Preference-Based Assistance Prediction for Human–Robot Collaboration Tasks

Elena Corina Grigore, Alessandro Roncone, Olivier Mangin, and Brian Scassellati[1]

***Abstract*— Human-Robot Collaboration (HRC) aims to develop robots that provide assistance to human workers while performing physical tasks. Such assistance comes in the form of supportive behaviors that are different from the actions part of the task, and that are meant to help a human worker more effectively accomplish the task. Learning how to provide useful behaviors that are tailored to a human peer represents a difficult challenge. This is due to the need of large amounts of training data in the form of real world observations that include information about such preferences. This data needs to encode not only the structure and progression of the task, but also the different workers' preferences with respect to when and what assistance the robot should provide. Our work separates the challenge of learning a model of the task (which requires a large amount of training data) from that of learning supportive behavior preferences for the interaction (which has obvious restrictions for the number of user-provided demonstrations to which we have access). We first learn a hidden Markov model (HMM) from a training set consisting of observed human workers performing the considered task in simulation. We then use this model to predict, while observing the human peer, what supportive behaviors a robot should offer throughout the task. Building upon the hidden state representation, our system is able to learn the supportive behaviors based on as few as five user-annotated demonstrations, learning a personalized supportive behavior model. We evaluate our system on a user study with 14 participants, and show results on par with human-level prediction for the task.**

I. INTRODUCTION

One of the main goals of robotics research is to develop robots that adapt to novel situations, without the need to be pre-programmed. Human-Robot Collaboration (HRC) is an area of robotics that aims to make such robots useful in the context of teamwork. Today, state-of-the-art robots in industry work in isolation from humans, performing precise, repetitive tasks. HRC pushes for a transition to adaptive, collaborative robots that are capable of offering assistance to a human worker throughout the execution of a task.

When developing adaptive robots for HRC, the goal is to provide *assistance* to the human, rather than aiming for the robot to autonomously execute the full task. This goal is motivated by two reasons. First, the type and level of knowledge, as well as the training required for the robot to complete the task on its own, is difficult to establish and collect. Second, despite significant advancements in areas of robotics such as manipulation [1], [2], robots are still far from having the fine manipulation capabilities required by tasks such as furniture assembly (e.g., using a screwdriver on a small screw). We thus desire robots that provide those types of behaviors well-suited for a robot, while allowing the human worker to perform actions better suited for a person. For example, a robot might provide supportive behaviors like stabilizing a component or bringing a heavy part required for assembly [3], while the human worker can execute actions such as screwing, which requires higher dexterity and a particular type of adaptation to the task.

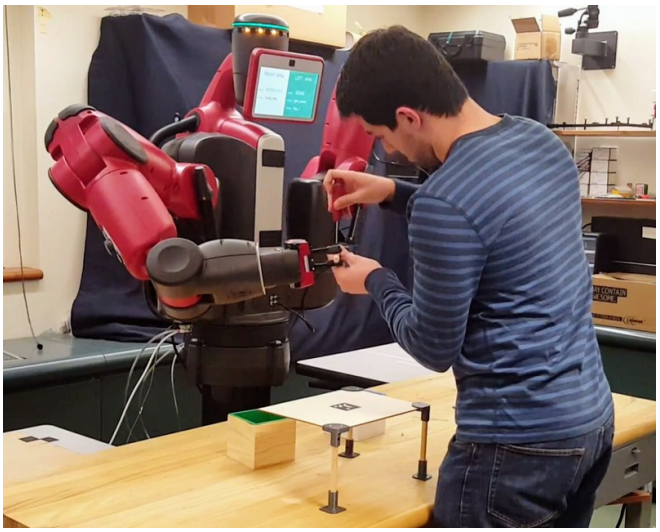

Fig. 1. The target domain is human–robot collaborative task execution.

There has been significant progress made in robotics, both for learning models of tasks, and for investigating how to integrate human preferences with respect to these tasks. The former includes work such as skill acquisition [4], obstacle avoidance [5], navigation [6], and so on. The latter investigates human anticipatory and preference-based models, such as human anticipatory control for robots [7], joint-action model learning in HRC [8], communication for task allocation [9], and others. This work allows us to build models of tasks and even take into account user preferences with respect to task structure. However, it is difficult to use such models to learn personalized models of what kind of *assistance* the robot should provide throughout the task.

Learning when and what kind of assistance to provide to different users poses non-trivial challenges. A human worker might prefer the robot to always stabilize a particular component like the seat of a chair while they are performing a screwing action. Another worker might instead wish for

This work has been supported by a subcontract with Rensselaer Polytechnic Institute (RPI) by the Office of Naval Research, under Science and Technology: Communication for enhancing Human–Robot Collaboration.

[1] The authors are with the Social Robotics Lab, Computer Science Department, Yale University, New Haven, CT 06511, USA name.surname@yale.edu.

the robot to utilize that time to bring a screwdriver for the next step of the assembly. These different options are not the typical task preferences referring the order in which task sub-states are executed (e.g., perform *assemble leg 1* before *assemble leg 2*); rather, they represent different supportive behavior preferences on the part of the users (e.g., provide supportive behavior *stabilize component* for the *assemble back* sub-state but not for the *assemble seat* sub-state). We can only learn such preferences when we collect data about the supportive behaviors themselves, and not by simply learning a model of the task via observing human workers perform the task without assistance. The proposed system is thus intended for scenarios that can feasibly provide us with relatively high amounts of trajectories of a worker completing the task on their own, but that make it difficult to collect many observations that include supportive behavior information. In this paper, we provide a solution for learning *personalized supportive behavior models* for different users while building upon a common model of the task across all users. This provides us with the flexibility to build upon existing techniques of modeling tasks in HRC, while only needing to train an extension of this task model in order to learn supportive behaviors tailored to different user needs.

We employ hidden Markov models (HMMs) to encode information about human action patterns during task execution, with no information about what supportive behaviors users would desire. By doing so, our method is able to implicitly yet flexibly represent the relevant structure of the task. We generate in simulation a pool of observations similar to what could be acquired by mounting sensors in a factory, while workers perform their usual routines. Such unsupervised trajectories would not include information about the assistance a robot would need to provide. We then assume access to a very small number of user-provided demonstrations (five) that annotate trajectories of the task with labels for the supportive actions the human would like the robot to provide. We employ these user labels to learn a *model of the supportive behaviors* with respect to the actions we have observed workers perform during training. This allows us to predict what supportive behaviors the robot should provide that are *tailored to an individual worker*.

We test the proposed system in a study with n=14 participants. We present results showing that we achieve predictions on par with human-level performance by using only five user-provided demonstrations. We also present the versatility of our system by asking participants to change their preferences for what supportive behaviors the robot should offer at different points throughout the task, and show that our model separation allows us to train on these new preferences quickly, with similar human-level results.

II. Related Work

In robotics, HMMs are widely utilized. Work in this area includes HMMs for acquiring behavioral models for robots [10], learning robot trajectories acquired from demonstration [11], and having a robot learn and reproduce gestures by imitation [12].

Work on model learning in robotics spans a wide range of topics. This includes learning high-level representations of navigation tasks for mobile robots from observations and use the learned model for the robot to engage humans for help during task execution [6]. Further work outlines the application of a neural network model to control an industrial robotic manipulator generation [13], and performing task-level learning that can refine the task command based on the system's error metrics, with an application to ball throwing [14]. Our work seamlessly builds upon these types of models and extends them to provide an important and novel capability for robots in HRC scenarios: building personalized models of supportive behaviors. With the current work, we build on the model we presented in [15], and present our results as part of a study conducted on a real robot.

A final and extremely important corpus of related work is composed of research in the HRC domain. This research makes significant contributions by integrating information into the models about user preferences and user-intention prediction. Relevant work in this area that is based on the use of HMMs includes, among others, gesture recognition for human-robot interaction [16], and understanding human intentions for autonomous mobile robots [17]. Reinforcement learning has been utilized for enabling collaborative learning between a robot and a human [18], and for leveraging the way in which people teach robots [19]. Further important work includes interactive learning in HRC [20], manipulation planning for HRC [21], anticipatory robot control for HRC [7], action-selection mechanisms for improving human-robot fluency [22], [23], learning models of joint actions for HRC [8], and improving robot performance as a collaborator in HRC [24], [9].

Such models do not provide us with the means to build supportive behaviors for different users. This is due to the fact that models that are trained on interaction data need a prohibitive amount of such data in order to learn anything meaningful, and models that are trained on task demonstrations only do not typically include interaction data. Our solution tackles these challenges by employing data hungry models only to represent the task. We then extend this task model to account for the supportive behaviors that can only be learned by collecting expensive user-provided labels. This allows us to train new supportive behavior models by re-utilizing a common task model and the need for a very low number of user-provided labels.

III. Problem Formulation

In this paper, we focus on developing robots able to offer support to a human worker throughout a task. Our goal is to enable such robots to predict, in a way that is tailored to individual users, when this support is needed. To this end, we target an HRC scenario where a robot works side-by-side with a human partner to achieve a shared goal, specifically the collaborative assembly of furniture. In this work, we employ a model set for HRC that has been developed in related work [25]. It is tailored to collaborative experiments, and it explicitly exposes a variety of experimental variables

for the evaluation of HRC algorithms (e.g., task complexity, respective roles of robot and human in performing the task, etc.). For the purposes of this work, we focus on a single assembly task, namely a chair (see Fig. 1).

We target the Baxter Research Robot (cf. Fig. 1), a robotic platform commonly used in HRC. These platforms are usually characterized by limited capabilities compared to those of humans; they can execute pick-and-place actions and holding of components to facilitate actions performed by the human. They cannot perform complex assembly tasks by themselves, but they are well-suited to support a human.

We consider that the robot can provide a set of supportive behaviors consisting of a total of 10 behaviors, including no action. Table I presents a sample trajectory generated from the task together with two different sets of supportive behavior preferences, listing the action performed by the human together with the supportive behaviors the robot should offer for that human action. For example, when the human executes *gather parts leg 3*, the robot should perform two supportive behaviors: *bring back bracket* and *bring dowel*. The robot should aim to execute the exact behaviors labeled for each step, without missing or adding anything. These supportive behaviors represent actions the robot can take throughout the task to help the human efficiently complete the task. In particular, we consider the types of actions the human takes to be different from those the robot can take. For example, an industrial arm can easily bring different labeled parts from a pool of assembly components onto the workspace, while the human arranges them and performs the actual assembly, tackling actions that need more dexterity. This represents a realistic HRC scenario, where each of the agents performs actions for which they are best suited.

In order for the assistance provided by the robot to be useful to an individual user, our aim is for the robot to learn how to do so in a personalized manner. One human worker might prefer the robot to *hold* both the seat and the back while they perform the *assemble* action for that part (depicted in Table I under the *assemble seat* and *assemble back* human actions). A different user might instead wish the robot not to provide a *hold* behavior for the seat or the back, preferring *hold* for *gather parts back top* and for the third time *gather parts leg* happens (depicted under the *assemble seat*, *assemble back*, *gather parts back top* and *gather parts leg 1* actions). In this work, we set forth to learn this spectrum of user preferences by using only a small amount of user-provided labels. This is a central concern in collaborative scenarios, as user demonstrations are expensive to obtain in large numbers. The results presented in this paper are obtained from only five user-labeled demonstrations.

Smooth collaboration requires that the robot provide supportive actions when the human needs them. This requires anticipation since actions and, specially, robot actions take time to happen. For instance, if the robot is to help a human perform a screwing action, we need to predict the action in advance such that the robot can bring a screwdriver to the human in time. Here, we focus on the objective of predicting the human's needs with an anticipation time of Δt.

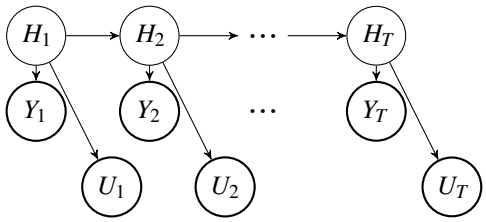

Fig. 2. The extended HMM model used in the paper. In addition to a regular model learned from observation of all the users ($(Y_t)_{1\leq t\leq T}$), we consider user preferences, represented by the variables U_t. We assume that the user preferences only depend on the hidden state and that they are only observed on a limited number of examples.

IV. Predictive Model

We propose a method of separating the problems of learning a task model from that of learning personalized models of supportive behaviors for the interaction. This technique allows us to leverage learning a single, robust task model that does not need interaction data to then tailor the supportive behaviors the robot should offer during the interaction to different users.

Our approach starts by first learning a limited set of activity patterns and transitions between them from observations of human behaviors. In this paper, we implement this approach with a hidden Markov model (HMM). During our second phase, we learn to predict (from the hidden states) what supportive behaviors are useful to the human worker. The choice to employ an HMM to model our task is based on the strong mix of the model's simplicity and power of predicting into the future given past observations. In particular, this means we can choose to model our task by using a higher or a lower number of hidden states. The former would provide us with a better representation of the task itself but would require more user preference labels in order to achieve high quality predictions for supportive behaviors, while the latter would have a more limited representational power while requiring a small amount of user-provided labels. This is a trade-off we analyze further in Section VI.

Our model of the personalized supportive behaviors consists of an extended HMM, presented in Fig. 2. This extended HMM is based on a basic HMM that models our task. This basic task HMM is trained by using only the task trajectories, without any information about supportive behavior labels (i.e., interaction data). The extended HMM that models our supportive behaviors is created by augmenting the task model with additional variables corresponding to user-preferred supportive behaviors, and is trained via adding a low number of task trajectories that are labeled with supportive behavior preferences. We denote by H_t the hidden state at time t, by Y_t the observations, and by U_t the preference of the user for some of the supportive behaviors.

A. Main Task HMM

To represent our task in a simple yet powerful way, we model our emission probabilities with Bernoulli distributions. We assume access to a training set consisting of observations, each in the form of a vector of features. In the following,

TABLE I

SUPPORTIVE BEHAVIOR LABELS FOR A SINGLE TRAJECTORY BASED ON TWO DISTINCT SUPPORTIVE BEHAVIOR PREFERENCE SETS

Supportive behavior preference set 1		Supportive behavior preference set 2	
Human action	Supportive behavior labels	Human action	Supportive behavior labels
gather parts leg 3	[bring back bracket, bring dowel]	gather parts leg 3	[bring back bracket, bring dowel, bring screwdriver]
gather parts leg 4	[bring back bracket, bring dowel]	gather parts leg 4	[bring back bracket, bring dowel]
gather parts leg 2	[bring front bracket, bring dowel, bring screwdriver]	gather parts leg 2	[bring front bracket, bring dowel]
gather parts leg 1	[bring front bracket, bring dowel]	gather parts leg 1	[bring front bracket, bring dowel, hold]
gather parts seat	[bring seat]	gather parts seat	[bring seat]
assemble seat	[hold]	assemble seat	Ø
gather parts back right	[bring top bracket, bring dowel]	gather parts back right	[bring top bracket, bring dowel]
gather parts back left	[bring top bracket, bring dowel]	gather parts back left	[bring top bracket, bring dowel]
gather parts back top	[bring back, bring long dowel]	gather parts back top	[bring back, bring long dowel, hold]
assemble back	[hold]	assemble back	Ø

we assume demonstrations are of length T and that the features are binary, although the proposed approach is not at all limited to such constraints. We denote as n the number of such features so that each observation $y_t \in \{0,1\}^n$. We employ features that are based on the presence versus absence of objects in the workspace, with a 1 denoting presence, for a total of 19 features. We train the model with the Baum-Welch algorithm, implemented on top of the Python *hmm_learn* library. Baum-Welch is the standard HMM algorithm for learning the model parameters (i.e., in our case, the Bernoulli probabilities and state transition probabilities). In Fig. 2, this corresponds to learning the emission probabilities $P(Y_t|H_t)$, as well as the transitions $P(H_{t+1}|H_t)$ and the initial distribution over hidden states.

B. Personalized Supportive Behavior Model

We represent our supportive behavior preferences by introducing variables U_t, which depend only on the hidden state at time t (i.e., they do not depend on the observed Y_t). The U_t are binary vectors of dimension $n_{\text{preferences}}$ in which each coordinate is $U_{t,j} = 1$ if the preference is that the robot provide the supportive behavior j and 0 otherwise. The separation in the model enables learning the preference probabilities from a different number of samples than the HMM. In particular these are user-provided labels that are more expensive to collect than pure observations of a worker's behavior. If the HMM correctly captures the relevant information about the task progression, only a few examples of user labels are needed to predict the preference, since we are relying on the quality of the decoded hidden state.

In this experiment we further approximate the behavior preferences as being independent knowing the hidden state. We model these preferences with Bernoulli distributions, like we do for the observations. We, however, assume that the HMM has already been learned when we proceed to train the preference model. Each labeled interaction i provided by the user consists in a trajectory of observations together with user preferences for the supportive behaviors, that is a list of feature vectors $(y_t^i)_{1\leq t\leq T}$ and a list of preference vectors $(u_t^i)_{1\leq t\leq T}$. We start by decoding the trajectories with a soft Viterbi algorithm (equivalent to an E-step in Baum-Welch), and then estimate the probabilities for the preferences similarly to an M-step, by using the probabilities over the decoded states. More precisely, if we have n trajectories:

$$P(U_t = u|H_t = k) = \frac{\sum_{i=1}^{n}\sum_{t=1}^{T} P(H_t^i = k|y_{1\ldots T}^i) \times \mathbb{1}_{u_t^i = u}}{\sum_{i=1}^{n}\sum_{t=1}^{T} P(H_t^i = k|y_{1\ldots T}^i)}. \quad (1)$$

C. Predicting supportive behaviors

We are interested in predicting the supportive behaviors that the robot can provide to the human worker with an anticipation time of Δt. In order to predict into the future, we employ the standard soft Viterbi algorithm to decode the hidden state until the current time step, knowing the observations until that step. In other words, we compute $P(H_t = k|y_{1\ldots t})$, namely the probabilities of being in each hidden state given only the past observations. We can then compute the likelihood of each hidden state at a given Δt in the future using the HMM, and from there the user preferences for that time step. In particular, if $P_{H_{t+1}|H_t}$ is the matrix of transition probabilities, the transition probabilities after Δt steps are given by

$$P_{H_{t+\Delta t}|H_t} = P^{\Delta t}_{H_{t+1}|H_t} \quad (2)$$

and hence

$$P(h_{t+\Delta t}|y_{1\ldots t}) = \sum_{h_t} P(h_{t+\Delta t}|h_t) P(h_t|y_{1\ldots t}) \quad (3)$$

$$P(u_{t+\Delta t}|y_{1\ldots t}) = \sum_{h_{t+\Delta t}} P(u_{t+\Delta t}|h_{t+\Delta t}) P(h_{t+\Delta t}|y_{1\ldots t}). \quad (4)$$

V. MODEL EVALUATION

To evaluate our model we follow four steps: i) we generate a training set consisting of task trajectories without supportive behavior preference data in order to train the main task HMM, and we train the main task model on this set. We refer to this training set as set T henceforth, and explain the details in Section V-A; ii) we choose a small number of trajectories from the training set generated for the main task, and label

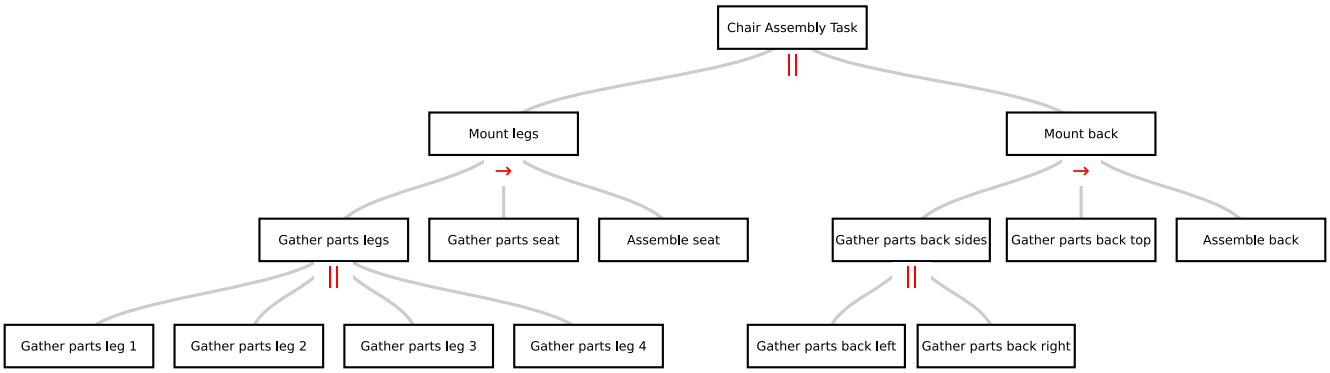

Fig. 3. Hierarchical task model (HTM) representing the chair assembly task. Each leaf is an atomic subtask; each node composes subtasks that need to be achieved in sequence (in a given order, $\rightarrow$) or in parallel (in any order, $\|$). For each atomic subtask, the human worker has preferences over supportive behaviors that the robot might provide.

them with supportive behavior preferences, experimenting with the exact number of trajectories to pick. We use this training set to train our personalized supportive behavior model. We refer to this training set as set *SB* henceforth, and explain the details in Section V-B; iii) we decide a mechanism for computing errors for our model (detailed in Section V-C), and we optimize parameters for all of the above during our simulation experiments (cf. Section V-D); iv) we use the optimized model from our simulated experiments (i.e., the trained HMM and the trajectories from set *SB*) to test with real users (Section V-E).

A. Training the main task HMM

In order to study the behavior of the learning algorithm on consistent and intuitive trajectories, we consider a task to be based on a hierarchical task model (HTM), as defined in [26]. HTMs represent tasks as trees of subtasks of varying complexities and abstraction. Each node represents a subtask and is itself a combination of subtasks following a sequence or parallel operation. Subtasks composed in sequence have to be executed in the order in which they appear as children of the node. Parallel combinations of subtasks can be executed in any order. Leaves of the tree are atomic subtasks. Typically, HTMs are compact representations of a task but they can correspond to complex constraints on task execution orders. In particular, parallel combinations of n nodes enable $n!$ execution orders. Our task represents a real world chair assembly task that is composed of a sequence of two main blocks, each of which has children that can be executed in parallel. The network of our task can be seen in Fig. 3.

To train our HMM in simulation, we construct training set T consisting of 350 trajectories generated in accordance with the presented HTM, which can generate a total of 96 distinct trajectories. In particular, given the HTM, we generate valid trajectories that are composed of leaves from the tree. We further assume that each leaf corresponds to a human activity, which generates the feature values based on object presence in the workspace. We only use the HTM in order to generate trajectories consistent with the structure of the task, which would be naturally observed in a factory environment. We do not learn, or insert knowledge about the HTM into the system at any point. In a real-world scenario, this step corresponds to gathering observations of human workers performing assembly tasks in their natural environment, without a robot providing assistance.

B. Training our personalized supportive behavior model

To train our personalized supportive behavior model, we need to choose a small number of trajectories from training set T and obtain user-provided labels for the supportive behaviors desired for these trajectories. We call this training set *SB*. For our simulation experiments, we consider a user with particular behavior preferences, and label our trajectories in accordance with these preferences. This provides us with set *SB*. During our user study, we perform trials in accordance with this set *SB*, and we also allow participants to choose their own preferences, by labeling the same trajectories and creating a personalized set *SB'*.

For each set *SB*, we assume stationarity of preferences (e.g., if a user prefers the screwdriver be brought for *leg 3*, we assume this to always be true for that preference set). Our goal is to minimize the number of trajectories for *SB*, since these always need to be labeled by users. We thus experiment with different numbers for the size of *SB*. When training with three or five, we use a decided-upon set of trajectories. These are chosen to cover different trajectories from our HTM, although our results are similar when we pick these at random. When training with more than five trajectories for parameter optimization, we pick these at random from our training set. In a real-world scenario, this step corresponds to asking human workers to label a set of task trajectories with their preferences for what supportive behaviors they would want a robot to provide during the task.

C. Decision model and computing errors

We use a model of an agent that takes all the supportive behaviors with probability greater than 0.5 at each time step. The probabilities are computed as explained in Section IV-C, with $\Delta t = 1$, meaning the agent predicts one time step in advance. In this work, we choose $\Delta t = 1$ because we predict

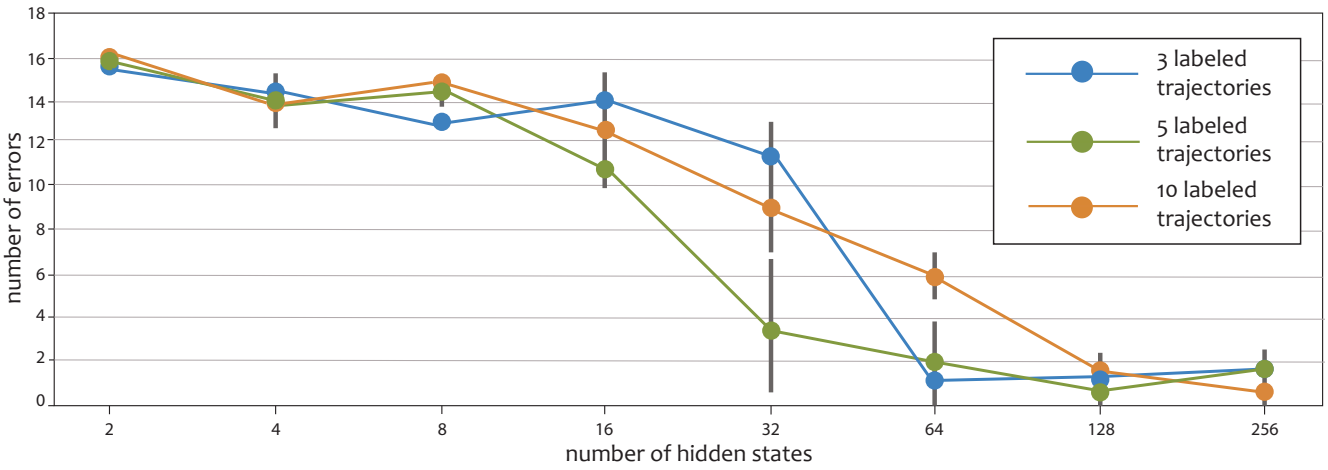

Fig. 4. HMM parameter optimization procedure. This graph presents training eight different models corresponding to $2^1 = 2$, $2^2 = 4$, ..., $2^8 = 256$ total number of hidden states, and running the training for each model 10 different times, each tested with a different HTM trajectory. The procedure was repeated for 3, 5, and 10 user-labeled trajectories, respectively. The x-axis presents the number of states equidistant for better visualization. We picked the HMM that provided us with the best trade-off regarding number of user demonstrations vs. number of hidden states.

for discrete time steps in terms of task progression, and each task step is essential for task completion. However, our model extends to any Δ value, and interesting investigations are possible, for example where we combine predicting more steps into the future with using macro-actions that span multiple time steps. We do not tell our model how many supportive behaviors it should choose at each time step. Rather, we automatically choose only those behaviors that have high certainty (i.e., greater than 0.5).

We count both false positive and false negative errors when computing the total number of errors per episode. If the robot missed bringing the dowel when it should have and brought the top bracket when it should not have, we count 2 errors for that step. For any parallel nodes (e.g., we can either start the assembly with the back or with the legs), either choice the agent performs is correct. We also allow users to change the order for parallel actions whenever they choose to do so.

D. Choosing model parameters

HMMs are prone to local optima. To choose our parameters for the user study, we ran a total of eight different HMMs, with increasing number of hidden states, using a single *SB* set composed of trajectories distinct from those we test on during our user study. We chose the number of hidden states as increasing powers of 2, starting with $2^1 = 2$ and ending with $2^8 = 256$ hidden states. Fig. 4 shows our training process with 3, 5, and 10 user-provided labels.

As the figure highlights, the higher the number of hidden states, the more power the HMM has to model the task. However, when we reach a large number of hidden states, we need more and more user-provided labels in order to scale up to the number of model parameters. This is revealed by the two curves seen in the figure for 3 and 5 labeled trajectories, where we notice the number of errors going down when the number of states initially increases (due to the higher modeling power of the HMM), but starts to increase again when the number of hidden states increases beyond the modeling power of the labeled trajectories. The curve for 10 labeled trajectories does not show this pattern in the figure, but we posit that with 10 trajectories, the turning point will happen with a higher number of hidden states.

We thus chose the HMM that provided us with the best trade-off in terms of necessitating a low number of labels and a low number of hidden states, resulting in 128 hidden states and the use of 5 user-labeled trajectories. Compared to choosing 256 hidden states with 10 labeled trajectories or 64 hidden states with 3 labeled trajectories, this choice results in the lowest average number of errors (0.20) and lowest standard deviation (0.60).

E. Testing personalized supportive behavior models on robot

We run our experiments with the task presented in Fig. 1, with the HTM from Fig. 3. Our experiment consists of two phases, as follows.

During the first phase, we provide an interactive survey to each participant. We provide them with three labeled trajectories, two of which can be seen in Table I. The participants are given a few minutes to consider the trajectories, and they are then asked to label five new trajectories with supportive actions based on the three examples, by thinking of the behaviors to be offered for the next time step as the task progresses. This task represents the equivalent of the prediction problem for our robot, with the same amount of information given (when we train on three trajectories). We then offer two more examples to the participants, which are based on the same task and same preferences. The new examples do not contain any additional information, but are meant to solidify the information already present in the previous examples. The initial three examples cover all the relevant information for the task and are not chosen to obscure any particularities of the preferences. The final provided set of five trajectories represents the same set *SB* we use to train our model. Finally, the participants are asked to label another set of five new trajectories. We consider the

first set to be a habituation phase with the task, and so we present only the results from the second set.

During the second phase, we confirm with participants the structure of the task and the set of preferences they just predicted for. We now ask them to play the part of the human worker, and allow the robot to make the predictions, while they assemble the chair. We have each participant assemble the chair twice with this set *SB* of preferences. We then allow participants to switch their preferences and choose a new set *SB'*, and perform two more assembly tasks on the new set. The trajectories are the same five ones the participants were given during the previous phase. We need only label these five trajectories according to the new preference set expressed by each participant in order to tailor the experience to each participant's liking. We perform this phase with a different set of preferences to show that our system allows for facile and quick labeling of the necessary trajectories to tailor assistance to individual preferences.

During the assembly phase with the robot, participants can always switch to a different action whenever a parallel alternative exists. For each time step during our task, the robot first makes its predictions for the following time step, and then executes the predicted behaviors in sequence. If the participant is satisfied with the behavior, he or she presses a green button on the robot arm that causes behavior to be completed; otherwise, the participant can press a red button on the arm that causes the behavior to be cancelled (i.e., the robot takes back any objects to their original pick-up location or stops executing the behavior in the case of *hold*). Participants are told to press the red button either in the case of a desired switch (we do not count the press as an error), or in the case of an incorrectly predicted and executed behavior on the part of the robot (we count the press as an error). Whenever the green button is pressed and an object is successfully released, the system automatically updates the features based on the new object presence.

After the robot finishes executing the predicted behaviors for a step, it asks the participant if it has missed any behaviors. The participant has three options: 1) state that the robot did not miss any behaviors, 2) ask for a new action due to a previous switch to a parallel action, or 3) ask for an action the robot missed due to incorrect predictions. In the first case, the interaction proceeds to the next step. In the latter two the experimenter, through a web interface, gets the robot to execute the missing actions before proceeding to the next step. The errors counted during the task execution are used only to evaluate the model, and not for online learning, although this would be an interesting future research direction.

VI. Results and Discussion

In this section, we present our results on the main set of user preferences, on the new per-user basis preference set, and compare them with human-level prediction for our task. Fig. 5 shows the average number of errors for the robot trials on the main set of preferences and on the user-specified set of preferences, as well as the human-level prediction in

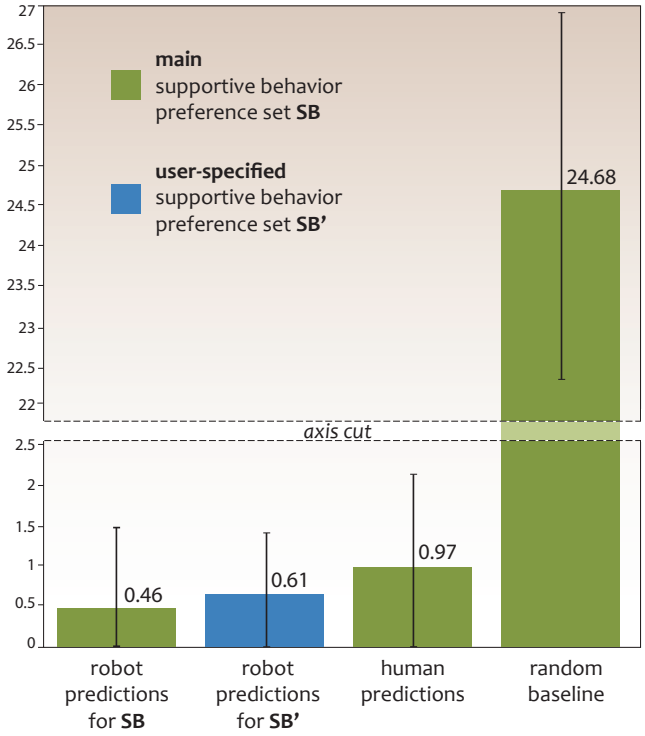

Fig. 5. Average number of errors for the main supportive behavior preference set **SB**, the user-specified set **SB'**, the human predictions (interactive survey), and a random baseline.

comparison to a random baseline for our task. We computed the random baseline by using information about the task structure in the following way. For a total of 10 time steps for one task trajectory, we used the number of times we need to choose 1, 2, or 3 supportive actions per time step to compute an estimate of how many behaviors to choose at that step. Although our predictive system does not receive this information and automatically chooses only those predictions it has high confidence about, we wish to show the power of our model by computing a baseline that uses some task knowledge. After choosing the number of behaviors for the current step, we proceed to randomly choose those behaviors from the available set. For each step, we sample without replacement, since our system only considers different supportive behaviors and cannot pick the same one twice for a single step. We then sample with replacement for the next time step. We compute the total number of errors by following the same procedure as described in Section V. The average for the baseline is 24.68 (s.d. 2.84).

We use the same evaluation in order to compute human-level prediction for our task: we only utilize the five trajectories participants have predicted after having seen the five examples that are also provided to our system. We compute errors similarly to the robot condition: when the person incorrectly predicts a supportive behavior (in accordance with the preference set), we count this as an error. Since these trajectories represent the second set participants go through, they have practice performing the prediction; hence this represents a high bar to compare our system against.

The average for human-level performance is 0.97 (s.d. 1.12). Fig. 5 shows that our system performs on par with this human-level prediction both for the initial set of preferences we ask users to operate with, and for the newly specified preference sets. The main set of preferences has an average of 0.46 (s.d. 1.05), and the average for the number of errors averaged over all of the newly specified preference sets across all participants is 0.61 (s.d. 0.86).

To make sure that many possible distinct trajectories were covered, all participants were reminded at the beginning of each assembly that they can always choose to switch to a different order for parallel actions. The data reveals that participants switched 11 out of 28 times and 10 out of 28 for the main and specified preference sets, respectively. These switches were performed for the main *chair assembly* parallel node from our HTM (Fig. 3). We also experienced 10 switches for the *mount legs* parallel node. The total number of different newly specified preference sets for the last two runs with the robot amounted to 7 different new sets.

VII. Conclusions

Our research focus is to develop robots that can adapt to HRC settings in a robust and seamless way. In this paper, we present a way of learning personalized supportive behavior models that leverage a single, robust model of the task, and show that our system achieves human-level prediction for a real HRC task via a user study.

Although we allowed our system to occasionally miss object-presence features when manipulation errors occurred during our study, we are excited to extend our model for handling noisier observations like the ones provided by a motion capture or computer vision system. We also look forward to experimenting with different tasks. We have already started testing how well we can transfer knowledge from one task to a related, but previously unseen task, and are excited to present these results in future work.

References

[1] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1503–1510.

[2] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4019–4026.

[3] O. Mangin, A. Roncone, and B. Scassellati, "How to be Helpful? Implementing Supportive Behaviors for Human-Robot Collaboration," *arXiv preprint arXiv:1710.11194*, 2017.

[4] G. E. Hovland, P. Sikka, and B. J. McCarragher, "Skill acquisition from human demonstration using a hidden markov model," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 3. Ieee, 1996, pp. 2706–2711.

[5] Q. Zhu, "Hidden markov model for dynamic obstacle avoidance of mobile robot navigation," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 390–397, 1991.

[6] M. N. Nicolescu and M. J. Mataric, "Learning and interacting in human-robot domains," *IEEE Transactions on Systems, man, and Cybernetics-part A: Systems and Humans*, vol. 31, no. 5, pp. 419–430, 2001.

[7] C.-M. Huang and B. Mutlu, "Anticipatory robot control for efficient human-robot collaboration," in *Human-Robot Interaction (HRI), 2016 11th ACM/IEEE International Conference on*. IEEE, 2016, pp. 83–90.

[8] S. Nikolaidis, R. Ramakrishnan, K. Gu, and J. Shah, "Efficient Model Learning from Joint-Action Demonstrations for Human-Robot Collaborative Tasks," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '15. New York, NY, USA: ACM, 2015, pp. 189–196.

[9] A. Roncone, O. Mangin, and B. Scassellati, "Transparent Role Assignment and Task Allocation in Human Robot Collaboration," *Robotics and Automation (ICRA), IEEE International Conference on*, 2017.

[10] M. Fox, M. Ghallab, G. Infantes, and D. Long, "Robot introspection through learned hidden markov models," *Artificial Intelligence*, vol. 170, no. 2, pp. 59–113, 2006.

[11] A. Vakanski, I. Mantegh, A. Irish, and F. Janabi-Sharifi, "Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1039–1052, 2012.

[12] S. Calinon, F. D'Halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation: An approach based on hidden markov model and gaussian mixture regression," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, 2010.

[13] M. Kawato, Y. Uno, M. Isobe, and R. Suzuki, "Hierarchical neural network model for voluntary movement with application to robotics," *IEEE Control Systems Magazine*, vol. 8, no. 2, pp. 8–15, 1988.

[14] E. W. Aboaf, C. G. Atkeson, and D. J. Reinkensmeyer, "Task-level robot learning," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*. IEEE, 1988, pp. 1309–1310.

[15] E. C. Grigore, O. Mangin, A. Roncone, and B. Scassellati, "Predicting supportive behaviors for human-robot collaboration," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 2186–2188.

[16] N. Nguyen-Duc-Thanh, S. Lee, and D. Kim, "Two-stage hidden markov model in gesture recognition for human robot interaction," *International Journal of Advanced Robotic Systems*, vol. 9, no. 2, p. 39, 2012.

[17] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis, "Understanding human intentions via hidden markov models in autonomous mobile robots," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*. ACM, 2008, pp. 367–374.

[18] U. Kartoun, H. Stern, and Y. Edan, "A human-robot collaborative reinforcement learning algorithm," *Journal of Intelligent & Robotic Systems*, vol. 60, no. 2, pp. 217–239, 2010.

[19] A. L. Thomaz, G. Hoffman, and C. Breazeal, "Reinforcement learning with human teachers: Understanding how people want to teach robots," in *The 15th IEEE International Symposium on Robot and Human Interactive Communication, 2006. ROMAN 2006*. IEEE, 2006, pp. 352–357.

[20] T. Ogata, N. Masago, S. Sugano, and J. Tani, "Interactive learning in human-robot collaboration," in *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, vol. 1. IEEE, 2003, pp. 162–167.

[21] J. Mainprice, R. Hayne, and D. Berenson, "Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 885–892.

[22] G. Hoffman and C. Breazeal, "Cost-based anticipatory action selection for human–robot fluency," *IEEE transactions on robotics*, vol. 23, no. 5, pp. 952–961, 2007.

[23] C. Breazeal, G. Hoffman, and A. Lockerd, "Teaching and working with robots as a collaboration," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*. IEEE Computer Society, 2004, pp. 1030–1037.

[24] B. Hayes and B. Scassellati, "Effective Robot Teammate Behaviors for Supporting Sequential Manipulation Tasks," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 2015.

[25] S. Zeylikman, S. Widder, A. Roncone, O. Mangin, and B. Scassellati, "The HRC model set for human-robot collaboration research," in *Intelligent Robots and Systems (IROS), 2018 IEEE/RSJ International Conference on*. IEEE, Oct 2018.

[26] B. Hayes and B. Scassellati, "Autonomously constructing hierarchical task networks for planning and human-robot collaboration," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 5469–5476.