Towards Meaningful Human-Robot Collaboration on Object Placement

Advik Iyer Guha
Advisor: Stefanie Tellex
Reader: Jeff Huang
Department of Computer Science, Brown University

***Abstract*—Human-robot collaboration is a complex interaction set involving speech, gesture, and feedback. We propose a *pomdp* framework for the task of collaborative object placement in a tabletop environment. The framework models discrete space, performs high-level reasoning for objective estimation, and provides continuous feedback to the human collaborator, to achieve faster and more accurate object placement. We also present a pilot study evaluation of the framework suggesting that all three forms of interaction mentioned are necessary for a consistent and accurate robot collaborator.**

I. Introduction

Countless object-centric, collaborative tasks between human and robot involve *object placement*. Once a robot is holding an object of interest, there is still the complex, but critical problem of collaborating to correctly place or deliver this object.[1] For example, a kitchen assistant robot might be asked to place a box of flour near the red mixing bowl; a workshop assistant robot might be required to deliver a specific tool to the mechanic's outstretched hand; a nurse assistant robot might be asked to apply a bandage on the wound of a patient; a house assistant robot might be enlisted to lay the dinner table. An essential first step to each of these tasks is understanding *where* they need to be done.

To perform this first step and collaborate on such a task, a robot must first be able to understand how humans reference space in the physical world. Humans reference space using expressions that combine speech, gesture, and body language, just as they do objects (Eldon, Whitney, and Tellex 2016). However, unlike objects, space is a *continuous* variable, and this both changes how humans construct referring expressions, and makes the inference more complex.[2] In addition to this, in any such collaborative task, the human provides their referring signals continuously over time, and these signals evolve as the human's objective changes. Therefore, the robot must not only be able to understand how humans reference space, but do so in real-time. Furthermore, when two humans collaborate on a task, they monitor each other to get feedback on the other person's current understanding of the goal, and alter their future actions based on this feedback, such that the end goal may be achieved (Clark and Krych 2004). Therefore, to emulate a human collaborator, a robot must also be able to provide this continuous feedback (termed *social feedback* by Wu et al. 2015). These three capabilities appear to be critical for a truly collaborative framework for object placement.

Recent robotics research in this field has primarily focused on (a) object reference rather than space/location reference, and (b) on the *how* of object placement, rather than the *where*. In the area of object reference, current approaches have explored the static learning of object referring expressions

[1] Canonically called the pick-and-place problem

[2] For example, the robot must have a notion of how humans discretize space in their minds.

(Matuszek et al. 2014), as well as the dynamic interpretation of object referring expressions (Eldon, Whitney, and Tellex 2016). In the area of enabling object placement, recent work includes an inverse motion planner for object trajectories (Holladay et al. 2013), and learning algorithms for calculating the optimal positions for placement in prior-known areas (Jiang et al. 2012a, Jiang et al. 2012b). While this work is useful and part of the puzzle, it assumes knowledge of *where* the human wants the object placed. There is limited current research that directly addresses the critical *first step* of determining the human's placement objective.

As a foundation for such work, we propose a computational framework for collaborative object placement in tabletop environments. The framework integrates the three capabilities for collaboration outlined above. It maintains a discretized spatial model of the placement environment to understand space/location referring expressions. It uses a `POMDP` that receives multimodal observations and chooses multimodal actions, to interact with the human and continuously estimates their placement objective. It displays continuously updating visualizations of the `POMDP` objective estimate, as forms of real-time feedback to the human collaborator.

To evaluate our proposed framework, we conducted a pilot study where participants used speech and gesture to ask our robot to place an object in a particular location on the tabletop. We measured speed and accuracy of these tasks for multiple configurations of our framework, to test the relative importance of language versus gesture, and one form of continuous feedback over another.

II. Related Work

A. Planning Frameworks

Considerable work has been done in developing generalized frameworks for solving planning problems in artificial intelligence.

(Kaelbling, Littman, and Cassandra 1998) advocate for the modelling of sequential planning problems in partially observable domains as Partially Observable Markov Decision Processes (`POMDPs`). Within this general framework, the agent maintains and periodically updates, through observations and actions, a *belief state* or probability distribution over possible states, in lieu of concrete knowledge of the current state. Solutions to `POMDPs` balance the priorities of information-gathering actions, with exploitative, reward-seeking actions. This property is particularly applicable to a collaborative setting such as the problem of object placement, since we want to develop a non-rule-based approach for the robot to decide how and when to gather information from the human about their objectives. Our work will use this general `POMDP` framework to model and solve specific task of collaborative object placement.

In (Gmytrasiewicz and Doshi 2005), a framework for sequential planning in multiple-agent settings is described, as an extension of the `POMDP`. This framework, the `Interactive POMDP`, replaces the simple *belief state* described above, with *hierarchical belief systems* that comprise beliefs of the physical world, of other agents, and of other agent's beliefs. Modelling the human as a second agent in the environment (as opposed to simply an observation generator), is an interesting prospect, and this is one approach to doing so. However, there are two problems with the `I-POMDP` that makes it an unsuitable candidate for our work. First, the introduction of hierarchical belief statements compounds state and belief spaces, considerably increasing the complexity of finding solutions in an already difficult state space. A succeeding paper (Doshi and Gmytrasiewicz 2009) describes alternate methods for solving `I-POMDPs`, but it only partially addresses the issues of state and policy space complexity. Second, the utility of modelling typical human collaborative communication with more than two layers of belief systems is unclear. Our work therefore searches for a better balance between computational tractability and an effective model of collaboration.

In (Wu et al. 2015), the general `POMDP` framework

is applied to the task of object pickup and handoff in a tabletop environment. Though the intended approach and environment are similar, their focus is on a conceptually and computationally different problem than that of object placement. The latter task is more complex, due to the exponential increase in state space complexity, as described already. Therefore, our work poses different challenges and requirements, and the opportunity to understand how to support complex collaborative tasks between humans and robots.

B. Human-Robot Communication

Work has also been done on the subject of human-robot communication towards better human-robot collaboration.

In (Fang, Doering, and Chai 2014), two collaborative models are designed for human-robot referential communication, motivated by human-human referential communication. These models show good success in comparison to non-collaborative models, but they only consider language within the realm of referential communication. In a succeeding work (Fang, Doering, and Chai 2015), the role of robot gesture and human eye gaze (defined as *embodiment*) in the collaborative process is explored. Though exploring *embodiment* in referential communication, this work did not explore more critical human referential gestures, which will be central to our work.

In (Matuszek et al. 2014), a corpus of unscripted language and gesture object-referring expressions is used to train models to identify objects. Though the models are successful, they are static and do not consider the relationship of referring expressions over time.

In (Eldon, Whitney, and Tellex 2016), object-referral expressions *are* treated as multimodal and dynamic. A `Bayesian Filter` approach is used to interpret natural language and human gestures together, in real time. However, this model is mor rule-based than collaborative, since robot actions are predefined to occur at certain observation thresholds. It cannot support higher reasoning about the state of the world, nor represent robot actions as affecting the world and subsequent human actions. Our intended `POMDP` framework allows us to do exactly that, thus moving towards more meaningful collaboration experiences.

III. Technical Approach

The goal of this work is to enable the robot to meaningfully collaborate with a human on the task of object placement.

A. POMDP Overview

A Markov Decision Process is a mathematical framework for planning and decision making in environments with full agent observability, but stochastic transition dynamics. At each timestep of such a process, the decision maker or *agent* is in a particular state s, and can choose from a range of available actions, to transition into a new state s', and receive some corresponding reward r. The name Markov, and the related term Markovian, is critical because it specifies that the next state of this stochastic process (s') is dependent only on the current state, s, and the action taken a.

A `POMDP` is a generalization of the `MDP` that introduces an additional constraint in the form of *partial observability* to the environment. The agent can no longer directly see what state it is in; instead, it receives observations, and has a model for how observations arise from different states and optionally, actions. The underlying dynamics are still Markovian.

The `POMDP` is formally defined as a seven-tuple $(S, A, T, R, \Omega, O, \gamma)$, where:

- S is the set of states
- A is the set of actions
- T $(S \times A \to S')$ is the function of transition probabilities between states
- R $(S \times A \to \mathbb{R})$ is the reward function

- Ω is the set of all observations
- O $(S \times A \rightarrow [0, 1])$ is the function describing the probability of an observation, given the current state and previous action
- $\gamma \in [0, 1]$, is the discount factor, dictating how we value future rewards

B. Abstracting the Environment

The underlying hidden variable of our placement framework - the spatial coordinates of the human's placement objective - is a continuous variable. This means that the associated state space of the domain would be infinite. Since POMDPs are computationally complex already, an essential first step was discretizing our state space through environment abstraction.

We abstract our environment by first redefining the tabletop as a two-dimensional grid $\mathcal{G}$, with dimensions $\mathcal{G}_x \times \mathcal{G}_y$. With the grid in place, we redefine our hidden variable as the coordinates of the grid cell containing our human's placement objective.

This environment abstraction is referenced several times throughout the paper. For convenience, the Table I defines the parameters associated with our environment abstraction.

TABLE I
ENVIRONMENT ABSTRACTION PARAMETERS

Parameter	Significance
$\mathcal{T}$	Tabletop in the environment
$\mathcal{G}$	Grid abstraction of the framework
$\mathcal{T}_x$	Size of the table along the x axis
$\mathcal{T}_y$	Size of the table along the y axis
$\mathcal{G}_x$	Size of the grid along the x axis
$\mathcal{G}_y$	Size of the grid along the y axis
$\mathcal{ADE}$	Accepted discretization error due to $\mathcal{G}$

Our environment abstraction is useful for the following reasons:

- It makes the framework computationally tractable, trading centimeter-level accuracy for a substantial computational gain. The exact trade-off (or level of accepted error due to the discretization), can be defined as:

$$\mathcal{ADE} = \sqrt{(\frac{\mathcal{T}_x}{\mathcal{G}_x})^2 + (\frac{\mathcal{T}_y}{\mathcal{G}_y})^2} \tag{1}$$

- It allows us to model a spectrum of placement tasks with a single framework. A coarser grid $\mathcal{G}$ significantly speeds up the per-step computation time of the framework, while a finer $\mathcal{G}$ offers much higher accuracy in placement.
- It provides a natural foundation to the language and gesture models of the framework, described in sections III-D and III-E below.

C. Placement Framework

The proposed POMDP framework is formally defined by the seven-tuple, $(S, A, T, R, \Omega, O, \gamma)$.

States S is the set of states in this domain, each of which is a tuple $(\mathcal{O}, \chi, \kappa, \tau, \lambda)$:

- $\mathcal{O}$ is the set of objects on the tabletop
- χ is the object being held by the robot
- κ, is the count of the time-steps since the last physical action
- τ is the boolean value representing whether the object χ has been placed
- λ is the table-grid location where the human collaborator wants χ to be placed (partially-observable)

$\mathcal{O}$ is useful in allowing the human to make referring speech expressions in conveying their objective. χ is useful for the robot to frame expressions or questions of its own, to prompt for further objective inference. κ is useful in managing how frequently the robot acts physically. τ is useful in detecting the terminal state of the framework.

Actions A is the set of actions that can be taken by the robot.

For each cell l of grid $\mathcal{G}$, we define the following actions:

- PLACE, which places the object χ in l
- HOVER, which hovers the held object over l, and asks the user a question

We also define a strictly information-gathering action:

- WAIT, which simply waits (and probabilistically asks) for further input from the user

See III-F1 for details on how the framework decides which actions are applicable to be performed in a particular state of the process.

Transitions T is the transition function governing the probability of moving to a new state, given a current state and a particular action. It is assumed that the set of objects on the table $\mathcal{O}$, and the object being held, χ, do not change until an object is actually placed. The WAIT action causes a transition to a new state with the value of κ incremented by one. The HOVER action causes a transition to a new state with the value of κ reset to zero. The PLACE action transitions to a terminal state (i.e., a state with τ true) since the object χ has been placed.

Most of the time, a human collaborator has a specific placement location for the object χ in mind throughout the collaboration task. This can be modelled in the transition dynamics by deterministically keeping the λ value of our state constant through each transition. On occasion however, human collaborators may change their mind about where they want the object χ placed, mid task. To keep our decision-making framework robust in these situations, we model a slow decay in our transition dynamics.

Specifically, we assume that the human collaborator has a high probability k of referring to the same goal location λ during the task, and therefore a probability $(1-k)$ of changing their mind. We further assume that the probability of switching to any other desired location is equally probable. These assumptions lead to the following transition model, for λ:

$$p(\lambda_{t+1}|\lambda_t, a_t) = \begin{cases} k & \text{if } \lambda_t == \lambda_{t+1} \\ \frac{1-k}{n-1} & \text{otherwise} \end{cases} \quad (2)$$

where $k > 0.99$, a_t is WAIT or HOVER, and n is the number of states in our belief, i.e, the dimensions of our grid abstraction, G.

Reward Function R is the reward function, governing the positive or negative reinforcement received by the robot based on the current state and action taken. The ultimate objective of the framework is for the robot to place the object in the human collaborator's desired location; we model this by return a reward of $+5$ when the robot performs a correct placement. The worst outcome for the framework would be an incorrect placement; we model this by returning a heavy negative reward of -20 for bad placements.

During the interaction, we have several secondary objectives that we model into our reward function. We want to generate confidence in the human collaborator that the robot is receptive to changing language and gesture; we model this by assigning a negative reward (-3) to the most passive action, WAIT. We also want to convey straightforwardly what we believe to be the goal location; we model this by assigning a relatively higher reward to a HOVER over what we believe to be the human's goal location, than a HOVER over some other location.

The exact rewards assigned to *good* and *bad* HOVERs depends on the planning horizon being used to solve the POMDP. For a finite planning horizon higher than 1, rewards of -2 and -1 for *bad* and *good* HOVER actions respectively allow the framework to converge quickly. For a finite planning horizon of 1, values of -2 and 4 have shown in experiment to encourage both sufficient interaction and quick convergence.

A summary of reward function (for a planning horizon of 1) is available in table II.

TABLE II
REWARD FUNCTION

Action/Outcome	Reward	Reasoning (should...)
Incorrect Place	-20	Avoid premature placement
Wait/Do Nothing	-3	Interact with collaborator
Hover Over Incorrect Location	-2	Explore collaborator's goal
Hover Over Human's Objective	4	Generate confidence in collaborator
Correct Place	5	Achieve collaborator's goal

Observations Ω is the set of observations, each of which is a tuple $o = (l, g)$, representing the human collaborator's language, and the human collaborator's gestures. For experiments run on our Baxter robot, language is received through the connected microphone and the Google Speech Recognition platform, and gestures are received through the robot's retrofitted Microsoft Kinect and OpenNI Tracking Software.

The observation l is the natural, unaltered transcription received from *GSR*. The gesture observation g is a set of four vectors representing the world (x, y, z) coordinates of the human collaborator's left and right shoulders and wrists. We use this information to calculate the targets of their pointing gestures, as explained in section III-E below.

Observation Function O is the observation function, governing the probability of witnessing a particular observation, based on the action taken and the resulting state in which said observation occurred. That is,

$$O = p(o_{t+1}|s_{t+1}, a_t) \tag{3}$$

which can be expanded to

$$O = p((l_{t+1}, g_{t+1})|s_{t+1}, a_t) \tag{4}$$

We assume[3] the conditional independence of the language and gesture components of our observations. This makes development and configuration testing much easier, and simplifies the observation function to be:

$$O = p(l_{t+1}|s_{t+1}, a_t).p(g_{t+1}|s_{t+1}, a_t) \tag{5}$$

[3]Similar to Wu et al. 2015.

Discount γ is the discount factor used when calculating future discounted rewards. The framework sets $\gamma = 0.9$, to value future rewards higher, to encourage longer-term thinking in the framework.

Solution & Policy Generation

For our work, we chose to solve our `POMDP` using the technique of Belief Sparse Sampling, a finite-horizon algorithm that provides approximations of the optimal policy of the `POMDP`. The `POMDP` is solved by first converting it into a `Belief MDP`, so called because we consider every belief to be a Markovian state in this process. The resulting `BMDP` is no longer partially observable, since the agent always knows its belief. This related `BMDP` is then solved using Sparse Sampling. The framework described in this paper has been implemented using the Brown-UMBC Reinforcement Learning and Planning Library (BURLAP, MacGlashan 2015).

D. Language Model

Language is a key component of any human-human collaborative effort, and therefore for this human-robot collaborative framework. Two types of speech were focused on:

- Simple Affirmative/Negative Expressions
- Relative Location Referring Expressions

Affirmative/Negative Expressions are the set of expressions used to express positive or negative responses. These expressions support question-answer dialogue between robot and human.

Relative Location Referring Expressions are speech expressions that refer to a location in *relation* to other known locations in the environment. These expressions support more complex, instructive dialogue between human and robot, and are very important in command/collaborative tasks such as object placement.

An assumption made here is that the two types of speech are conditionally independent of each other.

This simplifies the observation function considerably, and also support configuration testing, which will be elaborated on below.

Affirmative/Negative Expressions These are expected to occur after the HOVER action, since this action is accompanied by a question that explicitly prompts a positive or negative response from the human.

Affirmative expressions are handled as follows:

$$p(l_{t+1} \textit{ is aff. } | s_{t+1}, a_t = hover) = \begin{cases} 0.99 & \text{if } a_t.l == s_{t+1}.\lambda \\ \frac{0.01}{n-1} & \text{otherwise} \end{cases} \quad (6)$$

Negative expressions are handled as follows:

$$p(l_{t+1} \textit{ is neg. } | s_{t+1}, a_t = hover) = \begin{cases} 0.01 & \text{if } a_t.l == s_{t+1}.\lambda \\ \frac{0.99}{n-1} & \text{otherwise} \end{cases} \quad (7)$$

where n is the number of states, or grid cells in the grid abstraction $\mathcal{G}$.

Referring Expressions Referring expressions are expected to occur either after the HOVER action or after the WAIT action, since both actions are accompanied by questions that prompt further input from the human collaborator.

A simple location referring expression (aimed at location l, say) is defined as:

$$\phi^l = (d, r)$$

where d is the direction specified and r is the object/point used as reference; the goal location l is in direction d from r. It follows that complex location referring expressions can be defined as the intersection of multiple, conditionally independent simple expressions, as:

$$\Phi^l = (\phi_1^l, \phi_2^l, \phi_3^l, ...)$$

These definitions together form an extensible model on which the observation function of the framework can be specified. Under this model, we need only specify the observation function for a set of simple referring expressions. We can incorporate increasingly complex (or overloaded) referring expressions without requiring redesigns of the observation function.

In short, this Φ specification becomes a layer of abstraction between all language parsing modules and our framework's observation function, and offers three advantages:

- the observation function can be defined
- the observation function is unaffected by the varying speech patterns of human collaborators
- the language parsing module can be substituted or changed without affecting the observation function

Assuming that the language parsing module being used is configured to output these referring expressions, we can rewrite the language model as:

$$p(l_{t+1}|s_{t+1}, a_t) = p(\Phi_{t+1}|s_{t+1}, a_t) = \prod_i p(\phi_i|s_{t+1}, a_t) \quad (8)$$

where $a_t =$ WAIT *or* HOVER

Where the conditional probability for a particular referring expression, ϕ, is based on whether the state's hidden variable location (λ) is actually in the direction $\phi.d$ from the reference object $\phi.r$. Formally:

$$p(\phi|s_{t+1}, a_t) = \begin{cases} 0.99 & \text{if } (s_{t+1}.\lambda).from(\phi.r) == \phi.d \\ \frac{0.01}{n-1} & \text{otherwise} \end{cases} \quad (9)$$

where $a_t =$ WAIT *or* HOVER, and n is the dimensions of G.

As a start, we support for movement along the positive and negative$x$ and y axes, and in the negative z. This gives us considerable flexibility to

define directions like *left, right, forward, backward, in front, behind, on, downward*.

For example, if the human collaborator produces the referring expression, "place the object to the left of the spoon, but in front of the fork", the language parsing module might produce the following complex referring expression:

$$\Phi = ((\texttt{NEG\_Y}, spoon), (\texttt{POS\_X}, fork))$$

The observation function can readily interpret the probability of the getting such a referring expression from every state in S.

Attentive Model A special case of the referring expressions model is the direct location referring expression, i.e., location referring expressions made with respect to the robot's current end-effector position itself. These fit cleanly within the referring expression model, as:

$$\phi^{direct} = (\texttt{d}, robot_{EE})$$

This creates an attentive model for the robot, allowing it to use the current position of its end-effector (and therefore the object in its hand, χ), to interpret language observations from the human collaborator, and respond to direct referring expressions such as "to the left", or "just in front", from human collaborators.

E. Gesture Model

The gesture model of the framework revolves around pointing gestures. These are defined as the vector v collinear with our human collaborator's raised forearm, directed towards the tabletop, with the point of intersection p of this vector with the tabletop considered the intended target.[4]

For the observation function, the human collaborator's pointing target p is considered to be sampled from a bivariate Gaussian (normal) distribution centered on the human's true goal placement location.

[4]Similar to Eldon, Whitney, and Tellex 2016

In terms of the variables defined above and from the `POMDP` framework described earlier, this is

$$p(g_{t+1}|s_{t+1}, a_t) \propto \mathcal{N}_{2,(\mu,\Sigma)}[(p.x, p.y)] \tag{10}$$

where the mean of the distribution, μ is:

$$\mu = (\lambda.x, \lambda.y) \tag{11}$$

and the bivariate distribution has a covariance matrix of the form:

$$\Sigma = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \tag{12}$$

where σ_1 and σ_2 are a function of the dimensions of G. These variables represent the range of accuracy we expect our human collaborators to have while pointing at their target location. With our tracking software and experimental setup, the following values produced promising and reliable results:

$$\sigma_x = \sigma_y = 0.12 \tag{13}$$

The parameters of the distribution selected should also be used to select the dimensions of our grid abstraction, $\mathcal{G}_x$ and $\mathcal{G}_y$. In particular,

$$\frac{\mathcal{T}_x}{\mathcal{G}_x} \in [\sigma_x, 2\sigma_x] \tag{14}$$

and similarly for the y dimension. This ensures that the each grid cell roughly fits the shape of the bivariate normal distribution, balancing accuracy with fast convergence of the `POMDP` framework.

F. Framework Considerations

1) Preconditions on BMDP

As described earlier, the `POMDP` is solved by converting it into a `BMDP`. To significantly improve the quality and solvability of the framework, certain preconditions are placed on the `Belief Actions` of the `BMDP`. These preconditions provide a significant speedup to the per-step solving

time of the framework, allowing us to increase the finite horizon of the solver, as well reduce our $\mathcal{ADE}$.

We specify that the `HOVER` and `PLACE` `Belief Actions` are only *applicable* (i.e., can be performed) on belief states where the target of these actions matches the highest probability state of the belief state. This precondition can easily be relaxed to the top x percentile of states within the belief state.

The net result of the preconditions is that the framework more often chooses to `HOVER` and `PLACE` in the locations that it has most beliefs to be the human collaborator's goal, rather than spending time hovering over other locations, for example. It chooses confirmation over elimination while performing its inference, and this has shown to work very well in experiment.

2) Continuous Social Feedback

Heat Map A *heatmap* belief state visualizer is the first form of continuous feedback provided by the framework to its human collaborator. It visualizes the continuously updating belief state of the underlying `BMDP` being used to solve the `POMDP`, as the framework is being run.

In particular, the visualizer displays a scaled version of the actual table in the environment; it overlays the belief state as a grid of cells on the table, each representing a potential placement location (identical to our grid abstraction $\mathcal{G}$). The color fill or *heat* of each grid cell is defined on a color spectrum ranging from green (cold) to red (hot); it is a function of the current belief (probability value) that that cell is the human's goal placement location (λ).

The visualizer communicates to the collaborator what the robot currently believes is their goal, based on the past language and gestural communications of the human. If the belief matches the human collaborator's desire, it increases the human's confidence in the robot. If the belief does not match the human collaborator's desire, then they have the opportunity to alter their speech or gesture as they think is most appropriate.[5]

The heatmap updates at roughly $5-10Hz$, depending on $\mathcal{G}_x$ and $\mathcal{G}_y$, and other parameters of our implementation. Improving this frequency will be a focus in our future work.

Figure 1 shows three frames of the heatmap belief state visualizer, at different stages of a placement task.

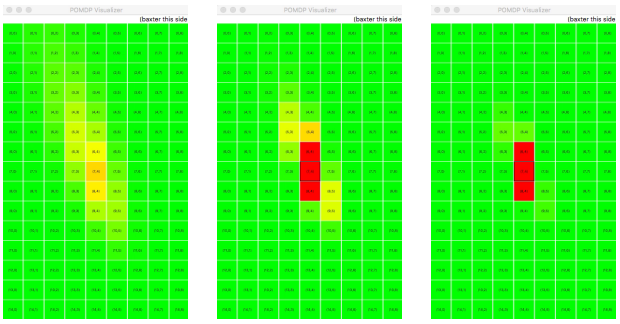


Fig. 1. Heatmap Progression

Robotic Emotions Robotic emotions are the second form of continuous feedback provided by the framework to its human collaborator. They are also continuously updated based on the belief state of the underlying `BMDP`, though using a different approach.

Our robotic framework is capable of expressing several emotions, using custom-designed animated faces on its head display. Each emotion has a library of roughly 120 frames, representing a wide range of intensity of that emotion. For our framework, we chose to focus on the emotion of *confusion*.

Each time the belief state (b) of the underlying `BMDP` is updated, the entropy, H of the associated probability distribution is also calculated.

$$H(b) = -\sum_{i}^{n} P(b(i)).\log P(b(i)) \qquad (15)$$

where n is the total number of states in the belief state, $\mathcal{G}_x * \mathcal{G}_y$.

[5]Based on the findings from Clark and Krych 2004

The more distributed the probability distribution is over the belief states, the higher the entropy. This value H is divided by the maximum entropy of the belief state, H_{max} (which is the entropy of the initial belief state uniform over all possible locations.).

$$H_{max} = -\sum_{i}^{n} P_{init}(b(i)).\log P_{init}(b(i)) = -\log \frac{1}{n} \quad (16)$$

where n is the total number of states in the belief state, $\mathcal{G}_x * \mathcal{G}_y$.

This result is a measure of how unsure the framework is about its collaborator's goal. The result $\frac{H}{H_{max}}$ is used to express the *degree* of confusion on the robot's head display, with higher degrees displaying more confusion. Figure 2 elucidates degrees $0, 60,$ and 119 of confusion.

Fig. 2. Robotic Emotion

Comments on Feedback Both forms of continuous feedback are rooted in the framework's belief state, since that is the *current understanding* of the goal (Clark and Krych 2004). The heat map is an unprocessed, explicit form of feedback; mathematically, it offers more information about the state of the robot's belief. By contrast, the robotic emotions the framework chooses to display are a more processed, implicit form of feedback, summarizing the robot's belief of the world, but open to subjective interpretation.

3) Safer Actions

An optional feature of the framework is to use safer physical `HOVER` and `PLACE` actions. These actions use the clearance map of the inverse kinematic solver of the robot, to make sure that all hover and place actions occur with target locations that are known to be reliably accessible by the end-effector of the robot. The inferred target hover and placement locations are mapped to the closest valid locations with the clearance maps, before these actions are carried out.

The motivation behind this optional feature is to avoid unexpected movements from the `IK` solver (a separate piece of software also undergoing improvement), enabling the framework to perform more safely when used with fragile objects.

IV. Evaluation

The framework was evaluated in simulation, as well as in a real-world pilot study. The objective of our evaluation was to gather preliminary results of the performance of different configurations of our framework. These results would allow us to conclude the relative importance of language versus gesture, and different forms of continuous social feedback, to the process of human-robot collaborative object placement.

A. Framework Configurations

The set of configurations of the framework used for testing are available in Table III.

TABLE III
Framework Configurations

Code	Configuration
`G`	Gesture
`G-H`	`G` + Heatmap
`GY`	`G` + Affirmative/Negative Speech
`GY-H`	`GY` + Heatmap
`GYR`	`GY` + Referring Speech
`GYR-H`	`GYR` + Heatmap
`GYR-HF`	`GYR-H` + Facial Expressions

The remaining configurations of our framework (particularly those supporting `F` without `H`) were excluded due to time and other constraints of the study, but will be included in future user studies.

B. Metrics

The following per-trial, quantitative metrics were selected:

- *placement distance error* - the distance in centimeters from the target to the actual point of placement
- *time until completion* - the time in seconds from the start to the end of the trial
- *steps until completion* - the number of steps taken by the framework from start to end of the trial
- *placement success* - whether the *placement distance error* was within the $\mathcal{ADE}$ of the framework

C. Placement Tasks

Two types of placement tasks were defined for the evaluation:

- A *simple placement task*, in which the target location is communicated to the robot from the beginning of the task, with coordinated speech and gesture as chosen. Success is defined as placing the object in the indicated target location.
- A *complex placement task*, in which an initial target location is communicated to the robot at the beginning, followed by a change of intent part-way through the task and the communication of a new, final target location, both with coordinated speech and gesture as chosen. Success in this task is defined as accounting for the change of intent, and placing the held object in the final target location.

D. Simulation

1) Procedure

Trial Trials were run on a *2.7 GHz Intel Core i7* workstation. Pointing gestures were simulated from a bivariate normal distribution of similar design to that used in the observation function. Speech (affirmative/negative expressions and referring expressions) were simulated using text constructed using the known target location. Both language and gesture were generated probabilistically, to match situations where the user did not produce either one or both observations. The *simple* and *complex* tasks defined above, were implemented as described using simulated environments in `BURLAP`. Since there was no human collaborator in simulation, only the configurations without continuous feedback were evaluated.

Experiment Each experiment involved 50 simple and 50 complex trials. Each experiment was run for the configurations `G`, `GY`, and `GYR`.

Measurements Measurements for the quantitative metrics were made automatically within the simulation setup, using a system of organized logs.

2) Results

The results of the simulation tests are reported using 95% confidence intervals in tables IV and V.

TABLE IV
SIMULATION RESULTS (SIMPLE TASK)

Code	Success (%)	Error (cm)	Time (s)	Steps (#)
G	80 ± 19.20	7.51 ± 4.19	39.60 ± 7.02	32.50 ± 5.14
GY	100 ± 0	3.13 ± 0	7.75 ± 1.02	8.60 ± 0.65
GYR	100 ± 0	3.13 ± 0	6.05 ± 0.72	7.35 ± 0.49

TABLE V
SIMULATION RESULTS (COMPLEX TASK)

Code	Success (%)	Error (cm)	Time (s)	Steps (#)
G	61.82 ± 13.25	17.77 ± 5.78	105.47 ± 17.39	72.16 ± 5.67
GY	82.00 ± 11.03	17.60 ± 9.39	24.86 ± 2.71	23.96 ± 2.18
GYR	46.00 ± 14.31	53.90 ± 13.45	14.90 ± 2.15	16.04 ± 1.92

Discussion The simulation results show that the `GY` and `GYR` configurations perform considerably better, both in terms of accuracy as well as consistency, than `G` on the simple placement task. The results of the complex placement task are not as easily interpretable or reliable, due to the wide confidence intervals of multiple metrics. It is possible that

the environment for the complex task did not adequately simulate hesitant speech before the change of intent, leading the heavily speech dependent configuration, GYR to be adversely affected. The possibility of this is also supported by the short *time until completion* numbers of the latter two frameworks - they might have converged before the change of intent even occurred, due to affirmative speech. Improving the quality of this simulation environment will be focused on in the future, so that more statistically useful results can be gathered for this task. Nevertheless, it useful to note that the framework is capable of handling changes of intent in many trials.

E. Real World

The real-world pilot study was conducted on two external subjects selected through convenience sampling. For the purposes of consolidating our data and limiting the complexity of the study for the external participants, only the simple placement task was used. The study was also conducted on one internal subject (the author of this paper), for the purpose of gathering more data under time constraints. (It also allowed us to make a first approximation on the learning curve involved in using this collaborative framework).

The pilot study produced promising preliminary results that we view as a good proof of concept of our work. A major objective of our future work will be to conduct a larger and more comprehensive user study, so that we may more rigorously evaluate the strengths and weaknesses of our framework.

1) Procedure

Framework The values of all parameters of the framework (whose motivations were discussed above) were constant throughout the study. For our study environment, we defined $\mathcal{G}$ as a 15×9 grid over a table of dimensions roughly $1.88m \times 1.135m$, giving us the following scales:

$$\frac{\mathcal{T}_x}{\mathcal{G}_x} = 0.1253m \quad \frac{\mathcal{T}_y}{\mathcal{G}_y} = 0.1261m$$

Given the dimensions of our table, this gives us an *accepted discretization error* ($\mathcal{ADE}$) of about 8.89 cm.

Trial Each trial was performed with a Rethink Robotics Baxter Robot set up in a tabletop environment. A small rectangular test object about $6cm$ wide was placed between the grippers of one of Baxter's hands. A square piece of tape about $2cm$ wide was randomly placed on the table to indicate the target placement location (for the center of the test object) to the subject, completely unknown to Baxter.

Subject Each subject was instructed to stand in front of the table and collaborate with Baxter on placing the held object in the specified target location. They were informed that Baxter might ask them questions about their objective, which they could answer with yes/no responses. They were given a microphone to wear, so their speech could be recorded and transcribed by our speech recognition platform. They were informed that the Microsoft Kinect on top of the robot was tracking the movements of both their arms. Subjects were asked to communicate in the most natural way possible, given the capabilities of the particular configuration being tested. Informing the subjects about the capabilities of each configuration was an active choice made for the study, to exclude differences linked to how subjects go about learning about the abilities and limitations of their collaborators.[6]

Experiment Each experiment involved a subject running at least four trials of the simple placement task for each configuration of the framework in the table III, giving a total of at least 20 trials per subject.

Measurements During the trial, the time and steps until completion were measured within the framework implementation. The remaining metrics were measured at the end of each trial.

[6]Although this is also an interesting question, and something we will likely explore in the future

2) Results

The pilot study produced preliminary results that are promising. The results and implications are summarized in the tables and figures below.[7]

Statistics Across Trials

Table VI shows the overall statistics across all 64 trials of the framework. On average, the framework converges in under a minute of continuous interaction. It has a median *placement distance error* that is less than the framework's stated $\mathcal{ADE}$. The *placement success* rate across all trials is over 57.8%.

TABLE VI
PILOT STUDY OVERALL STATISTICS

Metric	Mean	Median
Placement Distance Error (cm)	9.486	7.75
Time until Completion (s)	45.08	38.5
Steps until Completion (#)	21.17	18

Figure 3 elucidates the distribution of *placement distance error* across trials. It is slightly right-skewed (with a mean of 9.48 and a median of 7.75, as previously stated).

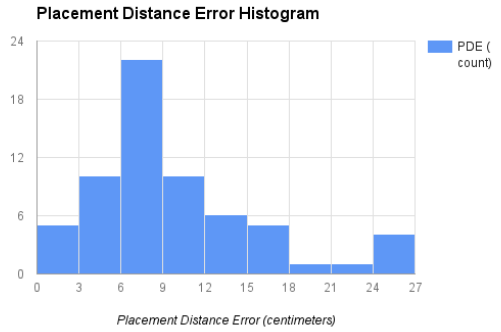


Fig. 3. PDE Histogram

Statistics Across Configurations

Table VII gives the mean-value metrics for the 7 configurations used in the pilot study. We note a rise in the time taken between configuration `G` and almost all other configurations. Since `G` is one of two configurations not using speech, this (undesirable) increase in time could be attributed to delays our speech recognition platform. We also note here that the average *placement distance error* drops by almost 70% from trials with configuration `G` to those with configuration `GYR-HF`.

TABLE VII
PILOT STUDY CONFIGURATION MEANS

Configuration	Error (cm)	Time (s)	Steps (#)
`G`	14.19	38.25	23.375
`G-H`	9.27	47.63	28
`GY`	11.21	49.83	18.75
`GY-H`	11.31	40.78	17.33
`GYR`	7.51	53.38	23.75
`GYR-H`	5.97	35.5	14.25
`GYR-HF`	5.94	47.38	21.88

Figure 4 shows the *placement distance error* across all configurations. We note the spikes in error for configurations `GY` and `GY-H`; these could (again) be attributed to delays in the speech recognition platform, where an affirmative response received by the framework in the wrong step (or vice-versa) could lead to a completely incorrect placement. We also note the lower and more consistent *pde* for the last two configurations of the framework, `GYR-H` and `GYR-HF`.

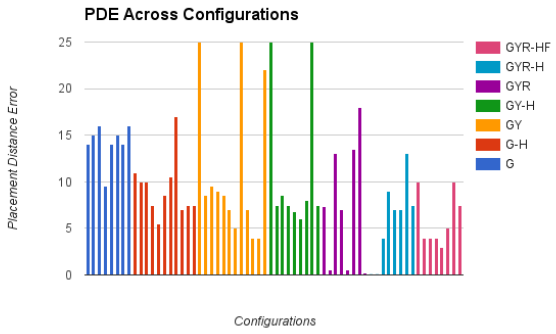


Fig. 4. PDE Across Configurations

Figure 5 shows the scatter of *placement distance error* versus *steps until completion* across all configurations. Speed and accuracy in the collaborative

[7]See end of paper for full-size figures

framework is thus represented by the lower left corner of the chart.

We note that the fastest trial in terms of steps that qualifies as a *placement success* is from configuration GYR−H. We also note that the entire *blue* G-cluster hovers near the middle of the chart, while the *magenta* GYR−HF-cluster occupies strictly the length of the bottom two-fifths of the chart.

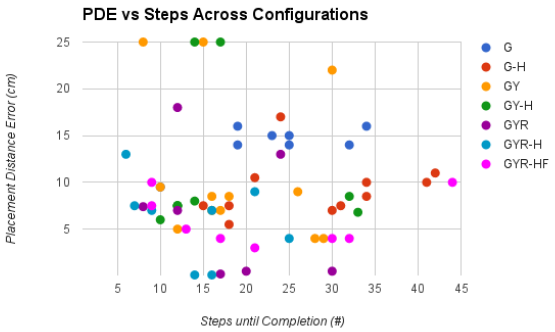


Fig. 5. PDE Across Configurations

Final Discussion

TABLE VIII
PILOT STUDY CONFIGURATION SUCCESS

Configuration	Success (# trials)	Success (%)
G	0/8	0%
G−H	6/11	54.55%
GY	7/12	58.33%
GY−H	7/9	77.78%
GYR	5/8	62.5%
GYR−H	6/8	75%
GYR−HF	6/8	75%

Table VIII summarizes the performance of the 7 configurations of our framework. We see that the *placement success* rate is approximately increasing from the basic configuration G to the full (action + feedback) configuration GYR−HF. From the previous tables and figures, we have also seen that the latter two configurations perform more accurately than the others. These results serve as a proof of concept of the required capabilities of a collaborative framework in object placement, as we discussed in our introduction.

The GYR−H and GYR−HF configurations both:

- model how human collaborators reference space, with language (two types) and gesture (one type)
- offer low and/or high-level continuous feedback to their collaborator
- interact in real-time (though this is true of all configurations)

The results of our pilot study have given us tangible insights into our collaborative framework. They form a good foundation on which to design our full user study, and gain more statistically significant evaluations of our work.

V. CONCLUSION

To meaningfully collaborate with humans, robots must understand how humans collaborate with each other. Collaboration is founded on communication, which starts with language and gesture, but extends far beyond into implicit and explicit feedback, and shared knowledge and experience. This research proposes an approach to the task of collaborative object placement on tabletops: a framework that models discrete space in the environment, performs high-level reasoning for estimation of the human's objective using a POMDP, utilizes speech and gesture for direct interaction, and provides continuous feedback to its human collaborator. Our results, though preliminary, suggest that each of these components contribute to a more consistently accurate and meaningful collaboration between human and robot.

VI. FUTURE WORK

Practical Improvements We plan to practically improve our framework by optimizing the current implementation in BURLAP as well as by replacing the speech recognition platform being used. Both of these improvements would increase the response time and reliability of our framework.

Evaluation As mentioned earlier in this paper, a comprehensive user study is an important objective in our future work.

Theoretical Approach We would like to explore several other ways that we can gather observations from our human collaborators. During the pilot study, both external subjects attempted gestures to mimic controlling the robot (*pushing* or *pulling* it towards the goal), as well as other forms of speech (”hot” and ”cold”, as notions of distance from the goal). Incorporating these within our underlying markovian process will be an interesting challenge, and could substantially improve the accessibility of the framework. Additionally, sentiment analysis on our speech observations might make the framework more sensitive to the mental state its human collaborator.

VII. Acknowledgments

This undergraduate thesis is the culmination of a year-long project in the H2R Laboratory at the Brown University Department of Computer Science.

I owe tremendous thanks to my advisor Professor Stefanie Tellex for her guidance, wisdom, and vision; to my second reader Professor Jeff Huang, for his immediate support and his ideas on effective user studies; to Prof. Thomas Doeppner, for his endless patience and understanding; to James MacGlashan, for his patient help with the `BURLAP` library and my framework design; to John Oberlin for his keen insight into robotic interfaces; to David Whitney for his handy advice on user study protocol and more; to Yuxin Han, for designing emotions for robots; to Emily Wu for her continuous readiness to share her knowledge and experience in the world of robotic social feedback; to my family, for their unwavering support; and, of course, to many other wonderful people at this wonderful university.

References

[1] Miles Eldon, David Whitney, and Stefanie Tellex. “Intrepreting Multimodal Referring Expressions in Real Time”. In: *International Conference on Robotics and Automation* (2016).

[2] Herbert H. Clark and Meredyth A. Krych. “Speaking while monitoring addressees for understanding”. In: *Journal of Memory and Language* 50 (2004), pp. 62–81.

[3] Emily Wu et al. “Robotic Social Feedback for Object Specification”. In: *AAAI: Artificial Intelligence for Human-Robot Interaction* (2015).

[4] Cynthia Matuszek et al. “Learning from Unscripted Deictic Gesture and Language for Human-Robot Interactions”. In: (2014).

[5] Anne Holladay et al. “Object Placement as Inverse Motion Planning”. In: *IEEE International Conference on Robotics and Automation Automation (ICRA) Karlsruhe, Germany* (2013).

[6] Yun Jiang et al. “Learning to Place New Objects”. In: *International Conference on Robotics and Automation (ICRA)* (2012).

[7] Yun Jiang et al. “Learning to Place New Objects in a Scene”. In: *International Journal of Robotics Research (IJRR)* (2012).

[8] Leslie P. Kaelbling, Michael L. Littman, and Anthony R. Cassandra. “Planning and acting in partially observable stochastic domains”. In: *Artificial Intelligence* 101.1-2 (1998).

[9] Piotr J. Gmytrasiewicz and Prashant Doshi. “A Framework for Sequential Planning in Multi-Agent Settings”. In: *Journal of Artificial Intelligence Research 24* (2005).

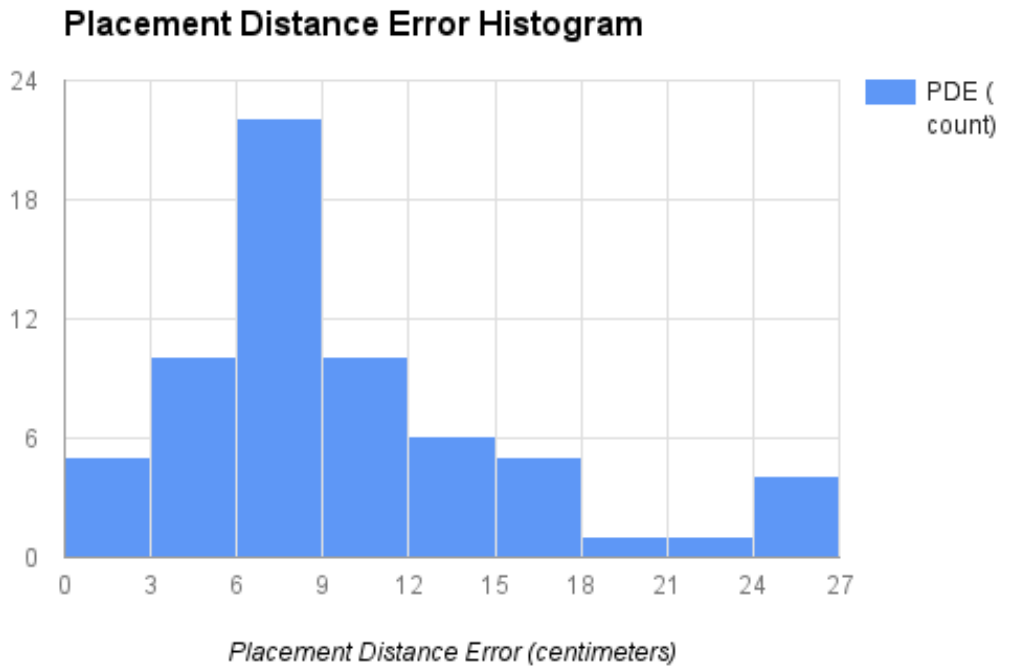
[10] Prashant Doshi and Piotr J. Gmytrasiewicz. “Monte Carlo Sampling Methods for Approximating Interactive POMDPs”. In: *Journal of Artificial Intelligence Research 34* (2009).

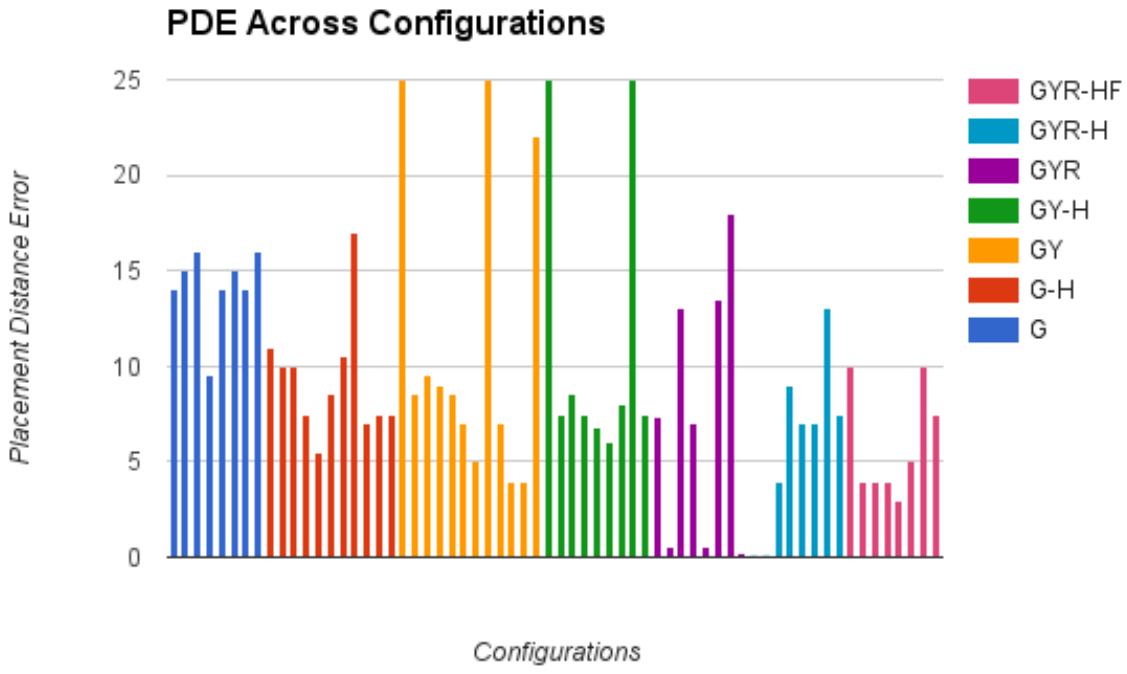
[11] Rui Fang, Malcolm Doering, and Joyce Y. Chai. “Collaborative Models for Referring Expression Generation in Situated Dialogue”.

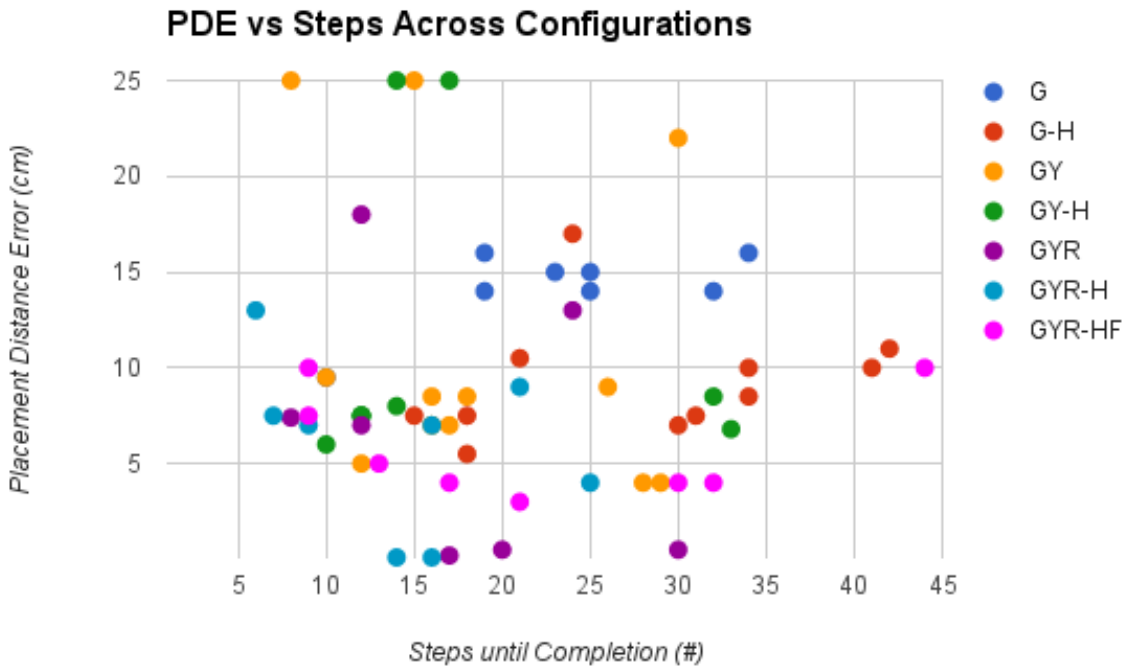
In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*. Aug. 2014.

[12] Rui Fang, Malcolm Doering, and Joyce Y. Chai. "Embodied Collaborative Referring Expression Generation in Situated Human-Robot Interaction". In: *Proceedings of the 10th ACM/IEEE Conference on Human-Robot Interaction (HRI)*. Mar. 2015.

[13] James MacGlashan. Brown UMBC Reinforcement Learning and Planning Library. burlap.cs.brown.edu. 2015.


Placement Distance Error Histogram
24
18
12
6
0
0
3
6
9
12
15
18
21
24
27
PDE (count)
Placement Distance Error (centimeters)


PDE Across Configurations
25
20
15
10
5
0
Placement Distance Error
Configurations
GYR-HF
GYR-H
GYR
GY-H
GY
G-H
G


PDE vs Steps Across Configurations
Placement Distance Error (cm)
25
20
15
10
5
5
10
15
20
25
30
35
40
45
Steps until Completion (#)
G
G-H
GY
GY-H
GYR
GYR-H
GYR-HF