amount of time CoBot spent completing the meeting schedule, broken down by navigation time, time to ask for help, and time to stop for replanning. While the question-asking conditions spent about 1500 seconds navigating, many took longer overall to complete the

Condition	Uncertainty (cm)	Nav. Time	# Replans
Autonomous	80.77	1666	3349
Ask-D-Only	64.67	1523	64
Ask-π-Only	71.89	1495	38
Ask-D-or-π	64.24	1511	61
Ask-D-and-π	74.72	1547	1601

Table 4: Median Uncertainty (measured with $D(\mathbb{P}^*)$ in cm), Navigation time, and Number of Replanning steps for each condition

Condition	# Help	Help Time (s)	B/w Help (s)
Ask-D-Only	25	200	68
Ask-π-Only	38	304	49
Ask-D-or-π	62	496	32
Ask-D-and-π	8	64	214

Table 5: Number requests for help, Time spent stopped for help (seconds), and Time between questions (seconds)

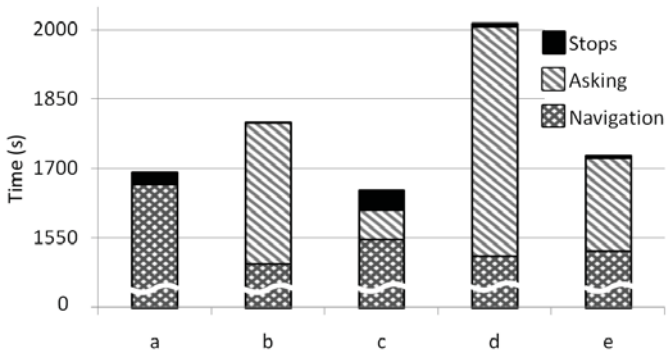

Figure 3: Time taken for Stops, Asking and Navigation for a) Autonomous b) Ask-π-Only c) Ask-D-and-π d) Ask-D-or-π e) Ask-D-Only

schedule compared to autonomous navigation because they spent a significant amount of time asking for help. In total, CoBot took 1692 seconds to complete the schedule autonomously - stopping to replan for 26 seconds and navigating for the remainder of the time. CoBot took the longest to complete the schedule using the Ask-D-or-π algorithm (2015 seconds total), spending 1495 seconds to navigate and an additional 498 seconds waiting for help. The Ask-D-and-π algorithm resulted in the shortest completion time in 1653 seconds, 39 seconds shorter than autonomous navigation.

4.6 Where to Ask

Table 6 shows the number of questions each asking function asks by building location. Because the Ask-D-and-π algorithm requires both measures of uncertainty to be true before asking for help, CoBot asks few times compared to the other asking algorithms and can take advantage of mobile replanning. While other algorithms ask for help in the hallways at least 10 times, the Ask-D-and-π algorithm never does, because the policy in the hallway is always MoveDownCorridor and there is never a conflict. Compared to the algorithms that ask whenever there is a policy conflict, the Ask-D-and-π algorithm asks for help in intersections and by meeting rooms at least 20 fewer times because it often is certain of its location in these areas.

Overall, the Ask-D-and-π function asks the fewest questions and spends the least amount of time waiting for help, resulting in the longest time between questions compared to the other question-asking conditions. Although it has more uncertainty on average than the other question-asking functions, Ask-D-and-π has lower uncertainty than autonomous navigation, spending less time navigating and cutting the replans in half. The algorithm takes advantage of mobile replanning to continue moving without requesting help.

Condition	Halls	Intersects	Meetings	Total
Ask-D-Only	16 (64%)	7 (28%)	2 (8%)	25
Ask-π-Only	10 (26%)	24 (63%)	4 (11%)	38
Ask-D-or-π	19 (31%)	35 (56%)	8 (13%)	62
Ask-D-and-π	0 (0%)	4 (50%)	4 (50%)	8

Table 6: Number of times CoBot asked for help in each area of the building - hallways, intersections, and meeting rooms.

5. ROBOT HELP TO THE HUMAN

We have shown that when CoBot makes decisions about when to ask for based on its capabilities it can vary the number of questions it asks and the time it takes to complete the subgoal. However, we have not discussed the impacts of the questions as well as the CoBot's other abilities on the visitor. Next, we present the results of five visitors using our CoBot and show that most visitors would use the robot again despite the requests for help.

5.1 Visitor Experiences

In order to test the CoBot's capabilities in assisting visitors through their meeting schedules, we invited five participants to participate in a four-meeting schedule over the same floor. The rooms were all constrained to a single floor but were spread out in all four hallways. The participants were true visitors and had never been in the building before. Participants were told that the CoBot could assist them in the following ways on the way to their meetings:

- bring drinks to meetings
- providing additional information about meeting hosts (by displaying the host's website)

They were told they should request the CoBot perform each of these 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. They were told also CoBot will offer information about different rooms and labs as they walk between meetings. Additionally, they were told that sometimes the robot got lost and would ask for localization help.

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.

At the end of 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). Table 7 shows the ratings each participant (P1-5) gave for the robot's abilities. We found that each participant rated the usefulness of the meeting information differently, showing that each participant had different expectations for each ability and subsequent state. Additionally, we asked each participant to rate the number of questions CoBot asked from -2 (too many) to 2 (too few).

For each participant, CoBot gave the same information and asked nearly the same number of questions. While participants mostly felt the robot could have asked fewer questions, they had different opinions about how many were too many - reflecting different costs associated with the questions. When we combine the robot's abilities and the questions into a complete experience, 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.

This finding necessitates modeling the expectations with user-dependent costs for the visitor's state. The robot can

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 7: Participants ratings from -2 (not useful) to 2 (very useful) of CoBot's abilities and questions.

use utility functions with the costs to determine which actions to take so that the visitor finds value in them.

6. CONCLUSION

In this work, we contribute a robot agent capable of being in a symbiotic relationship with humans. We introduce the Visitor-Companion Task as an example of a task where a human visitor and a companion robot agent have joint goals and coordinate as a team of human and robot, but interact asynchronously. The robot can ask for help from the human to overcome some of its limitations, including possible location confusion, or need to open a door. The visitor is helped by the robot in the navigation to meetings, and for occasional needs, such as coffee. We presented a formalization of the robot state and action spaces, where the effects of the robot actions are probabilistic and can trigger invoking the help of the human.

We presented our implementation of the task on our robot CoBot and showed how CoBot could ask for localization help. Our results show that asking for help can reduce localization uncertainty as well as the number of replanning steps the robot must take compared to autonomous navigation. As a result, the robot backtracked less and took less time to navigate without asking many questions. We showed that while CoBot performed the same tasks to help each visitor, different visitors have different expectations for the robot and reacted differently to the questions it asked. Most of the visitors were satisfied with the human-robot relationship and would use CoBot again. Our formalization and our analysis enable our future work further experimenting with human subjects.

Acknowledgments

This work was partially supported by a Fellowship from the National Science Foundation, by the Computational Thinking Center at Carnegie Mellon, and by the Lockheed Martin, Inc. under subcontract 8100001629/1041062, and by Intelligent Automation, Inc, under subcontract 654-1. The views and conclusions contained in this document are those of the authors only.

7. REFERENCES

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

[2] H. Asoh, S. Hayamizu, I. Hara, Y. Motomura, S. Akaho, and T. Matsui. Socially embedded learning of the office-conversant mobile robot jijo-2. In *International Joint Conference on Artificial Intelligence*, volume 15, pages 880–887. Citeseer, 1997.

[3] J. Biswas and M. Veloso. Wifi localization and navigation for autonomous indoor mobile robots. In *IEEE International Conference on Robotics and Automation*, 2010.

[4] P. Bradley and U. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the 15th International Conference on Machine Learning*, volume 727, 1998.

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

[6] S. Chernova and M. Veloso. Multi-thresholded approach to demonstration selection for interactive robot learning. In *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, pages 225–232. ACM New York, NY, USA, 2008.

[7] T. W. Fong, C. Thorpe, and C. Baur. Robot, asker of questions. In *Robotics and Autonomous Systems*, volume 42, No. 3-4, pages 235–243, 2003.

[8] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL: The Planning Domain Definition Language, Version 1.2. *Yale Center for Computational Vision and Control, Tech Report CVC TR98003/DCS TR1165*.

[9] F. Heger, L. Hiatt, B. Sellner, R. Simmons, and S. Singh. Results in sliding autonomy for multi-robot spatial assembly. In *8th International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, pages 5–8. Citeseer, 2005.

[10] J. Hong, Y. Song, and S. Cho. A hierarchical bayesian network for mixed-initiative human-robot interaction. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3808–3813, 2005.

[11] T. Kanda, M. Shiomi, Z. Miyashita, H. Ishiguro, and N. Hagita. An affective guide robot in a shopping mall. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 173–180. ACM, 2009.

[12] C. McMillen and M. Veloso. Thresholded rewards: Acting optimally in timed, zero-sum games. In *Proceedings of The National Conference on Artificial Intelligence*, volume 22, page 1250, 2007.

[13] I. Nourbakhsh, J. Bobenage, S. Grange, R. Lutz, R. Meyer, and A. Soto. An affective mobile robot educator with a full-time job. *Artificial Intelligence*, 114(1-2):95–124, 1999.

[14] S. Rosenthal, A. K. Dey, and M. Veloso. How robots' questions affect the accuracy of the human responses. In *Proceedings of The International Symposium on Robot - Human Interactive Communication*, pages 1137–1142, 2009.

[15] R. Siegwart, K. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, et al. Robox at Expo. 02: A large-scale installation of personal robots. *Robotics and Autonomous Systems*, 42(3-4):203–222, 2003.

[16] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, et al. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *The International Journal of Robotics Research*, 19(11):972–999, 2000.