Social Feedback For Robotic Collaboration

Emily Wu
Professor Stefanie Tellex
Professor Jeff Huang
Brown University
May 2016

Abstract

Robotic collaboration requires not just communication from the human to the robot, but also from the robot to the human. We call this robot-to-human communication *social feedback*. In order to flexibly and intelligently generate social feedback, we describe a Partially Observable Markov Decision Process that incorporates the human's belief about the robot's intent. We then engineer a system that can respond in near-real time that is supplemented with a multithreaded interaction model that allows for more responsive human-robot interactions. Initial evaluation in simulation shows improvement in interaction length in a toy domain and improvement in accuracy in the full domain.

Contents

Social Feedback For Robotic Collaboration

Emily Wu

April 14, 2016

1 Introduction

Collaboration is a process that relies heavily on communication for its success. When humans collaborate, this communication is both obvious and implicit—we not only instruct and request aid from each other, but we also tacitly monitor our partners for signs of approval and understanding while producing these signals ourselves. However, in robotics, many of these vital components of successful communication are missing or lost. The absence of these signals likely account for numerous failures in human-robot collaborative tasks. In order to give robots these missing communicative skills, we employ *social feedback* signals to provide human-like communication to both inform the human participant of the robot's state and request information from the participant.

Feedback refers to the responses that are received by an agent when it takes some action. These feedback responses inform the agent about the success or failure of its actions. *Social feedback* therefore refers to social signals that convey this information. These can be explicit signals, such as "I want this object", or implicit signals, such as a perplexed expression. When humans interact with another human, they use this feedback to improve the flow and clarity of the collaboration, which in turn improves its success. For my thesis work, I have developed a framework that describes how robotic agent should likewise generate social feedback for its human partner in human-collaborative tasks, and show that it improves the speed and accuracy of human-robot interaction.

Many existing works focus on the task of interpreting human communications[Tellex et al., 2011, Matuszek et al., 2012, Tellex et al., 2012, Misra et al., 2014], but provide no response if the interpretation fails. As a consequence, it becomes difficult to tell whether the robot is still making progress in interpreting the request, or has failed. Some works do allow for robotic responses to ambiguous situations, but the manner of their responses are hard coded, often limited to generic requests, such as "please repeat the question". In this work, we provide a means to generate these requests in the form of language and gesture that can flexibly represent the robot's state as well as inform the human in such a way so that allows the human to help the robot as much as possible.

Specifically, we address the object delivery task. In this task, a set of objects are laid out on a table within a Baxter robot's reach. A human participant requests an object from the robot using speech and gesture (pointing). The robot must interpret the human's speech and gesture and deliver the requested object to the human. The task then repeats with the remaining objects on the table. This task is achievable without social feedback; the robot need only wait until enough information is given and then deliver the correct object. However, we will show that by adding social feedback actions, such as asking questions, looking at objects, and pointing at objects, we will achieve better accuracy and speed as well as improved user experience.

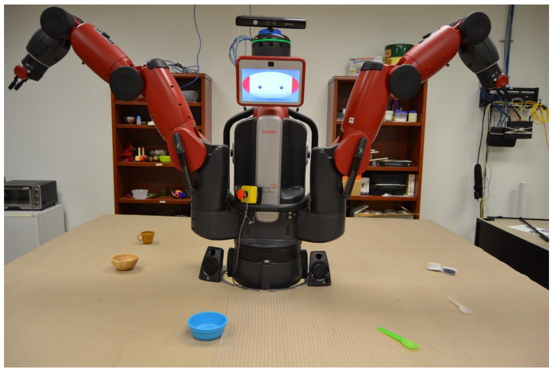

Figure 1: The experimental setup from the participant's perspective: a Baxter robot has six objects within reach. The participant stands in front of the table to provide speech and gesture requests

We will solve this problem by formulating it as a Partially Observable Markov Decision Process (POMDP)[Kaelbling et al., 1998], which will allow us to dynamically and flexibly determine how to choose social feedback actions. First, we will describe a two agent model and use this to motivate the construction of a POMDP. The crucial improvement we make over existing robot interaction models is that we maintain a state variable that tracks the human's interpretations of the actions the robot takes. We will next discuss how we solved this POMDP and other measures we took to allow our system to respond dynamically and fluidly.

As an initial evaluation, we run our model in simulation to showcase improved speed and accuracy in the object delivery task.

2 Related Work

This work is primarily built off of Eldon [2015], which describes a system for incremental speech and gesture recognition in an object-delivery domain. In previous work [Wu et al., 2015], the model from Eldon [2015] is expanded from a bayesian inference model into a POMDP model that can ask yes-no questions. In this work, we generalize the idea of the affect of robot's actions on humans state within a POMDP framework.

Work demonstrating the importance of social feedback in human-human communication has been done in the field of psycholinguistics. In Clark and Krych [2004], one human (labeled the builder) builds a Lego model according to instructions given by another human (labeled the director). In the feedback-free trials, the director's instructions were prerecorded, and the resulting models were very inaccurate (in fact no model was completely correct). In the feedback trials, errors were reduced by a factor of eight. Our goal is to enable a robot to collaborate with a human in this way.

Other work with collaborative robots exists, for example, Foster et al. [2012] have done research with a bar-tending robot. This robot follows a rule-based state estimator, and delivers drinks from fixed positions behind the bar to multiple users based on their speech and torso position. We expand the scope of the problem: we do not use a rule-based state planner, our items are not in fixed positions, and our gesture model uses pointing instead of torso position.

In Bohus and Horvitz [2014], a robotic building guide directs guests to find specific rooms. Our project addresses a similar domain, requiring the interpretation of users' requests, but differs in the task and the type of communication necessary to accomplish that task.

Other work involving robotic object delivery also exists. Some approaches have no social feedback and will either deliver the wrong item or do nothing if given a request it does not understand [Tellex et al., 2011, Matuszek et al., 2012, Tellex et al., 2012, Misra et al., 2014]. Language only feedback models also exist [Chai et al., 2014, MacMahon et al., 2006, Tellex et al., 2011, Matuszek et al., 2012, Guadarrama et al., 2014, Hewlett et al., 2011, Misra et al., 2014], and several gesture only models [Waldherr et al., 2000, Marge et al., 2011].

Matuszek et al. [2014] shows promising work in fusing language and complex gesture to understand references to multiple objects at once. We build off this work by including social feedback.

In the field of computational linguistics, previous work exists in resolving referring expressions incrementally, such as Schlangen et al. [2009], Zender et al. [2008], Gieselmann [2004]. Other work in that community also incorporates gesture, and/or eye gaze [Kennington et al., 2013, 2015], but the given work does not incrementally update gesture along with speech. Chai et al. [2011] provides work towards resolving referring expressions in a different domain, but does not address the task of acting on the results of these referring expressions. In Kruijff et al. [2008], they propose a system for planning to ask for clarifications, which covers a wide scope of knowledge failures. In this work, we are interested only in a small subset of these clarifications, and address the problem of how and when these clarifications should be used in a concrete human-robot collaboration task.

POMDP approaches to dialog [Young et al., 2013] are quite common, but treat dialog as a discrete, turn-taking interaction. The Dialog State Tracking Challenge [Williams et al., 2013] a notable driving force for computer dialog understanding, treats dialog in this turn-based way. Our model is also based off a turn-based interaction, but we provide a systems solution to allow our model to behave more incrementally.

Interactive POMDPs (I-POMDPs) [Gmytrasiewicz and Doshi, 2005] describe a multi-agent POMDP that allows agents to anticipate each other's actions by forming beliefs over action and observation histories. This work attempts to accomplish the same, but makes more specialized assumptions that allow for a more tractable state space.

Alternative approaches to POMDPs include cognitive architecutres such as SOAR [Laird, 2012] or DIARC [Schermerhorn et al., 2006]. By taking a probabilistic approach, we can seemlessly fuse information from multiple sources and explicitly reason about the robot's uncertainty when choosing actions.

3 Technical Approach

3.1 POMDP Overview

Markov Decision Processes (MDPs) are models that describe how an agent can take actions to transition between states, receiving rewards for its actions. Partially Observable Markov Decision Processes (POMDPs) are used to model MDPs where the true state is not known. Instead, the agent receives observations that are generated by the true state, and must infer what the true state is from the observations. Thus, the agent maintains a belief over true states which is updated as it receives new observations. In subsequent sections, we will refer to this belief over hidden states as b. The agent must then use this belief state to determine which action to take to maximize its expected value of the reward over time. This splits the POMDP into two main components, a state estimator and a policy generator. A typical graphical model for a POMDP is provided in figure 2.[1]

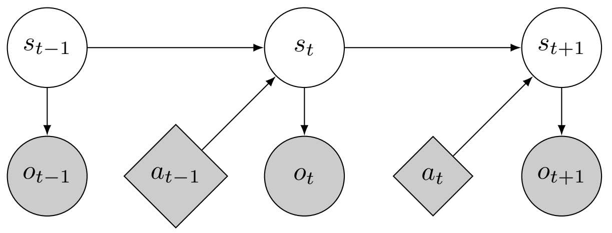

Figure 2: A graphical model representation of a POMDP

3.2 Model Description

To model this human-robot interactive task, we will use a POMDP. To motivate the construction of our model, consider two-agent model constructed depicted in figure 3.

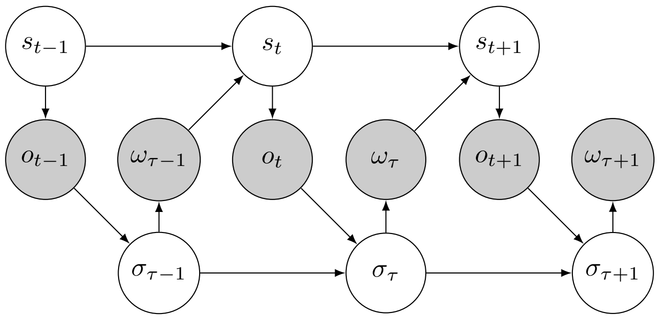

Figure 3: A two-agent model based off two POMDPs

We represent human states s_t, robot states σ_τ, and observations generated by the robot ω_τ and by the human o_t. Observe that the structure of this model resembles two POMDPs combined together at their actions and observations. The lower POMDP is the POMDP from the human's perspective: the robot has some hidden states σ_τ, which the human observes by means of ω_t. The human takes actions o_t to influence the robot's state σ_t. The upper POMDP models the interaction from the robot's perspective: the human's state s_t is hidden from the robot, and the robot must infer it from observations o_t. When the robot takes action $\omega_{\tau-1}$, it affects the human's state s_t. Crucially, each agent treats the other agent's action as an observation that influences their belief about the other agent's hidden state. Thus, the human's actions affect the robot's belief about the human's state, which is what we call b above. Importantly, the reverse is also true: the robot's actions affect the human's belief about what the robot's hidden state is. We will call the human's belief over the robot's hidden state β.

In the following section we will use this dual structure to inform the construction of our POMDP as applied to our object delivery domain.

3.3 POMDP Definition

We define our object-delivery POMDP as a tuple $\{S, A, T, R, \Omega, O\}$:

- Each $s \in S$ is a tuple of $\langle \iota, \beta, \mathcal{I} \rangle$
  - $\mathcal{I}$ is the set of all objects that the robot can deliver. Each object is parameterized by a name, a unigram vocabulary, and a position; for example: a red bowl would be represented ⟨redBowl, [red, red, bowl, bowl, plastic], (1.0, 2.0, 0.0)⟩. We assume the set of all objects is known.
  - $\iota \in \mathcal{I}$ is the object the human desires. This is a hidden variable.
  - β is a distribution over the robot's hidden states, as defined above. In this domain, the robot's hidden state contains which object the robot will hand the human, or which object the robot believes the human wants. We assume the human estimates the distribution over the hidden state by observing the actions the robot has taken. Specifically, $\beta_t(\iota_t) = p(\iota_t | a_{1:t-1})$. While this variable is technically unknown, we make some assumptions about the transition functions and initialization so β_t's value is known at every time step.

  Since our state is composed of both hidden and known states, it resembles a Mixed Observability MDP [Ong et al., 2010], and we leverage the computational benefits of having both hidden and known states.

- The set of actions A consists of both social and non-social actions. Non-social actions are `pick(i)` (picking up and delivering an object i) and `wait`. Social actions are `point(x)`, pointing at a location x; `look(x)`, looking at a location x;[2] and `say(p)`, informing the human that the robot believes the desired object has property p, where p is an element of some object's vocabulary.

[1]In this model, we have omitted the dependency between an action and the observation from the next timestep. In the POMDP discussed in this paper, we will not use this dependency for any action. Rather, the affect of the actions on the observation is encapsulated in the state.

[2]x is chosen from the set of points that describe the surface of the table. For computational reasons, in this work, we restrict it to the set of object locations

- $T(s, a, s') = p(s'|s, a)$. We make the assumption that the human's desired object ι does not change unless the robot picks up object ι. The set of all objects $\mathcal{I}$ transistions to $\mathcal{I} \setminus \{i\}$ when the robot chooses action `pick(i)`. For each action in A, we define a "reverse observation function" that describes our assumptions about how the human imagines the robot generates actions (which the human sees as observations) given a hidden robot state. This allows us to define a transition function for β. The transition dynamics are described in greater detail in section 3.3.2.

- $R(s, a, s')$ returns a numeric reward from transitioning from s to s' by taking action a. In this domain, we incentivize our robot to pick the correct object by giving it a +10 reward if it delivers the correct object and a -50 reward for picking the incorrect object. We also give negative rewards for taking various actions: `wait` receives a -1 reward (to incentivize the robot to finish the task quickly); `look(x)` receives a reward of -2; `say(p)` receives a reward of -3; `point(x)` receives a reward of -4. These additional penalties for social actions reflect the penalty for "bothering the user", as well as the time it takes to execute these actions. These reward values were chosen experimentally to result in a high rate of pick accuracy.

- Observation $o \in \Omega$ represents an observation generated by the human. These are tuples of language and gesture: $\langle l, g \rangle$. Language utterance l is represented by a string of any number of words, obtained by transcribing microphone input using webkit's speech recognition API through google chrome. A gesture g is represented by a vector from the participant's shoulder to their wrist, and all gestures are interpreted as a straight-armed point. This vector is obtained using the Kinect's tracking software.

- $O(o, s, a) = p(o|s, a)$ describes the probability of of seeing an observation o from the human given their state s and the robot's last action a (though assume all observations are independent of a). We choose an observation function that reflects that the human is an agent attempting to communicate which object they desire to the robot, and thus chooses to generate observations that are more likely to result in the robot delivering the correct object. This is gone into more detail in the next section, section 3.3.1.

3.3.1 Observation Function

According to our double-agent model, the human emits observations as though it were an agent interacting with our robotic agent. Thus, we choose an observation model that depends on the human's belief about the robot's state, β. Specifically, the human will choose an action according to its estimate that the robot will hand them their desired object. In order to define this function, we will first have to define a base-level observation function.

Base-Level Observation Function The base level observation function describes the probability of an observation conditioned only on the object: $p(o|\iota)$. For our object delivery domain, we will define two base-level observations, one for language and one for gesture. These observation functions are the ones defined in Eldon [2015].

Speech Model: Language is interpreted according to a smoothed unigram speech model. An utterance l is broken down into individual words, $w \in l$:

$$p(l|\iota) = \prod_{w \in l} p(w|\iota) = \prod_{w \in l} \frac{\texttt{count}(w, \iota.\texttt{vocab}) + \alpha}{|\iota.\texttt{vocab}| + \alpha|\texttt{words}|}$$

where $\texttt{count}(w, \iota.\texttt{vocab})$ is the number of times word w appears in ι's vocabulary. This allows for repeated words to have greater probability of being spoken.[3] α is a smoothing value chosen to be 0.05. $|\texttt{words}|$ is the total number of words in all object vocabularies.

Gesture Model: All gestures are interpreted as a straight armed point. These pointing gestures are selected from a normal distribution centered at the object's location.

Define the angle between the vector defined by the pointing gesture and the vector from the human's shoulder to the object ι to be θ_ι. The probability of a particular gesture is then

$$p(g|\iota) = \mathcal{N}(\theta_\iota|0, v)$$

where v is a hand-tuned variance. We choose $v = 0.4$.

Posterior Observation Function We will use the base-level observation function defined above to define a posterior observation function that considers how the base-level observation function will affect the robot's belief. Specifically, the human chooses an action proportional to the probability that the robot will hand them the desired object ι if the human had chosen that

[3]In the future, object vocabularies will be collected from data; repeated words are therefore meaningful as ways that are commonly used to describe objects.

action. The robot receives this action as an observation.

$$
\begin{aligned}
p(o|s) = p(o|\iota, \beta) &\triangleq \eta p(\iota|o) \\
&= \eta \frac{p(o|\iota)p(\iota)}{\sum_{\iota'} p(o|\iota')p(\iota')}
\end{aligned}
$$

where η is a normalization factor.

Next, we will set $p(\iota) = \beta(\iota)$. $p(\iota)$ describes the robot's belief in ι. This value is represented in the robot's belief-state vector, b. But since the human does not know b, it uses its estimate of b, β. Our new expression is

$$p(o|s) = \eta \frac{p(o|\iota)\beta(\iota)}{\sum_{\iota'} p(o|\iota')\beta(\iota')}$$

Using β as a prior for our observation function incentivizes the robot to take actions that modify β in such a way that increase the probability of useful observations, while in turn decreasing the probability of getting observations that tell the robot things it already knows. This result falls out of our assumption that the human is more likely to provide observations that communicate its intentions to the robot well.[4] For a worked example in a toy domain, see the appendix.

Modification for Object Delivery Domain In the object delivery domain, we have both speech and gesture, which we assume are conditionally independent given the state.

$$p(o|s) = p(l, g|s) = p(l|s)p(g|s)$$

In addition, it is possible to observe no speech or no gesture input. Let L be a random variable that is 1 if the agent receives a language observation and 0 otherwise. Similarly, let G be a random variable that is 1 if the agent receives a gesture observation and 1 otherwise. For all states s:

$$
\begin{aligned}
p(L = 1|s) &= c_L \\
p(L = 0|s) &= 1 - c_L
\end{aligned}
$$

$$
\begin{aligned}
p(G = 1|s) &= c_G \\
p(G = 0|s) &= 1 - c_G
\end{aligned}
$$

where c_L and c_G are constant values less than 1.

We will use a separate posterior observation function for language and gesture. Combined with the probability to receive a null speech or null gesture observation, the full expression is as follows:

$$
\begin{aligned}
&p(o|s) = \eta p(l|s)p(g|s) \\
&= \eta p(L|s) \left[\frac{p(l|\iota)\beta(\iota)}{\sum_{\iota'} p(l|\iota')\beta(\iota')}\right]^L \cdot p(G|s) \left[\frac{p(g|\iota)\beta(\iota)}{\sum_{\iota'} p(g|\iota')\beta(\iota')}\right]^G
\end{aligned}
$$

3.3.2 Transition Function

We make the assumption that every component of our state transitions conditionally independent of the other components given the previous state. In addition, each state variable depends only on its previous value and the action taken.

$$
\begin{aligned}
&p(\iota_{t+1}, \beta_{t+1}, \mathcal{I}_{t+1}|\iota_t, \beta_t, \mathcal{I}_t, a_t) \\
&\qquad = p(\iota_{t+1}|\iota_t, a_t)p(\beta_{t+1}|\beta_t, a)p(\mathcal{I}_{t+1}|\mathcal{I}_t, a_t)
\end{aligned}
$$

Object Transition Function As previously stated, we assume the object the human desires does not change unless its desired object is picked up. If the desired object is picked, it transitions uniformly at random between the remaining objects.

If a_t is `pick(`ι_t`)`,

$$p(\iota_{t+1}|\iota_t, a_t) = \begin{cases} 1/(|\mathcal{I}| - 1) & \text{if } \iota_{t+1} \in \mathcal{I} \text{ and } \iota_{t+1} \neq \iota \\ 0 & \text{otherwise} \end{cases}$$

If a_t is any other action:

$$p(\iota_{t+1}|\iota_t, a_t) = \begin{cases} 1 & \text{if } \iota_{t+1} = \iota_t \\ 0 & \text{otherwise} \end{cases}$$

Object Set Transition Function The set of objects $\mathcal{I}$ changes only when an object is picked up. The picked object is removed from the set.

If a_t is a `pick(i)` action:

$$p(\mathcal{I}_{t+1}|\mathcal{I}_t, a_t) = \begin{cases} 1 & \text{if } \mathcal{I}_{t+1} = \mathcal{I}_t \setminus \{i\} \\ 0 & \text{otherwise} \end{cases}$$

If a_t is any other action:

$$p(\mathcal{I}_{t+1}|\mathcal{I}_t, a_t) = \begin{cases} 1 & \text{if } \mathcal{I}_t = \mathcal{I}_{t+1} \\ 0 & \text{otherwise} \end{cases}$$

[4]It is perhaps more accurate to say that the human does not pick proportional to the probability, but picks the action with the greatest value of $p(\iota|o)$. However, this deterministic observation function would lead to irregularities when faced with real human users who do not pick perfectly. Using a softmax function with a tuned temperature parameter would allow us to balance the strong guarantees of of a deterministic max with the flexibility of "proportional to probability".

Belief Transition Function The human's belief about which object the robot will hand over, β_t, is informed by the actions the robot takes. In the same way that the robot's belief state b is a summary of all the observations the robot has made [Kaelbling et al., 1998], β_t is a summary of all observations the human has made of the robot's actions, and reflects the human's estimate about the robot's true state. We therefore update β_t according to Bayesian probabilities in the same way that b is updated. The following is a standard Markovian update:

For a particular state ι_t, its probability after t observations is given as follows:

$$p(\iota_{t+1}|o_{1:t+1}) = p(o_{t+1}|\iota_{t+1}) \sum_{\iota_t} p(\iota_{t+1}|\iota_t) p(\iota_t|o_{1:t})$$

Recall that β is a distribution over states (objects) ι_t, so we can use this rule to write an update for each entry of β.

$$\begin{aligned}\beta_{t+1}(\iota_{t+1}) &= p(\iota_{t+1}|a_{1:t}) \\ &= p(a_t|\iota_{t+1}) \sum_{\iota_t} p(\iota_{t+1}|\iota_t) p(\iota_t|a_{1:t-1}) \\ &= p(a_t|\iota_{t+1}) \sum_{\iota_t} p(\iota_{t+1}|\iota_t) \beta_t(\iota_t)\end{aligned}$$

This requires us to specify $p(a_t|\iota_{t+1})$. Recall that actions are operating as observations from the perspective of the human, making this expression a *reverse observation function.* We must provide a reverse observation function for each action. These will often be very similar to the base-level observations described in Section 3.3.1, though adapted slightly to suit being performed by a Baxter robot. See the appendix for details.

We must also specify a transition function, $p(\iota_{t+1}|\iota_t)$. We will use the object transition function described earlier.

We can rewrite our element-wise update for the whole β vector using a matrix multiplication:

$$\beta_{t+1} = A_t T \beta_t$$

where A_t is a matrix representing the reverse observation function $p(a_t|\iota_{t+1})$ and T is a matrix representing the transition function $p(\iota_{t+1}|\iota_t)$. If the set of items $\iota = \{\iota_1, \iota_2, \ldots, \iota_n\}$, A is a diagonal such that

$$a_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ p(a_t|\iota_{t+1} = \iota_i & \text{otherwise} \end{cases}$$

and T is a matrix with entries

$$t_{ij} = p(\iota_{t+1} = \iota_i | \iota_t = \iota_j)$$

This gives us a deterministic transition function for β:

$$p(\beta_{t+1}|\beta_t, a_t) = \begin{cases} 1 & \text{if } \beta_{t+1} = A_t T \beta_t \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

If we assume that β_0 begins as the uniform distribution over objects, we can now calculate β_t for any given time step deterministically. However, there are many assumptions that were made, and there is additional expressiveness that can be used by allowing for non-deterministic updates to β. However, we lead the exploration of this space for future work.

3.4 Policy Generation

It is possible to calculate an optimal policy to solve our POMDP, however, due its size, doing so is intractable. Moreover, we would like to solve our POMDP at speeds suitable for interaction. Fortunately, approximate solutions exist which allow us to trade off accuracy for computational time. One such solver is the Partially Observable Sparse Sampling (POSS) algorithm [Gopalan et al., 2016], which uses Monte-Carlo Tree Search [Coulom, 2006] and Upper Confidence Bound exploration [Kocsis and Szepesvári, 2006] to derive a policy for our POMDP. In order to improve the accuracy of the planner, we also provide a heuristic for the random rollouts that prevents the rollouts from selecting pick actions that are unlikely to select the correct objects.

We implement this model within the Brown UMBC Reinforcement Learning And Planning [MacGlashan] framework, which allows for many standard AI and Reinforcement learning algorithms to be used with our domain.

3.5 Incremental System Design

Interactive systems that involve speech often make an assumption that the interaction can be split neatly into turns: first one speaker provides a complete utterance, and then the other. This is often not the case, and it is a subtle matter to determine when one turn is completed and the other agent can begin to speak. As a consequence, many systems involving dialogue, including earlier iterations of this project, do not "hear" utterances or gestures while the agent is speaking or acting. Any speech or gestures the human made while the robot was executing its action were dropped and never incorporated into the robot's belief about the state. An additional problem was that while the robot was planning its next action, it would also not incorporate any observations made. This was a significant effect, as it could take up to 10 seconds of planning for the robot to make its next move. As a consequence, most observations would be dropped unless we incorporated a system of turn-taking where we waited a set amount of

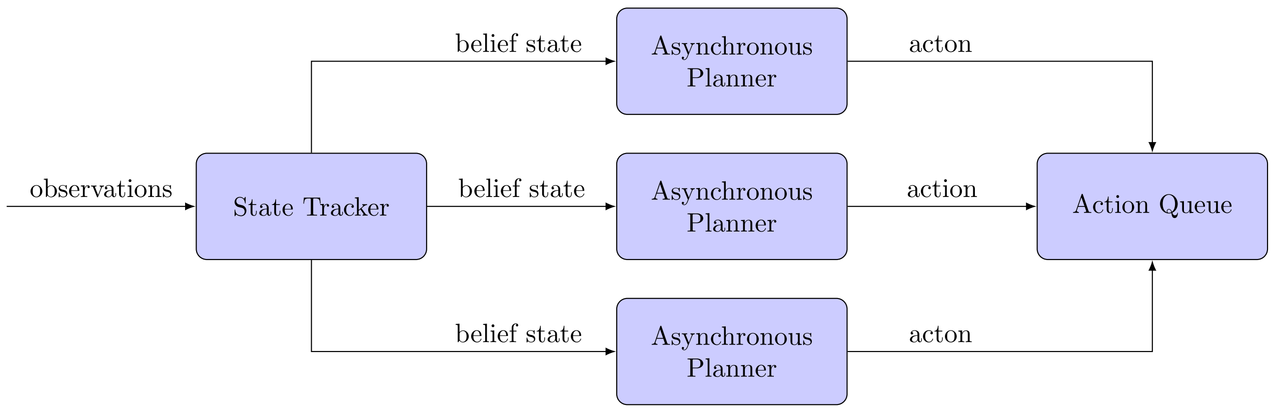

Figure 4: An overview of multithreaded robot interaction framework. Each node is run on a separate thread. Edges are labeled with the type of data transmitted

.

time for the human to provide any speech and gesture input.

In order to create more fluid interactions, we employ a multithreaded solution that separates the state estimation, planning, and executions of actions. This allows our robot to simultaneously track the human's belief, plan its next move, and interact with the human. A diagram of the system is available in figure 4

State Tracker The state tracker incorporates observations from the human at a very high rate, allowing the tracker to always have access to the most up to date state estimation. However, our domain description requires the robot to choose an action between each consecutive observation. We address this by feeding the model a `wait` action unless the planning thread has provided a different action. This requires minimal dependency between actions and observations, which our model avoids.

Asynchronous Planner A separate thread runs the planning code. Planning can take several seconds, during which the robot is not only idle, but also deaf and blind. Running the planning code on a separate thread allows the robot to continue observing and interacting with the human while the robot plans its next move. In addition, we can use several planning threads at once, with separate policies to control the robot's face, arms, head angle, etc. In our work, we use one thread to plan for the POMDP domain described above, and another to animate the Baxter's face to show facial expression of increasing confusion with the entropy of the belief state.

Action Queue The planning threads push their actions onto a queue which executes them one by one, allowing the planning threads to resume planning if the actions they produce take some time to execute. An additional advantage is that this allows the robot to be constantly acting, with minimal idle time.

4 Evaluation

Due to difficulties with the POSS algorithm, we return to Belief Sparse Sampling [Kearns et al., 2002] (BSS) to evaluate the efficacy of our model (for more details about POSS, see section 4.3). BSS, however, takes several minutes to generate an action, which is too slow for user interaction. Instead, we evaluate the performance of our model in simulation. A procedure for user studies with human participants is also described.[5]

4.1 Results in Simulation

To evaluate the performance of our model, we run simulated trials with two different domains: first with a toy domain, and second with the full social feedback domain.

4.1.1 Toy Domain Results

The toy domain is a simplified version of the full social feedback domain. Objects are represented by binary string bits of length 3, and observations tell the robot that a particular bit has some value. There is no noise in observations. The agent can take `wait` and `pick` actions in the non-social trials, and in the social trials, can take non-social actions as well as `inform(x, y)`

[5]We do note that it would be possible to run the full domain in real time to test with users by precomputing and caching a policy for each belief state that describes the optimal action to take from that state. This is the scheme that we used in Wu et al. [2015]

actions, which represent informing the human that the robot believes that the xth digit has value y.

We run 25 trials in simulation. Results are presented in figure 5. We see a sharp reduction in the number of actions needed until the robot can successfully pick the correct object, while accuracy remains the same. In this simple domain, it is easy to identify the correct object with enough observations, accounting for the high accuracy in both trials. However, taking social feedback actions allows the agent to request the observations that it needs from the human to complete the interaction as quickly as possible. Notice also that the variance in the number of actions the robot takes until the first correct pick is very small in the social trials, while it is very large in the non-social trials. Social feedback actions allow the agent to control the interaction so that it is as brief as possible.

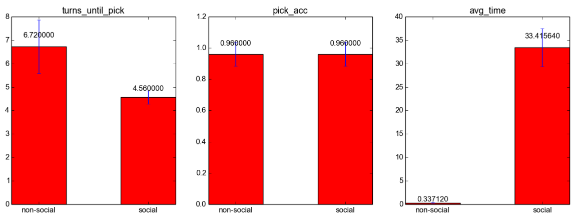

Figure 5: Results in simulation for the toy domain. `turns_until_pick` is the number of timesteps until first correct pick, `pick_acc` is the ratio of correct picks to total picks, `avg_time` is the the time to compute a single action

4.1.2 Full Domain Results

We run two simulated trials: one without any social feedback actions (only `pick(i)` and `wait`), and one with non-social feedback actions and a random subset of six social feedback actions (to reduce computation time). There are four objects on the table for each trial. Each trial uses a different set of randomized actions (to reduce computation costs). Results are presented in figure 6.

We see a small, but statistically significant improvement in accuracy at the cost of slightly less than one additional action for the first correct pick. The less pronounced effects of social feedback are likely due to the greater number of observations that are possible, and the relative probability of seeing a useful observation is reduced. This more complicated domain is also hampered by our reduced horizon of 2, allowing the planner to only consider up to two actions in the future. In the full domain, the agent is much less likely to receive enough observations to correctly pick the object within two timesteps, even using social feedback actions.

To see more improvement in speed and accuracy, we would likely need to use a larger planning horizon, which is only possible with approximate solvers such as POSS. In addition, increasing the number of objects, i.e., the complexity of the task, would likely result in a sharper difference between the number of turns required to give the human the correct object. The robot is unlikely to receive a useful observation unless it can signal to the human which observation it requires.

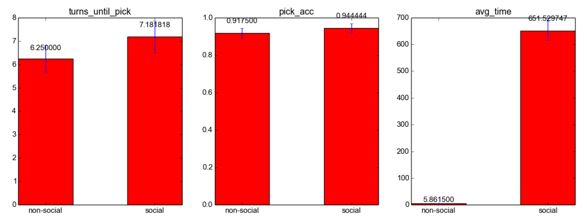

Figure 6: Results in simulation for the full domain. `turns_until_pick` is the number of timesteps until first correct pick, `pick_acc` is the ratio of correct picks to total picks, `avg_time` is the the time to compute a single action

4.2 User Studies Procedure

Users are acquired from convenience sampling by inviting volunteers from the Brown University CIT building. For each user, two trials are run: a baseline trial without any social feedback actions, and a social-feedback trial with social feedback actions enabled. For each trial, the user is given a microphone and invited to use speech and gesture to ask the robot for a specific item on the table, and to not change that object until it was picked up. After their requested object is picked up, they request a new object and repeat the procedure until the robot has handed them all objects on the table. It is decided randomly whether the social or non-social trial is performed first for each user. Between the two trials, they are given a survey to assess the qualitative aspects of the interaction, including metrics such as the robot's perceived intelligence, friendliness, etc.

This procedure is based off a previous user study run for an earlier iteration of this project. In this user study, we noticed a small effect showing the increased speed of interaction with social feedback actions enabled. However, this study was run with only six participants, so these effects were likely not statistically significant. When user studies are run in the future, we will have a larger sample size, but also increase the difficulty of the task by adding more objects, which should further highlight the necessity of social feedback.

Interesting metrics to measure include are how the perception of the robot is correlated to the actions the robot takes, as well as the accuracy and speed of the interaction. We may also consider a longer-running user study to investigate how people learn how to interact with the robot.

4.3 What happened to POSS?

Initially we planned to use POSS to solve our POMDP. As an approximate solver, it would have been possible to solve the POMDP at speeds quickly enough for real-time user interaction. In addition, we would not be limited by short planning horizons, which prevent our robot from planning far into the future. However, despite our efforts to integrate POSS, the nature of the algorithm was shown to be incompatible with our domain.

POSS is a Monte Carlo Tree Search algorithm that uses Upper Confidence Bounds (UCB) to decide which action to take. Specifically, selecting an action is treated as a multi-armed bandit problem, where the reward is the discounted reward for taking that action. The downside to this approach is that if taking an action results in an incorrect pick (which gives the agent a reward of -50) anywhere down the tree, then the planner is unlikely to ever try that action again. Because the POSS algorithm relies on random rollouts to generate actions when it has no estimates for the best action, it is extremely likely to choose an incorrect pick action. In attempt to rectify this behavior, we created a heuristic for the random rollouts that prevents the algorithm from choosing pick actions that are unlikely to be correct. Unfortunately, this heuristic was not enough (see figure 7. Using POSS with this heuristic, we achieve only 25% accuracy, which is only about twice as good as random chance. The algorithm only waits an average of about one action before deciding to pick. But, at the same time, the computation time is extremely fast: much less than a second, sometimes more than 100 times faster than Belief Sparse Sampling.

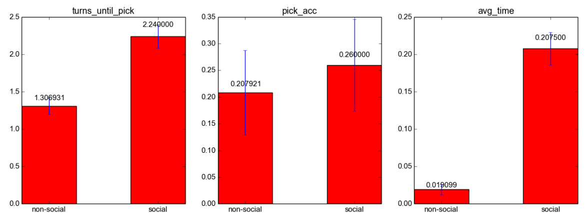

Figure 7: Results for POSS with Heuristic function in the toy domain.`turns_until_pick` is the number of timesteps until first correct pick,`pick_acc` is the ratio of correct picks to total picks, `avg_time` is the the time to compute a single action.

Despite removing the main source of negative reward, the randomness of the rollout was still obscuring the true utility of taking most actions. We then attempted to remove the randomness by forcing the rollout to choose the action according to Belief Sparse Sampling. This removed much of the variance in the q-values of the actions (which describe the utility of a given action), but this still did not produce a consistent policy for the following reason: Upper Confidence Bound methods tend to underestimate the q-values of actions. However, for terminal actions like `pick(i)` in the toy domain, it will give fairly accurate results. As a result, the more beneficial `inform(x, y)` actions can be given lower q-values than terminal `pick(i)` actions, resulting in very incorrect policies.

In order to solve our POMDP, we will need to rely on approximate solvers. Whether the domain or POSS can be tweaked to be compatible must be investigated. If they cannot be made compatible, we will explore other approximate solving methods, including policy improvement and non-UCB-based Monte Carlo methods.

5 Future Work

The next task will be to run user-studies with human participants to compare the performance of the system with and without social feedback. This will require the use of approximate solvers to allow us to solve the POMDP in real time, or to precompute policies.

We would also like to expand the domain by allowing for more modalities of user input, such as other types of gestures as well as more subtle cues such as expression or emotion.

This thesis work relies on very simple observation functions. In the future, we will investigate more sophisticated observation functions, allowing for language interpretation that can understand prepositional phrases and other ways of composing referring expressions.

Other areas for further exploration can be found by removing many of the assumptions made in this thesis. For example, the set of objects on the table was assumed to be known. Attempting to learn from the human more about the set of objects, such as their vocabularies and location, would be an interesting expansion.

Use of the β variable can be expanded to include uncertainty over the value of β_t, allowing for ambiguity in how the human interprets the robot's action. In the same vein, additional modes of interaction from both the robot and human can be explored.

The object delivery domain can also be abstracted to a domain where the goal is not to determine which object the human wants, but the human's intent in general. This system of tracking the human's belief about the robot's communicated intent can be widely applied to other human-robot interaction tasks.

6 Acknowledgements

This thesis is the result of work that has been ongoing since December 2014, including a AAAI workshop paper, Wu et al. [2015]. I would like to acknowledge my many collaborators, Advik Guha, Miles Eldon, David Whitney, Horatio Han, Nakul Gopalan, Lawson Wong, James MacGlashan, and John Oberlin.

This project is part of an ongoing initiative towards social robotics by Brown University's H2R laboratory under Professor Stefanie Tellex.

7 Appendix

7.1 Posterior Observation Function: Toy Example

In order to demonstrate and motivate the posterior observation function, we will define a toy domain. In this toy domain, the states are the following: $\{AA, AB, BA, BB\}$ as well as β, which is a distribution over these states. The observations are $\{A\_, \_A, B\_, \_B\}$, which mean “the first character is an A”, “the second character is an A”, “the first character is B”, “and the second character is a B” respectively. These observations are provided by the human. The agent can take actions CX which is informing the human that the agent believes the Xth character is a C, as well as “picking” the object or waiting. For the basel level observation, the human always gives truthful observations, and has equal probability of generating an observation pertaining to a particular character. We will see how this affects the POMDP observation function, which incorporates β.

Consider the following situation:

Intially both b, the robot’s belief about the human’s desired object and β, the human’s belief about the robot’s belief, are uniform. The true state is AA.

t	b	β	a	o
0	[0.25, 0.25, 0.25, 0.25]	[0.25, 0.25, 0.25, 0.25]		

The robot then receives an observation $A\_$. The new beliefs are:

t	b	β	a	o
0	[0.25, 0.25, 0.25, 0.25]	[0.25, 0.25, 0.25, 0.25]		
1	[0.50, 0.50, 0, 0]	[0.25, 0.25, 0.25, 0.25]		$A\_$

Next, the robot can choose to take an action. If it chooses to wait, this is the resulting state

t	b	β	a	o
0	[0.25, 0.25, 0.25, 0.25]	[0.25, 0.25, 0.25, 0.25]		
1	[0.50, 0.50, 0, 0]	[0.25, 0.25, 0.25, 0.25]		$A\_$
2	[0.50, 0.50, 0, 0]	[0.25, 0.25, 0.25, 0.25]	wait	

Examine the probabilities of each observation.

$$\begin{aligned}
p(A\_|AA,\beta) &= \frac{p(A\_)\beta(AA)}{p(A\_)\beta(AA)+p(A\_)\beta(AB)+p(A\_)\beta(BA)+p(A\_)\beta(BB)} \\
&= \frac{0.5*0.25}{0.5*0.25+0.5*0.25+0+0} \\
&= 0.5
\end{aligned}$$

$$\begin{aligned}
p(\_A|AA,\beta) &= \frac{p(\_A)\beta(AA)}{p(\_A)\beta(AA)+p(\_A)\beta(AB)+p(\_A)\beta(BA)+p(\_A)\beta(BB)} \\
&= \frac{0.5*0.25}{0.5*0.25+0+0.5*0.25+0} \\
&= 0.5
\end{aligned}$$

The probabilities of all other actions are 0, since we only give truthful observations.

Notice that this situation is not ideal. The agent has equal probabilities of receiving either observation, even though the agent already knows that the first character is an A. Now, consider what would happen if the robot chose the action $A0$. We would get the following belief states:

t	b	β	a	o
0	[0.25, 0.25, 0.25, 0.25]	[0.25, 0.25, 0.25, 0.25]		
1	[0.50, 0.50, 0, 0]	[0.25, 0.25, 0.25, 0.25]		$A\_$
2	[0.50, 0.50, 0, 0]	[0.5, 0.5, 0, 0]	$A0$	

If we examine the probabilities again:

$$\begin{aligned}
p(A\_|AA,\beta) &\propto \frac{p(A\_)\beta(AA)}{p(A\_)\beta(AA)+p(A\_)\beta(AB)+p(A\_)\beta(BA)+p(A\_)\beta(BB)} \\
&\propto \frac{0.5*0.5}{0.5*0.5+0.5*0.5+0+0} \\
&\propto 0.5 \\
&= \frac{1}{3}
\end{aligned}$$

$$\begin{aligned}
p(\_A|AA,\beta) &\propto \frac{p(\_A)\beta(AA)}{p(\_A)\beta(AA)+p(\_A)\beta(AB)+p(\_A)\beta(BA)+p(\_A)\beta(BB)} \\
&\propto \frac{0.5*0.5}{0.5*0.5+0+0+0} \\
&\propto 1 \\
&= \frac{2}{3}
\end{aligned}$$

Now, we are more likely to get the observation $\_A$, which is more useful to us than the observation $A\_$, since it gives us the information we need to pick the correct object, AA. This provides an incentive for the robot to choose an action that conveys something about its current state of belief over doing nothing.

7.2 Reverse Observation Functions

We define a reverse observation function for each action the robot can take. These define how β is updated when a robot takes an action.

For `look(x)` actions, we define an angle ϕ_l to be the angle between a vector from the robot's face to the location x and the vector from the robot's face to the location of object ι_{t+1}. We assume this angle is chosen from a normal distribution centered at 0 with a hand tuned variance, v_l.

$$p(a_t = \texttt{look(x)}|\iota_{t+1}) = \mathcal{N}(\phi_l|0, v_l)$$

For `point(x)` actions, we define an angle ϕ_p to be the angle between the vector from Baxter's wrist to location x and the vector from Baxter's wrist joint to the object ι_{t+1}. This gesture is chosen according to a normal distribution with hand-tuned variance v_p.

$$p(a_t = \texttt{point(x)}|\iota_{t+1}) = \mathcal{N}(\phi_p|0, v_p)$$

Actions `say(p)` take the form of a statement "I believe that the object you want is p", where p is some property from some object's vocabulary. The probability of this action given some ι is identical to the speech model for the human, given in section 3.3.1.

References

Stefanie Tellex, Thomas Kollar, Stephen Dickerson, Matthew R. Walter, Ashish Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI*, 2011.

C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In *Proc. of the 13th Intl Symposium on Experimental Robotics (ISER)*, 2012.

Stefanie Tellex, Pratiksha Thaker, Robin Deits, Thomas Kollar, and Nicholas Roy. Toward information theoretic human-robot dialog. 2012.

DK Misra, Jaeyong Sung, Kevin Lee, Ashutosh Saxena, Jaeyong Sung, Bart Selman, Ashutosh Saxena, Jaeyong Sung, Colin Ponce, Bart Selman, et al. Tell me dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *Robotics: Science and Systems, RSS*, 2014.

Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2), 1998.

Miles Eldon. Incrementally Interpreting Multimodal Referring Expressions in Real Time. Undergraduate Thesis, May 2015.

Emily Wu, Yuxin Han, David Whitney, John Oberlin, James MacGlashan, and Stefanie Tellex. Robotic social feedback for object specification. *AAAI: Artificial Intelligence for Human-Robot Interaction*, 2015.

Herbert H Clark and Meredyth A Krych. Speaking while monitoring addressees for understanding. *Journal of Memory and Language*, 50(1):62–81, 2004.

Mary Ellen Foster, Andre Gaschler, Manuel Giuliani, Amy Isard, Maria Pateraki, and Ronald P. A. Petrick. Two people walk into a bar: dynamic multi-party social interaction with a robot agent. In *International Conference on Multimodal Interaction, ICMI '12, Santa Monica, CA, USA, October 22-26, 2012*, pages 3–10, 2012. doi: 10.1145/2388676.2388680. URL `http://doi.acm.org/10.1145/2388676.2388680`.

Dan Bohus and Eric Horvitz. Managing human-robot engagement with forecasts and... um... hesitations. In *Proceedings of the 16th International Conference on Multimodal Interaction*, ICMI '14, pages 2–9, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2885-2. doi: 10.1145/2663204.2663241. URL `http://doi.acm.org/10.1145/2663204.2663241`.

Joyce Y Chai, Lanbo She, Rui Fang, Spencer Ottarson, Cody Littley, Changsong Liu, and Kenneth Hanson. Collaborative effort towards common ground in situated human-robot dialogue. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pages 33–40. ACM, 2014.

Matthew MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1475–1482. AAAI Press, 2006.

Sergio Guadarrama, Erik Rodner, Kate Saenko, Ning Zhang, Ryan Farrell, Jeff Donahue, and Trevor Darrell. Open-vocabulary object retrieval. In *Robotics: Science and Systems*, 2014.

Daniel Hewlett, Thomas J Walsh, and Paul Cohen. Teaching and executing verb phrases. In *Development and Learning (ICDL), 2011 IEEE International Conference on*, volume 2, pages 1–6. IEEE, 2011.

Stefan Waldherr, Roseli Romero, and Sebastian Thrun. A gesture based interface for human-robot interaction. *Autonomous Robots*, 9(2):151–173, 2000.

Matthew Marge, Aaron Powers, Jonathan Brookshire, Trevor Jay, Odest C Jenkins, and Christopher Geyer. Comparing heads-up, hands-free operation of ground robots to teleoperation. *Robotics: Science and Systems VII*, 2011.

Cynthia Matuszek, Liefeng Bo, Luke Zettlemoyer, and Dieter Fox. Learning from unscripted deictic gesture and language for human-robot interactions. 2014.

David Schlangen, Timo Baumann, and Michaela Atterer. Incremental reference resolution: The task, metrics for evaluation, and a Bayesian filtering model that is sensitive to disfluencies. In *Proceedings of the SIGDIAL 2009 Conference*, page 30–37, London, UK, September 2009. Association for Computational Linguistics, Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W09/W09-3905`.

H. Zender, Martinez O. Mozos, P. Jensfelt, G. J. M. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56:493–502, 2008.

Petra Gieselmann. Reference resolution mechanisms in dialogue management, 2004.

Casey Kennington, Spyros Kousidis, and David Schlangen. Interpreting situated dialogue utterances: an update model that uses speech, gaze, and gesture information. In *Proceedings of the SIGDIAL 2013 Conference*, page 173–182, Metz, France, August 2013. Association for Computational Linguistics, Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W13-4030`.

Casey Kennington, Livia Dia, and David Schlangen. A discriminative model for perceptually-grounded incremental reference resolution. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 195–205, London, UK, April 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W15-0124`.

Joyce Yue Chai, Zahar Prasov, and Shaolin Qu. Cognitive principles in robust multimodal interpretation. *CoRR*, abs/1109.6361, 2011. URL `http://arxiv.org/abs/1109.6361`.

G.M. Kruijff, M. Brenner, and N. Hawes. Continual planning for cross-modal situated clarification in human-robot interaction. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pages 592–597, Aug 2008. doi: 10.1109/ROMAN.2008.4600731.

Steve Young, M Gašić, Blaise Thomson, and Jason D Williams. POMDP-based statistical spoken dialog systems: A review. 2013.

Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, 2013.

Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, pages 49–79, 2005.

John Laird. *The Soar cognitive architecture*. MIT Press, 2012.

Paul W Schermerhorn, James F Kramer, Christopher Middendorff, and Matthias Scheutz. Diarc: A testbed for natural human-robot interaction. In *AAAI*, pages 1972–1973, 2006.

Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *Int. J. Rob. Res.*, 29(8):1053–1068, July 2010. ISSN 0278-3649. doi: 10.1177/0278364910369861. URL `http://dx.doi.org/10.1177/0278364910369861`.

Nakul Gopalan, James MacGlashan, and Stefanie Tellex. Solving human-robot collaborative tasks using pomdps. In *International Conference on Robotics and Automation*, 2016. in submission.

Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and games*, pages 72–83. Springer, 2006.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293. Springer, 2006.

James MacGlashan. Brown UMBC Reinforcement Learning and Planning Library. http://burlap.cs.brown.edu/.

Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning*, 49(2-3):193–208, 2002.