

# What Lies Ahead? Expectation Management in Human-Robot Collaboration

Guy Hoffman and Cynthia Breazeal

MIT Media Laboratory  
20 Ames Street  
Cambridge, MA 02139  
{guy,cynthiab}@media.mit.edu

## Abstract

We aim to build robots that go beyond command-and-response and can engage in *fluent* collaborative behavior with their human counterparts. This paper discusses one aspect of collaboration fluency: *expectation management* — predicting what a human collaborator will do next and how to act on that prediction. We propose a formal time-based collaborative framework that can be used to evaluate this and other aspects of collocated human-robot teamwork, and show how expectation management can enable a higher level of fluency and improved efficiency in this framework. We also present an implementation of the proposed theoretical framework in a simulated human-robot collaborative task.

## Introduction

Two people performing an activity together naturally display a high level of coordination and flexibility. Their timing is for the most part impeccable, and this behavior emerges often without exchanging much verbal information. On the other hand, human-robot interaction (HRI) is usually unintuitive, restrictive, and limited to a rigid command-and-response structure. If robots are to enter human environments — be it homes, offices, hospitals or schools — they must steer away from this trend and display a significantly more fluid meshing of their actions with ours. To that end, robot designers must address many issues, such as mutual responsiveness, intention-reading, the concoction of shared plans, and the mastery of nonverbal communication.

We argue that one of the skills enabling fluent collaboration, and a prerequisite to true mutual responsiveness, is the gradual forming of expectations of other team members' actions, and the subsequent acting on these expectations. If we are to go beyond stop-and-go interaction, the agent must take into account not only past events and current perceived state, but also its expectations from the human collaborator.

Clearly, acting on expectations holds a risk of misstepping in cases where expectations are not fulfilled. This should, however, result in realistic mistakes on the robot's part, and the frequency of these mistakes should decrease

as the agent's model of the human improves, and as the human's adaptation to the robot's behavior becomes more polished.

In order to investigate the effect of expectation management on the efficiency and fluency of human-robot collaboration, we propose a formal framework for examining fluency in collocated human-robot teamwork. We investigate a model of preemptive action based on expectation, and compare a collaborative robotic agent implementing this framework to a purely reactive agent acting within a traditional perception-action loop. We discuss the integration of expectations based on a learned behavior model for the human's action, as well as a structural model of the environment. We then present a software environment we are currently using to compare the performance of these agents when working with untrained human collaborators.

We are currently in the process of acquiring results from user studies using our software, which we hope to report in the near future.

## World Description

We frame the collaboration timing problem as a discrete time-based deterministic decision process:

$$W = \langle \Sigma, D, A_H, A_R, T \rangle$$

The world includes two agents, a robot and a human, working together on a shared task.

Both robot and human share a common workspace, which at any time point  $t$  is in one of a finite number of states  $\Sigma_W = \{s_0^w, \dots, s_n^w\}$ , and is initially in state  $s_0^w$ . A subset  $G_W \subset \Sigma_W$  includes states denoted as goal states. The state at time  $t$  is denoted  $\sigma_t^w$ .

The agents also have a number of states  $\Sigma_H = \{s_0^h, \dots, s_n^h\}$  (the human's states) and  $\Sigma_R = \{s_0^r, \dots, s_n^r\}$  (the robot's states). In our model the robot can only perceive the state of the workspace if it is in state  $s_0^r$  (the *perceptive* state). At time  $t$  the human's state is  $\sigma_t^h$ , and the robot's state is  $\sigma_t^r$ .

We denote full state of the system at time  $t$  as

$$\sigma_t \equiv \langle \sigma_t^w, \sigma_t^h, \sigma_t^r \rangle$$

Similarly,

$$\Sigma = \Sigma_W \times \Sigma_H \times \Sigma_R$$

Human and robot have distinct abilities, described as two sets of actions,  $A_H = \{a_1^h, \dots, a_k^h\}$  for the human, and  $A_R = \{a_1^r, \dots, a_l^r\}$  for the robot.

$T : ((A_H \cup A_R) \times \Sigma) \mapsto \Sigma$  is a transition function that maps certain state-action pairs to new states. We denote a particular state-action pair transition in  $T$

$$\tau_i^x(s_k) = s_l \equiv \langle a_i^x, s_k \rangle \mapsto s_l$$

meaning that if agent  $x \in \{h, r\}$  (human or robot) performed action  $i$  while the world is in state  $s_k$ , the world would transition into state  $s_l$  after applying the action.

However, a central motivation of our model is to investigate aspects of *time* associated with actions of two collaborating agents. Therefore, state transitions are not atomic, and the decision to take a particular action does not result in an immediate state transition. Instead, moving between states takes time, and is associated with a known discrete cost, which is a function of the states before and after the action. This cost can be thought of as the ‘distance’ between states, or more generally — the duration it takes to transition between states. We denote the costs of the system with  $D$ :

$$D : \left( \bigcup_{i \in W, H, R} \langle \Sigma_i \times \Sigma_i \rangle \right) \mapsto \mathbb{N}$$

Thus when, at time  $t$ , agent  $x \in \{h, r\}$  decides to take action  $a_i^x$  on state  $s_k$  and  $\tau_i^x(s_k) = s_l$ , the world will be in state  $s_l$  only at time  $t + d(s_k, s_l)$ . While the other agent may take more actions during this time, the next time step agent  $x$  will be able to take another action is  $t + d(s_k, s_l)$ .

It can be useful to depict the state transitions as a directed graph, with the nodes representing the states and the edges the transitions between the states, weighted by the duration/cost function  $D$  (see Figure 3). For sake of simplicity, we will sometimes denote  $d(s_k, s_l)$  as  $d_{kl}$  as indicated in the figure.

Agents cannot change the other agent’s states with their actions, but they operate on a common workspace. Therefore, our model is clearly ill-defined with regards to race conditions on the  $\Sigma_W$  state space. There are several possible solutions (such as the use of semaphores and other synchronization mechanisms). In this paper, for the sake of simplicity, we will assume that actions that change the workspace are locking with regards to actions that operate on the common workspace *for both agents*, i.e. neither of the two agents can take an action that causes a state transition in  $\Sigma_W$  while another such action is being performed by one of the agents. In the implementation of our model described below, we solve this race condition by making all state transitions effecting the workspace atomic.

## The Factory World

We have instantiated the above model in a simulated factory setting (see Figure 1). The goal of the team is to assemble a cart made of a *Body*, a *Floor*, two *Axles*, and four *Wheels* (see Figure 2). The various parts have particular ways to be

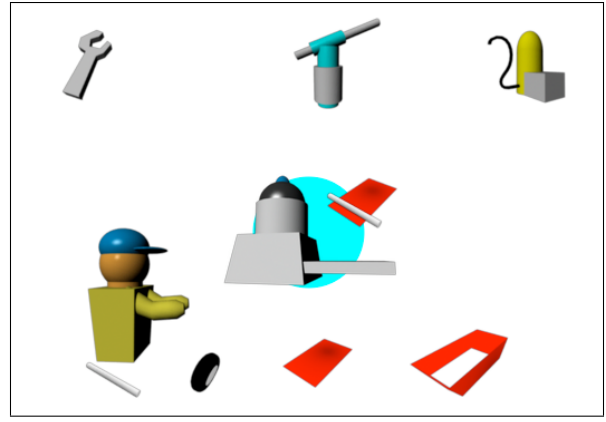


Figure 1: Simulated factory setting with a human and a robot sharing a workspace, but dividing their tasks. The robot has access to the tools (top, left to right: *Wrench*, *Rivet-Gun*, *Welder*), whereas the human is responsible to bring the parts (bottom, left to right: *Axle*, *Wheel*, *Floor*, *Body*) to the workspace.



Figure 2: The goal of the task is to assemble a cart made of one floor, one body, two axles, and four wheels.

attached to each other — the *Body* is welded to the *Floor*, *Axles* are riveted to the *Floor* and the *Wheels* are attached to the *Axle* using a wrench. A *component* is a partially or fully assembled cart segment that includes one or more individual parts, all attached to each other, denoted by the parts names delimited by +, for example *Axle + Axle + Body + Floor*.

The labor is divided between the human and the agent as follows: the human has access to the individual parts, and is capable of carrying them and positioning them on the workspace. The robot, on the other hand, is solely responsible for fetching the correct tool and applying it to the currently pertinent component configuration in the workspace. Each part has a stock location (with an infinite supply of parts), and each tool has a storage location, to which it has to be returned for the robot to be able to find it again. The workspace can, at any one time, contain at most two components.

In the above described framework, the factory workspace

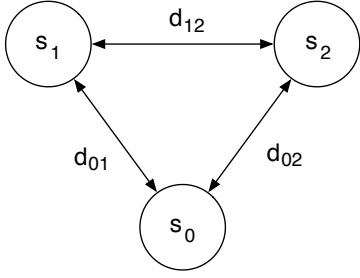


Figure 3: Sample state transition cost graph

state space  $\Sigma_W = Comps^2$ , where *Comps* is the space of all possible components. Note that this is not  $2^{Parts}$ , since not all parts can be attached to each other, and some parts can appear multiple times in a component. In our case,  $|Comps| = 24$ , and thus  $|\Sigma_W| = 24 \times 24 = 576$ .

The robot's state space includes its position at one of the three tools' storage areas or at the workspace, and whether the robot is holding one of the tools or not:

$$\begin{aligned} \Sigma_R &= L \times H \\ L &= \{Workspace, Welder, RivetGun, Wrench\} \\ H &= \{\emptyset, Welder, RivetGun, Wrench\} \end{aligned}$$

$|\Sigma_R| = 16$ . Similarly, with 4 kinds of parts,  $|\Sigma_H| = 25$ . Thus, the size of the whole state-space of our simulation is  $|\Sigma| = 576 \times 25 \times 16 = 230400$ .

The action-space of the robot is

$$A_R = \{Workspace, Welder, Rivet, Wrench, PickUp, PutDown, Use\}$$

The effects of each of these actions is dependent on the current state of the robot and the workspace. *Use* is the only action that changes the workspace, if the robot is holding a tool and is located at the workspace. The first four actions are mobility actions, moving to one of the four locations in the factory. *PickUp* and *PutDown* are operational only in the tool locations, with the latter only available at the correct storage location of the currently held tool.  $A_H$  is a similar space with one more navigation action, and no *Use* action.

Actions have their expected effects. Below are some examples of transitions brought about by actions in  $A_R$ :

$$\begin{aligned} \tau_{Pickup}^r(\langle s_i^w, s_j^h, \langle Wrench, \emptyset \rangle \rangle) &= \\ &= \langle s_i^w, s_j^h, \langle Wrench, Wrench \rangle \rangle \end{aligned}$$

$$\begin{aligned} \tau_{Use}^r(\langle \langle Floor, Body \rangle, s_j^h, \langle Wrkspc, Welder \rangle \rangle) &= \\ &= \langle \langle Floor + Body, \emptyset \rangle, s_j^h, \langle Wrkspc, Welder \rangle \rangle \end{aligned}$$

$$\begin{aligned} \tau_{RivetGun}^r(\langle s_i^w, s_j^h, \langle Wrench, \emptyset \rangle \rangle) &= \\ &= \langle s_i^w, s_j^h, \langle RivetGun, \emptyset \rangle \rangle \end{aligned}$$

The duration cost of a robot state transition that involves navigation is the distance between the previous and the new location. The duration cost for state transitions involving the inventory of the robot, or changes to the workspace, are currently 1, but could theoretically be different for each tool.<sup>1</sup>

## Perception

As noted previously, the robot can perceive the state of the workspace only when it is in its perceptive space. In this instance of our model, the robot's perceptive state is when it is in the workspace. Therefore, in the robot's world representation, workspace state changes that happen while the robot is in any other state are not applied. This discrepancy between the real world state and the robot's internal representation of the world is realistic, and requires the robot to occasionally return to the perceptive state to update its world view.

Moreover, we assume that the robot has a function

$$\Phi : \Sigma_W \mapsto H - \emptyset$$

that maps the workspace state to the appropriate tool required to bond the two components in the workspace. For example:

$$\phi(\langle Floor + Axle, Wheel \rangle) = Wrench$$

This can be a lookup table, or a simple decision process. We modeled the components based on open and close male and female attachments to deduct  $\Phi$ . Also note that some workspace states do not warrant any tool, because they either have an empty component, or two components that cannot be attached. We mark these function values as  $\phi(s_i^w) = \emptyset$ .

## Naïve Agent

The baseline for our study is a naïve agent that is purely responsive to its environment and internal state. Its action policy  $\Pi$  is shown in Figure 4; or in words:

1. If in the workspace, holding nothing, and the components warrant a tool, go to that tool.
2. If in a tool area and holding nothing, pick up the tool.
3. If in a tool area and holding a tool, go to workspace.
4. If in the workspace and holding the correct tool for the components, use the tool.
5. If in the workspace and holding the incorrect tool, or the components warrant nothing, put the tool back.
6. If in a tool area and holding a tool, put it down.
7. If in a tool area and holding nothing, return to workspace.

<sup>1</sup>For example: welding can take longer than riveting, and picking up the wrench could be faster than picking up the welder.

$$\left\{ \begin{array}{l} \pi(\langle s_i^w, s_j^h, \langle Workspace, \emptyset \rangle \rangle) = \phi(s_i^w) \\ \pi(\langle s_i^w, s_j^h, \langle x \in L - Workspace, \emptyset \rangle \rangle) = PickUp^* \\ \pi(\langle s_i^w, s_j^h, \langle x \in L - Workspace, x \rangle \rangle) = Workspace^* \\ \pi(\langle s_i^w, s_j^h, \langle x \in H, s_j^h, \langle Workspace, x \rangle \rangle) = Use \\ \pi(\langle s_i^w, s_j^h, \langle x \in H, s_j^h, \langle Workspace, x \rangle \rangle) = x \\ \pi(\langle s_i^w, s_j^h, \langle x \in L - Workspace, x \rangle \rangle) = PutDown^* \\ \pi(\langle s_i^w, s_j^h, \langle x \in L - Workspace, \emptyset \rangle \rangle) = Workspace^* \end{array} \right.$$

Figure 4: Naïve policy for factory robot. \*Based on boolean action history variable. See note in text.

Note that the third and the sixth rules, as well as the second and the seventh rule in the policy are contradictory, and need to be disambiguated. In order to do so, the robot has one more internal state variable. This is a boolean action-history variable indicating whether the last action was *PickUp* or *PutDown*. If this variable is true, rule three (or seven) is invoked; if false, rule six (or two) is used. This prevents oscillating between picking up and putting down tools in the tool area.

### Conservative Tool Return

We have begun experimenting with this agent and several human collaborators. The most obvious fallacy of the naïve agent occurs when the same tool is needed twice in a row (which happens quite frequently with the wheels and axles). Since the agent makes no predictions as to the next step, a tool is always returned immediately after use, sometimes resulting in a superfluous sequence of returning and then using the same tool:

$$\begin{aligned} x \in L &\rightarrow PickUp \rightarrow Workspace \rightarrow Use \rightarrow x \rightarrow \\ &\rightarrow PutDown \rightarrow Workspace \rightarrow x \rightarrow PickUp \rightarrow \\ &\rightarrow Workspace \rightarrow Use \end{aligned}$$

If the distance between the workspace and the tool is  $d$ , and under the assumption that the time it takes for the human to bring the next part  $h$  is smaller than  $4d + 3$ , the total cost of this sequence is  $6d + 5$ .

In these cases, the naïve policy can be improved by changing the fifth rule to be replaced by the following two rules:

$$\begin{aligned} \pi(\langle s_i^w, s_j^h, \langle Wkspc, x \rangle \rangle) &= Wkspc \\ \pi(\langle s_i^w, s_j^h, \langle x \in H, \emptyset \rangle, s_j^h, \langle Wkspc, x \rangle \rangle) &= x \end{aligned}$$

or:

1. If in the workspace and the components warrant nothing, stay in the workspace
2. If in the workspace and the components warrant a different tool than the one the agent is holding, put the tool back

This prevents the agent from returning a tool before it knows for sure that it's not needed again in the next step. In the above case, the sequence would be:

$$x \in L \rightarrow PickUp \rightarrow Workspace \rightarrow Use \xrightarrow{\delta} Use$$

Where the time delay between the two *Uses* would be

$$\delta = \begin{cases} h - (2d + 2) & \text{if } h > 2d + 2, \\ 0 & \text{otherwise.} \end{cases}$$

The total cost of the sequence is  $2d + 3 + \delta$ . The gain in performance is  $4d + 2 - \delta$ .

However, depending on the cost function on the robot's state transitions, there might be a negative impact of this strategy in the case where the next tool will be different. It is easy to show that if the distances between the tools are as depicted in Figure 3, and the tool at location  $s_2^r$  is needed after the tool at location  $s_1^r$ , the cost effect of the conservative tool return strategy would be  $\delta - d_{01} - d_{02} + d_{12}$  (ignoring atomic action costs).

This comes to show that even a very intuitive policy improvement is not demonstrably more optimal than the naïve approach, and its cost effect in the two-tool case is dependent not only on the known world configuration, but also on  $\delta$ , which can not be known, but only estimated, by the robotic agent. Additionally, the overall expected cost effect is dependent on the probability distribution on the workspace configuration over time.

Estimating this probability distribution and acting upon it is the topic of the following section.

## Expectation Management

Given the theoretical analysis in the previous section, we can assume that — in a realistic scenario — the human collaborator will increasingly choose their action so as to minimize the discussed cost. Humans are remarkably adaptive and become increasingly effective when performing repetitive trials of an identical task collaborating with a consistent teammate. This can usually be seen even over a very short period of time engaging in a novel task. As argued in the introduction, we believe that the learning and managing of expectations of each other's collaborative behavior might be a key ingredient the achievement of fluency within the team.

Therefore we expect that the human collaborator will work to minimize the cost impact of the naïve agent policy, adapting to the agent's peculiarities. Indeed, we have

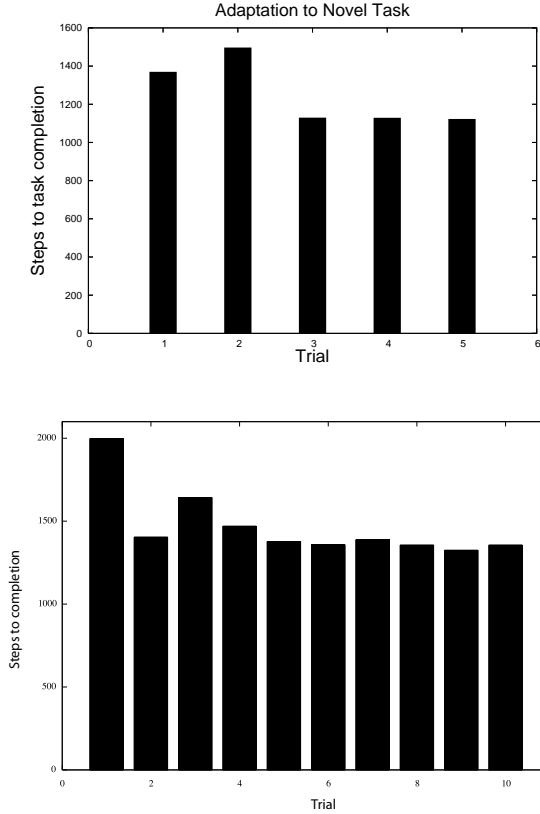


Figure 5: Human adaptation to a novel task. Two users interacting with the naïve agent increasingly adapt their behavior to match the robot’s process. As a result, task completion time drops and levels out at circa 20% lower than in the initial trials. This cost level is usually maintained.

been able to observe this behavior in anecdotal trials with users who have not been exposed to the interaction scenario beforehand. Figure 5 shows characteristic results.

However, in order to fully take advantage of the collaborative interaction, it is our belief that the robot should also engage in a form of expectation management, with the aim of achieving a more fluent interaction with the human collaborator. To that end, we model two kinds of expectation management: *history-based* and *situation-based*. History-based expectation estimates future states based on past states and a consistency assumption. Situation-based expectation estimates future states based on domain knowledge about the current state. Both approaches can be combined in the decision making process, as will be shown below.

### History-Based Expectation

History-based expectation assumes that over time, the human collaborator will act consistently, i.e. will make similar decisions under similar circumstances. Consequently, this approach estimates future states based on past states.

Under the most straightforward history-based expectation

framework, the agent models the workspace as a first-order Markov Process. The probability of workspace state at time  $t$ ,  $\sigma_t^w = s_i^w$ , is thus conditional on  $\sigma_{t-1}^w$  and denoted as

$$p_{i|j}^w \equiv Pr(\sigma_t^w = s_i^w | \sigma_{t-1}^w = s_j^w)$$

A presumably more realistic model would be to view the collaboration as a Hidden Markov Model, with the human state transitions being hidden, and the workspace transitions (which are perceivable to the robot) as the evidence layer of the model. However, since — on one hand — many of the human’s state transitions do not affect the visible workspace state, and — on the other hand — the probability of workspace transitions conditional on the human state transitions  $Pr(\sigma_t^w = s_i^w | \sigma_{t-1}^h = s_j^h)$  are not independent of  $\sigma_{t-1}^w$ , it is unclear whether such a model would indeed be of value in our domain. We therefore leave this extension to future investigation.

The agent can learn the parameters of this Markov process using a naïve Bayesian estimate. To do this, the agent keeps a one-step history of the state transitions of the workspace, whenever it is in state  $s_0^r = \langle Workspace, x \rangle$  and perceives a change. A change from state  $s_j^w$  at some time  $t-l$  to state  $s_i^w$  at time  $t$  increases the counter  $n_{i|j}$ . Consequently,  $p_{i|j}^w$  is computed as

$$p_{i|j}^w = \frac{n_{i|j}}{\sum_{k=1}^{|\Sigma_w|} n_{k|j}}$$

However, in order to estimate the cost of preemptive action as described in the following section (which is ill-defined for non-constructive workspace states), and also to reduce the decision state space, the robot in our factory domain can alternatively model the probability of the tool needed based on the previous state. If  $Q(x) = \{1 \leq i \leq n : \phi(s_i^w) = x\}$  is the set of indices of workspace states that warrant tool  $x$ , the new probability model learned is now

$$p_{x|j} \equiv Pr(\sigma_t^w \in Q(x) | \sigma_{t-1}^w = s_j^w)$$

We estimate this model as follows: a change from state  $s_j^w$  at some time  $t-l$  to state  $s_i^w \in Q(x)$  at time  $t$  increases the counter  $n_{x|j}$ . Using a Laplace correction of 1,  $p_{x|j}$  is then estimated by

$$p_{x|j} = \frac{n_{x|j} + 1}{\sum_{k=1}^{|H-\emptyset|} n_{k|j} + 1}$$

### Decision Making

However, even given the probability distribution, it is an open question how to use this distribution in an interactive scenario. In the proposed model, since the agent does not get additional information about the workspace state (or about the expectation distribution) in any state other than the perceptive state  $s_0^r$ , it makes sense to make decisions in terms of *action sequences* (in a different context also called *options* (Sutton, Precup, & Singh 1998)). The acquisition of these sequences is beyond the scope of this paper, but sufficed to say that in our scenario the agent needs only to consider action sequences that begin at  $s_0^r$  and terminate at the same state.

In the discussed factory domain we can identify four proto-sequences. The sequence transitions in  $\Sigma_R$  are given in parentheses.  $x, y \in H - \emptyset$

1. Pick up a tool and use it  
[ $\langle s_0^r, \emptyset \rangle \rightarrow \langle s_0^r, x \rangle$ ]
2. Return a tool and return to workspace  
[ $\langle s_0^r, x \rangle \rightarrow \langle s_0^r, \emptyset \rangle$ ]
3. Return a tool, bring a new tool, and use it  
[ $\langle s_0^r, x \rangle \rightarrow \langle s_0^r, y \rangle$ ]
4. Do nothing and wait  
[ $s_i^r \rightarrow s_i^r$ ]

The decision process operates as follows: at any time the robot is in  $s_0^r$ , it can evaluate the cost of each of the proto-sequences. Proto-sequence 1 needs to be grounded for each tools and proto-sequence 3 needs to be grounded for each one of the currently not used tools. Given the probability distribution, the robot can compute the expected cost for choosing each of the strategies, and select the lowest-cost sequence. In calculating the expected cost for proto-sequences 1–3, we will assume that  $\forall i. [h < 2d_{0i}]$ . This is the reasonable assumption that the robot will be able to perceive the correct state transition in  $\Sigma_R$  when it is back at the workspace. The cost in our calculations includes performing the correct action afterwards.

To illustrate, we will calculate the expected duration cost of proto-sequence 1–3. In the following derivations, the current state of the workspace is  $s_j^w$ .

$$Cost_1(x) = p_{x|j}(2d_{0x} + 2) + \sum_{y \neq x} [p_{y|j}(3d_{0x} + d_{xy} + d_{0y} + 4)]$$

$$Cost_2(x) = \sum_{y=1}^{|H-\emptyset|} [p_{k|j}(2d_{0x} + 2d_{0y} + 3)]$$

$$Cost_3(x, y) = p_{y|j}(d_{0x} + d_{xy} + d_{y0} + 3) + \sum_{z \neq y} [p_{z|j}(d_{0x} + d_{xy} + 2d_{y0} + d_{yz} + d_{z0} + 5)]$$

Action sequence 4 is unique insofar as it is dependent not only on the state transitions in  $\Sigma_W$ , but also on the behavior of the human teammate. If the human's next workspace-changing action is at time  $t+h$ , the cost of waiting is the cost of performing the correct action with complete confidence, plus  $h$ . For the case that the robot is holding a tool  $z$ :

$$Cost_4(x) = p_{x|j} + \sum_{y \neq x} [p_{y|j}(d_{0z} + d_{zy} + d_{y0} + 3)] + h$$

For the case that the robot is not holding a tool:

$$Cost_4(x) = p_{x|j}(2d_{0x} + 2) + h$$

However, Since  $h$  is not usually accessible to the robotic agent, its estimate could conceivably be used as a confidence parameter, adjusting between an aggressively predictive agent behavior and a more cautious approach.

**Rephrasing the Naïve Agents** It is interesting to note that using the above notation, we can now rephrase the previously discussed agent behaviors. The naïve agent's policy can be viewed as selecting proto-sequence 2 whenever it is holding a tool in the workspace, and selecting proto-sequence 1 whenever a tool is warranted. The agent employing conservative tool return can be rephrased as selecting proto-sequence 4 whenever it is in the workspace and no tool is warranted, and selecting proto-sequence 1 or 3 if a workspace state warrants a tool. This rephrasing enables comparison between the different policies, the conclusions of which are currently under investigation.

### Situation-Based Expectation

In addition to making predictions on the next state based on prior state transitions, the robot could also use domain specific knowledge to guide the expectation of the state transition. In our case, the pertinent domain specific knowledge is coded in  $\Phi$ . Based on all the possible state transitions, the transition probability  $p_{i|j}^w$  can be estimated as follows:

$$p_{i|j}^w = \begin{cases} \frac{1}{|\{s_k^w : \phi(s_k^w) \neq \emptyset, s_i^w \subset s_k^w\}|} & \text{if } \phi(s_i^w) \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

where  $s_i^w \subset s_k^w$  means that  $s_k^w$  has one component that is also in  $s_i^w$ .

In other words, the robot can assume that a reasonable human teammate will transition the workspace state into one that is useful for the progress of the task. This is in line with *joint intention theory* (Cohen & Levesque 1991) and *shared cooperative activity* (Bratman 1992). In the above computation we assume a uniform distribution over 'helpful' states.

Ideally, the robotic agent could combine history-based and situation-based expectation. This could be done via averaging the probability values for a given state transition. However, a possibly more useful approach is to use the situation-based probability distribution as priors to the history-based (learned) expectation distributions. A simple way in which this could be integrated into the above described framework is through manipulation of the Laplace correction of the Bayesian estimator, reflecting the situation-based expectation.

### Conclusion and Future Work

In the above work we have proposed a novel way to investigate an important aspect of fluency in a collaborative task between a human and a robot, namely expectation management and the preemptive action upon these expectations. As part of that effort, we have proposed a formal framework in which problems of timing in collocated human-robot teamwork can be investigated. We have begun to analyze strategies for the robotic agent, and implemented our conclusions in a simulated environment.

We have implemented the described policies on simulated robotic agents and are currently in the process of analyzing both their ideal performance (using simulated human decision making), as well as their performance with untrained humans engaging in a collaboration with these agents. We will be able to present initial performance results from user interaction with these agents at the symposium meeting.

From preliminary observation we note a significant performance improvement in task efficiency between the naïve and the conservative tool-return policy. History-based expectation usually starts out at the performance level of conservative tool-return and improves gradually until it settles on a baseline performance marking an improvement over both alternative policies.

We are additionally investigating other measures of fluency and the effect of our policies on these measures. We argue the need to take into account not only the total time to complete a task, but also the time wasted by either the human or the robot waiting for the other agent, as well as the time wasted on erroneous action predictions.

In future work, in addition to modeling state transitions, the agent can improve its decision making process by modeling the duration cost of the human's part in the collaborative task. Earlier we discussed modeling the human's state transitions as an HMM. However, a more useful approach would be to model the human cost function. This can result in an autonomous estimate of  $h$  for each workspace state, grounding the timing assumption in the described cost calculation, as well as making the arbitrary estimate of  $h$  used in the calculation of  $Cost_4(x)$  based on the robot's experience on the task.

Moreover, we are in the process of investigating the impact different cost functions have on the effects of preemptive action given each of our decision making policies.

It would be interesting to see whether the learned process model can be applied to role-reversal, enabling the robot to take on the human's role after engaging in an increasingly fluent collaboration on a specific task.

Finally, we believe that the agent's expectations, on the higher cognitive level as well as on the perceptual level, should play a significant role not only in the decision making, but also in the perception process, in line with the approach laid out in (Aloimonos 1993).

## References

- [Aloimonos 1993] Aloimonos, Y., ed. 1993. *Active Perception*. Lawrence Erlbaum Associates, Inc.
- [Bratman 1992] Bratman, M. 1992. Shared cooperative activity. *The Philosophical Review* 101(2):327–341.
- [Cohen & Levesque 1991] Cohen, P. R., and Levesque, H. J. 1991. Teamwork. *NOÛS* 35:487–512.
- [Sutton, Precup, & Singh 1998] Sutton, R.; Precup, D.; and Singh, S. 1998. Between mdps and semi-mdps: Learning, planning and representing knowledge at multiple temporal scales. *Journal of Artificial Intelligence Research* 1:1TH39.