Abstract

Learning Supportive Behaviors for Adaptive Robots in Human-Robot Collaboration

Elena Corina Grigore

2018

Robotic systems deployed in industry today work in isolation from humans, and perform precise, repetitive tasks, based on well-defined plans and known task structures. The field of human-robot collaboration (HRC) [1] focuses on endowing robots with capabilities to work alongside humans and help them achieve various tasks in different settings. Whether it be in manufacturing settings such as factories, public spaces such as restaurants, or in our homes, robots should adapt to the task at hand and learn from the user how to best assist. Although robotics has seen recent successes like quickly learning certain manipulation tasks [2, 3], robots are still far from autonomously carrying out high dexterity tasks. Even beyond manipulation issues, the level of task knowledge necessary for a robot to autonomously build a piece of furniture, for example, is extremely high and not straightforward to acquire.

We are interested in leveraging machine learning techniques to create adaptive robots capable of learning useful behaviors in dynamic and collaborative environments. This requires the robot to learn from both the people it interacts with and its environment. Learning models of the task, understanding a person's actions throughout the progression of the task, and learning and predicting user-based preferences for tailored assistance stand at the basis of adaptive robots in HRC. Applying such techniques to complex state and action spaces—raw sensor data at the low-level, and abstract states at the high-level—involves considerable challenges, especially when considering human-in-the-loop scenarios. To create a useful system, we turn our attention towards semi-autonomous robots that aim to learn how to provide supportive behaviors to a person during a task, and not complete the task

autonomously themselves. Such assistance is meant to help the person complete the task more efficiently, while allowing the human and the robot to perform those actions for which they are best suited.

In this thesis, we present novel models and paradigms for HRC that allow a robot to learn how to provide assistance to a human worker throughout the execution of a task. We first focus on learning about action primitives (building blocks of motion) that a human worker performs during a physical task. We present a framework that discovers whether a coarser- or a finer-grained level is better suited for a primitive given the task at hand, coining this concept the *granularity level* of a primitive.

We then present models for high-level learning, where the robot learns what supportive behaviors to offer throughout the execution of a task in which both the human and the robot are involved. To do so, we introduce personalized models of user *supportive behavior preferences*, which we build atop of a single, cross-users model of the task. The personalized models leverage this task representation, and only require as few as five demonstrations of the task labeled by the user in order to train.

We further compare this model-based technique with a model-free variant inspired by multi-agent reinforcement learning. We present two novel HRC paradigms that are multi-agent based, where we consider both the robot and the human as agents operating in the environment. These paradigms are multi-agent based since the human is indeed an agent in the system, but one whose actions we do not control. We introduce the *Multi-Agent Based Reinforcement Learning (MAB-RL)* and the *Hierarchical Multi-Agent Based Reinforcement Learning (HMAB-RL)* paradigms, and present a total of four algorithms based on these paradigms. We show that we can learn a supportive behavior preference set for the task on par with human-level performance from 40 episodes, with varying episode lengths dictated by whether we employ macro-actions.

Finally, we present work on the social aspect of interactions within HRC. Alongside gaining an understanding of how to act in a useful manner to help accomplish the task as

a team, social collaboration is another important facet of HRC. An adaptive robot needs to facilitate task progression by keeping the human engaged and motivated throughout. Thus, we present a series of studies that set out to explore different tools for a robot to maintain high engagement during collaborative tasks and engender positive user perceptions and reactions.

As we move towards an era where we envision robots becoming widely used in a variety of settings, developing robust techniques for such robots to learn how to usefully interact and collaborate with people becomes critical. The algorithms and paradigms presented in this thesis contribute to the aim of developing more intelligent, useful, and adaptive robots, capable of offering assistance tailored to humans' needs and preferences.

Learning Supportive Behaviors for Adaptive Robots in Human-Robot Collaboration

A Dissertation
Presented to the Faculty of the Graduate School
of
Yale University
in Candidacy for the Degree of
Doctor of Philosophy

by
Elena Corina Grigore

Dissertation Director: Brian Scassellati

December, 2018

Contents

List of Figures

List of Tables

To Names here

Acknowledgements

Add acknowledgements here.

Glossary

HRC Human-Robot Collaboration

HRI Human-Robot Interaction

Chapter 1

Introduction

One of the visions of robotics research is to integrate robots into everyday life, having them help humans in a variety of settings. We envision robots functioning smoothly in settings like public spaces, where we wish robots to act as guides in museums or waiters in restaurants, in our homes, where we wish robots to act as helpers, or in manufacturing settings, where we wish robots to provide assistance to human workers or execute tasks themselves. In order to achieve the wide array of functionality such robots would necessitate, we need to push the boundaries of state-of-the-art robotic systems extant today.

The status quo for such systems in industry today is for robots to work in isolation from people due to their lack of adaptation skills and task knowledge. For a robot to execute a particular task, it needs to be pre-programmed with specific knowledge about the task structure, and the manipulation needs that go alongside. Such pre-programming is costly, and necessitates highly trained roboticists, resulting in a high price for changing between contexts or tasks to be executed by the robot. Thus, 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) [1] is an area of robotics that focuses on robots that can work alongside humans, in settings such as factories, and that can learn how to be useful in these contexts by learning about the task and the humans' preferences.

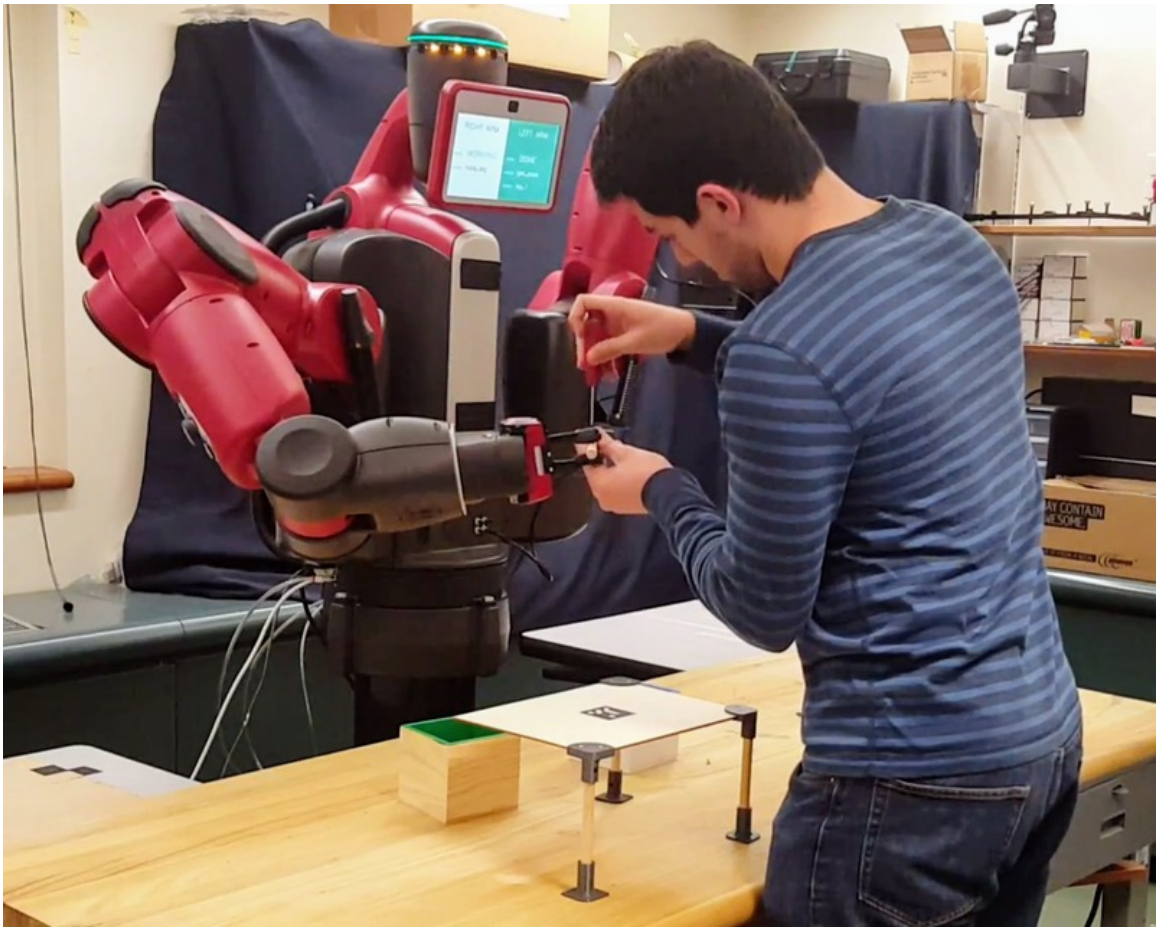


Figure 1.1: Target application domain for Human-Robot Collaboration (HRC) research. The human's goal is to assemble the chair, and the robot learns how to perform supportive behaviors that are helpful towards achieving this goal.

To create such adaptive robots, we need to develop robust and dependable techniques for the robot to learn both from its environment and from the people with whom it interacts. One of the main focuses in HRC is thus how to best learn useful information that will help the robot adapt in novel situations, as well as flexibly pivot to working with different humans, each of whom might have individual preferences for executing various tasks. In this thesis, we focus precisely on this aspect: *How do we leverage machine learning techniques in order to develop robots that are capable of learning about the structure of a task, about the human preferences for what supportive behaviors the robot should offer throughout this task, and about how to work alongside the human to accomplish a shared goal?* The typical HRC scenario we consider throughout this thesis is depicted in Figure 4.10.

1.1 Challenges

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 main 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 [2, 4], 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 aim for 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, while the human worker can execute actions such as screwing, which requires higher dexterity and a particular type of adaptation to 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 the robot to utilize that time to bring a screwdriver for the next step of the assembly. These different options are not the task ordering preferences referring the sequencing 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. We thus refer to preferences related to sub-state order as *task preferences* and to those related to the type of assistance to be provided as *supportive behavior preferences*.

Another critical aspect of HRC is the difficulty of modeling human behavior during

the interaction. On one hand, existing decision models for robots either rely on known models of the task, human behaviors, and their preferences, or are model-free but require a large amount of interactions to train. They are hence non-trivial to apply to realistic work environments, in which humans introduce even more complexity and uncertainty [5]. On the other hand, models of human motions or activities are designed to be trained on potentially high-frequency and high-dimensional observations, but their use as the basis for robot collaborative behaviors have been little studied. HRC thus suffers from both a very challenging modeling problem with respect to modeling human behavior, and from a data problem with respect to the expense of obtaining human labels or demonstrations to train different models.

1.2 Contributions and Thesis Outline

In order to address the difficulties surrounding the development of a robot that can complete a task on its own autonomously, we focus on robots that are meant to provide support to human workers throughout the execution of a task. This allows both the robot and the human to perform those types of behaviors and actions they are best suited for, and allows for the leveraging of the knowledge the human worker has. In a context such as manufacturing, the human worker is an expert at the task at hand, and so can teach the robot about the task structure in a multitude of ways.

To address (1) the challenge of learning *supportive behavior preferences* and (2) the tension between expensive interactions with users and the high volumes of data needed to model complex human behavior, we make the following main contributions:

- *We tackle the difficulty of learning useful information about what happens throughout the execution of a complex task in HRC:* **We learn action primitives (building blocks of motion) that a human worker executes throughout a physical task. We present a framework that discovers whether a coarser or a finer-grained level**

is better suited for a primitive given the task at hand, coining this concept the *granularity level* of a primitive.

- *We tackle scenarios where it is feasible to obtain a large pool of data consisting only of task trajectories executed by the human worker, with no information about preferences for robot assistance:* **We create a model of personalized supportive behavior preferences for different users that can be trained from as few as three task trajectories labeled by a user, built atop of a single, cross-users model of the task. This system achieves on par with human-level performance.**

- *We tackle scenarios where it is difficult to obtain large amount of data and thus desire a model-free approach:* **We introduce *MAB-RL* and *HMAB-RL*, a multi-agent based reinforcement learning paradigm and for HRC and its extension to hierarchical reinforcement learning, where we consider both the robot and the human worker as agents in the environment, but where we do not control the person's actions throughout the task. We compare algorithms based on these paradigms that employ different ways of synchronizing the robot and human's action during each time step, including an algorithm that utilizes *communicative* actions. We present results that show the robot can perform on par with human-level performance from training sessions composed of 40 episodes, with varying episode lengths.**

- *We explore the social dimension of collaboration:* **We perform a number of studies that present results supporting different behaviors we can endow collaborative robots with in order to keep their users more engaged and motivated throughout the task.**

The rest of this thesis is organized as follows. In **??**, we introduce the relevant background work for the main contributions of this thesis, including modeling robot decisions,

predicting human behavior, and work at the intersection of these areas. We further introduce the formalisms for different mathematical frameworks used throughout this thesis, including Hidden Markov Models (HMMs) and Markov Decision Processes (MDPs).

Chapter 3 presents the framework for discovering the granularity level of action primitives. In this chapter, we focus on learning about appropriate action primitives (building blocks of motion) that a human worker executes throughout a physical task. We present a framework that discovers whether a coarser or a finer-grained level is better suited for a primitive given the task at hand, coining this concept the *granularity level* of a primitive. This corresponds to the contribution mentioned as part of the first bullet point above.

Chapter 4 presents the model based paradigms for HRC, together with the associated algorithms, corresponding to the contribution mentioned as part of the second bullet point above. We present a model-based method for learning personalized supportive behavior preferences, by extending a single, cross-users model of the task. This separation between learning a model of the task from learning a model of supportive behavior preferences allows me to mitigate the need for a large training corpora that would need to include interaction data. The model-based method I present here not only helps take the burden off users with respect to labeling demonstrations, but it focuses on a core challenge in HRC that does not appear in typical machine learning domains?that of needing to model interactions that agents operating in this environment (be it humans or robots) have or wish to have with the environment and task.

Chapter 5 presents the model-free paradigms for HRC, together with the associated algorithms, corresponding to the contribution mentioned as part of the third bullet point above. The model-based method we propose in Chapter 4 is well-suited for HRC since interactions with a person or labels provided by users are expensive to obtain, but it still assumes access to a large pool of task trajectories of the human worker performing the task themselves. Although we can easily imagine mounting sensors in environments like factories in order to build up this dataset consisting of task trajectories while simply observing

human workers in their natural environment, and thus without putting an extra onus on workers, model-free techniques are still quite appealing for scenarios where acquiring such a dataset is not easily accomplished. We thus present a novel model-free algorithm based on the multi-agent setting, where we consider both the robot and the human as agents operating in the environment.

This paradigm is multi-agent based since it is the joint action of the robot and the human (and not just the robot action) that effects the transition from one state of the system to the next, as well as the reward received when transitioning. However, since there is no controlling the human's actions like in typical multi-agent settings, we term this paradigm *multi-agent based (MAB-RL)*. This allows maximum flexibility for specifying preferences with respect to what actions the human would like to themselves perform at vs. what actions they would like the robot to execute at each step. Alongside this valuable flexibility, this paradigm mitigates the need for a large pool of training data, but generally requires more interactions with the user or user-provided labels for demonstrations. Although it requires more interactions than the proposed model-based technique, compared to the single-agent paradigm (which we also apply to this scenario for comparison purposes) the multi-agent based model leverages the knowledge of the human worker, considered an expert at performing the task at hand, and thus allows both agents to take actions during a time step, which helps speed up learning. We present different variants of the algorithm that consider different ways of synchronizing the robot and human's action during each time step (including an algorithm that utilizes *communicative* actions), and how this affects quality and speed of learning. We also present a hierarchical variant of the paradigm, coined *Hierarchical Multi-Agent-Based Reinforcement Learning (HMAB-RL)*, which speeds up learning through the use of options. The algorithms introduced in this chapter achieve on par with human-level performance from training sessions composed of 40 episodes, with varying episode lengths.

Chapter 6 goes over the studies that focus on endowing robots with behaviors that help

to keep users engaged and motivated throughout the task, corresponding to the contribution mentioned as part of the fourth bullet point above. An adaptive robot need not only know what the different preferences of the user are in terms of executing physical actions, but it needs to facilitate task progression by keeping the human engaged and motivated throughout the task. We thus focus on endowing robots with useful behaviors that can help maintain high levels of engagement and motivation for the users throughout the task they are attempting. In this chapter, we present a series of studies that set to explore different valuable tools we can endow a robot with that can help maintain high engagement during collaborative tasks and engender positive user perceptions and reactions with respect to the robot. We present studies that center on conveying information effectively during interactions with robots that center on both teaching and rapport building. We further focus on verbal communication in collaborative environments, and on how a robot can improve users' perception of its social presence in such a context. Finally, we explore how different embodiments of the agent can engender higher perceptions of a robot in a collaborative environment.

Chapter 7 summarizes the key points in this thesis and discusses limitations, as well as future directions of research. We also add an extra analysis representing an incursion into modeling human behavior in HRC by predicting the amount of time required for different agents to complete actions during a task, which can be found in Appendix A.

Chapter 2

Background

Robotics can heavily benefit from machine learning in order to develop autonomous and adaptive robots that do not need to be pre-programmed for specific tasks. The typical robotics application, however, fundamentally differs in nature from the standard paradigm assumed in machine learning due to the effects robot actions have on the state of the world and the feedback loop this creates. This work is positioned between approaches that focus on modeling a robot's decision, and work on modeling and predicting human behaviors. In section 2.1, we review the most relevant research spanning these areas, starting with how work on predicting human behavior can be integrated into the type of Human-Robot Collaboration (HRC) scenarios we are considering is this dissertation (Subsection 2.1.1), continuing with work on modeling robot decisions relevant to HRC (Subsection 2.1.2), and concluding with work that is considered to be fully part of HRC and is at the intersection of these areas (Subsection 2.1.3). Finally, in Section 2.2, we present the formalisms for the main techniques used throughout this thesis.

2.1 Related Work

2.1.1 Predicting Human Behavior

We start by giving an overview of work on modeling human motions and activities. Many techniques within this area employ hidden Markov models (HMMs) [6] or variants thereof, like we do in our own work. Modeling human motions and activities represents a widely studied topic, and so we touch upon some of the most relevant work to our own. Ogawara et al. [7] model manipulation interactions, Bernardin et al. [8] recognize continuous human grasping sequences, Yang et al. [9] perform gesture recognition, and Fox et al. [10] model multiple time series with an application to motion capture segmentation, all via employing HMM-based models.

In robotics, HMMs are also widely utilized. Fox et al. [11] describe the use of an HMM to acquire behavioral models for robots, Zhu et al. [12] model dynamic obstacle avoidance for a mobile robot, Vakanski et al. [13] learn robot trajectories acquired from demonstration, and Calinon et al. [14] have a robot learn and reproduce gestures by imitation.

Work on model learning in robotics spans a wide range of topics. Nicolescu et al. learn high-level representations of navigation tasks for mobile robots from observations and then use the learned model for the robot to engage humans for help, when needed, during parts of the task execution [15]. Nguyen et al. provide a real-time online-learning method called local Gaussian process regression (LGP) and show they can learn in real time the inverse dynamics for controlling a robot arm [16]. Kawato et al. outline the application of a neural network model to control an industrial robotic manipulator, focusing on modeling the computational tasks of trajectory determination, coordinates transformation, and motor command generation [17]. Aboaf et al. [18] perform task-level learning that can refine the task command based on the system's error metrics, with an application to ball throwing.

Model learning has also been successfully used in inverse kinematics [19, 20, 21], robot locomotion [22], and navigation for autonomous robots [23]. Nguyen et al. survey the main

applications of model learning for control, including representative work for simulation-based optimization [24, 25, 26], approximation-based inverse dynamics control [27, 28], and learning operational space control [29, 30, 31].

2.1.2 Modeling Robot Decision

In this thesis, we present both single- and multi-agent methods for tackling the problem of learning supportive behaviors in HRC. Below, we present relevant work for each of these two paradigms. Many of these paradigms are based on Markov Decision Processes (MDPs), which we define in Subsection 2.2.2. MDPs are models used for sequential decision making when outcomes are uncertain [32], hence their heavy use in robotics.

Single-Agent Paradigms

In this category, we find model-based and model-free techniques. Hester et al. propose a model-based reinforcement learning algorithm called reinforcement learning with decision trees (RL-DT) that utilizes decision trees to learn a model in a sample efficient manner, and test it on a robot scoring goals scenario [33]. Theocharous et al. [34] learn hierarchical observable Markov decision process models for robot navigation. As an alternative to assuming access to expert knowledge in the form of demonstrations, Deisenroth et al. [35] model the observed dynamics by learning a probabilistic, data-efficient, non-parametric Gaussian process transition model of the system called probabilistic inference for learning control (PILCO). Doya et al. [36] tackle the issues of non-linearity and non-stationarity in real-world control problems by proposing a new reinforcement learning architecture that decomposes a non-linear or non-stationary control task in space and time, using multiple modules.

Multi-Agent Paradigms

In order to account for different agents operating within our environment, we need to consider the different types of actions performed by each, the effects their actions have on the world they operate in, and the goals they are working towards. A relevant paradigm for representing such a problem is multi-agent systems (MAS) [37], where several agents attempt to solve tasks together in order to reach specific goals, either shared or competing. A particularly useful method that has been widely applied to MAS is that of reinforcement learning (RL) [38]. In RL, the agent operates in an environment in which it receives rewards when performing different actions in each state of the world and subsequently aims to behave such that it maximizes the long-term reward received. The use of RL for MAS has engendered the multi-agent reinforcement learning (MARL) paradigm [39], where multiple learners can take different actions in the same environment, each trying to maximize long-term reward (such agents can be collaborative, competitive, or somewhere in between).

The field of MAS is relevant to a wide array of areas, including robotics, telecommunications, distributed control, economics, and so on. A considerable amount of research studies the use of RL for MAS, the intersection of which is known as multi-agent reinforcement learning (MARL). MARL presents important benefits, such as faster learning through experience sharing (e.g., exchange of local information, skilled agents teaching less skilled ones, etc.), speed-ups due to parallel computation when exploiting the decentralized structure of the task, robustness to failures of individual agents, and easy insertion of new agents into the system [39], [37]. However, it is also plagued by a number of challenges, including the curse of dimensionality (the state action-space grows exponentially with the number of agents), the difficulty in specifying goals in such multi-agent settings (the rewards different agents receive are correlated and thus cannot always be maximized independently), non-stationarity due to concurrent learning (for each agent, the best policy changes as other agents' policies change), and an exacerbated exploration-exploitation tradeoff (agents need to explore not only to reach new regions of the state space, but also

to gather information about other agents) [39], [37].

Even though there exist considerable challenges in the field of MARL, this framework is extremely relevant to our human-robot collaboration scenario. This is due to the fact that, even though it is not part of our goal to find policies for the human worker during the task, their actions affect the transition model of the world in which our robot needs to operate. Beyond directly impacting the world dynamics, the actions performed by the human also play a key role in the type of supportive behavior that is useful for the robot to offer during different subtasks. In order to address two of the main challenges mentioned above—the curse of dimensionality, and partial observability with respect to the policies of other agents—researchers have looked into the use of hierarchical reinforcement learning (HRL). Work in this area includes the use of options [40], HAMs [41], and MAXQ [42]. Other work looks at using task-level coordination in HRL settings, where all agents were given a hierarchical break down of the task at hand that was used as a basis for communication [43], [44].

One of the most important and widely used ways in which we can generalize MDPs to the domain of MARL is via stochastic games [45]. There are many different types of stochastic games, but for the purposes of this thesis, we consider general-sum stochastic games, where there are no constraints on the sum of the agents' rewards. This is due to the cooperative nature of our environment, where the two agents are working together towards the same goal. However, if we were to simply extend techniques used to solve MDPs for the single-agent case to the multi-agent case, we would loose optimality guarantees due to the loss of stationarity in MARL caused by concurrent learning. One of the solutions to this issue is using the Nash equilibrium (or other equilibria) to describe optimal policies in multi-agent systems. In this thesis, we build upon a variant of the Nash Q-Learning algorithm for general-sum stochastic games [46] called Friend-or-Foe Q-learning (FFQ) in general-sum games [47].

Work that addressed partial observability and the issue of how the actions of other

agents impact the policy of one agent includes a learning paradigm called team-partitioned, opaque-transition reinforcement learning (TPOT-RL). TPOT-RL considers opaque transitions to encompass multi-agent environments where each learner cannot rely on having knowledge of future state transitions after acting in the world [48]. Other relevant work in this area uses behaviors, defined as goal-driven control laws that achieve and/or maintain particular goals, to abstract away low level details of robot control and diminish the learning space, as well as tackling the credit assignment problem via heterogeneous reinforcement functions and progress estimators [49].

2.1.3 Work at the Intersection

A final and extremely important corpus of related work is composed of research in the Human-Robot Collaboration (HRC) domain. HRC is an area of robotics that focuses on endowing robots with the capabilities they need in order to operate efficiently and safely in natural, populated environments, as well as achieve higher levels of cooperation and communication with humans [1]. Because of the interdisciplinary nature of this field, we mention relevant work to this dissertation below, touching upon work that utilizes HMMs, work that employs interactive learning and model learning, and work that utilizes reinforcement learning-based techniques.

Relevant work in this area that is based on the use of HMMs includes gesture recognition for human-robot interaction [50], skill acquisition from human demonstrations [51], skill modeling for human-robot coordination [52], and understanding human intentions within the context of autonomous mobile robots [53]. Reinforcement learning techniques have been utilized by Kartoun et al. [54] for enabling collaborative learning between a robot and a human, and by Thomaz et al. [55] for leveraging the way in which people teach robots. Further important work includes interactive learning in HRC [56], manipulation planning for HRC [57,58], anticipatory robot control for HRC [59], action-selection mechanisms for improving human-robot fluency [60,61,62], learning models of joint actions for

HRC [63, 64], and improving robot performance as a collaborator in HRC for sequential manipulation tasks [65].

Further work in HRC centers on how to employ human feedback as a reward signal. Such work includes enabling a robot to improve its policy for a knot untying task through user-provided guidance. In this scenario, the robot performs a set of state-action transitions to shake the contents placed inside of a bag (with the goal of untying the knot and emptying the contents of the bag), and asks for a reward signal from the human at the end of each learning episode [54]. Other relevant work in the area of obtaining rewards from a human user explored the importance of understanding the human-teacher/robot learner system as a whole, with findings that people use the reward signal not only to provide feedback about past actions, but also to provide future directed rewards to guide subsequent actions [55]. Yet other work investigated discovering policies for improving collaborator performance through a task and motion planning approach [66], which differs from our MARL framework.

Another HRC contribution we wish to mention here is that of the hierarchical task model (HTM), as defined in [67]. Throughout this thesis, we employ HTMs to represent our tasks as a way of generating consistent and intuitive trajectories for our training sets. HTMs following the definition herein 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.

2.2 Formalisms

The main relevant formalisms the techniques presented in this thesis are built upon include Hidden Markov Models (HMMs) and Markov Decision Processes (MDPs), which we present below.

2.2.1 Hidden Markov Models (HMMs)

The following description closely follows the explanations from Rabiner et al. [68]. A Hidden Markov Model (HMM) is a doubly stochastic process with an underlying stochastic process that is not observable (it is hidden), but can only be observed through another set of stochastic processes that produce the sequence of observed symbols. An HMM has a finite number of states, N, with each state being characterized by a signal with measurable, distinctive properties. At each time step, t, a new state is entered based upon a transition probability distribution that depends on the previous state only, respecting the Markovian property. After each transition, an observation output symbol is produced according to a probability distribution which depends on the current state.

Formally, an HMM can be defined with notation $\alpha = (A, B, \pi)$, where:

- $A = a_{ij}, a_{ij} = Pr(q_j att + 1 | q_j att)$, state probability distribution
- $B = b_j(k), b_j k = Pr(v_k att | q_j att)$, observation symbol probability distribution in state j
- $\pi = \pi_i, \pi_i = Pr(q_i att = 1)$.

The HMM has N number of model states, M number of observation symbols, $Q = Q_1, Q_2, ..., Q_N$ states, and $V = v_1, v_2, ..., V_M$ discrete set of possible symbol observations. Using this model, an observation sequence $O = O_1, O_2, ..., O_T$ can be generated as follows:

- Choose an initial state, i_1, according to the initial state distribution, π

- Set $t = 1$
- Choose O_t according to $b_{i_t(k)}$, the symbol probability distribution in state i_t
- Choose i_{t+1} according to $a_{i_t i_{t+1}}, i_{t+1} = 1, 2, ..., N$, the state transition probability distribution for state i_t
- Set $t = t + 1$, and return to the third step if $t < T$ or terminate procedure.

The three main problems relevant to HMMs are as follows:

- *The evaluation problem:* Given the observation sequence $O = O_1, O_2, ..., O_T$, and the model $\alpha = (A, B, \pi)$, how do we compute $Pr(O|\alpha)$, the probability of the observation sequence?
- *The uncovering the hidden part of the model problem:* Given the observation sequence $O = O_1, O_2, ..., O_T$, how do we choose a state sequence $I = i_1, i_2, ..., i_T$that is optimal in some meaningful sense?
- *The optimizing the model parameters problem:* How do we adjust the model parameters $\alpha = (A, B, \pi)$ to maximize $Pr(O\ \alpha)$?

2.2.2 Markov Decision Processes (MDPs)

An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{I} \rangle$, where $\mathcal{S}$ represents the state space the agent operates in, $\mathcal{A}$ represents the action space the agent acts within, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a probabilistic transition function (representing the dynamics of the world) such that $P(s'|s, a)$ denotes the probability of transitioning to state s' when the agent takes action a in state s, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ represents the reward function with $r(s, a)$ being the reward received by the agent when it takes action a in state s, and $\mathcal{I} : \mathcal{S} \rightarrow [0, 1]$ constitutes the initial state distribution. The typical MDP formulation for infinite-horizon settings (i.e. where the agent can take an infinite number of steps) uses discounted sum of rewards $\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$, where $\gamma \in [0; 1)$ is the discount factor.

Chapter 3

Learning at the Low Level: What Can We Learn from Raw Data?

In this chapter, we explore what kind of information we can learn directly from raw data that is useful for the HRC context. This chapter is based on [69] [1].

In order to build adaptive robots capable of providing assistance to a human worker, the robot needs to understand the state of the environment and how this is changing. For complex tasks, state changes encompass both the movements of the person, and the motions of the objects manipulated by the human. To this end, there has been a great deal of attention on learning from observations of a person performing a particular task. Demonstrations can be leveraged to both teach the robot how to imitate certain human motions we wish to replicate, and to allow the robot to learn about the structure of the task and its progression.

To handle the complexity of human motion, researchers have formalized it to be composed of basic units of action, called action or motor primitives. Primitives can be sequenced or combined to generate more complex behavior [70]. A rich area of research focuses on learning action primitives from raw data, in order to avoid the need for hand-crafting, and to allow for scalability. The goal is to then feed these learned primitives into

1. E. C. Grigore and B. Scassellati, "Discovering action primitive granularity from human motion for human-robot collaboration", in Robotics: Science and Systems (RSS), Boston, USA, 2017, July 12–16.

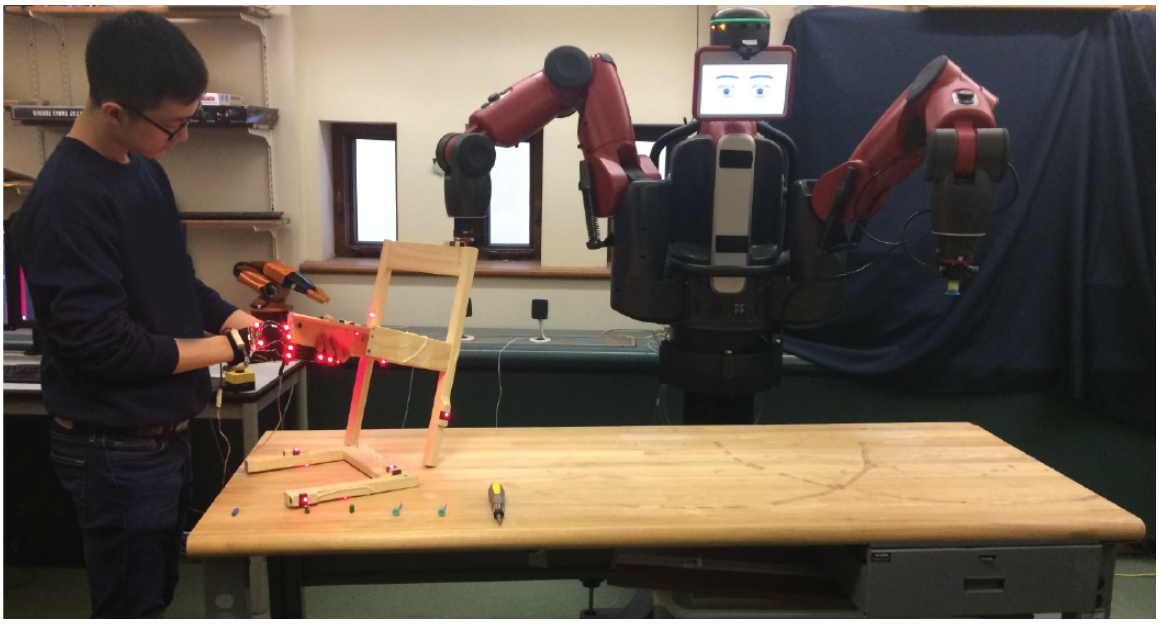


Figure 3.1: Participant assembling an IKEA chair during data collection. We use a motion capture system to record the participant's hand movements (via gloves fitted with sensors), and the trajectories of the chair components.

high-level planners that expect them as inputs. Such planners usually assume that a library of primitives is available for them to build upon [67], [71].

Obtaining clear and useful primitives necessary for existing planners is difficult to accomplish, and depends on the task at hand. If our task is to observe a person's general morning routine, we might benefit from primitives like *brush teeth*, *get dressed*, etc. However, what if we instead wish to have a robot learn how to imitate the person's physical movements during this routine? In that case, primitives like *grab toothbrush*, *add toothpaste*, *execute brushing motion* might be better suited, depending on the robot's physical capabilities. Other situations also show the need to investigate this level of granularity. For example, using primitives with low granularity such as *press gas pedal* or *release gas pedal* when driving might prove to be useful since the motion of pressing the pedal has an easily identifiable signature. Using a low level for a primitive like *move cup to mouth* when observing a person drinking, on the other hand, might cause problems due to the variability in the human's motion when executing this action.

My work aims to find a solution to the problem of discovering an appropriate level of

granularity of primitives for the task at hand. Existing methods of learning primitives from observations do not tackle this problem. Such techniques often aim to learn primitives that are clear and easy to label, such as *running*, *jogging* or other repetitive motions, or investigate robot motion, which is inherently different from human movement. In order to make use of existing techniques for learning primitives and use these as inputs for higher-level planners that have already been developed, discovering the appropriate level of granularity is of paramount importance.

In this chapter, I present a framework for primitive granularity discovery, and implement it for a human-robot collaboration (HRC) scenario involving the complex task of assembling an IKEA chair as follows. I record a set of 28 participants performing the task, using a motion capture system to acquire both hand motions and object movements. Fig. 3.1 shows a participant putting together the chair during the data collection phase. I learn primitives in the data sets via a beta process hidden Markov model (BP-HMM) [72]. Of the learned primitives, I set out to discover the appropriate level of granularity for object *pick up*, a widely used primitive in robotics. I use task-specific factors in order to establish reasonable minimum and maximum levels of granularity, and use these levels to test what classification performance I obtain when distinguishing them from the rest of the learned primitives. I then aim to find the lowest granularity level with high classification performance. My insight comes from the observation that high granularity levels can provide us with good classification performance when distinguished from the rest of the primitives, yet they can hide multiple subclasses of primitives that might be relevant to the considered task. Of course, when we reach a level that is too low, the classification performance will drop significantly, and thus this does not represent an appropriate level. The results show that the presented framework successfully finds a well-suited granularity level for the *pick up* primitive within the context of this HRC scenario.

The rest of this chapter is structured as follows. I give an overview of relevant work in Section 5.1. I present our framework in Section 3.2, detail the experimental setup in

Section 3.3, and go into the implementation of our framework in Section 3.4. We present our results in Section 3.5, and conclude with a discussion in Section 3.6.

3.1 Related Work

My work focuses on discovering the appropriate level of granularity for learned primitives from observations of human and object motion data. In this section, I first give a brief overview of the types of primitives high-level planners expect as their inputs. I then present relevant work in the area of action primitive discovery from motion data.

3.1.1 Primitives Expected by High-Level Planners

High-level planners can work with different levels of abstraction. Most planners assume as their input collections of primitives and do not concern themselves with the acquisition of these primitives. Furthermore, such planners do not specify the granularity level expected for the primitives they take in, and include information about this level only indirectly, through the handcrafted specification of the inputs. Different types of representations expected by planners include: action templates represented with preconditions and effects (e.g., "put-block-on-table(x y)" describes the action of moving block x from the top of block y to the table) [73], primitive skills based on actions that the agent can perform (i.e. specifying a start condition, a set of effects, and a set of executable actions) [74], or primitives with manually specified inputs and outputs (e.g., unscrew takes in a stud as input and returns a nut as output) [75]. Other work investigated symbolic descriptions for use in low-level environments for planning [76]. Although this technique goes beyond using handcrafted primitives, it also requires careful specifications of preconditions and effects of atomic components.

3.1.2 Primitive Action Detection

Understanding the motions that make up different tasks requires finding patterns in the data to facilitate the identification and characterization of actions. Thus, there has been considerable effort in the area of learning how to decompose motion data into atomic parts called motion or action primitives.

Methods and Approaches

One class of widely used techniques for the purpose of primitive detection is the Hidden Markov Model (HMM). An example of the use of HMMs within this context is work that learns primitives from observations of human motions by first segmenting the motion via an HMM [77], and then applying another HMM over the found segments to learn the primitives as clusters [77]. Another example uses Gaussian mixture models together with HMMs in order to represent the primitives and the transitions between them, respectively [78]. Yet other work employed parametrized HMMs over clusters of segments detected based on changes in object state rather than looking at the full body motion [79].

Other techniques employed for primitive detection include using Orthogonal Matching Pursuit on basis functions that model time series data [80], propagation networks that work on activities represented as partially ordered intervals [81], spatio-temporal non-linear dimension reduction [82], and automatic methods for segmenting long sequences of data into segments that usefully identify different primitives [83]. Non-negative matrix factorization has also been utilized to perform the primitive detection part and then use the learned representation to predict future motions [84], as well as to predict linguistic labels given as input for every motion data observation [85].

Building on Action Primitives

Work that explores ways of clustering or detecting higher abstractions based on learned primitives include contributions from different fields. In communities like computer vision,

much of the focus is placed on building up from learned primitives to intuitive behaviors that do not necessarily need to feed into higher-level planners, but are directly used by various applications, such as camera surveillance [86], or detection of actions, gestures and expressions [87]. This research makes contributions to building abstractions on top of learned primitives, but does not face the problem of needing to bridge the gap between such abstractions and inputs needed for higher-level planners. Hence, this work tackles the problem of finding primitive granularity levels only indirectly.

In robotics, there are a number of contributions that do look at what abstractions can be built onto learned primitives to tackle tasks at higher levels. This work is more relevant to our aim of finding the granularity of learned primitives. In this area, [88] address defining behavior vocabularies of human motion data for movements that include punching, dancing, handwaving, and so on. [77] look at mapping actions from full body human motion demonstrations of behaviors like raising arms and bending down to similar actions that a humanoid robot can perform. Other work focuses on learning about the structure of a task in a bottom-up approach that starts with automatic segmentation of learning from demonstration (LfD) data and continuing to build useful abstractions on top [89], [90], [91], and [92].

The work of [88], and [77] differs in nature from this work in that I consider different types of primitives that occur over the progression of a task and also take into account object trajectories, while the cited contributions consider actions that are repetitive in nature or look at the full body posture of a person. The LfD contributions cited above are more similar in nature to the attempts in this chapter, but they consider robot motions, whereas I investigate human motion data in the present chapter. Motions generated by humans and robots are inherently different. Thus, even when utilizing the same approaches and techniques for primitive discovery on both types of data, the learned human and robot primitives fit into the task differently.

Furthermore, this contribution focuses on a framework for discovering primitive gran-

ularity levels that are well-suited for the considered task. This work brings attention to the possible hidden structure that might exist in choosing an appropriate granularity level, even when obtaining good performance on classification tasks for the desired primitives.

3.2 Framework for Discovering Primitive Granularity Levels

In order to use high-level planners that expect well-defined primitives as their inputs, it is important to discover those primitives that (i) help capture the core patterns in the actions part of the task, and (ii) constitute the main building blocks of the observed motion without hiding further subclasses within. If the level we employ cannot capture the main differences we observe in the data, our reliable identification of primitives can be compromised. This is bound to happen with the choice of too low of a granularity level, when this level is not sufficient to capture the common basic actions present in the task at hand. For example, if we choose a low primitive level such as *move cup to mouth*, this might prove problematic due to the different ways in which a person can move their hand with the cup to their mouth. On the other hand, if we choose too high of a level, this can end up obscuring deeper structure present in the primitives. This can prevent us from generalizing well from the data and from reusing learned primitives in an effective way when we build higher-level behaviors. An illustrative example here is *brush teeth*, where we might benefit from learning lower-level primitives like *grab toothbrush*, *add toothpaste*, and *execute brushing motion*. These primitives can then be used for higher-level actions, such as *brush teeth* itself, but also for an action like *clean tongue*, that might require the same *grab toothbrush* primitive. My framework provides a method for discovering precisely this appropriate level of granularity for the considered application or scenario.

I introduce the framework for primitive granularity discovery in Framework 1. This outlines the general methodology to be applied for determining appropriate levels of gran-

Framework 1 Discovering learned primitive granularity levels

1: **Data collection and preparation:**
2: Collect raw data from scenario relevant to the considered task, from which primitives are to be learned. Data should be collected in conditions that simulate as best as possible the desired scenario.
3: Divide data set into training and test sets, depending on desired method of validation.
4: **Learn action primitives from raw data:**
5: Identify an appropriate algorithm for learning primitives from the raw data.
6: Apply algorithm from 5 to training set.
7: Apply algorithm from 5 to test set, using the learned parameters from 6.
8: **Classification experiments:**
9: **Classification method:**
10: Identify an appropriate method for classifying time series composed of the learned primitives.
11: Apply method from 10 to training set.
12: **Primitive choice:**
13: Choose what primitive(s) to investigate.
14: Identify factors that engender different granularity levels of primitives found at step 13. These factors rely heavily on the task.
15: **Establish granularity level interval:**
16: Identify the lowest reasonable level, and apply method from 10 using the learned parameters from 11 to test whether good performance is obtained.
17: Identify the highest reasonable level, and apply method from 10 using the learned parameters from 11 to test whether good performance is obtained.
18: **Find final granularity level:**
19: Decrease level and test with lower and lower levels starting from the highest, until a significant decrease in classification performance occurs.
20: Choose the lowest level with high classification performance as well-suited for the task: (i) high performance for high levels can hide different subtypes of primitives relevant to the task; (ii) when the significant decrease in performance occurs, it likely indicates that the level with poor performance is too low, and the one with high performance is appropriate.

ularity for a considered task. In Section 3.3, I introduce the experimental setup employed for the framework, and in Section 3.4, I present the implementation of each framework step.

3.3 Experimental Setup

In this chapter, I consider the task of assembling part of an IKEA chair. I use a children's chair for ease of tracking and moving objects around in the workspace, but I choose one that requires complex assembly, including several small components and the need to use a tool. I consider part of the assembly task, namely attaching the front frame onto the rest of the chair. This part contains similar steps as those required for the rest of the assembly. It is composed of the necessary and sufficient actions for showcasing all the different types of objects and movements that are part of the full chair building process. Fig. 3.2 shows all the components of the task: the front frame (which I refer to as the frame herein), the rest of the chair (which I refer to as the main component herein), two pegs, two nuts, two screws, and a screwdriver. To complete the task, participants need to: (i) place both pegs in either the frame or the main component, (ii) place both nuts, in any order, in the sides of the main component, (iii) attach the frame onto the main component, (iv) place the screws into the two holes of the frame, and (v) twist the screws in.

The motion capture system I employ throughout the data recording sessions is PhaseSpace [93], which utilizes active LED markers as its tracking technology. To acquire motion capture data, I hand fashioned a pair of gloves specifically for participants to be able to manipulate all the different objects during this task. I used fitted, stretching material gloves without the tips of the fingers, which I mounted with eight sensors each for tracking. I also tracked the frame and the main component. Fig. 3.3 presents a screen shot of the motion capture system software, highlighting the gloves and chair components as tracked by PhaseSpace. Each glove is fitted with eight sensors, and thus the two groups of eight sensors

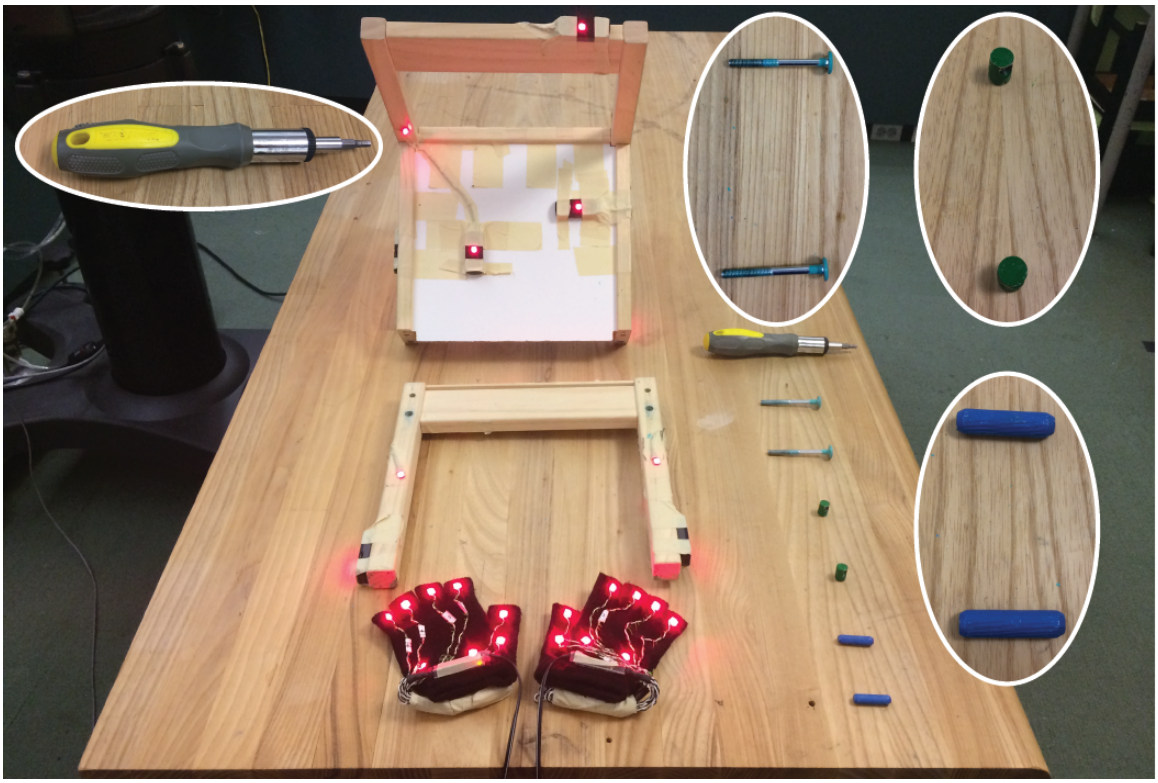


Figure 3.2: The components part of the assembly task: a frame, a main component, two pegs (blue), two nuts (green), two screws (cyan), and a screwdriver (yellow). The figure also includes the gloves participants wear for tracking hand movements.

represent the participant's left and right hand, respectively. For the chair components, we created rigid bodies. A rigid body represents a collection of markers that are fixed relative to each other. This allows for regarding the collection as one tracking target and making the tracking more robust to individual marker occlusions. Fig. 3.3 includes the two rigid bodies created for the frame and the main component, respectively. For each, I utilized the cartesian coordinates of the rigid body center, which allowed effective tracking of each of the two objects as single points in the workspace. I used redundant sensors placed on different planes, to make the tracking robust to occlusions occurring when participants would place the objects on the table in different positions, or cover some of the sensors. I did not create rigid bodies for the gloves, since these markers did not maintain a fixed position relative to each other when participants' hands moved.

I employ this experiment to emulate the complexities of assembly tasks humans en-

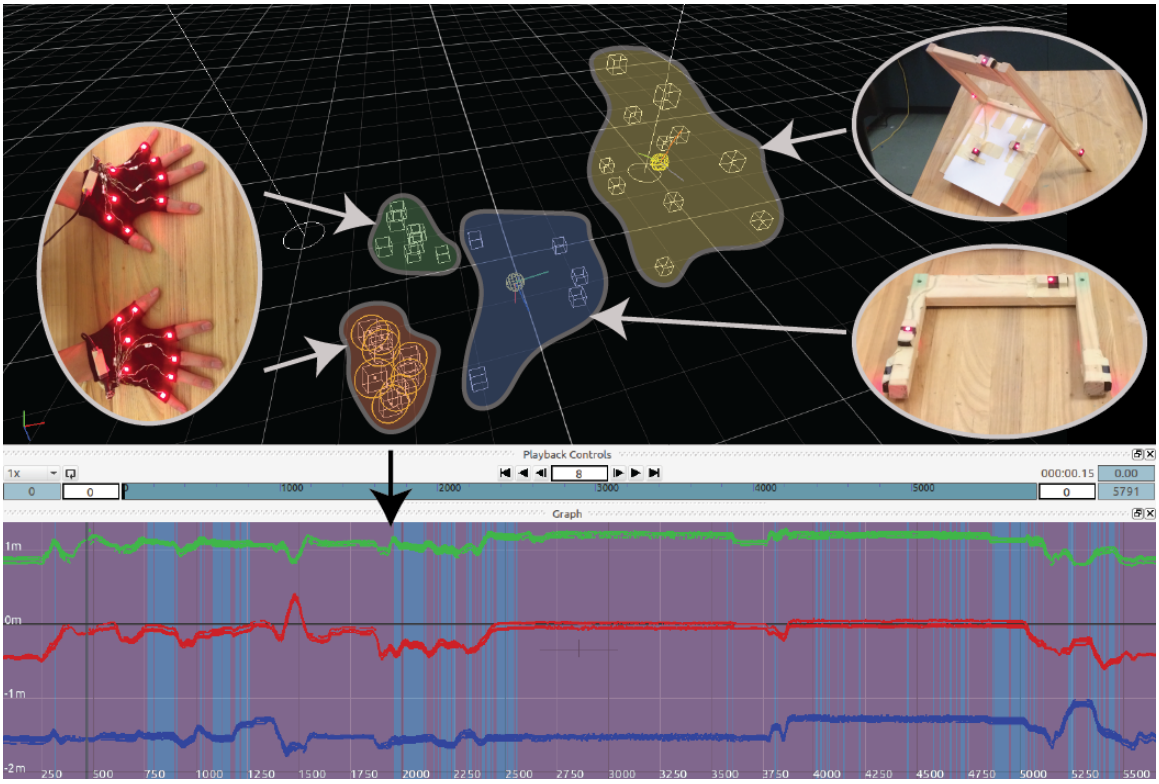


Figure 3.3: Screen shot of the motion capture system software highlighting: (i) green—the group of eight individual markers mounted on the left hand glove, (ii) orange—the group of eight individual markers mounted on the right hand glove, (iii) blue—-the individual markers mounted on the frame, together with the center of the frame rigid body, and (iv) yellow—the individual markers mounted on the main component, together with the center of the main component rigid body. The figure shows the individual markers visible at the time of the screen capture, with the two rigid body centers still being tracked even with marker occlusions. The figure also includes the (x, y, z) coordinates (red, green, and blue, respectively) of the right hand markers recorded during an assembly session. The x, y, and z signals show the values for all eight markers, highlighting the high correlation present for markers on the same glove.

gage in. I impose no restrictions on participants about how to perform the assembly. This provides me with a thorough understanding of what different levels of granularity we need to consider for primitives relevant to intricate HRC tasks. For such tasks, even primitives as intuitive sounding as *pick up* are not trivially defined. Given that we employ different types of objects, each with a different result on the motion data signal, it is not immediately apparent what level of granularity to use for such a primitive. In this scenario, *pick up* can be applied to a peg, a nut, a screw, the screwdriver, the frame, or the main component. The granularity level here refers to what types of object *pick up* I label as positive examples of this class of primitive. A low granularity level for our task would involve distinguishing between different small objects, like pegs, nuts, screws, and the screwdriver, while a high level would involve treating pick ups for all the small and big objects as positive examples of this class.

3.4 Framework Implementation for HRC Scenario

Here, I start with an outline our data collection phase, and then detail how action primitives are learned via a beta process hidden Markov model (BP-HMM). Finally, I describe the classification experiments for discovering a well-suited level of granularity, investigating the *pick up* primitive, in particular.

3.4.1 Data Collection and Preparation

I present the data collection phase that corresponds to lines 1–3 in Framework 1. To implement line 2, I recorded data from a total of 28 participants partly assembling a children's IKEA chair (mounting the frame onto the main component of the chair). I recorded each assembly as a different trial, with five-to-ten trials per participant, for a total of 158 trials across the data set. Trials varied in length, from $57''$ to $4'21''$, with an average length of $2'10''$. The motion capture system recorded at a rate of approximately $1.25\,\mathrm{Hz}$.

During each trial, I recorded motion capture data for each of the 16 markers mounted on the gloves (eight markers on each glove) and all the markers mounted on the rigid bodies. I considered (x, y, z) cartesian coordinates for the glove markers and the centers of the two rigid bodies. For each glove, I averaged over the eight values corresponding to the eight markers each was fitted with, resulting in an average (x, y, z) signal for each hand. I did so given the fact that the information in the signals from the different markers was redundant. This provided a good representation for each hand. This choice is supported by a principal component analysis (PCA) performed on the signals of the eight markers for the right hand, and similarly, on those of the eight markers for the left hand. For each hand, I performed PCA three times, for the x, y, and z signals. I included the average of the eight markers for the dimension in question, resulting in nine total variables for each dimension. For each dimension, the PCA found that the principal component composed of all nine variables approximately equally weighted accounted for at least 89% of the variance. This motivates the choice of representing each dimension with the average of the eight markers for one hand, showing the nine variables within one dimension are highly correlated.

To implement line 3 of the framework, I divided our data into a training set and a test set. Out of a total of 158 time sequences, I leave out a total of 28, one per person. I apply the action primitive algorithm on the training set, and then apply the estimated parameters of the model to the 28 held-out sequences to mimic learning primitives in real-time, after the model parameters have been learned. The primitive classification is applied in a similar manner, training the classifier on the training set, and using the learned parameters to mimic classifying on real-time data. I note here that each held-out trial per participant was randomly selected, and that each was segmented in accordance with the description in Subsection 3.4.3. This resulted in approximately 25 windows per each of the 28 held-out sequences. The results that I present in Section 3.5 constitute test results across the different 28 participants, supporting a strong generalization.

Table 3.1: Primitive Granularity Levels and Labels for Classification Experiments

	$granularity\ level = low$		$granularity\ level = intermediate$	$granularity\ level = high$	
	$2\ classes$	$3\ classes$	$2\ classes$	$2\ classes$	$3\ classes$
$label\ 1$	screws	screws	small objects	small and big objects	small objects
$label\ 2$	—	nuts	—	—	big objects
$label\ 0$	rest	rest	rest	rest	rest

[1] I employ 2-class and 3-class experiments to analyze at what granularity level we can distinguish between primitives labeled differently.
[2] $Label\ 1$ and $label\ 2$ (when it exists), refer to primitives at different granularity levels. A bar (—) appears under 2-class experiments, and indicates a missing label.
[3] $Label\ 0$ always refers to the "rest" of the primitives, i.e. all primitives that have not been labeled with 1 or 2.
[4] "Small objects" include the pegs, the nuts, the screw, and the screwdriver. "Big objects" include the frame, and the main component.

3.4.2 Learning Action Primitives from Raw Data

The approach I use to implement lines 4–7 from Framework 1 is based on the beta process hidden Markov model (BP-HMM) presented by [10] and [72]. I used the implementation made available online by [94]. This method uses a non-parametric Bayesian approach based on a stochastic process—the Beta Process (BP) [95]—to learn behaviors (primitives) that are shared among several related time series. Such Bayesian nonparametric techniques provide us with the advantage of learning from sequential data that does not require knowing the number of hidden states in advance. I pick the BP-HMM algorithm presented herein (Framework 1, line 5) due to its capability of handling multiple times series and learning a set of primitives shared among them, with tractable analysis of hundreds of series. Below, I present the algorithm, as described by [72].

In this case, a time series represents a recording of an assembly session. The i^{th} series is assigned a sparse binary vector $f_i = [f_{i1}, f_{i2}, ...]$, that indicates the presence or absence of each feature in the set of shared primitives. For N time series, matrix $F = [f_1; ...; f_i; ...; f_N]$ contains the binary features of all the time series, and is generated by the BP:

$$B|B_0, \gamma, \beta \sim BP(\beta, \gamma B_0), B = \sum_{k=1}^{\infty} b_k \delta_{\gamma_k} \tag{3.1}$$

Realization B of the BP contains potentially infinitely many features k. For each feature, $\theta_k \sim B_0$ represents the model parameters, and $b_k \in (0, 1)$ represents the feature's

inclusion probability in the respective i^{th} series. The binary feature vector for the i^{th} series is obtained by independent Bernoully draws $f_{ik} \sim Ber(b_k)$. Parameter γ determines the Poissson(γ) distribution that represents the number of active features in series i. Parameter β dictates the frequency of sharing features between series.

The BP is then combined with an HMM to form the BP-HMM. The binary vector f_i determines a finite set of primitives available for the i^{th} series. A single primitive $z_{it} = k$ from the set $\{k : f_{ik} = 1\}$ is then assigned to each time step t, determining parameters θ_k that generate the data value x_{it} at that time step.

Given that only one primitive is active at each time step, I applied the BP-HMM algorithm on three different signals composing our task: one representing the person's left hand (i.e. the (x, y, z) signal averaged over the eight markers mounted on the left-hand glove), one representing the right hand (similar to the left hand), and one representing the tracked objects (to more easily scale up to tracking several objects throughout a task, I learn primitives based on the signals provided by all the task components). Thus, I obtained three sets of primitives, corresponding to the left hand, right hand, and objects, respectively. Prior to applying the algorithm on the three different sets of time series, we normalized the data within each set to have mean zero and unit variance. To implement lines 6 and 7 from Framework 1, we applied the BP-HMM algorithm on the training set to learn the model parameters, and then used these parameters when applying the algorithm to the test set. The training and test sets are as defined in Subsection 3.4.1.

3.4.3 Classification Experiments

In this subsection, we detail the implementation of Framework 1, lines 8–20.

Classification method

The classification method I employ for the time series consisting of learned primitives is the k-Nearest Neighbors (k-NN) algorithm [96], with dynamic time warping (DTW) [97]

as the time series similarity metric. This constitutes the implementation of Framework 1, line 10. Although DTW is known to be computationally expensive, it is considered one of the best measures for use with time series across application domains [98] and has also been shown to be able to run quickly even on very large data sets [99]. Classifying time sequences represents a non-trivial task, and the choice of similarity metric in this context is particularly known to be an open research question [100]. The choice of the current classification method is based on the fact that DTW has proven a successful metric for measuring time series similarity, especially when used together with the k-NN algorithm [99]. The analyses in this chapter have also shown that, for the current learned primitive representation under the form of time series, other classification methods, such as support vector machines and multinomial logistic regression, have proved ineffective.

To apply the algorithm, I divided each time sequence into non-overlapping windows of fixed size 400 (approximately 5 seconds), starting from the beginning of each time series, and normalized all the data to have mean zero and unit variance. For cases when the series' length did not equally divide into the window length, I padded the sequence with the last encountered value. Since I applied the BP-HMM algorithm for primitive action discovery three times for each of the right hand, left hand, and the objects, this resulted in three time series per trial, each represented as primitive sequences. For each window of size 400, I concatenate the three primitive sequences corresponding to the right hand (RH), left hand (LH), and objects (Obj), respectively. This results in a sequence of length 1200 per window, with format $RH + LH + Obj$. Next, I implement Framework 1, line 11. Since k-NN is a lazy algorithm (i.e. does not do any generalization using the training data), I create a set consisting of all the 1200-length windows obtained from segmenting the learned primitive training data set, ready to use for the testing phase.

Primitive choice

To implement Framework 1, line 13, I choose to explore *pick up*, since it represents one of the most prevalent primitives in robotics. In addition, *pick up* presents strong relevance to HRC scenarios such as the one in this chapter. Both humans and robots need to frequently execute *pick up* actions during the assembly of the chair, and need to do so for different kinds of objects.

In order to represent the *pick up* primitive, I consider the factors that engender different types of pick ups for our task (Framework 1, line 14). I thus turn my attention towards the types of objects part of the chair assembly. As described in Section 3.3, there are four different types of small objects: pegs, nuts, screws, and a screwdriver, and two bigger objects: the frame and the main component. Looking at these objects, a low granularity would involve differentiating between *peg pick up*, *nut pick up*, *screw pick up*, and *screwdriver pick up*. This means treating each type of *pick up* as a separate class. A high level of granularity would involve simply distinguishing that an object is being picked up, whether small or large. These represent the established lowest and highest reasonable levels of granularity, as appears in Framework 1, lines 15–17. An intermediate level would be placed in between, distinguishing between *small object pick up* and *big object pick up*.

Find final granularity level

As stated in Framework 1, lines 16 to 17, I start with the low and high granularity levels, and use classification performance to gauge if they represent reasonable minimum and maximum levels for the task at hand. I then implement Framework 1, line 19. I work down from the highest level and test an intermediate value. I do so in order to ensure I am not fooled by choosing too high of a level, which might hide multiple types of primitives.

In order to find the lowest level with high performance, per Framework 1, line 20, I perform three main classification experiments and two additional ones. The three main experiments consist of a binary classification task. Each classification aims to distinguish

between the primitives at the considered level of granularity and the rest of the primitives present in the data set. I segment each time sequence into windows of 400 time steps. I then label the windows that include the considered *pick up* primitives as the positive class (labeled as 1), and the rest of the windows as the negative class (labeled as 0). I perform two additional experiments to better interpret the results at the intermediate and high levels, and to test exactly what types of *pick up* primitives we can distinguish between. These experiments are performed using three classes, with windows that include the first type of *pick up* primitive labeled as 1, the second type labeled as 2, and the rest labeled as 0. The different levels are presented in Table 3.1.

When applying the k-NN algorithm to the classification experiments described above, I employed a k value of 3 for the binary classification tasks and a value of 5 for the 3-class tasks. For the 3-class tasks, ties can appear when two class labels occur an equal number of times. In cases of ties, I decrease k until no tie is present.

3.5 Results

I present the results of implementing our framework for the considered HRC scenario. Subsection 3.5.1 details the results of applying the primitive learning algorithm, and Subsection 3.5.2 presents the discovery of the granularity level appropriate for our task. I present an additional classification of participants in Subsection 3.5.3, performed on the learned primitives to showcase the utility of this representation.

3.5.1 BP-HMM Algorithm Results

Fig. 3.4 shows an example of the learned primitives for a time sequences representing a single trial. The results highlight two sets of primitives: right hand (resulted in 55 global primitives), and objects (resulted in 86 global primitives). The primitives resulted for the left hand are not visualized so as not to overload the graph. For the left hand, we obtained

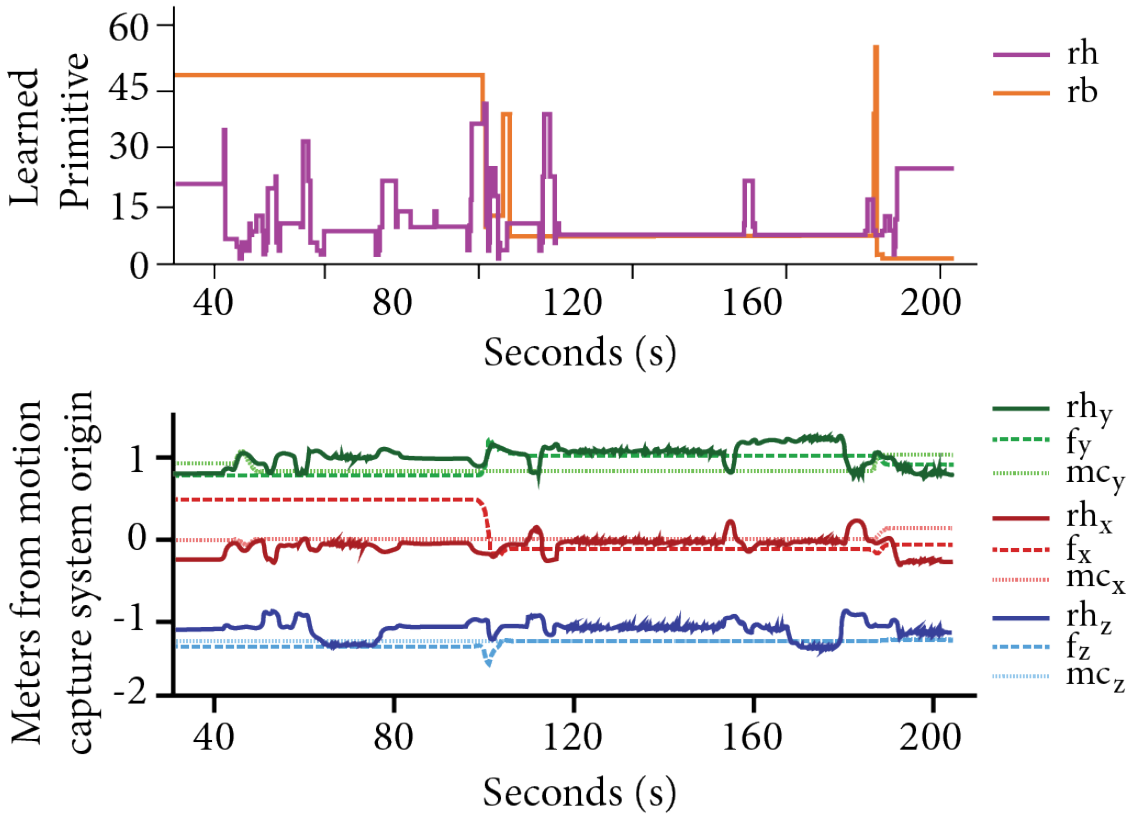


Figure 3.4: A time series from an assembly session including the raw motion capture data in the bottom subfigure, and the learned primitives in the upper subfigure. Here, I omitted the left hand signals for ease of visualization. The bottom subfigure presents the (x, y, z) coordinates for each of the: (i) right hand (*rh*), (ii) frame (*f*), and (iii) main component (*mc*). The x, y, and z values are represented in red, green, and blue, respectively. The coordinates represent displacement in meters from the motion capture system origin, set up in the center of the data collection room. The upper subfigure presents the learned primitives for the: (i) right hand (*rh*) in magenta, and (ii) the two rigid bodies (*rb*) in orange. I applied the primitive learning algorithm on the f and mc data together, resulting in a single set of primitives for the two rigid bodies.

51 global primitives. The fact that I obtained a similar number of primitives for the right and left hand indicates that the representation was able to pick up on similar types of movements a person can perform with the right and left hand. I posit that the number of primitives detected for the right hand is slightly higher than that detected for the left hand due to the fact that most people used their right hand far more often than they did their left hand (picking up the small objects placed to their right in the workspace, and even picking up bigger objects that were placed to the left of the small items).

3.5.2 Granularity Discovery Results

I evaluate the performance of the classification experiments by looking at **precision**, **recall** and **F1 score**. Table 3.2 presents the results of the experiments, and provides the definitions of these metrics. The table shows the results for the low, intermediate, and high granularity levels, including the 2-class and, where present, for the 3-class experiments.

Since I train the classifiers with unbalanced classes (i.e. positive labels occur much less frequently than negative ones), I do not place too much emphasis on the high scores for the negative labels. I present them in the table for completeness. The first, 2-class, low granularity level classification task provided poor results, with an **F1 score** of 0.31. This task represents training the classifier with only screw pick ups as positive labels. This suggests this granularity level is too low for our task, since there are many similar small-sized objects. The intermediate level classifier provides good performance, with an **F1 score** of 0.82. This level represents pick ups for similar-sized objects (i.e. pegs, nuts, screws, and screwdriver). The high granularity classifier also provides a good **F1 score**, of 0.81. However, since I employed a binary classification scheme, a similar result to that of the intermediate classifier might signify several subclasses are hiding under the positive label for the more general *pick up* action. In this high granularity case, I train the classifier with both small- and big-sized objects labeled as positive examples.

Since I expect the intermediate classifier to perform significantly worse if a high granularity would be appropriate for this task, I believe the high level classifier is now hiding two subclasses, namely *pick up* for small objects and *pick up* for big objects. To test this, I perform two additional classification tasks, employing three classes instead of two. The low level 3-class task labels screws with 1, nuts with 2, and the rest with 0. The high level 3-class task labels small objects with 1, big objects with 2, and the rest with 0. I obtain poor performance for the former (**F1** scores of 0.40 and 0.46 for classes screws and nuts, respectively) and good performance for the latter (**F1** scores of 0.78 and 0.82 for classes small objects and big objects, respectively). This indicates that, in the case of the low level

Table 3.2: Results of Classification Experiments

	$granularity\ level = low$		$granularity\ level = intermediate$	$granularity\ level = high$	
	$2\ classes$	$3\ classes$	$2\ classes$	$2\ classes$	$3\ classes$
Precision					
$label\ 1$	0.33	0.50	0.81	0.81	0.78
$label\ 2$	—	0.40	—	—	0.82
$label\ 0$	0.93	0.95	0.94	0.93	0.98
Recall					
$label\ 1$	0.29	0.40	0.84	0.81	0.88
$label\ 2$	—	0.43	—	—	0.75
$label\ 0$	0.94	1	0.93	0.81	0.95
F1 score					
$label\ 1$	0.31	0.40	0.82	0.81	0.78
$label\ 2$	—	0.46	—	—	0.82
$label\ 0$	0.93	0.98	0.93	0.93	0.97

[1] **Precision** is defined in the standard way, as $true\_positives/(true\_positives + false\_positives)$
[2] **Recall** is defined in the standard way, as $true\_positives/(true\_positives + false\_negatives)$
[3] **F1 score** is defined in the standard way, as $2 \times (\mathbf{precision} \times \mathbf{recall})/(\mathbf{precision} + \mathbf{recall})$
[4] A bar (—) appears under 2-class experiments, and indicates an absent value for the missing label.

classifier, it is not possible to differentiate between different types of small objects, and so the classifier cannot distinguish between labels 1 and 2. In the case of the high level classifier, it is indeed possible to differentiate between the pick ups for small objects and those for big ones.

These results show strong support for the intermediate level classifier having the appropriate level of granularity for the *pick up* primitive in the current scenario. This highlights the importance of evaluating different levels of primitive granularity in order to find the best suited one for the task at hand.

3.5.3 Participant Classification Results

As a complement to the classification tasks performed on windows resulted from segmenting the data sequences, I also performed a classification task per person in order to show the primitive action representation is also useful at a higher level. I applied the same k-NN with DTW algorithm described in Subsection 3.4.3, but this time by using the full sequences. I concatenated the sequences in a similar manner as above, with format RH_LH_Obj, where RH, LH, and Obj.

I divided the data in the same fashion, leaving out one time series per participant (ending up with 28 series in the test set) and aiming to classify participants correctly. The algorithm classifies 21 out of 28 participants correctly, giving us a percentage accuracy of 75%. I do not present usual $\mathbf{precision}$, $\mathbf{recall}$, and $\mathbf{F1\ score}$ metrics for this classifier since they are ill-defined for classes expecting a single example, which is missing for incorrectly classified participants.

3.6 Conclusions

In this chapter, I presented a framework for discovering the appropriate level of primitive granularity for the task at hand. I focused on an HRC scenario, for which learning from human demonstrations is important to develop adaptive robots. I recorded a motion capture data set of human and object motions during an IKEA chair assembly task, and used a BP-HMM algorithm to learn primitives. I presented our framework for discovering a well-suited level of granularity for the considered task, and applied it to *pick up*, a prevalent primitive in robotics, relevant to the HRC application presented here.

The results show that the framework is able to successfully discover the appropriate level of primitive granularity for the considered task. This method highlights that even when obtaining high classification performance, high granularities are not necessarily the best suited for the task at hand, as I uncover hidden structure in the data at this level. In this scenario, this occurs because it is not immediately obvious that picking up different types of objects should be regarded as different primitives. When I label different-sized object *pick up* primitives as all pertaining to the positive class of examples, we are mistakenly lead to believe this is a well-suited level of granularity. Such a decision can initially be made based on intuitive descriptions found in the expected inputs for high-level planners. However, the results in this chapter show that simply following the expected structure of such inputs without ensuring that the granularity level of the primitive is actually well-suited for the

task, can result in overlooking important differences in such primitives.

The framework introduced in this chapter represents a contribution towards providing existing methods in robotics with the types of inputs they expect to function as desired. As a direction of future research, I would like to automate the process of finding reasonable minimum and maximum granularity levels, as well as that of exploring the intermediate levels most likely to prove fruitful. Another worthwhile improvement is investigating how primitives learned from human motion relate to robot primitives for the same action. For example, the *pick up* primitive might have a granularity level better suited for the task if it more closely resembles the type of *pick up* the robot can perform.

Chapter 4

Learning at the High Level: How Can We Learn High-Level Supportive Behaviors? A Model-Based Approach

In this chapter, we assume capabilities to learn from raw data at the low level, which can give the system quality state estimations that can be used to make decisions at the high level about what supportive behaviors the robot should perform throughout the task. These types of capabilities are obtained from techniques such as that presented in Chapter 3. Here, we focus on presenting a model, together with afferent techniques, for HRC that enables the robot to decide what assistance to offer to the human worker in a personalized manner. We utilize states composed of actual features characterizing the task objects in order to run realistic studies on a robot interacting with human participants. These techniques, however, generalize to any state representation, as long as it is discretized in a similar way to the described states in this chapter.

We contribute a model-based model for HRC that is well-suited for situations where there is relatively easy access to a large pool of observations of human workers performing the task throughout which we ultimately wish a robot to provide assistance. We can

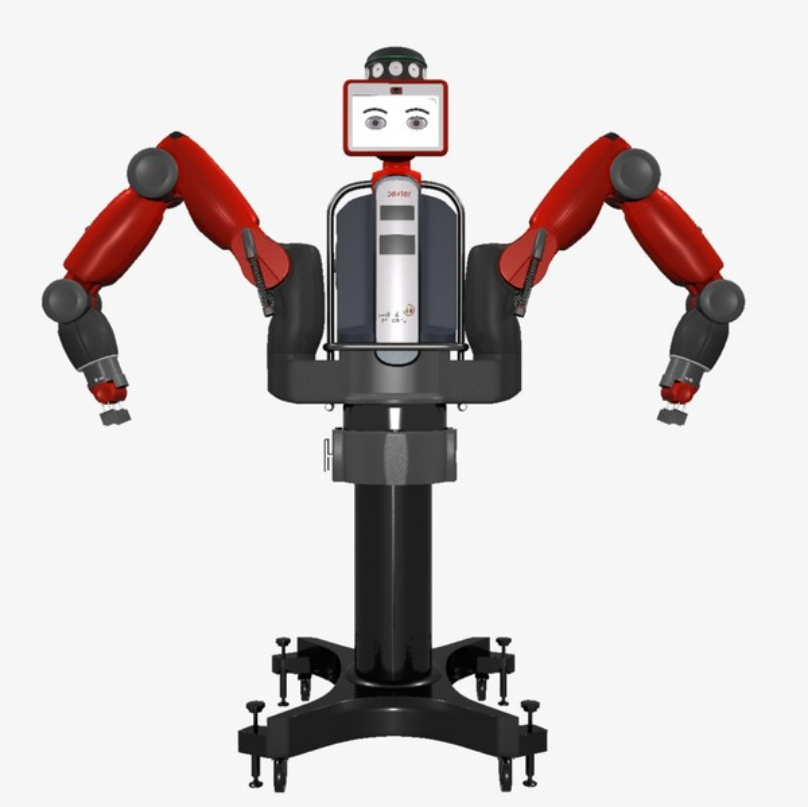


Figure 4.1: The Baxter Research robot used in this thesis for conducting user studies.

envision collecting such a dataset by mounting sensors (such as cameras or motion data capture systems) on the floor of a factory where employees perform their daily routines. This dataset would not contain any information about how the robot should assist the human worker, hence no information about supportive behavior preferences. We then present a model for building personalized models of supportive behaviors with as few as three trajectories either demonstrated by a human via interaction with the robot or labelled by the human.

The robot we utilize for the experiments is the Baxter Research robot (see Figure 4.1), a widely used robot platform 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 worker.

4.1 Learning with Noisy Observations

In this section, we present how to create personalized supportive behavior models for a robot to predict what supportive behaviors to offer throughout the execution of an assembly task. This section includes work performed in simulation, where we tackle noisy observations, and is an extension of the work from [101] [1].

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 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 collecting 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.

Furthermore, people are difficult to model and environments involving human workers are too complex to be fully observed or understood, even with state-of-the-art perception systems. Rather than observing discrete, well identified, human actions, this results in robots observing sets of (often high-dimensional) features of human activities, which are highly domain specific. Modern approaches like partially observable Markov decision processes (POMDPs) [102, 103] require a model of state transitions and rewards for the task. Another class of heavily used methods for learning in robotics that have model-free variants are reinforcement learning techniques [104]. Although this can alleviate the need to

1. 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 Multi-agent Systems (AAMAS), Extended Abstract. Stockholm, Sweden, 2018, July 10–15.

learn the dynamics of the world, model-free reinforcement learning still requires a large training set consistent with different users preferences and their interaction with the agent. Such models are often infeasible to learn in realistic HRC settings since we are severely limited by the number of demonstrations a person can feasibly provide.

To mitigate the need for a large corpora of user-provided training data, we propose a model that separates learning the task model from learning a model of user preferences for the interaction. We thus 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 model not only helps take the burden off users with respect to labeling demonstrations, but it focuses on a core challenge in HRC that does not appear in typical machine learning domains—that of needing to model interactions that agents operating in this environment (be it humans or robots) have or wish to have with the environment and task. In this section, we assume access to observations in the form of task trajectories performed by human workers. For example, this pool of observations can be acquired in a factory, while workers perform their usual routines. Such unsupervised trajectories would not directly include information about the assistance a robot would need to provide. They are however easy to obtain in large quantities without requiring user interactions, and they can be used to learn a model of the worker's behavior while achieving a task.

We employ hidden Markov models (HMMs) to encode information about the patterns observed in the actions taken by the human during task execution. By doing so, this method is able to implicitly, yet flexibly, represent the relevant structure of the task. We then assume access to a small number of user-provided demonstrations that annotate trajectories of the task with labels for the supportive actions the human would like the robot to provide, and we learn a model of the supportive actions with respect to the actions performed by workers, which were observed during training. This allows us to predict what supportive behaviors the robot should provide that are *tailored to an individual worker*. We present results on a realistic HRC task in simulation, achieving predictions with a small number of errors by

using as few as three user-provided demonstrations.

If our robots are to adapt to novel situations, another critical challenge in robot learning is to leverage the knowledge acquired on an initial task and user for generalizing to similar yet unseen tasks, users, and environments. In this work, we explore the question by training the system on an initial task and testing it on variations of this task's structure. I hence demonstrate how the current method is flexible enough to transfer to new scenarios.

The contribution is thus threefold. First, we explain how to learn a model of human behaviors from ambiguous observations and how use it to provide supportive behaviors to human workers throughout the execution of a task. Second, we demonstrate that this model can benefit from a large training set from several users, yet still account for individual preferences in the type of support a user would like to receive, with only a small set of provided labels. Third, we show that the method can perform well on new tasks, after having learned both the task and user preference models on a similar but differently structured task.

4.1.1 Related Work

Robotics can heavily benefit from machine learning in order to develop autonomous and adaptive robots that do not need to be pre-programmed for specific tasks. The typical robotics application, however, fundamentally differs in nature from the standard paradigm assumed in machine learning due to the effects robot actions have on the state of the world and the feedback loop this creates. This work is positioned between approaches that focus on modeling a robot's decision, and work on modeling and predicting human behaviors, presented in Chapter 2.

Compared to the relevant work presented in Chapter 2, the current technique relates in that I am interested in learning a model of the task, for which demonstrations of task trajectories can be acquired by observations in settings like, for example, factories. This allows a large number of demonstrations to be collected so as to allow us to use machine learning techniques such as HMMs to represent the task. However, the models presented

in the related work either need to model the full task (including any interactions with a human or a robot), or need a large amount of demonstrations that are used for model-free learning. In HRC scenarios, user preferences are expensive to obtain, and so we cannot straightforwardly employ either a very specific model, nor a technique that requires large amounts of user-provided labels. Thus, this idea of separating the task learning problem (from an assumed large set of relatively easy-to-obtain demonstrations of task trajectories) from learning user preferences positions our work in an impactful intersection relevant for the HRC application domain.

4.1.2 Problem Formulation

In this chapter, we focus on enabling robots for HRC to predict, in a way that is tailored to individual users' preferences, when this support is needed. To this end, we target a realistic human-robot collaboration scenario where a robot works side-by-side with a human partner to achieve a shared goal, and specifically the collaborative assembly of furniture. In this work, we employ a model set for HRC that has been developed in related work [105]. 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, amount of shared autonomy and so on). For the purposes of this work, we focus on a single assembly task, namely a chair (cf. Figure 4.2).

Current robotic platforms are limited in their perception capabilities, specially when considering realistic human working environments, and the fact that human workers' intentions are hidden, complex, and hence often not fully predictable. To account for this, we consider that the robot only has access to noisy ambiguous observations about the human activities. Such observations might directly come from sensors such as motion capture systems, or be the result of an algorithm that, for example, detects the presence of known objects in the workspace, or whether a particular tool is currently in use. In this chapter, we work under the generic assumption that a vector of features implicitly representing the

Figure 4.2: Chair assembly task utilized throughout Chapter 4 and Chapter 5. This task is representative of a typical Human-Robot Collaboration (HRC) scenario.

world is available to the robot. More specifically, we do not rely on domain- or hardware-specific perception; rather, we propose to exploit generic models on top of this potentially noisy and partial amount of low-level data. We thus target models that can scale to high-dimensional and noisy observations.

In order for the assistance provided by the robot to be useful to an individual user, the *first aim* is for the robot to learn how to do so in a personalized manner. One human worker might prefer the robot to *hold* each leg dowel for him or her to perform the *assemble* action for that leg (depicted in table 4.3). A different user might instead wish the robot to only provide this supportive behavior for a single leg instead of all legs. In this work, we

Table 4.1: Supportive Behavior Labels Provided by User 1 for Example Trajectories

Trajectory 1		Trajectory 2	
Human action	Supportive behavior labels	Human action	Supportive behavior labels
gather parts leg 3	[bring back brackets, bring dowel, bring screwdriver]	gather parts leg 2	[bring front brackets, bring dowel, bring screwdriver]
assemble leg 3	[hold]	assemble leg 2	[hold]
gather parts leg 4	[bring back brackets, bring dowel]	gather parts leg 1	[bring front brackets, bring dowel]
assemble leg 4	[hold]	assemble leg 1	[hold]
gather parts leg 2	[bring front brackets, bring dowel]	gather parts leg 4	[bring back brackets, bring dowel]
assemble leg 2	[hold]	assemble leg 4	[hold]
gather parts leg 1	[bring front brackets, bring dowel]	gather parts leg 3	[bring back brackets, bring dowel]
assemble leg 1	[hold]	assemble leg 3	[hold]
gather parts seat	[bring seat]	gather parts back	[bring back]
assemble seat	Ø	assemble back	Ø
gather parts back	[bring back]	gather parts seat	[bring seat]
assemble back	Ø	assemble sear	Ø

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 chapter are obtained from as few as three demonstrations.

Smooth collaboration requires that the robot supportive actions are provided when the human needs them. This however requires anticipation since actions and, specially, robot actions take time to happen. For instance, if the robot is to be truly helpful, it is essential to bring a screwdriver to a worker who is about to screw before actually reaching the point in the task when the screwing happens. In the following, we focus on the objective of predicting the human needs with an anticipation time of Δt.

The *second aim* with this work is—not only to devise a model that can be effectively used for supportive behavior prediction in HRC—but to also be able to make use of the knowledge learned during one task for different, related yet unseen tasks. This is in line with much of the aspirations in robot learning, where we wish to develop techniques that are flexible and versatile enough to be used for transfer learning. This way, we can leverage a large corpora of training samples from different subcomponents of a task, and utilize them for other tasks that might be composed of the same building blocks, but arranged

Table 4.2: Supportive Behavior Labels Provided by User 2 for Example Trajectories

Trajectory 1		Trajectory 2	
Human action	Supportive behavior labels	Human action	Supportive behavior labels
gather parts leg 3	[bring dowel]	gather parts leg 2	Ø
assemble leg 3	[hold]	assemble leg 2	[hold]
gather parts leg 4	[bring back brackets, bring dowel]	gather parts leg 1	[bring front brackets, bring dowel]
assemble leg 4	[bring back brackets, hold]	assemble leg 1	[hold]
gather parts leg 2	[bring screwdriver]	gather parts leg 4	[bring back brackets, bring dowel, bring screwdriver]
assemble leg 2	[hold]	assemble leg 4	[hold]
gather parts leg 1	[bring front brackets, bring dowel]	gather parts leg 3	[bring dowel]
assemble leg 1	[hold]	assemble leg 3	[hold]
gather parts seat	[bring seat]	gather parts back	[bring back]
assemble seat	Ø	assemble back	Ø
gather parts back	[bring back]	gather parts seat	[bring seat]
assemble back	Ø	assemble sear	Ø

differently. This challenge has particular importance in HRC since user-provided labels are difficult and expensive to obtain.

4.1.3 Predictive Model

We propose a method of separating the problems of learning a task model from that of learning user preferences for the collaborative interaction. This approach starts by first learning a limited set of activity patterns and transitions between them from observations of human behaviors. In this chapter, we implement this approach with a hidden Markov model (HMM). During the 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 the 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 subsection 4.2.4.

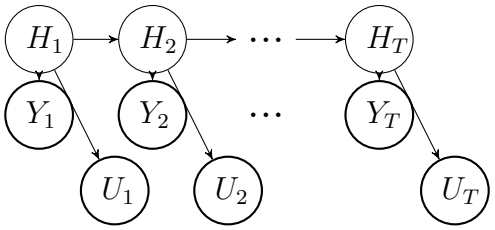


Figure 4.3: The extended HMM representing a personalized supportive behavior model. In addition to a regular model learned from observation of all the users ($(Y_t)_{1\leq t\leq T}$), we consider user preferences, represented by the variables U_t. We assume that the user preferences only depend on the hidden state and that they are only observed on a limited number of examples.

We present the extended HMM in fig. 4.3. 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. As detailed in the following, we consider binary observation features with independent emission probabilities. We hence use a Bernoulli HMM model which we detail in subsection 4.1.3. Similarly, we model the preference for each behavior, given a hidden state, to be independent and modeled as a Bernoulli distribution, as explained in subsection 4.1.3.

We assume access to a training set consisting of observations, each in the form of a vector of features. In the following, 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 $y_t \in \{0,1\}^n$. For example, the feature vector $[0,1,1,0,1]$ could represent [*human worker right hand moving object*, *human worker right hand moving without object*, *human worker left hand moving object*, *human worker left hand moving without object*, *human worker screwing*].

Main Task HMM

To represent the task in a simple yet powerful way, we model the 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, 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 $y_t \in \{0,1\}^n$. For example, the feature vector $[0,1,1,0,1]$ could represent [*human worker right hand moving object*, *human worker right hand moving without object*, *human worker left hand moving object*, *human worker left hand moving without object*, *human worker screwing*]. We train the model with the Baum-Welch algorithm, implemented on top of the *Python hmm_learn* library[2]. Baum-Welch is the standard HMM algorithm for learning the model parameters (i.e., in this case, the Bernoulli probabilities and state transition probabilities). In Fig. 4.3, 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.

Personalized Supportive Behavior Model

We represent the 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 the predictions rely 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

2. The API is available at: http://hmmlearn.readthedocs.io

learned when we proceed to train the preference model. Each labelled 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, for n trajectories:

$$P(U_t = u|H_t = k) = \frac{\sum_{i=1}^{n}\sum_{t=1}^{T} P(H_t^i = k|y_{1\ldots T}^i) \times \mathbb{1}_{u_t^i=u}}{\sum_{i=1}^{n}\sum_{t=1}^{T} P(H_t^i = k|y_{1\ldots T}^i)}. \tag{4.1}$$

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:

$$P_{H_{t+\Delta t}|H_t} = P_{H_{t+1}|H_t}^{\Delta t} \tag{4.2}$$

and hence

$$P(h_{t+\Delta t}|y_{1\ldots t}) = \sum_{h_t} P(h_{t+\Delta t}|h_t)P(h_t|y_{1\ldots t}) \tag{4.3}$$

$$P(u_{t+\Delta t}|y_{1\ldots t}) = \sum_{h_{t+\Delta t}} P(u_{t+\Delta t}|h_{t+\Delta t})P(h_{t+\Delta t}|y_{1\ldots t}). \tag{4.4}$$

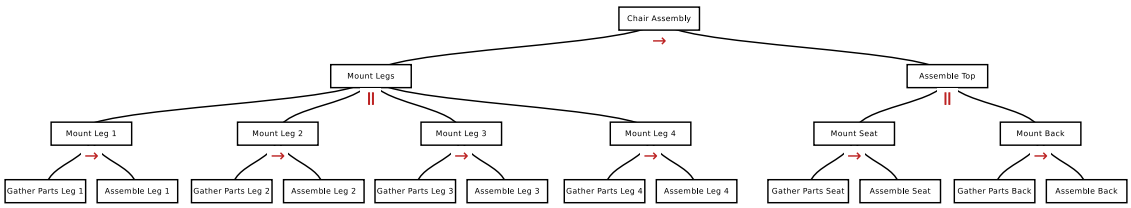


Figure 4.4: Hierarchical task model (HTM) representing the chair assembly task. Each leaf is an atomic subtask; each node composes subtasks that need to be achieved in sequence (in a given order, $\rightarrow$) or in parallel (in any order, $\|$). For each atomic subtask, the human worker has preferences over supportive behaviors that the robot might provide.

4.1.4 Model Evaluation

To show the versatility and applicability of the current model to a realistic collaborative chair assembly task, we conducted a series of simulation experiments for different sets of user preferences, as well as for transfer learning tasks.

Representing the task

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 [67]. HTMs following the definition herein 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. The current 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 Figure 4.4.

Generating the training set for the task

The experiments presented here consist of generating trajectories according to the presented HTM, which form the training set. In particular, given the HTM, we generate valid trajectories that are composed of leaves from the tree that are actually subtasks. We further assume that each leaf corresponds to a human activity, which we model as a distribution of patterns of observations. We simulate observations from the trajectory according to this distribution (the Y_t variables).

In particular, to simulate the noisy process of collecting partial observations in the real world, we generate a vector of $n = 5$ observation features for a given subtask based on a signature of probability values. For example, for the subtask *assemble leg 1*, we randomly generate a vector $[p_1, p_2, p_3, p_4, p_5]$, where $p_j \in [0; 1]$ and represents the probability of feature y_t^j having value 1. We then generate the actual observations, as binary vectors, drawing from Bernoulli distributions whose parameters are given in the vector. Whenever the same subtask appears, it will have the same signature of probabilities, but might have different values for its features depending on this generation process.

Generating the training set for user preferences

We assume access to a set of supportive behavior labels different users provide for a small number of generated task trajectories. We model two different users, with their own likes for what assistance the robot should provide for different actions the users perform. Table 4.3 shows the labels provided by *user 1* and table 4.2 those provided by *user 2*. The first set constitutes an easier learning task for the user preferences, while the second one represents a more difficult learning task. We discuss this aspect in subsection 4.2.4.

Since in a real-world scenario, these preferences are obtained by asking users to manually label a set of trajectories, we assume no noise when generating the labels. Thus, for the human action *gather parts for leg 3*, we will always have the same labels {*bring back brackets*, *bring dowel*} for the supportive behaviors that *user 1* would like the robot to offer.

To choose trajectories for labelling, we pick the number of trajectories to be labelled at random from the training set generated following the description in subsection 4.1.4. We then label them by the scheme in table 4.3 for *user 1* and that in table 4.2 for *user 2*.

Decision Model and Computing Errors

We use a model of an agent that only takes one supportive action at each time step based on the prediction of supportive behaviors for the next step, derived as explained in subsection 4.1.3, with $\Delta t = 1$. At each time step, the agent chooses the supportive behavior with the highest predicted probability as the supportive behavior to be executed.

To evaluate the agent, we generate a test set by using the same techniques described for the training sets, with user annotation unobserved from the agent but used as ground truth for its evaluation. Each different run, we choose a new trajectory at random from the test set, and we compute the hidden state estimation from past observations. We count the agent's choice as an error if the ground truth for the chosen test trajectory is not labeled 1 for this supportive behavior. In other words, the agent is evaluated on its ability to provide a relevant supportive behavior (evaluated based on this particular user's preferences) at each time step.

4.1.5 Results and Discussion

In this section, we present results on the main task for two different sets of user preferences, and on two different transfer learning scenarios.

Prediction Results on Chair Assembly Task

We present the results for the main task in Figure 4.5. This graph highlights the number of errors we obtain as a function of the number of hidden states of the HMM, averaged over ten runs for each, for increasing user preference training set sizes. Each run, we re-train the HMM from scratch to make sure we obtain consistent results over time. As we increase the

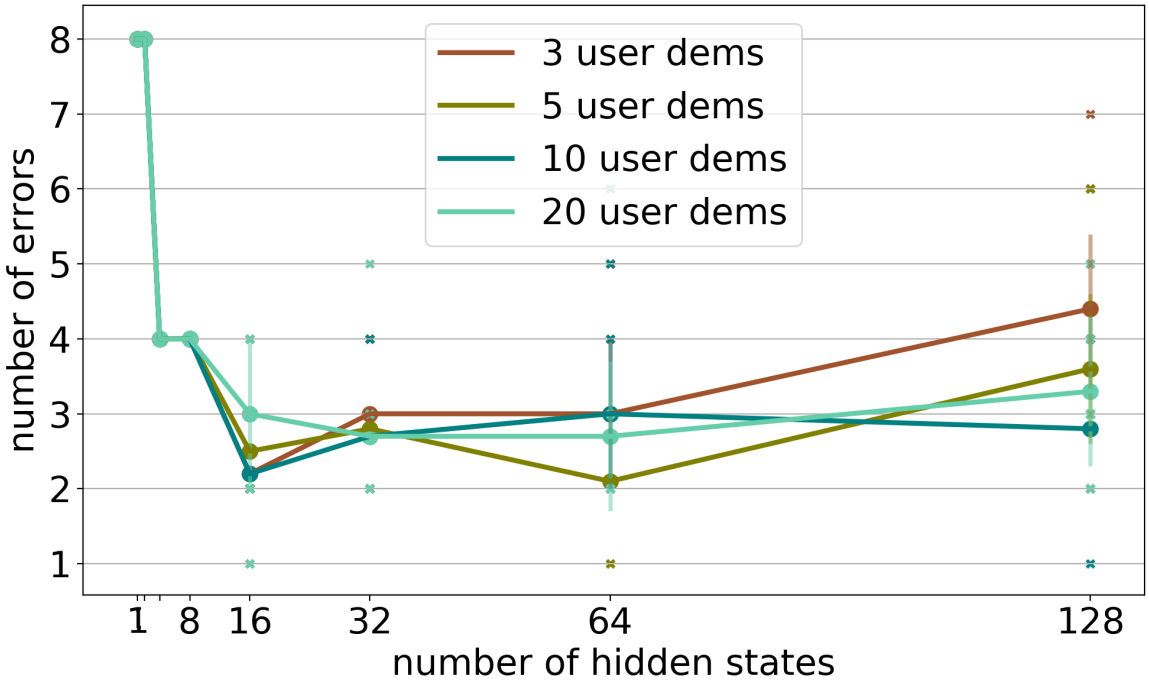


Figure 4.5: Number of errors as a function of the number of HMM hidden states for different user-provided number of labeled trajectories. The number of errors is computed with 500 training trajectories and labels provided by user 1 for 3, 5, 10, and 20 trajectories.

number of hidden states, the power of the model to represent the task increases. However, since we only employ a small set of user-provided labels, a higher number of hidden states effects the need to train more parameters with very few examples provided by users, and so we notice an error increase when maintaining the same number of user demonstrations. We notice this trend for all the different size label sets, testing with 3, 5, 10, and 20 user-labeled trajectories. This highlights a trade-off between the complexity of the model (the higher, the better it can represent the task), and the desired number of trajectories we are comfortable asking the user to label for the task.

In our scenario, this trade-off results in a minimum of 1 and an average number of 2 errors for a task with a total of 12 steps. Of course, this trade-off will differ for specific applications. Due to the nature of the task (i.e., composed of two parallel nodes), a person's lower bound on the number of errors when predicting what supportive behavior the robot should offer at each time step for this set of user preferences would be 1. Although such a

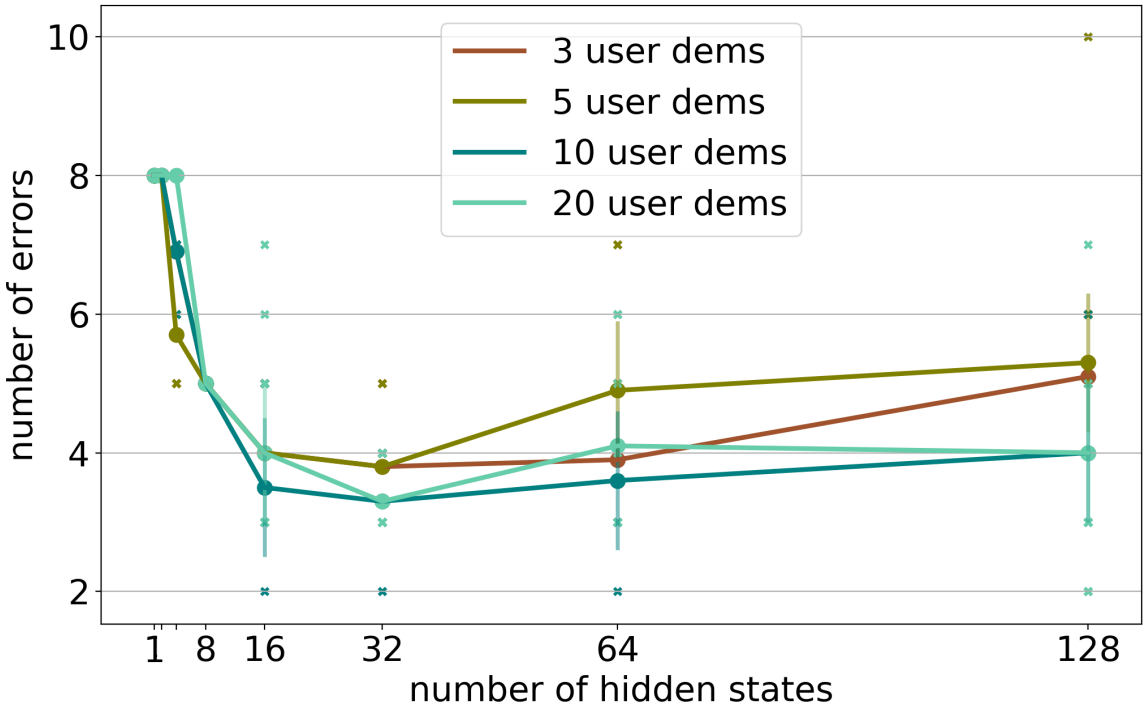
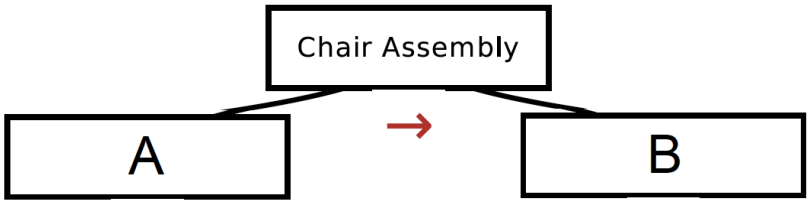


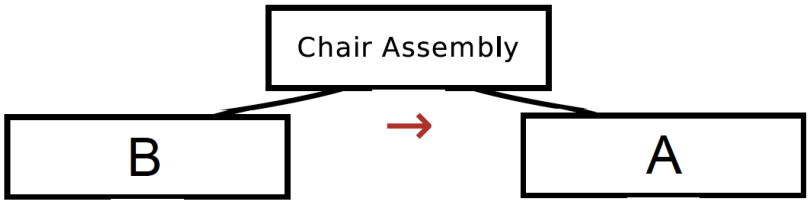
Figure 4.6: Number of errors as a function of the number of HMM hidden states for different user-provided number of labeled trajectories. The number of errors is computed with 500 training trajectories and labels provided by user 2 for 3, 5, 10, and 20 trajectories.

metric would be best thoroughly tested as part of a user study where humans would be asked to predict the supportive behaviors, if we are to assume the best scenario for a person's answers, this would be the accurate prediction for those behaviors that happen at the same time step within the progression of the task. For example, we assume people would be good at predicting *bring dowel* and *hold*, since they always happen at the same point in the task. The *bring seat* and *bring back* behaviors, however, are difficult to predict because of the parallel nature of the *assemble top* task node. Our model thus shows performance close to human level for this case.

Figure 4.6 presents the same analysis, but for a different set of user preferences (i.e., a different user who provided labels for a total of 5 trajectories). The preferences for what supportive actions the robot should offer are listed in table 4.2. Here, the preferences



(a) Task on which we train (*task 0*)



(b) Task composed of the sequence of building blocks A and B, but in reverse order (*task 1*)

(b) Task composed of the parallel combination consisting of building blocks A and B (*task 2*)

Figure 4.7: Structure of transfer learning tasks.

provided by the user represent a more challenging learning task, and so we notice a slightly higher error average, with more variability. In this scenario, it is also not clear what the lower bound on a human performance for predicting supportive behaviors would be. Our results are thus a promising avenue for predicting supportive behaviors in HRC scenarios, where we only have a small set of user-provided labels for task trajectories.

Transfer Learning

Another critical aspect in robot learning is that of transfer learning. Whenever a robot is faced with a new task, we would like it to be able to leverage what it has learned in the past

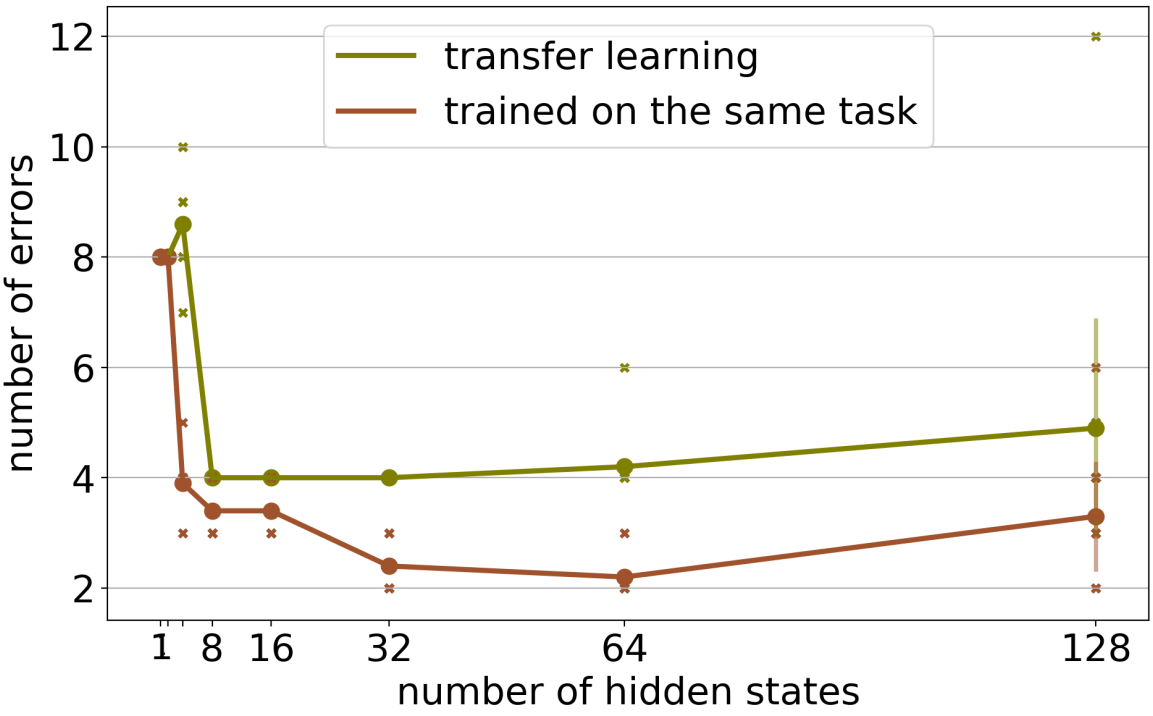


Figure 4.8: *Scenario 1* for transfer learning. This graph shows the number of errors as a function of the number of HMM hidden states when tested on a transfer learning task. Both the model and the user preferences are learned on *taks 0*, and tested on *task 1*.

about similar environments or situations, in order to speed up learning. The alternative is to necessitate to learn from scratch for each new task the robot faces. Transfer learning, however, is difficult to achieve. Typical machine learning techniques need to be considerably tweaked for even the smallest change in the task formulation. We show strong results on two transfer learning tasks. We learn both the task and the user preference models on an initial task, and test on two different tasks, composed of the same building blocks but arranged as different HTMs. Figure 4.7 shows the initial task we train on (which we call *task 0*), as well as the two tasks for which we report error metrics, namely *task 1* and *task 2*. All the metrics presented in this here have been computed with a training set size of 500 and user-provided labels for 5 trajectories in the training set, in accordance to *user 1*'s preferences.

For *task 1*, Figure 4.8 highlights that we obtain as low as 4 errors when trained entirely on the initial task. The baseline we compare against is training on *task 1* and testing on

itself. For our best results, we perform worse than training on the task itself by a difference of only 1 error, which happens in this case for a choice of 16 hidden states for the HMM.

This transfer learning result represents a success for HRC scenarios like the one we consider herein. Figure 4.7 (b) depicts the task we are performing transfer learning on as being composed of the same two building blocks *task 0* consists of, but in reverse order. What this amounts to in a real-world environment like a factory is being able to simply collect observations of particular subtasks of larger, ongoing tasks. We then only require users to label a single combination or way of arranging these subtasks together, and we are able to export this information in a way that serves many other possible tasks. Here, we have a combination of two blocks only, but it is immediately apparent how advantageous this technique can become when considering tens or hundreds of such subtasks. Such transfer learning is difficult to achieve for HRC scenarios, where we require user-provided labels of preferences. A model that would otherwise aim to learn both the task and worker preference models together for an HRC scenario needs to find ingenious ways to handle the scarcity of demonstrations a user can provide.

We also present our findings for a second transfer learning task. For *task 2*, depicted in Figure 4.7 (c), we obtain results highlighted in Figure 4.9. We compose this task from the same building blocks as above, but arrange them as part of a parallel combination. The metrics show that we perform as well as training on the task itself. This transfer learning problem constitutes an easier scenario than the previous one, since *task 2* represents a superset of *task 0*. Our results are still meaningful, however, since we work with so few user-labeled trajectories, and are thus not likely to observe many trajectories outside *task 0*.

4.1.6 Conclusions

In this section, we explore the problem of predicting supportive behaviors for a realistic chair assembly task, and show that we can learn accurate predictions from as few as three user-provided labels. We show there is a trade-off with respect to the representational

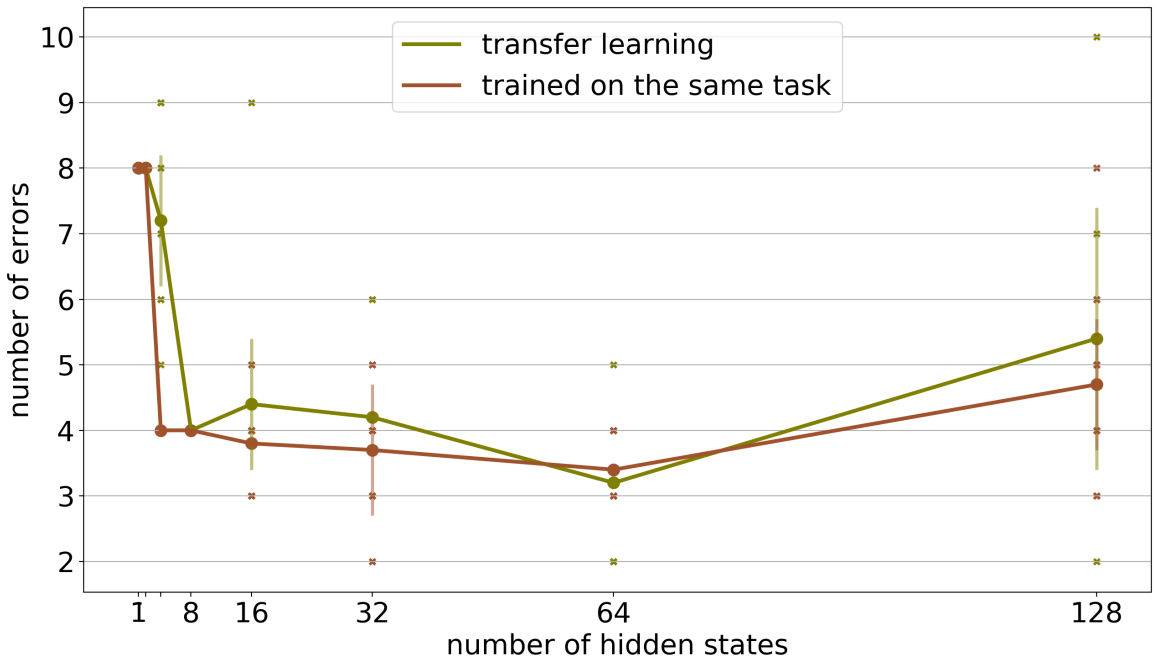


Figure 4.9: *Scenario 1* for transfer learning. This graph shows the number of errors as a function of the number of HMM hidden states when tested on a transfer learning task. Both the model and the user preferences are learned on *taks 0*, and tested on *task 1*.

power of the model and the costly number of user-provided labels we can obtain. Finally, we highlight strong results on two transfer learning tasks, where we train both our task and user preference models on an initial task, and test on two new tasks composed of the same building blocks, but arranged in a different structure. We are excited about continuing this line of research by implementing our model on a real robot, as well as by extending the model with layers of more advanced decision making processes. In Section 4.2, we implement this model on a robot, and perform a user study in order to validate our model. For our user study, we implement a simple decision making process that utilizes the predictions our model makes to autonomously choose the robot's actions at each step. Expanding in the direction of more sophisticated decision making processes represents a worthwhile future endeavor.

4.2 Learning with Object-Based Features on Real Robot

In this section, we optimize and implement the model presented in Section 4.1 on the Baxter robot and perform a user study where we demonstrate the practical use of the model. In this section, we use features related to the presence of task objects in the workspace in order to represent the state space. This work is based on [106] [3].

We test the system in a study with n=14 participants. We present results showing that the system achieves predictions on par with human-level performance by using only five user-provided labels. 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 model separation allows us to train on these new preferences quickly, with similar human-level results.

The contribution presented in this section is thus threefold. First, we explain how to learn a model of human behaviors from observations and how use it to provide supportive behaviors to human workers throughout the execution of a task. Second, we demonstrate that this model can benefit from a large training set from several users, yet still account for individual preferences in the type of support a user would like to receive, with only a small set of provided labels. Third, we show that the system performs on par with human-level prediction for our task by running a study with 14 participants.

4.2.1 Problem Formulation

In this section, we implement the model on the Baxter Research Robot (see Figure 4.10. We employ the same chair task as in Section 4.1, the chair seen in Figure 4.2, with a slightly modified HTM, which can be seen in Figure 4.11.

We consider that the robot can provide a set of supportive behaviors consisting of a

3. E. C. Grigore, A. Roncone, O. Mangin, and B. Scassellati, "Preference-based assistance prediction for human-robot collaboration tasks", in Proceedings of the 31st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Spain, Madrid, 2018, October 1–5.

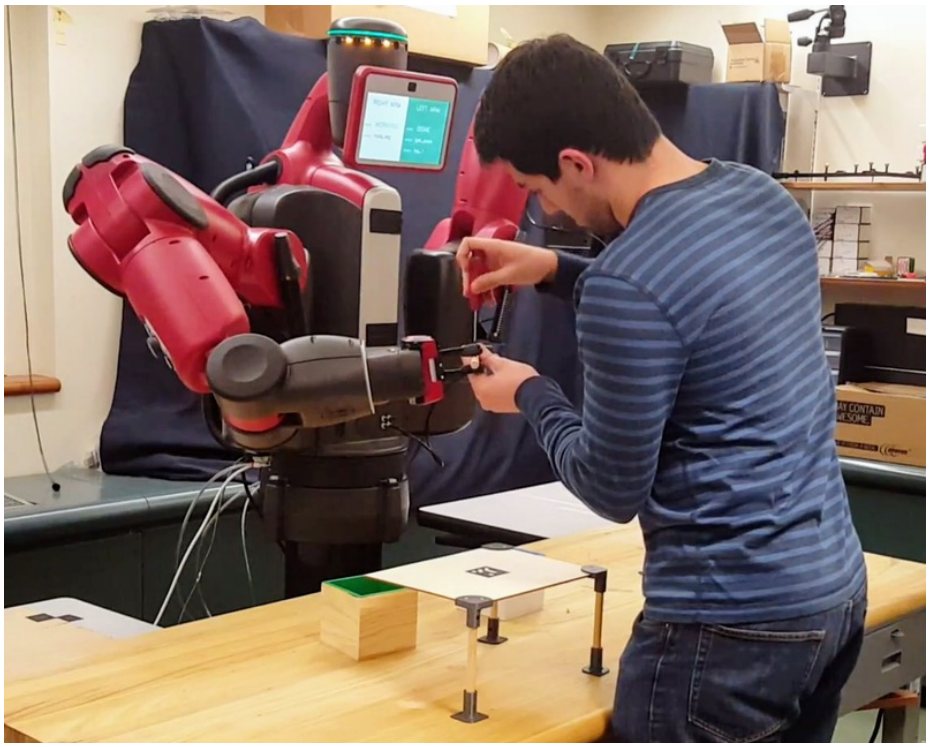


Figure 4.10: Human participant assembling the chair part of our user study. The robot is holding a component, allowing the participant to use both of his hands for the screwing action.

total of 10 behaviors, including no action. Table 4.3 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*. In contrast to Section 4.1, where the robot prediction consisted of only one supportive behavior per time step (the one with the highest predicted probability), in this work, the robot aims to execute all of the 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 order for the assistance provided by the robot to be useful to an individual user, the aim is for the robot to learn how to do so in a personalized manner. One human worker

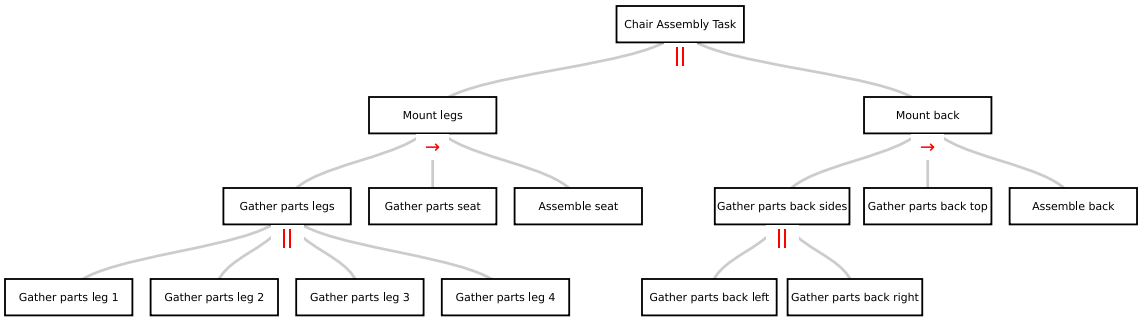


Figure 4.11: Hierarchical task model (HTM) representing the chair assembly task. Each leaf is an atomic subtask; each node composes subtasks that need to be achieved in sequence (in a given order, $\rightarrow$) or in parallel (in any order, $\|$). For each atomic subtask, the human worker has preferences over supportive behaviors that the robot might provide.

might prefer the robot to *hold* both the seat and the back while they perform the *assemble* action for that part (depicted in Table 4.3 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 section are obtained from only five user-labeled demonstrations, similar to Section 4.1.

Smooth collaboration requires that the robot supportive actions are provided when the human needs them. This however requires anticipation since actions and, specially, robot actions take time to happen. For instance, if the robot is to be truly helpful, it is essential to bring a screwdriver to a worker who is about to screw before actually reaching the point in the task when the screwing happens. Similarly to Section 4.1, in this section, too, we focus on predicting human needs with an anticipation time of $\Delta t = 1$.

Table 4.3: 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

4.2.2 Predictive Model

We implement the personalized supportive behavior model consisting of the extended HMM model presented in Section 4.1 in Figure 4.3, and perform a user study to evaluate this model with human participants. We use the same representations as those presented in Section 4.1, except for the observations used for the training set. In this section, each observation is in the form of a vector of features, with features based on the presence versus absence of objects in the workspace, with a 1 denoting presence. The values the features take on are still binary, like before, but the generation process is now based on the object presence in the workspace, without introducing noise during the training set generation process. We use a total of 19 features for our task.

4.2.3 Model Evaluation

Here, we detail the steps we followed to evaluate the model via the user study. 1. We generate a training set consisting of task trajectories without supportive behavior preference data in order to train the main task HMM (the single, cross-users model of the task), and I train the main task model on this set. We refer to this training set as set T henceforth,

and explain the details in Section 4.2.3. 2. We choose a small number of trajectories from the training set generated for the main task, and label them with supportive behavior preferences, experimenting with the exact number of trajectories to pick. We use this training set to train the personalized supportive behavior model. We refer to this training set as set *SB* henceforth, and explain the details in Section 4.2.3. 3. We decide a mechanism for computing errors for our model (detailed in Section 4.2.3), and we optimize parameters for all of the above during the simulation experiments (detailed in Section 4.2.3). 4. We use the optimized model from the simulated experiments (i.e., the trained HMM and the trajectories from set *SB*) to test with real users (detailed in Section 4.2.3).

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 [67] and detailed in Section 4.1. The current 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 the task utilized in this section can be seen in Fig. 4.11.

To train the 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.

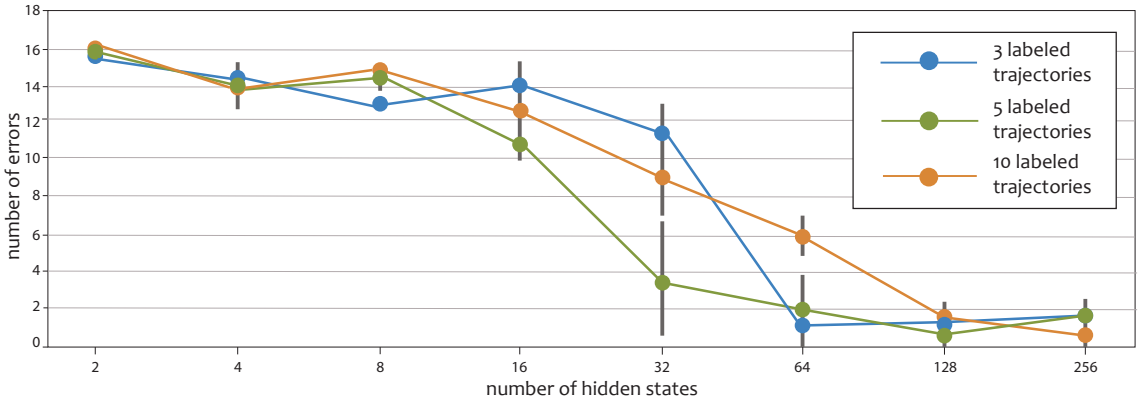


Figure 4.12: HMM parameter optimization procedure. This graph presents training eight different models corresponding to $2^1 = 2$, $2^2 = 4$, ..., $2^8 = 256$ total number of hidden states, and running the training for each model 10 different times, each tested with a different HTM trajectory. The procedure 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.

Training the personalized supportive behavior model

To train the 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 the simulation experiments, we consider a user with particular behavior preferences, and label the trajectories in accordance with these preferences. This provides me with set *SB*. During the 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). The 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, and are the same trajectories we give our human participants

during the user study in order to compare system performance with that of human-level performance (detailed in Subsection 4.2.3), although the results are similar when picking these trajectories at random. When training with more than five trajectories for parameter optimization, we pick these at random from the 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.

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 4.1.3, with $\Delta t = 1$, meaning the agent predicts one time step in advance. In this work, we choose $\Delta t = 1$ because the system predicts for discrete time steps in terms of task progression, and each task step is essential for task completion. However, the contributed 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 the 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.

Choosing model parameters

HMMs are prone to local optima. To choose the parameters for the user study, we ran a total of eight different HMMs, with increasing number of hidden states, using a single *SB* preference set. 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. 4.12 shows the 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).

Testing personalized supportive behavior models on robot

We run our experiments with the task presented in Figure 4.2, with the HTM from Figure 4.11. The 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 4.3. 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 the 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.

4.2.4 Results and Discussion

In this section, we present the results on the main set of user preferences, on the new per-user basis preference set, and compare them with human-level prediction for the task. Fig. 4.13 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 comparison to a random baseline for the 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 needed to choose 1, 2, or 3 supportive actions per time step to compute an estimate of how many behaviors to choose at that

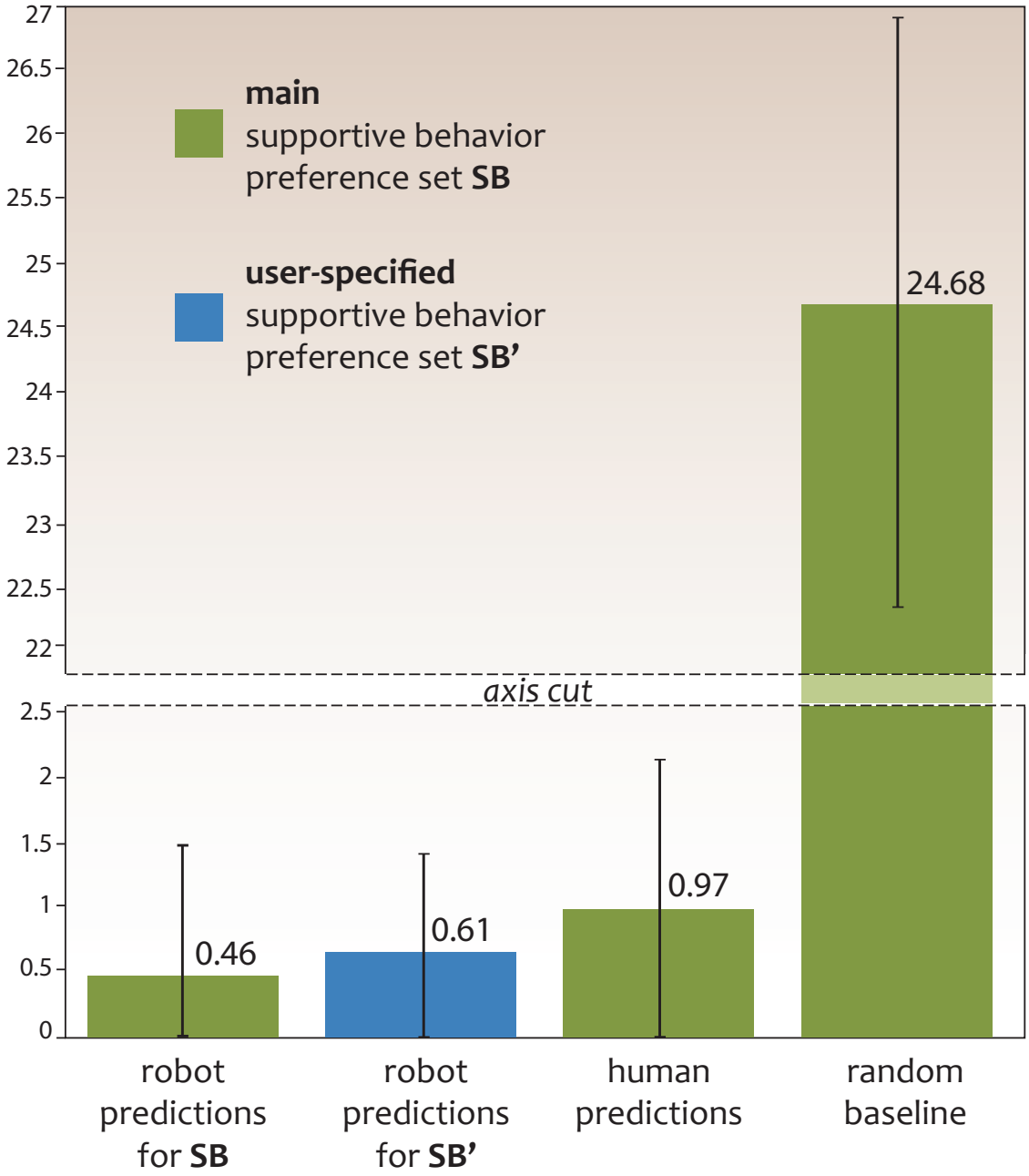


Figure 4.13: Average number of errors for the main supportive behavior preference set **SB**, the user-specified set **SB'**, the human predictions (interactive survey), and a random baseline.

step. Although the 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 the 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 the 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 Subsection 4.2.3. 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 the task: we only utilize the five trajectories participants have predicted after having seen the five examples that are also provided to the 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 the system against. The average for human-level performance is 0.97 (s.d. 1.12). Figure 4.13 shows that the 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 the HTM (**??**). 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.

4.2.5 Conclusions

In this section, we present a way of learning personalized supportive behavior models that leverage a single, robust model of the task, and show that the system achieves human-level prediction for a real HRC task via a user study.

Although we allowed the system to occasionally miss object-presence features when manipulation errors occurred during our user study, we are excited to improve the system by utilizing our model that can handle noisy observations presented in Section 4.1, integrated with camera or motion capture system input.

Chapter 5

Learning at the High Level: How Can We Learn High-Level Supportive Behaviors? A Model-Free Approach

In Chapter 4, we introduced an approach well-suited for situations when we have reasonable access to a relatively large pool of task trajectories, which can be used to learn a model of the task. We leveraged this by then building an extension of this model, and needing to obtain user-provided labels or demonstrations for only a very small number of trajectories in order to obtain a personalized model of user preferences with respect to what kind of support a robot should provide in an HRC scenario.

In this chapter, we investigate model-free approaches for the same HRC task, and compare this solution to the model-based one. Model-free solutions are better suited for situations where it is difficult or impossible to obtain the task trajectories needed to learn a model of the task. This chapter is based on [107] [1], and on further work building upon the

1. E. C. Grigore and B. Scassellati, "Hierarchical multi-agent reinforcement learning through communicative actions for human-robot collaboration", in Proceedings of the Future of Interactive Learning Machines (FILM) Workshop at the 30th Annual Conference on Neural Information Processing Systems (NIPS), Full paper, Barcelona, Spain, 2016, December 5–10.

concepts presented in this paper.

Here, we present a novel model-free paradigm based on the multi-agent setting, where we consider both the robot and the human as agents operating in the environment. This paradigm is multi-agent based since we consider the person to be an agent in the environment, and it is the joint action of the robot and the human (and not just the robot action) that effects the transition from one state of the system to the next, as well as the reward received when transitioning. However, since there is no controlling the human's actions like in typical multi-agent settings, we coin this paradigm *multi-agent based*. This allows maximum flexibility for specifying preferences with respect to what actions the human would like to perform themselves vs. what actions they would like the robot to execute at each step. Alongside this valuable flexibility, this paradigm mitigates the need for a large pool of training data, but generally requires more interactions with the user or user-provided labels for demonstrations. Although it requires more interactions than the proposed model-based technique, compared to the single-agent paradigm (which I also apply to this scenario for comparison purposes) the multi-agent based model leverages the knowledge of the human worker, considered an expert at performing the task at hand, and thus allows both agents to take actions during a time step, which can help speed up learning in some cases.

In order to account for different agents operating within our environment, we need to consider the different types of actions performed by each, the effects their actions have on the world they operate in, and the goals they are working towards. A relevant paradigm for representing such a problem is multi-agent systems (MAS) [37], where several agents attempt to solve tasks together in order to reach specific goals, either shared or competing. A particularly useful method that has been widely applied to MAS is that of reinforcement learning (RL) [38]. In RL, the agent operates in an environment in which it receives rewards when performing different actions in each state of the world and subsequently aims to behave such that it maximizes the long-term reward received. The use of RL for MAS has engendered the multi-agent reinforcement learning (MARL) paradigm [39], where multiple

learners can take different actions in the same environment, each trying to maximize long-term reward (such agents can be collaborative, competitive, or somewhere in between). This setup, however, assumes that we can control all the agents present in the system, and that each has a clear set of defined actions it can take in the shared world. The current HRC scenario, however, involves a human whose actions we neither control, nor define. Yet, these actions are important since, not only do they cause the state of the task to change over time, but they indirectly influence and are influenced by any supportive behaviors provided by the robot, and need to be taken into account.

We thus introduce a variant of the MARL framework, in which we know the state of the world (i.e. the state of the task), as well as the actions each of the agents are taking (i.e. we know what the robot's actions are, and we know what the human worker is doing during each state of the task), but we can only directly control the actions of the robot. Given that we consider the human to be knowledgeable about the task, we consider them an "expert" agent, which knows how to act within the task and has common sense knowledge about how different supportive behaviors could be useful within the task. The actions of the expert agent may be indirectly influenced by those performed by the robot, but this indirect impact is accounted for through the world dynamics (i.e. the joint actions performed by the person and the robot both affect state transitions). We present different variants of the paradigm that consider different types of communication between agents, and how this affects quality and speed of learning. We term these algorithms *Multi-Agent Based RL (MAB-RL)* algorithms. We further use a hierarchical reinforcement learning (HRL) framework [108] in order to re-use different types of supportive behaviors for different subtasks within the same domain, as well as to help with abstracting away from the state of the subtask details about the implementation of the specific supportive behaviors. We coin the hierarchical algorithm that we present in this chapter *Hierarchical Multi-Agent-Based Reinforcement Learning (HMAB-RL)*.

The *MAB-RL* and *HMAB-RL* algorithms can learn a valid trajectory for the task (i.e.

valid in accordance with both the task HTM and with the supportive behavior preferences for the robot) from one learning episode only. Of course, we wish to learn how to provide assistance in a robust way, and so test our algorithms by randomly choosing episodes from the HTM and ensuring our algorithms have successfully learned what actions are to be performed in *any* state of the task. Our algorithms are able to perform on par with human-level performance (as measured in Section 4.2) from 30 to 50 episodes worth of training, and start exceeding this level with increasing the number of episodes from 50 to 100. We also present an algorithm that uses *communicative actions* the robot can take when the cost of synchronizing to the other agents' actions by observations is too high. We show that we are able to learn the key moments throughout a task when the robot should take these communicative actions.

5.1 Related Work

The field of MAS is relevant to a wide array of areas, including robotics, telecommunications, distributed control, economics, and so on. A considerable amount of research studies the use of RL for MAS, the intersection of which is known as multi-agent reinforcement learning (MARL) [109]. MARL presents important benefits, such as faster learning through experience sharing (e.g., exchange of local information, skilled agents teaching less skilled ones, etc.), speed-ups due to parallel computation when exploiting the decentralized structure of the task, robustness to failures of individual agents, and easy insertion of new agents into the system [39], [37]. However, it is also plagued by a number of challenges, including the curse of dimensionality (the state action-space grows exponentially with the number of agents), the difficulty in specifying goals in such multi-agent settings (the rewards different agents receive are correlated and thus cannot always be maximized independently), non-stationarity due to concurrent learning (for each agent, the best policy changes as other agents' policies change), and an exacerbated exploration-exploitation tradeoff (agents need

to explore not only to reach new regions of the state space, but also to gather information about other agents) [39], [37].

Even though there exist considerable challenges in the field of MARL, this framework is extremely relevant to our human-robot collaboration scenario. This is due to the fact that, even though it is not part of our goal to find policies for the human worker during the task, their actions affect the transition model of the world in which our robot needs to operate. Beyond directly impacting the world dynamics, the actions performed by the human also play a key role in the type of supportive behavior that is useful for the robot to offer during different subtasks. In order to address two of the main challenges mentioned above—the curse of dimensionality, and partial observability with respect to the policies of other agents—researchers have looked into the use of hierarchical reinforcement learning (HRL). Work in this area includes the use of options [40], HAMs [41], and MAXQ [42]. Other work looks at using task-level coordination in HRL settings, where all agents were given a hierarchical break down of the task at hand that was used as a basis for communication [43], [44].

One of the most important and widely used ways in which we can generalize MDPs to the domain of MARL is via stochastic games [45]. There are many different types of stochastic games, but for the purposes of this thesis, we consider general-sum stochastic games, where there are no constraints on the sum of the agents' rewards. This is due to the cooperative nature of our environment, where the two agents are working together towards the same goal. However, if we were to simply extend techniques used to solve MDPs for the single-agent case to the multi-agent case, we would loose optimality guarantees due to the loss of stationarity in MARL caused by concurrent learning. One of the solutions to this issue is using the Nash equilibrium (or other equilibria) to describe optimal policies in multi-agent systems. In this chapter, we build upon a variant of the Nash Q-Learning algorithm for general-sum stochastic games [46] called Friend-or-Foe Q-learning (FFQ) in general-sum games [47].

Work that addressed partial observability and the issue of how the actions of other agents impact the policy of one agent includes a learning paradigm called team-partitioned, opaque-transition reinforcement learning (TPOT-RL). TPOT-RL considers opaque transitions to encompass multi-agent environments where each learner cannot rely on having knowledge of future state transitions after acting in the world [48]. Other relevant work in this area uses behaviors, defined as goal-driven control laws that achieve and/or maintain particular goals, to abstract away low level details of robot control and diminish the learning space, as well as tackling the credit assignment problem via heterogeneous reinforcement functions and progress estimators [49].

The related work presented above considered the use of MARL for scenarios where multiple robots or agents need to collaborate with each other in order to reach a common goal. We now mention research performed in the area of HRC, specifically. Such work includes enabling a robot to improve its policy for a knot untying task through user-provided guidance. In this scenario, the robot performs a set of state-action transitions to shake the contents placed inside of a bag (with the goal of untying the knot and emptying the contents of the bag), and asks for a reward signal from the human at the end of each learning episode [54]. Other relevant work in the area of obtaining rewards from a human user explored the importance of understanding the human-teacher/robot learner system as a whole, with findings that people use the reward signal not only to provide feedback about past actions, but also to provide future directed rewards to guide subsequent actions [55]. Yet other work investigated discovering policies for improving collaborator performance through a task and motion planning approach [66], which differs from our contribution.

MAB-RL and *HMAB-RL* distinguish themselves from previous work by considering a unique view of the (hierarchical) MARL paradigm, where *one of the agents* is an expert and thus *does not learn* its policy simultaneously with the robot learner, but *performs actions that have a direct effect* on the dynamics of the world. We thus model this scenario as a special case of a MARL problem, where the robot needs to learn policies that adapt not

only to the state of the task, but to the uncontrolled actions of the person as well. The nature of this problem is also inherently different from the above work due to the fact that we aim to learn supportive behaviors, which can (although need not) be different from the types of actions the human worker performs, and different from the types of actions a robot would learn for performing a task autonomously. To this end, we also take advantage of the interaction with a human, who can use common sense knowledge about how different types of supportive behaviors might be useful as the task progresses. For one of the variants we present, we also include this in the reward signal via employing *communicative actions*.

5.2 Existing Paradigms in Reinforcement Learning

Given that the presented framework lies at the intersection of existing paradigms in RL, we define these paradigms by going through the formalisms used for each.

5.2.1 Single-Agent RL

A typical RL formulation relies on the Markov Decision Process (MDP) formalism, which we present in Subsection 2.2.2. The MDP formulation is very well suited for the RL paradigm, where we have an agent that operates in a world composed of the states defined above, chooses one of several actions available to it in each state, receives a reward, and then moves into a different state based on the transition probability. The agent seeks to maximize its long-term discounted reward and thus seeks an optimal way to behave. The goal is then to solve the MDP by finding an optimal policy, π^* (a mapping $\pi : \mathcal{S} \rightarrow \mathcal{A}$ from states to actions in the deterministic case, or a probability distribution $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ over state-action pairs in the stochastic case), representing a policy that, for all states, is at least as "good" as any other policy (i.e. has a value or action-value function [2] as high as

2. The value of a state under a policy is the expected discounted sum of future rewards starting from that state and following that policy thereafter. The action-value is similar, but also considers the action taken in that state.

that of any other policy).

There exist several ways of solving MDPs, but here we mention Q-learning [110], an off-policy algorithm (i.e. the agent does not need to follow the policy for which it is learning the action-value function) that is also model-free, meaning that the agent does not need to learn the transition probabilities. Model-free algorithms are temporal difference (TD) techniques [38], which allow the agent to update the estimate of the value or action-value function based on other estimates, without needing to wait for the true value.

5.2.2 HRL

HRL is typically formulated based on a semi-Markov Decision Process (SMDP), an extension of the MDP in which actions can take a variable amount of time to complete [111], [112]. An SMDP is defined in a similar manner, as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{I} \rangle$. The only two elements that are different from those in an MDP are the transition probability and reward functions. The former is now defined as $\mathcal{P} : \mathcal{S} \times \mathbb{N} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, where $P(s', N|s, a)$ is the probability of transitioning from state s to state s' in N time steps when the agent takes action a (the moments when transitions occur are called decision epochs). The latter is now defined as $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, with $r(s, a)$ representing the expected total reward between two decision epochs, with the agent being in state s at the first epoch and taking action a (unlike in the MDP case, here rewards can accrue over the course of one action).

HRL is a framework for scaling RL problems with large state spaces by re-using policies learned as individual actions [108]. Such abstraction allows a learning system to ignore details irrelevant to each level of the task, and reduces computational costs. Most of the algorithms used for solving MDPs can be extended to solve SMDPs, including Q-learning. HRL generalizes the idea of a "macro-operator"—a sequence of operators or actions that can be invoked by name just like a primitive action—to closed-loop partial policies, defined for subsets of the state set [108]. An important paradigm proposed and studied in the HRL community is the formulation of partial policies into options [40].

Options generalize primitive actions to include temporally-extended courses of action [40]. An option is defined as a tuple $\langle \mathcal{I}_{\mathcal{O}}, \pi, \beta \rangle$, where $\mathcal{I}_{\mathcal{O}}$ is the initiation set of the option (the option is available in a state s if and only if $s \in \mathcal{I}_{\mathcal{O}}$), $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the policy to be executed, and $\beta : \mathcal{S}^{+} \rightarrow [0, 1]$ is the termination condition. When the agent takes the option, it chooses actions according to the policy π until the option terminates according to β. *HMAB-RL* uses options as temporally-extended options with variable number of primitive actions composing them.

5.2.3 Hierarchical MARL

The SMDP model has been further extended to the multi-agent domain, resulting in the multi-agent SMDP (MSMDP) model. An MSMDP is defined following [113], as a tuple $\langle \Upsilon, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{I}, \mathcal{T} \rangle$. Υ is a finite set of n agents. Each agent $j \in \Upsilon$ has at its disposal a set of actions, A^j, together forming the joint-action space $\mathcal{A} = \prod_{j=1}^{n} A^j$. An element of $\mathcal{A}$ represents the simultaneous execution of actions a^j by all agents j and is denoted $\vec{a} = \langle a^1, a^2, ..., a^n \rangle$. Elements $\mathcal{S}$, $\mathcal{P}$, $\mathcal{R}$, and $\mathcal{I}$ are defined like in the SMDP case, with the note that $P(s', N|s, \vec{a})$ is the probability that the system transitions from state s to state s' in N steps when the agents take joint-action $\vec{a}$. Finally, $\mathcal{T}$ represents the termination scheme that specifies how decision epochs are defined, to handle the fact that temporally-extended actions can have variable lengths.

Different types of termination strategies can be classified into synchronous schemes, where all agents make a decision at each epoch (agents need to be synchronized in a centralized way), and asynchronous schemes, where only a subset of agents make decisions at each epoch (de-centralized decision making). Examples of three termination strategies, τ_{any}, τ_{all}, $\tau_{continue}$, have been researched in [114]. τ_{any} has all agents interrupt their actions as soon as the first action of the joint-action that has been executing terminates, at which point the agents choose a new joint-action. τ_{all} involves waiting until all the actions part of the joint-action terminate for the agents to choose the next joint-action, with agents who

need to wait until completion taking an "idle" action. These two strategies are examples of synchronous schemes. Finally, $\tau_{continue}$ is an example of an asynchronous scheme, and involves choosing the decision epoch based on the earliest terminating action in a joint-action (like τ_{any}) but having only those agents who have terminated their actions choose new ones.

5.3 Algorithms Relevant to MARL

The algorithms we describe in this work utilize game theory concepts, which represent a powerful mathematical tool for modeling MARL scenarios and allows agents to make rational decisions in these contexts. A necessary paradigm for this work is that of one-stage general-sum n-player games, which we include here following closely the definitions from [47].

5.3.1 Stochastic Games

A one-stage general-sum n-player game is based on n players. Like in an MSMDP, each agent $j \in \Upsilon$ has at its disposal a set of actions, A^j, together forming the joint-action space $\mathcal{A} = \prod_{j=1}^{n} A^j$. The reward is defined similarly, with each agent $j \in \Upsilon$ receiving its own reward r_j when the joint-action $\vec{a} = \langle a^1, a^2, ..., a^j, ..., a^n \rangle$ is executed in state s. Each reward function $\mathcal{R}_j$ maps an action choice for each of the players to a scalar reward value.

A one-stage policy for player j, π_j is a probability distribution over its actions $\mathcal{A}_j$. The expected payoff to player j when players execute one-stage policies $\pi_1, \pi_2, ..., \pi_j, ..., \pi_n$ is $\mathcal{R}_j(\pi_1, \pi_2, ..., \pi_j, ..., \pi_n)$, which represents the expected value of the values of $\mathcal{R}_j$ weighted by the probabilities under the given one-stage policies.

Every one-stage general-sum n-player game has a Nash equilibrium, which is a set of one-stage policies $\pi_1, \pi_2, ..., \pi_j, ..., \pi_n$ such that no player can improve its expected payoff by unilaterally changing its one-stage policy: $\mathcal{R}_j(\pi_1, \pi_2, ..., \pi_j, ..., \pi_n) \geq \mathcal{R}_j(\pi_1, \pi_2, ..., \pi_{j-1},$

$\pi'_j, \pi_{j+1}, ..., \pi_n)$, for all one-stage policies π'_j and $1 \leq j \leq n$. A game can have more than one Nash equilibrium, and the expected payoff to player j can vary depending on the equilibrium in use. The Friend-or-Foe Q-learning (FFQ) algorithm [47] utilizes both adversarial and coordination equilibriums because it considers settings that are not purely cooperative. For our purposes, we make use only of the coordination equilibrium, in which all players achieve their highest possible value:

$$\mathcal{R}_j(\pi_1, \pi_2, ..., \pi_n) = \max_{a_1 \in \mathcal{A}_1, a_2 \in \mathcal{A}_2, ..., a_n \in \mathcal{A}_n} \mathcal{R}_j(a_1, a_2, ..., a_n). \tag{5.1}$$

A Markov game generalizes the one-stage game to multiple-stages, consisting of a finite set of states $\mathcal{S}$, with each state $s \in \mathcal{S}$ having its own payoff function. Markov games also have a transition function that maps between a state and an action choice for each player and a probability distribution over next states. This is equivalent to the policy in an MSMDP. The value for a player in a game is computed as the discounted sum of payoffs, with a discount factor $0 \leq \gamma \leq 1$. The $\mathcal{Q}$ function for player j is defined as:

$$Q_j(s, \vec{a}) = R_j(s, \vec{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \vec{a}) Q_j(s', \pi_1, \pi_2, ..., \pi_n), \tag{5.2}$$

where $Q_j(s', \pi_1, \pi_2, ..., \pi_n)$ is a weighted sum of the values of $Q_j(s', \vec{a'})$ according to the πs. This value represents the value to player j when the state is s and the players choose the joint-action $\vec{a}$, continuing to follow their respective policies afterwards.

5.3.2 Nash-Q and Friend-or-Foe Q-Learning (FFQ)

As detailed in [47], Filar and Vrieze [115] showed how the Q function links Markov games and one-stage games. Treating the Q functions at each state as payoff functions for individual one-stage games, the policies at the individual states are in equilibrium if and only if the overall multistage policies are in equilibrium. Therefore, an equilibrium for a Markov

game can be found by finding a Q function with the appropriate properties.

The main Nash-Q update rule for learning a Q function for a player from its experience interacting with other players in the game is:

$$Q_j(s, \vec{a}) = (1 - \alpha_t)Q_j(s, \vec{a}) + \alpha_t(r_j + \gamma Nash_j(s, Q_1, Q_2, ..., Q_n)), \tag{5.3}$$

each time a new experience $\langle s, \vec{a}, s', r_1, r_2, ..., r_n \rangle$ occurs. Here, $Nash_j(s, Q_1, Q_2, ..., Q_n) = Q_j(s, \pi_1, \pi_2, ..., \pi_n)$, where the πs are a Nash equilibrium for the one-stage game defined by the Q functions $Q_1, Q_2, ..., Q_n$ at state s, and the α_t values are a sequence of learning rates.

In single-player games, the Nash function reduces to maximization for one agent, which makes the above equivalent with single-agent Q-learning [116]. In zero-sum games, which are games that restrict the sum of all of the agents' rewards to be equal to zero, the Nash function reduces to a minimax function. The FFQ algorithm introduces the use of two different rules for computing the Nash function, based on whether the opponent is considered a friend or a foe. Since we only consider cooperative scenarios herein, we make use only of the friend variant:

$$Nash_j(s, Q_1, Q_2, ..., Q_n) = \max_{a_1 \in \mathcal{A}_1, a_2 \in \mathcal{A}_2, ..., a_n \in \mathcal{A}_n} Q_j(s, \vec{a}). \tag{5.4}$$

This is essentially the equivalent of utilizing Q-learning in the combined action space of all the players.

5.4 Problem Formulation

In this chapter, we employ the same chair task as in Section 4.2, the chair seen in Figure 4.2, with the same HTM that can be seen in Figure 4.11. We use the same HTM to compare the main results of the current model-free approach with those of the model-based approach.

However, we also create different variants of this HTM in order to test the nuances that are more interesting and relevant for the different algorithms and approaches we introduce here, and to show their versatility in being capable of learning more useful behaviors for the robot.

5.5 Paradigms and Algorithms for HRC

In this section, we present a single-agent RL baseline we utilize as a first comparison to our model-based approach, followed by two paradigms for HRC, namely *Multi-Agent Based RL (MAB-RL)* and *Hierarchical Multi-Agent-Based Reinforcement Learning (HMAB-RL)*. We introduce three algorithms stemming from the *MAB-RL* paradigm and one from the *HMAB-RL* paradigm, and we present data that compares these algorithms in terms of their effectiveness and speed of learning.

5.5.1 Single-Agent Baseline

In order to compare with the results presented in Chapter 4, we start by using the same HTM as in Section 4.2, for the same chair assembly task that can be seen in Figure 4.2. We use the same main preference set *SB* as in Section 4.2, which we present here in Table 5.1.

Here, we implement the algorithm that can be seen in Algorithm 2, using simple Q-learning. In this case, we are assuming the robot should be executing every supportive behavior specified, like in Chapter 4. This algorithm is the standard Q-learning algorithm [116], with no changes implemented. We specify the values used in the pseudocode presented in Algorithm 2. Since we utilize features based on object-presence to represent our states, the single-agent case is deterministic, and so we use a learning rate $\alpha = 1$, which is optimal for deterministic scenarios. For all of our multi-agent based algorithms, this does not hold since we do not distinguish between an action taken by the human and an action taken by the robot when we update the state. The algorithms we present in the following

Table 5.1: Example of two trajectories based on the main set of preferences *SB* used to test the single-agent reinforcement learning baseline

Trajectory 1		Trajectory 2	
Human action	Supportive behavior labels	Human action	Supportive behavior labels
gather parts leg 3	[bring back bracket, bring dowel] bring screwdriver]	gather parts leg 2	[bring back bracket, bring dowel,
gather parts leg 4	[bring back bracket, bring dowel]	gather parts leg 1	[bring back bracket, bring dowel]
gather parts leg 2	[bring front bracket, bring dowel,	gather parts leg 3	[bring front bracket, bring dowel] bring screwdriver]
gather parts leg 1	[bring front bracket, bring dowel]	gather parts leg 4	[bring front bracket, bring dowel]
gather parts seat	[bring seat]	gather parts seat	[bring seat]
assemble seat	[hold]	assemble seat	[hold]
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 right	[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	[hold]

subsections allow for complete flexibility for specifying what actions the robot should take and allow the user to take whatever actions they desire.

In Algorithm 2, we also present how we compute the reward starting with line 13. We essentially punish the robot for taking an action that is considered invalid given the current HTM (i.e. an action that cannot make the chair assembly task progress), and remain neutral when the action is valid (i.e. the action makes the task progress to another valid state). We also introduce the ability of specifying *ordering preferences* (not *supportive behavior preferences*), such as building the back part of the chair first vs. the legs first. To do so, we need only provide a high reward when the current state s reflects the fact that the preferred ordering was accomplished (we can check this easily because we use features based on object presence to represent the states). For example, if the current state s encodes our position after we have brought all the legs and the back, we know that the legs were executed first. Learning the ordering preferences requires more episodes, but this does not effect the amount of time it takes to learn the supportive behavior preferences. We present this analysis in Section 5.6.

We also present our procedure for computing the best path and choosing the actions the robot should take at each step throughout the task in line 17. This procedure can be

Algorithm 2 Single-Agent Q-learning Baseline: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:

Sates $\mathcal{S} = \{s_1, \ldots, s_{n_s}\}$

Actions $\mathcal{A} = \{a_1, \ldots, a_{n_a}\}, \qquad A : \mathcal{S} \Rightarrow \mathcal{A}$

Reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

Black-box transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

Learning rate $\alpha \in [0, 1]$, we set $\alpha = 1$

Discounting factor $\gamma \in [0, 1]$, we set $\gamma = 0.9$

Preference for build order $PREF \in \{'legs\_first', 'back\_first', 'none'\}$

procedure Q-LEARNING($\mathcal{X}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{P}$, α, γ)
  Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
  while Q is not converged or max number of iterations has not been reached **do**
    Start in state $s \in \mathcal{S}$
    while s is not terminal **do**
      Calculate π according to Q and exploration strategy, we use $\epsilon - greedy$ with $\pi(s) \leftarrow \arg\max_a Q(s, a)$ 70% of the time, and $\pi(s) \leftarrow$ random action a 30%
      $a \leftarrow \pi(s)$
      $R(s, a) \leftarrow$ REWARD(s, a) ▷ Compute the reward
      $r \leftarrow R(s, a)$ ▷ Receive the reward
      $s' \leftarrow P(s, a)$ ▷ Receive the new state
      $Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$
      $s \leftarrow s'$
  return Q

procedure REWARD(s, a)
  if s is a final state **then return** 10
  if s' is NOT a valid next state in accordance with the HTM **then return** -1
  if ($PREF == 'legs\_first'$ and s reflects $'legs\_first'$) or ($PREF == 'back\_first'$ and s reflects $'back\_first'$) **then return** 5
  return 0

procedure BEST_PATH(Q)
  $s \leftarrow start\_state$
  Append s to $best\_path$
  $actions \leftarrow \emptyset$
  while s is not terminal **do**
    $a \leftarrow \arg\max_a Q(s, a)$
    $s' \leftarrow P(s, a)$
    $s \leftarrow s'$
    Append s to $best\_path$
    Append a to $actions$
  return $best\_path$

used to recover the full path containing the states, together with the actions the robot is to execute by only making use of the learned Q matrix. If we always choose the action that maximizes the Q-value of the current state, we will always recover a correct set of actions in accordance with the HTM of the task, even after only one learning episode. This is because we penalize all invalid actions. If we do not have any preference whatsoever about the ordering of parallel actions, applying the standard Q-learning algorithm and training for one episode only would work well. However, this does not allow for the flexibility of switching to a different order for the parallel actions, since the robot has not experienced any other path after training for a single episode. It also does not allow the flexibility of being able to choose a valid action when the state changes for various other reasons. Thus, we compute the number of errors by choosing episodes at random the valid HTM set of trajectories and counting errors whenever the robot cannot produce a valid action at each step. We present this further in Section 5.6.

5.5.2 *MAB-RL*: Multi-Agent Based RL

Here, we introduce a novel paradigm for HRC scenarios like the ones we tackle in this dissertation. We introduce three algorithms that represent variants of this paradigm, each utilizing a different method for synchronization of actions in a MAS context. We coin this paradigm *Multi-Agent Based Reinforcement Learning (MAB-RL)* since we consider the person as an agent in the task, alongside the robot, but we cannot control their actions. This allows full flexibility in specifying preferences for what actions the robot should take vs. what actions the user prefers to take at each step throughout the task. Since we target settings like manufacturing domains where human workers are highly skilled in performing the kinds of tasks we are considering herein, we consider the human an expert agent.

We present three algorithms with different techniques for synchronizing the person and the robot's actions during each time step of the task. The first assumes the human watches closely what the robot does, and can observe the action the robot starts to execute at the

Table 5.2: Supportive Behavior Preference Set for *MAB-RL* and *HMAB-RL* Paradigms

Supportive behavior preference set	
HTM Leaf	Robot and human action pair
gather parts leg 1	[(bring front bracket, bring dowel)]
gather parts leg 2	[(bring front bracket, bring dowel)]
gather parts leg 3	[(bring dowel, no action human), (no action robot, bring screwdriver), (bring back bracket, no action human)]
gather parts leg 4	[(bring dowel, no action human), (bring back bracket, no action human)]
gather parts seat	[(bring seat, no action human)]
assemble seat	[(hold, no action human)]
gather parts back right	[(bring dowel, bring top bracket)]
gather parts back left	[(bring top bracket, bring dowel)]
gather parts back top	[(bring long dowel, bring back)]
assemble back	[(no action robot, no action human)]

start of each time step, and can choose his or her action for the same time step accordingly. We present this algorithm in Algorithm 3. The second assumes the robot can observe the start of the action the human is taking at the beginning of each time step, and thus takes its action accordingly. We present this second algorithm in Algorithm 4. The third does not assume any observation of the start of an action, and instead adds a *communicative* action to the robot's set of actions, which it can take in order to ask the person what action he or she is intending to take during the next time step. This is an ideal scenario since we do not need to depend on observing the start of an action, but it comes at the cost increased training time. We present this third algorithm in Algorithm 5.

In order to specify the preferences for the robot's supportive actions, we encode each leaf of the HTM with a list of pairs of the form $(robot\_action_i, human\_action_i)$, where i represents the time step. Each leaf can have a variable number of such pairs, without any restrictions. We utilize a total of eight actions that both the robot and person can perform, an extra two actions only the robot can execute, and an extra single action only the human can execute. For the algorithm that employs communicative actions, the robot has available an extra of three actions at its disposal. Each $(robot\_action_i, human\_action_i)$ pair gets executed during one course of a single time step part of the task, with the assumptions about observing the other agent's actions or about being able to take communicative actions dictating the way in which the actions are synchronized. In Table 5.2, we present the sup-

portive behavior preference set we utilized to perform the experiments herein. The order of the leaves as part of a trajectory is generated in accordance with the HTM. This represents the same set of supportive preferences used in Section 4.2, as well as in the single-agent scenario described in Subsection 5.5.1. The main and most important difference is that in the MAB-RL case, we can specify any preference with respect to what the robot vs. the human should do at each step, whereas we could only train the robot to perform a fix set of actions beforehand. Thus, the $(robot\_action_i, human\_action_i)$ pairs can be modified in any way based on different preferences (including subtle changes in terms of switching two actions between the robot and the human for different steps), and the algorithms will learn the new preferences without the need for any adjustment.

MAB-RL: Human Watch

The first *MAB-RL* variant we present is based on an action synchronization scheme that assumes the human observes the start of the robot action from the $(robot\_action_i, human\_action_i)$ pair during one time step, and will then perform an action that is valid in accordance with the HTM and with the preference set. This turns out to be the most efficient algorithm out of the three we present under the *MAB-RL* paradigm, and we can speed up learning by using the hierarchical version. Of course, the ideal scenario is one in which the robot does the observing or in which the robot takes communicative actions so as to remove the onus places on the human, but these come at the cost of increased learning in terms of the number of episodes needed. We present the *MAB-RL* Human Watch algorithm in Algorithm 3.

In this algorithm, we use the Q-value update rule Subsection 5.5.3 based on the FFQ update rule Equation (5.3), and compute the Nash value Subsection 5.5.2 utilizing the friend variant from FFQ Equation (5.4). We compute the reward similarly to the single-agent case, except that we need both the robot and human actions in order to compute the next state Subsection 5.5.2.

Algorithm 3 *MAB-RL* Human Watch Q-learning: Learn function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Require:

Sates $\mathcal{S} = \{s_1, \ldots, s_{n_s}\}$

Agents $i \in \Upsilon$, we have $\Upsilon = \{robot, human\}$

Joint-actions $\mathcal{A} = \{\vec{a_1}, \ldots, \vec{a_{n_a}}\}$, with action $\vec{a_i} = (a_{i_{robot}}, a_{i_{human}})$ $\quad A : \mathcal{S} \Rightarrow \mathcal{A}$

Actions $\mathcal{A}_{robot} = \{a_{1_{robot}}, \ldots, a_{n_{robot}}\}$

Actions $\mathcal{A}_{human} = \{a_{1_{human}}, \ldots, a_{n_{human}}\}$

Reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Black-box transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$

Learning rate $\alpha \in [0, 1]$, we set $\alpha = 0.2$

Discounting factor $\gamma \in [0, 1]$, we set $\gamma = 0.9$

Preference for build order $PREF \in \{'legs\_first', 'back\_first', 'none'\}$

procedure Q-LEARNING($\mathcal{X}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{P}$, α, γ)

  Initialize $Q : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ arbitrarily

  while Q is not converged or max number of iterations has not been reached **do**

    Start in state $s \in \mathcal{S}$

    while s is not terminal **do**

      Calculate π according to Q and exploration strategy, we use $\epsilon - greedy$ with $\pi(s) \leftarrow \arg\max_{a_{robot}} Q(s, a_{robot}, a_{human}), \forall a_{human} \in \mathcal{A}_{human}$ 70% of the time, and $\pi(s) \leftarrow$ random action a_{robot} 30%

      $a_{robot} \leftarrow \pi(s)$

      $a_{human} \leftarrow$ correct action in accordance with the HTM and the preference set based on having knowledge of a_{robot}

      $R(s, a_{robot}, a_{human}) \leftarrow$ REWARD$(s, a_{robot}, a_{human})$ ▷ Compute the reward

      $r \leftarrow R(s, a_{robot}, a_{human})$ ▷ Receive the reward

      $s' \leftarrow P(s, a_{robot}, a_{human})$ ▷ Receive the new state

      $Q(s', a_{robot}, a_{human}) \leftarrow (1 - \alpha) \cdot Q(s, a_{robot}, a_{human}) + \alpha \cdot (r + \gamma \cdot Nash_{robot}(s', Q)$

      $s \leftarrow s'$

  return Q

procedure $Nash_{robot}(s', Q)$

  return $\max_{a_{robot} \in \mathcal{A}_{robot}, a_{human} \in \mathcal{A}_{human}} Q_{robot}(s, \vec{a} = (a_{robot}, a_{human}))$

procedure REWARD(s, $\vec{a} = (a_{robot}, a_{human})$)

  if s is a final state **then return** 10

  if s' is NOT a valid next state in accordance with the HTM **then return** -1

  return 0

MAB-RL: Robot Watch

The second *MAB-RL* variant we present is based on an action synchronization scheme that assumes the robot observes the start of the human action from the $(robot\_action_i, human\_action_i)$ pair during one time step, and will then perform an action that is valid in accordance with the HTM and with the preference set. This turns out to be less efficient than the *MAB-RL Human Watch* algorithm, but it takes the onus away from the person to observe the robot's actions. We present the algorithm in Algorithm 4.

MAB-RL: Robot Communicative Actions

The third *MAB-RL* variant we present is based on an action synchronization scheme that assumes no observation of the start of an action on either the part of the robot or the human. Instead, we add a *communicative* action to the set of actions the robot can perform, and we learn when it is ideal to execute this communicative action, together with the rest of the supportive behaviors. This flexibility comes at the cost of increased learning time, however it affords us with learning not only how to act in accordance with the task and the supportive behavior preference set, but also to employ communicative actions at only key moments during the task. These key moments represent parallel nodes (i.e. when the robot would truly benefit from knowing in advance what the intention of the human is for the next time step, even at test time). This is because when a node is truly parallel, the human can choose to take either one of its children, and any option would be valid in this context. Hence, communication at these moments is crucial. We present our algorithm in Algorithm 5.

In order to tackle this challenge, we introduce a simple model of the human, and keep track of their actions throughout the learning phase. For each state we visit, we keep track of the frequency of the human having taken that action by incrementing a counter for the action the person is taking in the current state (line 19). We then reward the robot when it chooses a *communicative* action only when it does so in states of high uncertainty. We compute the uncertainty of a state based on the computed frequency, by computing the

Algorithm 4 *MAB-RL* Robot Watch Q-learning: Learn function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Require:

Sates $\mathcal{S} = \{s_1, \ldots, s_{n_s}\}$

Agents $i \in \Upsilon$, we have $\Upsilon = \{robot, human\}$

Joint-actions $\mathcal{A} = \{\vec{a_1}, \ldots, \vec{a_{n_a}}\}$, with action $\vec{a_i} = (a_{i_{robot}}, a_{i_{human}})$ $\quad A : \mathcal{S} \Rightarrow \mathcal{A}$

Actions $\mathcal{A}_{robot} = \{a_{1_{robot}}, \ldots, a_{n_{robot}}\}$

Actions $\mathcal{A}_{human} = \{a_{1_{human}}, \ldots, a_{n_{human}}\}$

Reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Black-box transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$

Learning rate $\alpha \in [0, 1]$, we set $\alpha = 0.2$

Discounting factor $\gamma \in [0, 1]$, we set $\gamma = 0.9$

Preference for build order $PREF \in \{'legs\_first', 'back\_first', 'none'\}$

procedure Q-LEARNING($\mathcal{X}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{P}$, α, γ)

  Initialize $Q : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ arbitrarily

  while Q is not converged or max number of iterations has not been reached **do**

    Start in state $s \in \mathcal{S}$

    while s is not terminal **do**

      $a_{human} \leftarrow$ correct action only in accordance with the HTM and the preference set, and not the a_{robot}

      Calculate π according to Q and exploration strategy, we use $\epsilon - greedy$ with $\pi(s) \leftarrow \arg\max_{a_{robot}} Q(s, a_{robot}, a_{human})$ 70% of the time, and $\pi(s) \leftarrow$ random action a_{robot} 30% ▷ a_{human} as chosen above

      $a_{robot} \leftarrow \pi(s)$

      $R(s, a_{robot}, a_{human}) \leftarrow$ REWARD$(s, a_{robot}, a_{human})$ ▷ Compute the reward

      $r \leftarrow R(s, a_{robot}, a_{human})$ ▷ Receive the reward

      $s' \leftarrow P(s, a_{robot}, a_{human})$ ▷ Receive the new state

      $Q(s', a_{robot}, a_{human}) \leftarrow (1 - \alpha) \cdot Q(s, a_{robot}, a_{human}) + \alpha \cdot (r + \gamma \cdot Nash_{robot}(s', Q)$

      $s \leftarrow s'$

  return Q

procedure $Nash_{robot}(s', Q)$

  return $\max_{a_{robot} \in \mathcal{A}_{robot}, a_{human} \in \mathcal{A}_{human}} Q_{robot}(s, \vec{a} = (a_{robot}, a_{human}))$

procedure REWARD(s, $\vec{a} = (a_{robot}, a_{human})$)

  if s is a final state **then return** 10

  if s' is NOT a valid next state in accordance with the HTM **then return** -1

  return 0

ratio between the number of past actions equal to the action the human is currently taking and the total number of all actions that have been taken so far in the current state (line 25). Given that in our case, states of low uncertainty are those that are part of sequential nodes, this ratio is close to 1 for such states, and thus straightforward to recognize.

In order to implement the *communicative* action and update the state accordingly, we introduce a feature in the representation of the state that represents the human's response to the question "What action are you intending to perform next time step?" This feature takes on value -1 whenever the *communicative* action is not being exercised, and takes on the value of the human's response when the robot decides to utilize this action. Once the human gives their response, the feature takes on the value of the action that is intended to be executed during the next time step by the human, and the Q matrix gets updated for the $(communicativeaction, humanresponse)$ pair for the current state. During the next time step, the human takes the action they previously stated was their intention, and the new state that now contains this action as one of its features now gets updated with the reward resulting from this transition. During this phase, the robot need only search those values of its Q matrix that correspond to the human action that the person shared as part of their reply previously.

In Table 5.3, we show a successful example of an episode that utilizes the Q matrix maximum value for choosing a_{robot}, with the human choosing a_{human} only in accordance with the HTM. We can see that the *MAB-RL Robot Communicative Actions* algorithm learns to take *communicative* actions only at key points during the task, and in particular in high uncertainty states which happen for parallel nodes in our HTM.

5.5.3 *HMAB-RL*: Hierarchical Multi-Agent Based RL, Human Watch

We coin the final paradigm from our contribution Hierarchical Multi-Agent Based RL (*HMAB-RL*), where we use options in order to speed up learning. We implement the options on the basis of the *MAB-RL Human Watch* algorithm, although this can be extended

Algorithm 5 *MAB-RL* Robot Communicative Actions Q-learning: Learn function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$

Require:

Sates $\mathcal{S} = \{s_1, \ldots, s_{n_s}\}$, Agents $i \in \Upsilon$, we have $\Upsilon = \{robot, human\}$
Joint-actions $\mathcal{A} = \{\vec{a_1}, \ldots, \vec{a_{n_a}}\}$, with action $\vec{a_i} = (a_{i_{robot}}, a_{i_{human}})$ $A : \mathcal{S} \Rightarrow \mathcal{A}$
Actions $\mathcal{A}_{robot} = \{a_{1_{robot}}, \ldots, a_{n_{robot}}\}$, Actions $\mathcal{A}_{human} = \{a_{1_{human}}, \ldots, a_{n_{human}}\}$
Reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, Black-box transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$
Learning rate $\alpha \in [0, 1]$, we set $\alpha = 0.2$, Discounting factor $\gamma \in [0, 1]$, we set $\gamma = 0.9$
Preference for build order $PREF \in \{'legs\_first', 'back\_first', 'none'\}$

procedure Q-LEARNING($\mathcal{X}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{P}$, α, γ)
  Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
  Initialize $freq\_human\_actions : \mathcal{S} \to \mathbb{R}^{size(\mathcal{A}_{human})}$ to $[0] * size(\mathcal{A}_{human})$
  while Q is not converged or max number of iterations has not been reached **do**
    Start in state $s \in \mathcal{S}$
    $took\_communicative\_action \leftarrow False$, $human\_reply \leftarrow -1$
    while s is not terminal **do**
      $a_{human} \leftarrow$ correct action in accordance with the HTM and preference set
      Calculate π according to Q and $70\%-30\%$ exploration-exploitation strategy
      if exploiting via $\pi(s) \leftarrow \arg\max_{a_{robot}} Q(s, a_{robot}, a_{human})$ **then**
        if $took\_communicative\_action$ is $True$ **then**
          $\pi(s) \leftarrow \arg\max_{a_{robot}} Q(s, a_{robot}, human\_reply)$
        else
          $\pi(s) \leftarrow \arg\max_{a_{robot}} Q(s, a_{robot}, a_{human}), \forall a_{human} \in \mathcal{A}_{human}$
      $a_{robot} \leftarrow \pi(s)$
      $took\_communicative\_action \leftarrow False$, $human\_reply \leftarrow -1$
      if a_{robot} is *communicative_action* **then**
        $took\_communicative\_action \leftarrow True$, $human\_reply \leftarrow a_{human}$
      $freq\_human\_actions[s][a_{human}] \leftarrow freq\_human\_actions[s][a_{human}]+1$
      $R(s, a_{robot}, a_{human}) \leftarrow$ REWARD$(s, a_{robot}, a_{human}, freq_human_actions)$
      $r \leftarrow R(s, a_{robot}, a_{human})$ ▷ Receive the reward
      $s' \leftarrow P(s, a_{robot}, a_{human})$ ▷ Receive the new state
      $Q(s', a_{robot}, a_{human}) \leftarrow (1 - \alpha) \cdot Q(s, a_{robot}, a_{human}) + \alpha \cdot (r + \gamma \cdot Nash_{robot}(s', Q)$
      $s \leftarrow s'$
  return Q
procedure REWARD(s, $\vec{a} = (a_{robot}, a_{human}, freq\_human\_actions)$)
  if a_{robot} is *communicative_action* **then**
    $state\_certainty \leftarrow freq\_human\_actions[s][a_{human}]/sum(freq\_human\_actions[s])$
    if $state\_certainty < 1$ **then** ▷ 1 represents low uncertainty states
      return 15 ▷ Encourage *communicative* actions in high uncertainty states
    else
      return -0.002 ▷ Discourage them in low uncertainty states
  if s is a final state **then return** 10
  if s' is NOT a valid next state in accordance with the HTM **then return** -100
  return 1

Table 5.3: Example trajectory recovered after the *MAB-RL Robot Communicative Actions* algorithm is trained

HTM Possible Leaves	Robot and human action pair
gather parts leg 1, gather parts leg 2, gather parts leg 3, gather parts leg 4	(communicate, no action human)
gather parts back right, *or* gather parts back left	
gather parts leg 3, or gather parts leg 4	(bring dowel, no action human)
gather parts leg 4	(communicate, no action human)
gather parts leg 4	(bring back bracket, no action human)
gather parts leg 1, *or* gather parts leg 2, gather parts leg 3	(communicate, bring dowel)
gather parts leg 1, *or* gather parts leg 2	(bring front bracket, bring dowel)
gather parts leg 2, *or* gather parts leg 3	(communicate, bring dowel)
gather parts leg 2	(bring front bracket, bring dowel)
gather parts leg 3	(bring dowel, no action human)
gather parts leg 3	(no action robot, bring screwdriver)
gather parts leg 3	(bring back bracket, no action human)
assemble seat	(bring seat, no action human)
assemble seat	(hold, no action human)
gather parts back right, *or* gather parts back left	(communicate, bring back bracket)
gather parts back right	(bring dowel, bring back bracket)
gather parts back left	(bring back bracket, bring dowel)
gather parts back top	(bring long dowel, bring back)
assemble back	(no action robot, no action human)

to any of the *MAB-RL* algorithms presented in this chapter. Hierarchical reinforcement learning (HRL) [108] helps speed up learning by allowing the re-use of different types of supportive behaviors for different subtasks within the same domain. HRL also helps with abstracting away from the state of the subtask details about the implementation of the specific supportive behaviors. We present our algorithm in Algorithm 6.

As we present in Subsection 5.2.2, an important paradigm proposed and studied in the HRL community is the formulation of partial policies into options [40]. Options generalize primitive actions to include temporally-extended courses of action [40]. An option is defined as a tuple $\langle \mathcal{I}_{\mathcal{O}}, \pi, \beta \rangle$, where $\mathcal{I}_{\mathcal{O}}$ is the initiation set of the option (the option is available in a state s if and only if $s \in \mathcal{I}_{\mathcal{O}}$), $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ is the policy to be executed, and $\beta : \mathcal{S}^{+} \rightarrow [0, 1]$ is the termination condition. When the agent takes the option, it chooses actions according to the policy π until the option terminates according to β. *HMAB-RL* uses options as temporally-extended options with variable number of primitive actions composing them.

Although options can be learned [117] and this is an exciting area of exploring in our future, it is common that designers of RL systems use their prior knowledge about the task to add a specific set of options, together with their policies (these can also instead be learned) [108]. Learning such options is beyond the scope of this thesis, but we find this an interesting future research direction. In Table 5.4 and Table 5.5, we define the set of options, together with their policies, which we add onto the primitive actions used in the *MAB-RL Human Watch* algorithm.

Table 5.4: Robot options and option policies for the *HMAB-RL Human Watch* algorithm

Option name	Initiation set	Policy	Termination condition
gather parts leg back type 1	$\{s \mid$ one of the *gather parts legs* states can be accessed from $s\}$	[*bring dowel, no action robot, bring back bracket*]	*bring back bracket* has been executed
gather parts leg back type 2	$\{s \mid$ one of the *gather parts legs* states can be accessed from $s\}$	[*bring dowel, bring back bracket*]	*bring back bracket* has been executed
assemble seat	$\{s \mid$ the *assemble seat* state can be accessed from $s\}$	[*bring seat, hold*]	*hold* has been executed
assemble back	$\{s \mid$ the *assemble back* state can be accessed from $s\}$	[*bring long dowel, no action robot*]	*no action robot* has been executed

Table 5.5: Human options and option policies for the *HMAB-RL Human Watch* algorithm

Option name	Initiation set	Policy	Termination condition
gather parts leg back type 1	$\{s \mid$ one of the *gather parts legs* states can be accessed from $s\}$	[*no action human, bring screwdriver, no action human*]	*no action human* has been executed
do nothing	$\{s \mid$ one of the *gather parts legs* states or the *gather parts seat* state can be accessed from $s\}$	[*no action human, no action human*]	*no action human* has been executed
assemble back	$\{s \mid$ the *assemble back* state can be accessed from $s\}$	[*bring back, no action human*]	*no action human* has been executed

5.6 Results

In this section, we present the results for the algorithms we contributed in Subsection 5.5.1, Subsection 5.5.2, and Subsection 5.5.3. Since all of the algorithms presented in this chapter can learn a valid trajectory for the task from one learning episode only, we evaluate our algorithms by computing the number of errors on random trajectories generated from the

Algorithm 6 *HMAB-RL* Human Watch Q-learning: Learn functions $Q_{primitives} : \mathcal{S} \times \mathcal{P} \rightarrow \mathbb{R}$ and $Q_{options} : \mathcal{S} \times \mathcal{O} \rightarrow \mathbb{R}$

Require:

Sates $\mathcal{S} = \{s_1, \ldots, s_{n_s}\}$

Agents $i \in \Upsilon$, we have $\Upsilon = \{robot, human\}$

Joint-primitives $\mathcal{P} = \{\vec{p_1}, \ldots, \vec{p_{n_p}}\}$, with joint-primitive $\vec{p_i} = (p_{i_{robot}}, p_{i_{human}})$

Joint-options $\mathcal{O} = \{\vec{o_1}, \ldots, \vec{o_{n_o}}\}$, with option $\vec{o_i} = (o_{i_{robot}}, o_{i_{human}})$

Primitives $\mathcal{P}_{robot} = \{p_{1_{robot}}, \ldots, p_{n_{robot}}\}$, Options $\mathcal{O}_{robot} = \{o_{1_{robot}}, \ldots, o_{n_{robot}}\}$

Primitives $\mathcal{P}_{human} = \{p_{1_h}, \ldots, p_{n_h}\}$, Options $\mathcal{O}_{human} = \{o_{1_h}, \ldots, o_{n_h}\}$

Reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, Black-box transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$

Learning rate $\alpha \in [0, 1]$, we set $\alpha = 0.2$, Discounting factor $\gamma \in [0, 1]$, we set $\gamma = 0.9$

Preference for build order $PREF \in \{'legs\_first', 'back\_first', 'none'\}$

procedure Q-LEARNING($\mathcal{X}$, $\mathcal{A}$, $\mathcal{P}_{robot}$, $\mathcal{P}_{human}$, $\mathcal{O}_{robot}$, $\mathcal{O}_{human}$, $\mathcal{R}$, $\mathcal{P}$, α, γ)
  Initialize $Q_{primitives} : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ and $Q_{options} : \mathcal{X} \times \mathcal{A} \rightarrow \mathbb{R}$ arbitrarily
  while Q is not converged or max number of iterations has not been reached **do**
    Start in state $s \in \mathcal{S}$
    while s is not terminal **do**
      Calculate π according to $70\% - 30\%$ exploration-exploitation strategy
      $option\_flag \leftarrow False$
      if exploiting **then**
        $\pi(s) \leftarrow \arg\max_{p_{robot}\ OR\ o_{robot}} (\arg\max_{p_{robot}} Q_{primitives}(s, p_{robot}, p_{human}), \arg\max_{o_{robot}} Q_{options}(s, o_{robot}, o_{human}))$
      else
        $\pi(s) \leftarrow$ random primitive p_{robot} or random option o_{robot}
      if picked option **then**
        $option\_flag \leftarrow True$
      $a_{robot} \leftarrow \pi(s)$
      $a_{human} \leftarrow$ correct action in accordance with the HTM and the preference set based on having knowledge of a_{robot}
      $R(s, a_{robot}, a_{human}) \leftarrow$ REWARD$(s, a_{robot}, a_{human}, option_flag)$ ▷ Compute the reward
      $r \leftarrow R(s, a_{robot}, a_{human})$ ▷ Receive the reward
      $s' \leftarrow P(s, a_{robot}, a_{human})$ ▷ Receive the new state
      if $option\_flag$ is $False$ **then**
        $Q_{current} \leftarrow Q_{primitives}$
      else
        $Q_{current} \leftarrow Q_{options}$
      $Q_{current}(s', a_{robot}, a_{human}) \leftarrow (1 - \alpha) \cdot Q_{current}(s, a_{robot}, a_{human}) + \alpha \cdot (r + \gamma \cdot Nash_{robot}(s', Q_{current})$
      $s \leftarrow s'$
  return Q

procedure REWARD(s, $\vec{a} = (a_{robot}, a_{human})$, $option\_flag$)
  if s is a final state **then return** 20
  if s' is NOT a valid next state in accordance with the HTM **then return** -100
  if $option\_flag$ is $True$ **then return** 5
  return 1

HTM. As in Chapter 4, we only use the HTM for evaluating the model. In this chapter, we also use it for giving out rewards to the system because we need to compute the validity of the next state starting from a current state and taking a (a_{robot}, a_{human}) pair, but this is only encoded in the reward, and it constitutes the reward signal that the human would give the robot in a real-world setting.

The way in which we compute errors is as follows. For each number of episodes, we train each of our algorithms three times, and for each, we randomly select five trajectories generated from the HTM. For each trajectory we test if the action the robot would take, together with the human action, leads the task to progress into a valid state (validity is computed taking into account both the HTM, and the supportive behavior preference set). If this is an invalid action, we count it as an error, otherwise we do not. We then proceed to the next state of the selected episode regardless of the next state to which the action pair would have led us. This ensures that our system is tested for any possible state we might end up in if, for example, the human decided to change something throughout the course of the task, or switch to a parallel node. The only exception to how we compute errors is the *HMAB-RL* algorithm, since the reason why we obtain speed-up for HRL in general is that we learn when we can use options instead of primitive actions. This means that in different intermediate states where options apply overall, the algorithm would not know what primitives to employ instead. Testing knowledge of primitives in this case would go against the goal of speeding up learning via HRL. Thus, in this case, when the algorithm decides to take an option, and this is a valid option, we allow the algorithm to proceed to the end of the option. We only do so if the part of the actual episode that was selected at random coincides with the states visited while taking the option, and move from that point onward. By doing this, we still follow the episode, and still allow for checking that the algorithm knows what to choose at each step. In terms of selecting the human action for testing, we do so in exactly the same manner that the individual algorithms do based on their respective action synchronization schemes.

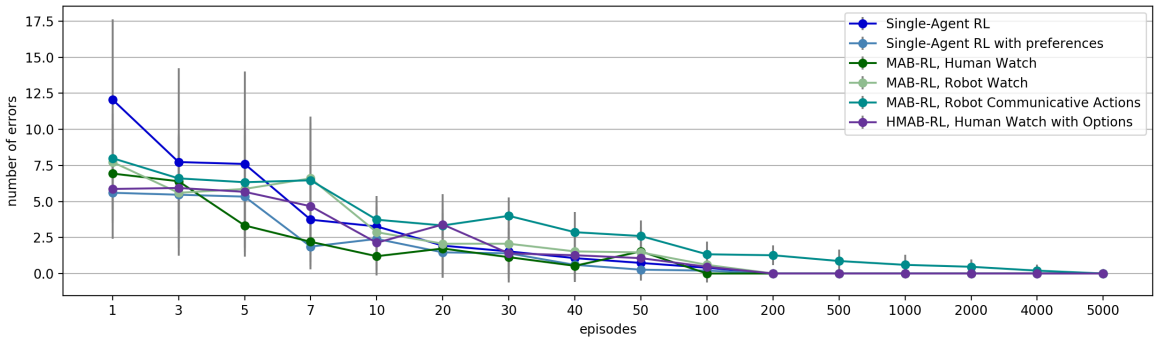


Figure 5.1: Comparison between the Single-Agent RL Baseline, the *MAB-RL*: Human Watch, Robot Watch, Robot Communicative Actions, and the *HMAB-RL* Human Watch algorithms. This graph shows the average number of errors per length of training sessions. Each session was ran three different times and the results were averaged. The error bars represent standard deviation.

In Figure 5.1, we present the results of all of our algorithms, over the course of learning phases consisting of 1 up to 5000 episodes. We notice that the best performing one is *MAB-RL Human Watch*, in terms of number of episodes. This algorithm achieves an average number of errors of 1.13 (s.d. 1.45) and 0.53 (s.d. 0.88) for 30 and 40 episodes' worth of training, respectively. This is on par with the human-level performance reported in Section 4.2, with an average of 0.97 (s.d. 1.12) for the same supportive behavior preference set. Once we reach 200 episodes for the training phase, our error is consistently 0. Compared to the model-based technique presented in Chapter 4 that needed a pool of either 500 or 350 task trajectories (depending on the presented model) in order to build a model of the task and 5 user demonstrations or labels to build the personalized supportive behavior model, this model-free technique necessitates more user-provided demonstrations or labels, but less demonstrations overall. Compared to the model-based HMM technique that only allows the robot to learn its own actions, both the *MAB-RL* and *HMAB-RL* paradigms allow for full flexibility in assigning different actions to the robot vs. the human, and changing these with ease and without any constraints. Below, we dive into deeper analyses and present sub-components of the data shown in Figure 5.1.

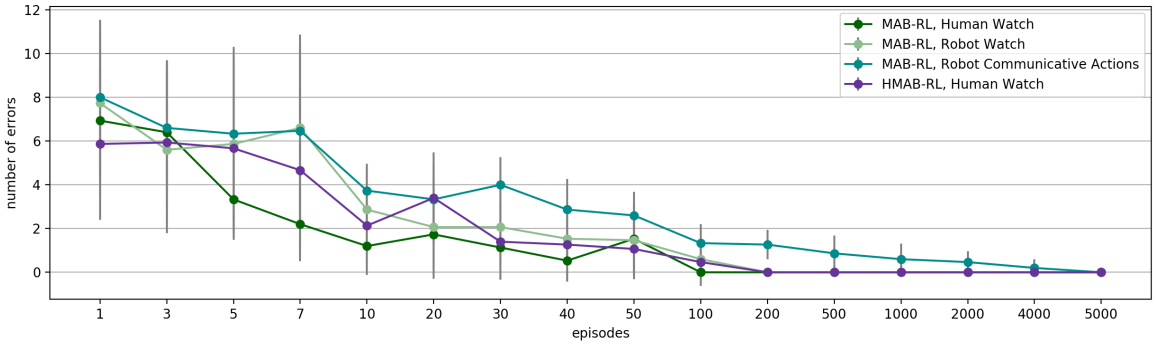


Figure 5.2: Comparison between the *MAB-RL*: Human Watch, Robot Watch, Robot Communicative Actions algorithms. This graph shows the average number of errors per length of training sessions. Each session was ran three different times and the results were averaged. The error bars represent standard deviation.

In Figure 5.2, we investigate more closely at the algorithms part of the *MAB-RL* paradigm. The curves suggest that the algorithms perform, from best to worst, in the following order: *MAB-RL Human Watch* (average 0.53, s.d. 0.88 for training 40 episodes), *HMAB-RL Human Watch* (average 1.27, s.d. 1.69 for training 40 episodes), *MAB-RL Robot Watch* (average 1.53, s.d. 1.02 for training 40 episodes), and *MAB-RL Robot Communicative Actions* (average 2.87, s.d. 1.41 for training 40 episodes). The algorithms reach performance on par human human-level performance at 40, 100, 100, and 500, respectively. The *MAB-RL Robot Communicative Actions* algorithm is the slowest, since it needs many more iterations in order to converge fully and learn exactly when it should apply the communicative actions. It is, however, the one that allows for the highest amount of flexibility in terms of action synchronization since it does not assume that one of the agents is able to observe the start of the other agent's action during one time step.

We now turn our attention towards the single-agent baseline, where we explore the effects of learning an added preference about the order of parallel nodes alongside the supportive behavior preferences. In Figure 5.3, we notice that the Single-Agent RL algorithm in which we incorporated ordering preferences (these are preferences with respect to the ordering of parallel nodes) performed better than the one with no preferences. We posit this

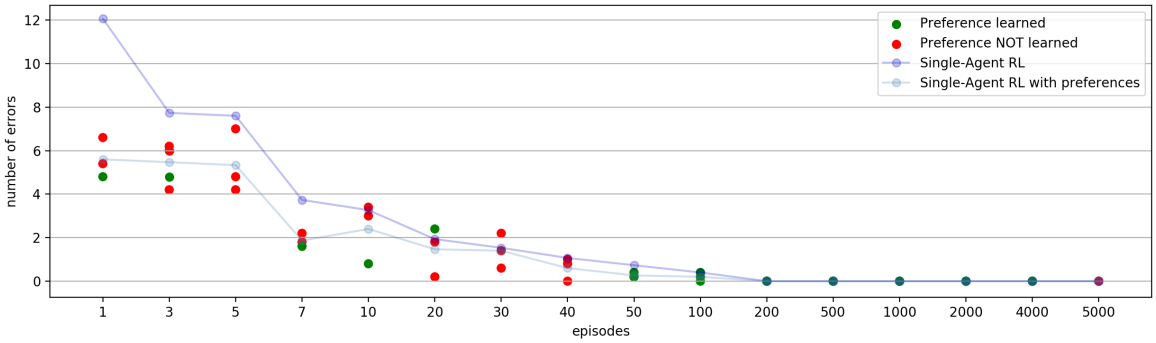


Figure 5.3: Comparison between the Single-Agent Vanilla Q-Learning and the Single-Agent Q-Learning with Preferences algorithms. This graph shows the average number of errors per length of training sessions. Each session was ran three different times and the results were averaged. The error bars represent standard deviation. For each of the training session lengths, we also plotted each of the three averages resulted from training for that length three times. We plotted with green those points where the ordering preferences were successfully learned, and with red those points where the ordering preferences were not successfully learned.

is because we introduce more information in the reward signal during the training phase, and this helps even in the cases where we are not successful at learning the actual ordering preference. The ordering preference we set out to learn in this experiment was executing the legs before the back part of the chair. In the figure, we plot with green those trainings that resulted in successfully learning this ordering preference, and with red those that resulted in failure. The Single-Agent RL with no preferences executes on par with human-level performance at 40 episodes, with an average of 1.07, s.d. 1.57, and the Single-Agent RL with Ordering Preferences does so at 40 episodes, with an average of 0.6, s.d. 1.20.

Finally, we investigate the difference between the *MAB-RL* and *HMAB-RL* Human Watch algorithms as part of Figure 5.4. Although when looking only at the training session length in terms of the total number of episodes, *MAB-RL* seems to perform better, adding options actually speeds up learning in terms of the length of the episodes themselves and the final total length of the training sessions. As the figure highlights, the *MAB-RL Human Watch* algorithm performs on par with human-level performance with an average of 0.53,

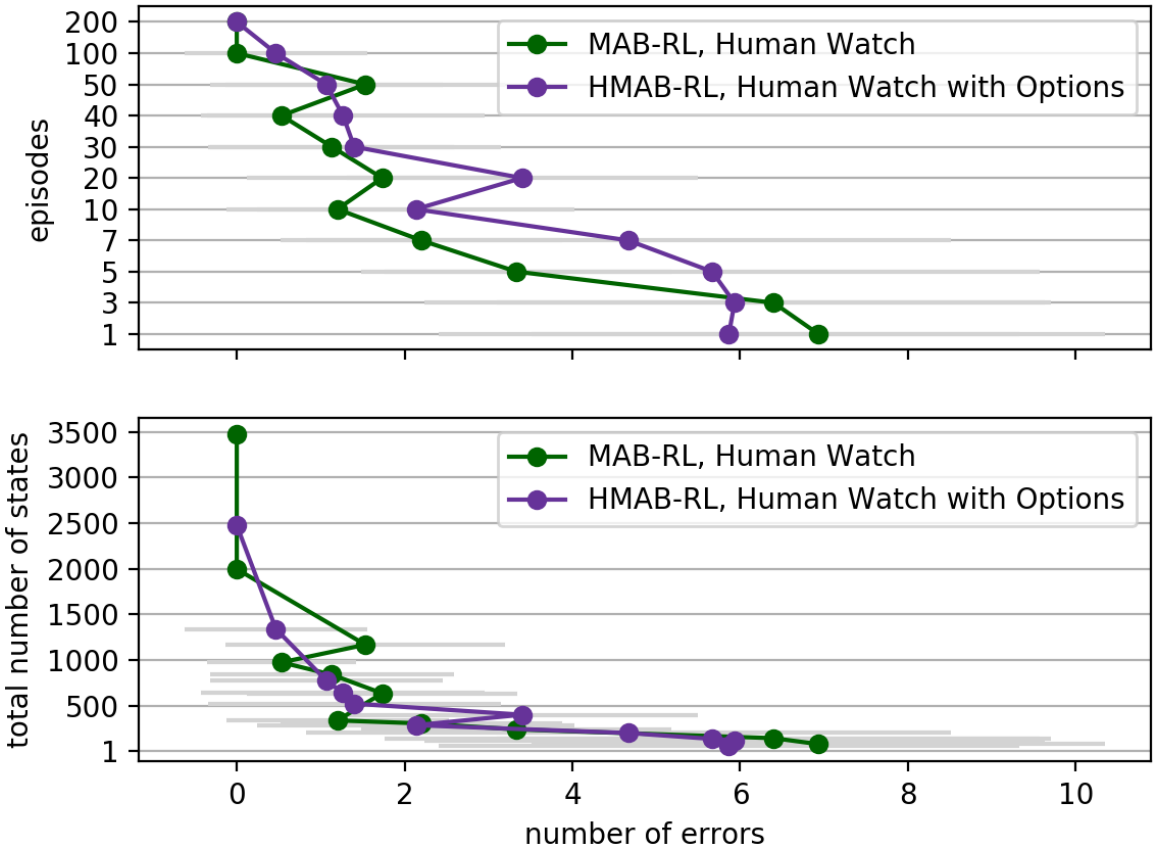


Figure 5.4: Comparison between the *MAB-RL* Human Watch and the *HMAB-RL* Human Watch algorithms. This graph shows the average number of errors per length of training sessions. The top graph shows the length of the training sessions in terms of the total number of episodes, while the bottom graph shows the length in terms of the averaged number of states visited (it takes into account the length of each epsiode). Each session was ran three different times and the results were averaged. The error bars represent standard deviation.

s.d. 0.88 when trained for 40 episodes, resulting in an average of 974.33 states visited throughout the 40 episodes. However, the *HMAB-RL Human Watch* algorithm does so with an average of 1.07, s.d. 1.39 when trained for 50 episodes, but resulting in an average of only 776.00 states visited throughout the 50 episodes.

These results show the power of employing macro-actions that can be re-used across the task. This improvement is caveated by the fact that the types of options present in the system affect how much speed-up we end up experiencing. If the options do not get re-used enough, the added complexity of implementing them and learning when to use an

option vs. a primitive action might not be worth a potentially marginal improvement in the total number of states visited. Furthermore, in some situations, this improvement is not present. In our case, this manifests when the *MAB-RL* algorithm performs well with a lower number of episodes due to a particularly good run in covering a wider variety of trajectories. We constructed our options so that they do not render the primitive actions completely unhelpful; however, the more primitive actions the options encompass and the more often the options can be re-used as part of the task, the higher the benefit. Thus, employing such macro-actions represents a trade-off between the improvement we hope to experience and the added complexity of the model.

5.7 Conclusions

In this chapter, we have introduced two novel paradigms we coin *MAB-RL* and *HMAB-RL*, and presented four algorithms representing different variants based on these paradigms. We compared this model-free technique with the model-based HMM we presented in Chapter 4, and showed that depending on the scenario we are faced with, we can trade off number of user demonstrations or labels for total number of demonstrations. Our techniques in this chapter make use of the multi-agent idea, but represent a variation on this, in which we consider the human an agent in the environment, but one whose actions we do not control. We also introduced a hierarchical approach that speeds up learning, which extends our multi-agent based paradigm.

Some of the limitations of this work, which represent interesting directions for future research, include implementing these algorithms on the Baxter research robot and performing another user study similar to Chapter 4, and looking into learning the options and the option policies so that we can speed up learning even further, and move towards researching transfer learning employing the learned options.

Chapter 6

The Multiple Facets of HRC: Social Collaboration

In this chapter, we focus on the social aspect of collaboration and on the importance of endowing collaborative robots with capabilities for maintaining users engaged and motivated throughout the interaction. We present studies that show how robots that aim to offer assistance to users, whether during verbal or physical interactions, can employ a variety of tools and behaviors, ranging from verbal communication for improved social presence (Section 6.1), to using different embodiments to engender improved perceptions from users (Section 6.2), to studying behaviors that engender effective interactions with robots (Section 6.3). Lastly, we present an application of a robot that exhibits such tools during an extended interaction, which presents evidence for relationship-building (Section 6.4). [1]

1. In Appendix B, we present a reinforcement learning system intended for a long-term study aimed at maintaining motivation through collaborative interaction with a robot. The studies we present in Section 6.1 and Section 6.2 explore particular tools that we can effectively use for such a robot to successfully and smoothly interact with a human over an extended period.

6.1 Verbal Communication for Social Presence in Collaborative Environments

The ability of social agents, be it virtually-embodied avatars or physically-embodied robots, to display social behavior and interact with their users in a natural way represents an important factor in how effective such agents are during interactions. In particular, endowing the agent with effective communicative abilities, well-suited for the target application or task, can make a significant difference in how users perceive the agent, especially when the agent needs to interact in complex social environments. This can be extremely valuable in contexts such as HRC, where we wish the robot to provide effective assistance to users throughout the execution of a task. The work we present in this section is based on [118] [2].

In this work, we consider how two core input communication modalities present in human-robot interaction—speech recognition and touch-based selection—shape users' perceptions of the agent. We design a short interaction in order to gauge adolescents' reaction to the input communication modality employed by a robot intended as a long-term companion for motivating them to engage in daily physical activity. A study with n = 52 participants shows that adolescents perceive the robot as more of a friend and more socially present in the speech recognition condition than in the touch-based selection one. The results highlight the advantages of using speech recognition as an input communication modality even when this represents the less robust choice, and the importance of investigating how to best do so.

Artificial agents, such as virtually-embodied characters or physically-embodied robots, can take on the role of help givers, companions, teachers, coaches, and so on [119], [120]. In this work, we focus on a physically-embodied robot intended as a long-term companion

2. E. C. Grigore, A. Pereira, I. Yang, D. Wang, and B. Scassellati, "Talk to me: verbal communication improves perceptions of friendship and social presence in human-robot interaction", in Proceedings of the 16th International Conferences on Intelligent Virtual Agents (IVA), Los Angeles, USA: Springer, 51–63. Best paper finalist.

that keeps adolescents motivated to engage in daily physical activity throughout time. The importance of developing methods for keeping adolescents engaged in physical activity is stressed by data revealing alarmingly low numbers of adolescents who engage in levels recommended by health guidelines [121]. Furthermore, this occurs at an age when physical activity holds essential benefits for healthy growth and development [122]. Given that the constant support needed for behavior change [123] is either not readily available or costly, employing a social agent as a companion would provide valuable social support to keep adolescents motivated to exercise routinely.

In order for such a companion to successfully provide its users with social support over time, enabling smooth and natural interactions through an effective communication modality is essential. We thus investigate the effects of the communication modality on users' perceptions of the agent. To this end, we conduct a study to observe adolescents' reactions to different ways of communicating with the robot (Figure 6.1 shows a participant during a session). Investigating how users' perceptions of the robot are affected by communication modality can guide us towards creating better and more effective agent interactions. In particular, we are interested in how users' perceptions of the agent change in terms of friendship and social presence factors. These factors are particularly relevant for creating engaging and compelling interactions, and are important to assess prior to the long-term deployment intended for the robot companion.

Given that verbal communication is at the core of how people interact with each other, one of the most natural ways of designing a human-robot communication interface is employing speech. Speech recognition provides a basic and natural way of conveying information to the robot. Even though it has seen serious advances in recent years, speech recognition technology still suffers from unreliable performance in noisy acoustic environments and when faced with speaker and language variability (especially when working with children [124]), as well as from difficulty handling free-style speech [125]. Due to the challenges faced by utilizing speech recognition reliably, many interfaces for communica-

Figure 6.1: Participant interacting with the robot using our smartphone interface.

tion with agents use touch-based inputs via tablet or smartphone devices [126], [127]. The use of such interfaces makes for more robust and dependable communication, but might take away from the benefits of having a more natural interaction.

This work investigates how adolescents' perceptions of the robot change when speech recognition versus touch-input selection is used to communicate with the agent. Results reveal that, even though the former was more error-prone and less robust than the latter, users perceived the robot as more of a friend and more socially present in the speech recognition condition than in the touch-based selection one. This highlights the importance of employing a more natural way of communication, even when this represents the less robust choice.

6.1.1 Related Work

Speech recognition has advanced significantly in recent years, with numerous applications deployed on a large scale. Smartphones represent some of the most widely used devices that employ this capability, with 3.4 billion subscriptions in 2015 set to double by the year 2021 [128]. Most smartphones come equipped with intelligent personal assistants that help users complete day-to-day activities, from using social media platforms to searching for travel information. Such assistants take inputs in the form of voice, images, or contextual information to provide useful answers, typically in the form of natural language. Through the widespread use of these devices, owners have become acquainted with the power provided by speech recognition, making for smooth and natural interactions with their devices.

Speech recognition systems, however, still suffer from lack of robustness beyond constrained tasks and are not reliable when used in noisy environments. The highest performing systems in research struggle to obtain word error rates (WER) lower than 10%, and employ strategies that might prove to be unfeasible for most real-world applications [125]. Furthermore, data on the performance of commercially available speech recognition services show considerable variability with respect to performance metrics, with WERs ranging from 15.8% to 63.3% [129]. In this work, we wish to investigate whether employing speech recognition to enable communication between a robot and its users is of value even when this represents the less robust choice.

Speech Recognition Communication Interfaces

Speech interfaces have been used for communicating with social agents in different contexts and application domains, given the natural way of interaction such interfaces can provide. A salient research area in which speech recognition is used is that of creating multimodal human-robot interaction interfaces. Such interfaces include verbal communication and visual perception modules, among others, and employ speech recognition as an important part of the interface allowing users to communicate with the robot [130], [131].

In a study focusing on teaching a robot a complex task [132], the robot learns the task by both observing a human perform it and by interpreting the speech used by the person while doing so. The authors emphasize the utilization of speech recognition in human-robot interfaces to create natural and familiar ways for people to interact with robots, which could more quickly lead to their acceptance in human environments like homes and workspaces. Alongside providing a natural mode of communication, other advantages include more easily handling situations when a person's hands or eyes are occupied, benefits for enabling communication between robots and handicapped persons, and the use of the ubiquitous mobile devices discussed above that allow for two-way voice communication [133].

Although there exist a number of studies investigating the effect of output communication modalities on users' perceptions of agents (i.e. using different interfaces to convey information from the agent to the user such as text-to-speech, text, etc.) [134], [135], research on input communication interfaces is more scarce and is the area where we focus our current efforts.

Touch-based Selection Communication Interfaces

Faced with the challenges surrounding the reliable use of speech recognition, a growing number of studies have started utilizing touch-based selection input as the main form of interaction between humans and robots. In the past ten years, we can observe a clear trend in the increasing number of studies that make use of mobile devices or tablet touchscreens for this purpose. For example, the Nao or DragonBot robots have commonly been paired with touchscreen tablets to provide a context for human-robot interaction [126], [120].

Other studies have used mobile devices and touch interfaces to communicate with robots. In such a study, authors used a smartphone as a means of interacting with Nao and teaching it about new objects [127]. In a study exploring the concept of enjoyment in a human-robot interaction scenario with the elderly, authors used a touchscreen interface in place of a speech recognition system due to increased reliability and smoother interac-

tions [119].

The research presented above provides interesting insights into using different kinds of devices and interfaces for communication in human-robot interactions. However, researchers have not yet explored the tradeoff between employing a more natural versus a more robust interface for enabling communication between a user and a robot with respect to users' perceptions of the agent. This work explores this tradeoff by investigating perceptions of friendship and social presence engendered by the use of different input communication modalities.

6.1.2 Methodology

Interaction Context

The interaction context for this study is that of a robot companion motivating adolescents to engage in daily physical activity. The companion is intended for long-term use, and the single-session study presented here constitutes the first interaction users would be going through during a longer-term study. During this session, the robot walks participants through its back-story and explains the different motivational strategies it would be using over time. The back-story consists of the agent taking on the role of a "robot-alien" whose space ship broke down on Earth and who needs help from the user in order to return home. The user can help the robot by exercising routinely and transferring "energy" to the robot by doing so. In a long-term scenario, this is accomplished by providing users with a wristband device that measures the physical activity level they engage in daily, which the robot can connect to in order to gain "energy points".

The back-story is created in order to build an engaging, compelling, and persuasive interaction by linking elements of the story (e.g., the fact that all inhabitants of the robot's planet have a high level of knowledge of physical activity) to ways in which the robot would help the user. In a long-term deployment, the agent can accomplish this by employ-

ing four different motivational strategies—cooperative persuasion, competitive persuasion, conveying information about physical activity through lessons and quizzes, and promoting self-reflection [136]. These motivational strategies are reinforced by providing users with a smartphone they can carry around with them during the day, while away from the physical robot. Participants also use the smartphone during their short, daily interactions with the physical robot in order to communicate with it.

Study Design

The user study we present herein examines the effects of speech recognition versus touch-based selection on users' friendship and social presence perceptions of the robot within the interaction context described in Subsection 6.2.2. Each session involves a participant interacting with the robot for six-to-nine minutes and being engaged in a dialogue on the topic of physical activity motivation for adolescents. The interaction is structured in the form of a dialogue controlled by the agent in which the robot either speaks to the participant or asks a question and waits for a response. Participants interact with the robot via a smartphone application that displays the appropriate interface based on whether the robot or the person is speaking, as well as based on the condition to which the participant is assigned. The conditions represent the use of two different input communication modalities: (1) speech recognition—when they are prompted through the smartphone interface, participants respond to the robot's questions by saying one of the answer choices displayed on the screen and their answers are evaluated through speech recognition, and (2) touch-based selection—participants respond to the robot's questions by using their finger to drag a button onto one of the answer choices displayed on the smartphone screen. We hypothesize that:

H1: Users would perceive the robot as more of a friend in the speech recognition than in the touch-based selection condition.

H2: Users would perceive the robot as more socially present in the speech recog-

nition than in the touch-based selection condition.

Measures

The two main dependent variables (DVs) we examine are users' perceptions of the robot in terms of friendship and social presence. The two measures are based on standardized questionnaires measuring friendship [137] and social presence [138]. We examine subscales of both measures to investigate the effects of the input communication modality on users' perceptions of the agent. The two main DVs constitute important factors to consider in any human-robot interaction scenario in which we wish to create an engaging and compelling interaction and engender a positive rapport with an artificial agent. This also extends to robots that we envision providing physical assistance to human workers throughout physical tasks.

Friendship. We employ the McGill Friendship Questionnaire (MFQ) [137] concerning a subject's assessment of the degree to which someone fulfills six friendship functions. Although friendships, like other relationships, vary in quality, research suggests it is possible to assess specific qualities, and the questionnaire used herein was created to define theoretically distinct friendship functions that distinguish between friends and non-friends, and that are associated with affection and satisfaction [137]. We believe that it is fundamental to engender the existence of such elements during interactions with social agents, whether they be long- or short-term. These factors are key to creating positive impressions of the agent, which is especially important during initial rapport development in relationships [139]. The more users perceive that the agent can fulfill friendship functions, the more likely they are to start building positive rapport with the agent. This, in turn, can help with providing effective assistance to a user throughout a task, by maintaining a positive interaction alongside the physical support offered.

The relevance of using friendship as a measure of the ability of an agent to establish meaningful relationships with its users has already been highlighted in a short-term study

[140]. We employ this measure in a similar way to inform this ability of the agent for the intended long-term use of the robot companion. The MFQ consists of 30 zero-to-eight Likert items in total, with six subscales composed of five questions each. The six subscales included in the questionnaire are: stimulating companionship, help, intimacy, reliable alliance, self-validation, and emotional security. Although not straightforward to delineate, some friendship subscales such as intimacy might appear more relevant to long-term interactions. Nevertheless, these subscales could provide insights into how verbal communication might affect a long-term interaction. Additionally, including the full set of subscales allows for future comparisons with long-term interaction results.

Social presence. Social presence is another core aspect of human-human interactions that we should take into account when developing interactions between users and robots or social agents. Social presence was initially defined as "the degree of salience of the other person in the interaction and the consequent salience of the interpersonal relationships" [141]. It is immediately apparent that users' perceived social presence of the robot is of high importance, given that we strive to develop robots that interact with their users in a convincing and social manner. An example of the power of creating strongly socially present robots is [142], where social presence is used to create believable and enjoyable board game opponents. It has been shown that an agent is more effective at being persuasive when perceived as socially present [143]. A robot companion that aims to keep adolescents motivated over time by employing persuasive motivational strategies would thus benefit greatly from being perceived as a strongly socially present party in the interaction.

The questionnaire used herein is based on [138], which consists of 36 zero-to-eight Likert items in total and defines social presence to include six important factors for the interaction. These factors of social presence represent the subscales used in the questionnaire: co-presence, attentional allocation, perceived message understanding, perceived affective understanding, perceived affective interdependence, and perceived behavioral interdependence. Social presence is directly applicable to short-term interactions and is routinely

employed in their assessment [144], [145].

Apparatus and Experimental Setup

In this study, we use a modified (to make programmable) version of MyKeepon, a commercially available variant of the Keepon robot. Keepon is a non-mobile robot with four degrees of freedom, designed for interaction with children [146]. The modified MyKeepon can be seen in Figure 6.1, presenting a green "alien" hat with antennae, consistent with the robot's back-story during the interaction.

Throughout the study, each participant sat at a table, with the robot placed on the table. A Kinect was positioned in front of the robot and covered with a black cloth to look integrated with the robot. The Kinect tracked the position of the user's head so that the agent would turn left or right, as needed, in order to mimic placing its attention on the participant as he or she moved. The user was also given a smartphone device with a custom-built application displaying the interface based on the condition. Figure 6.1 shows the experimental setup during an interaction session with an adolescent.

Participants We conducted a study with n = 52 participants, 26 male and 26 female, in local high schools. The subject population consisted of early adolescents and adolescents aged 14-to-16, with an average age of 15. Apart from the age, no other recruitment criteria were applied. Recruitment was managed with the help of the schools' staff and participants did not receive monetary compensation. We obtained ethical approval from the Human Subjects Committee at Yale University.

Procedure Participants were randomly assigned to either the speech recognition or the touch-based selection condition. The former included 26 participants, while the latter included the other 26. Users were given an assent form to read and sign prior to participation. Each participant was seated at a table, in front of the robot that was initially still and silent. We first gave the user a brief overview of the physical activity scenario and then handed

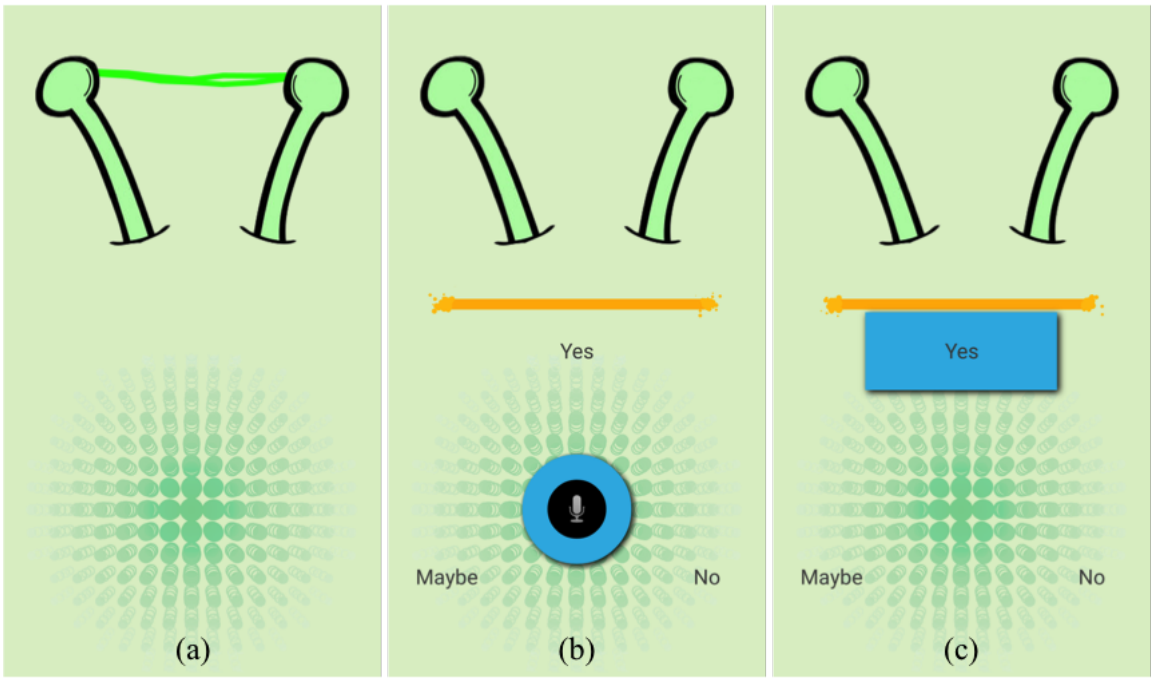


Figure 6.2: Interface for communication with the robot. Figures show the interface displayed for: (a) the robot speaking during both conditions, (b) the speech recognition condition, and (c) the touch-based selection condition.

him or her the smartphone.

Each participant was given a brief explanation on how to use the interface to communicate with the robot. Figure 6.2(a) shows the interface displayed across conditions when the robot is speaking to the user. Figure 6.2(b) shows the interface displayed in the speech recognition condition. Users in this condition would say one of the options displayed on the screen to answer a question. We employed the Google Speech Recognition API [147]. The system could handle synonyms and expressions with similar meaning to the words shown (e.g., "yeah", "sure", and "of course" would all be recognized as "yes"). When the system could not correctly identify the input or the participant would not use the interface correctly, the robot would prompt the user with a brief instruction based on the particular mistake. Figure 6.2(c) shows the communication interface displayed in the touch-based selection condition. Participants in this condition would use their finger to drag the button displayed on the screen onto one of the answer choices in order to respond. As long as they

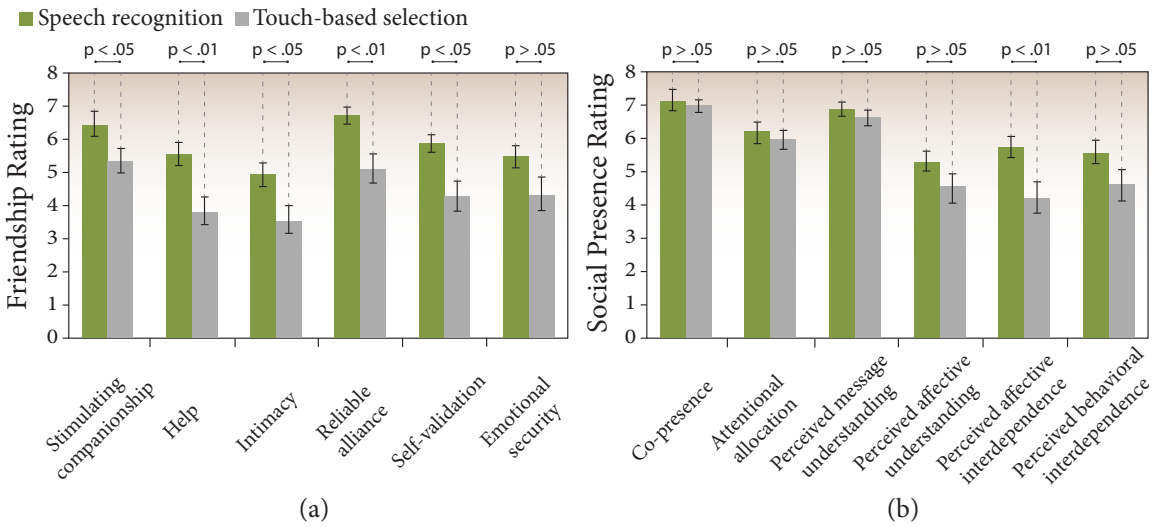


Figure 6.3: Mean ratings for (a) friendship subscales (p-values based on the Mann-Whitney U test statistic, with means and error bars depicted to visualize results for this data), and (b) social presence subscales (p-values based on the independent-samples t-test statistic). We consider Bonferroni-adjusted α levels with $p < .008$ for significance and $p < .017$ for marginal significance. All error bars represent $\pm 1SE$.

kept the button pressed, participants could change their mind and move the button from one choice to another. In this condition too, users were prompted with a brief instruction if they would not utilize the interface correctly.

The interaction itself consisted of the robot unfolding its back-story and talking about physical activity motivation. The interface would change from Figure 6.2(a) to either Figure 6.2(b) or Figure 6.2(c) (depending on the condition) when the robot would go from speaking to waiting for the participant's answer. The robot asked a total of six questions during each interaction, and users could choose among three simple entries to respond. The robot's replies to the different answer choices differed slightly to give users feedback that their particular answer was understood, but otherwise the script was fixed. The interaction was approximately seven minutes long. At the end of the interaction, participants were asked to fill out the friendship and social presence questionnaires.

6.1.3 Results

Here, we present the analysis of the friendship and social presence subscales, and discusses the robustness of the two conditions employed in the current study.

Friendship. We first evaluated the internal consistency for each of the six friendship subscales, resulting in highly reliable values. We thus computed the scores for the six subscales by averaging over values within each. The Shapiro-Wilk test of normality revealed the data is not normally distributed ($S - W = .93, df = 52, p = .005$). We thus employed the Mann-Whitney U test to compare users' ratings of the robot between the two conditions.

We performed the Mann-Whitney U test for each of the six subscales, using a Bonferroni-adjusted α level of .008 $(.05/6)$ for significance and .017 $(.1/6)$ for marginal significance. Participants rated the robot significantly higher in the speech input condition than in the touch input one for help ($U = 183, p = .004, z = -2.84$) and reliable alliance ($U = 186, p = .005, z = -2.79$), and marginally significantly higher for self-validation ($U = 200.5, p = .012, z = -2.52$). Based on the Bonferroni correction, there was no statistical significance in ratings for stimulating companionship ($U = 215, p = .024, z = -2.26$), intimacy ($U = 228.5, p = .045, z = -2.01$), or emotional security ($U = 246, p = .094, z = -1.68$), although trends do suggest higher ratings in the speech recognition than in the touch-based selection condition. These results are highlighted in Figure 6.3(a) and they reveal the strong effect of employing speech recognition within the context of fostering positive rapport during human-robot interactions. Here, we consider significance levels based on the Bonferroni correction, and display the actual p-values in Figure 6.3(a) for clarity of results.

Social Presence. We computed social presence subscale scores in a similar manner to the friendship scores, with internal consistency evaluations resulting in reliable and highly reliable values for each subscale. The Shapiro-Wilk test of normality suggested that normality of data is a reasonable assumption for the social presence data ($S - W = .97, df =$

$52, p = .146$), and so we employed the independent-samples t-test to compare users' ratings between the two conditions.

We performed the independent-samples t-test for the six subscales, using a Bonferroni-adjusted α level of $.008$ $(.05/6)$ for significance. Participants rated the robot significantly higher in terms of perceived affective interdependence in the speech recognition condition ($M = 5.72, SD = 1.58$) than in the touch-based selection one ($M = 4.17, SD = 2.31$), $t(50) = 2.83, p = .007$ (t-value reported for unequal variances), while the other five subscales (co-presence, attentional allocation, perceived message understanding, perceived affective understanding, and perceived behavioral interdependence) were not significant (Figure 6.3(b)). Although the friendship ratings reveal a stronger impact of the communication interface employed, social presence results also bolster the importance of employing a more natural and familiar mode of communication with robots.

Robustness. In order to assess the reliability of the two input communication modalities employed, we computed error rates for each. For both conditions, each session constitutes of the robot asking the user a total of six questions, for which the participant can choose among three simple, one-word entries to respond. In the speech recognition condition, we encountered two types of errors: (1) the system evaluates the user's speech as something other than the options available on the smartphone screen, and (2) there is no audible speech that can be evaluated by the system. Since each question response consists of one-word answers, we computed the word error rate (WER) for the system by dividing the total number of errors encountered (23) by the total number of questions asked (six questions per participant, with 26 participants in the speech recognition condition), and obtained a value of 14.74%. In the touch-based selection condition, the interface performed without errors, given the tailored nature of the interface design and the reliability of using a smartphone touch interface.

To ensure that the errors encountered in the speech recognition condition did not impact users' perceptions of the robot, we analyzed this data more in depth. Out of the 26

total participants in this condition, the system presented errors for 14 (with an average of 1.64 errors per user), and worked without errors for the remainder of 12. We thus applied the same statistical tests employed above for this condition only, using a binary coding scheme to divide the group into participants with and without errors. We again employed the Mann-Whitney U test for friendship subscales and the independent-samples t-test for social presence subscales, with the same significance levels used above. We obtained no significant effect for errors for either of the friendship or social presence subscales. This shows that the difference in users' perceptions of the robot is not influenced by errors present in the speech recognition condition.

Compared to available commercial systems, the speech recognition WER represents a fairly low error rate, but this is due to the nature of the constrained task and simple answers employed during the interaction. To note, however, is the fact that the speech recognition condition constitutes the less robust choice out of our two input communication modalities, and that this condition still engenders higher perceptions of the robot for participants. These results stress that it is important to consider the tradeoff between utilizing a more natural versus a more robust interface when deciding which input communication modality to employ for interactions with robots, or artificial agents more broadly.

6.1.4 Discussion and Conclusions

This work examines the effects of employing speech recognition versus touch-based selection as an input communication modality on users' perception of an agent. We conducted a study to investigate how the two modalities shape users' friendship and social presence perceptions of the robot within the context of physical activity motivation for adolescents. Our intuition was that, as a more natural means of communication, speech recognition would prove to engender stronger perceptions of friendship and social presence than touch-based selection, even when this represents the less robust choice.

Hypothesis H1 predicted stronger friendship perceptions of the robot in the speech

recognition than in the touch-based selection condition. We obtained statistically significant differences between conditions for help and reliable alliance, and marginally statistically significant for self-validation. Although we did not obtain statistically significant results for stimulating companionship, intimacy, and emotional security, we did notice a trend in the same direction. A possible explanation for why these subscales are only showing a trend could be due to their limited relevance to short-term interactions. We suspect that a longer exposure to our agent could result in significant results for these subscales as well, and consider this an interesting avenue for future research.

The second hypothesis, H2, is partly supported by the results. Co-presence, attentional allocation, and perceived message understanding engendered high perceptions of the agent for users across conditions, and so these subscales did not yield statistically significant differences. We did not obtain any significant differences for perceived message understanding and perceived behavioral interdependence. However, participants in the speech recognition condition perceived stronger affective interdependence than those in the touch-based selection one. Perceived affective interdependence is defined as "the extent to which the user's emotional and attitudinal state affects and is affected by the emotional and attitudinal states of the interactant." It represents an important facet of creating an engaging, persuasive, and compelling agent that aims to motivate adolescents. Having users perceive high affective interdependence when interacting with a social robot is fundamental to any type of agent, be it one that aims to entertain, motivate, help physically, or simply provide information.

The significant differences observed in users' perceptions of the robot are within the context of employing a smartphone as part of the input communication modality. We do so deliberately, in order to design a robot that can realistically be used as a long-term companion in adolescents' homes and employ widespread, commercially available technology (smartphones) to this end. Although comparing the conditions in this study to using speech recognition without a smartphone would help put the significant differences obtained into

context, it would introduce another independent variable, i.e. the presence or absence of the device. Nevertheless, we believe this comparison to be an interesting future direction.

The results yielded by the current single-session study inform us that, for a long-term application scenario, or a HRC context during which we aim for effective support, using natural language for communication with a social agent is worthwhile despite the presence of speech recognition errors. Given the general pertinence of the two scales we employed, we believe that these results can be applied to many other human-agent interaction domains where social interactions are key.

6.2 Differently Embodied Agents for Collaborative Environments

The question of whether to use a robot or a virtually-embodied character for applications in need of a socially intelligent agent depends on the requirements of the task at hand. To overcome limitations of both types of embodiment and benefit from advantages provided by both, we can complement a physical robot with a virtual counterpart. In order to link the two embodiments such that users perceive they are interacting with the same entity, the concept of “migration" from one embodiment to the other needs to be addressed. The work in this section based on [148] [3].

In this work, we investigate a particular aspect of this concept, namely how to best perform the triggering of migration, within the context of a physical activity motivation scenario for adolescents. We design two methods, a proximity-based method and a control, and compare their effects on adolescents’ perceptions of our agent. Results show that users perceive the agent as more of a friend and more socially present in the proximity-based than

3. E. C. Grigore, A. Pereira, J. J. Yang, I. Zhou, D. Wang, and B. Scassellati, "Comparing ways to trigger migration between a robot and a virtually embodied character", in Proceedings of the 8th International Conference on Social Robotics (ICSR), Kansas City, USA: Springer, 2016, November 1–3, 839–849. Best student paper finalist.

in the control condition. This emphasizes the importance of investigating different facets of entity migration for systems in need of employing both a physical and virtual embodiment for an artificial agent.

The field of Human-Robot Interaction (HRI) has highlighted the importance of a physical embodiment in contexts where we wish to use an artificial agent to provide supportive behaviors to a person. Research shows that having a physically embodied agent (as opposed to a virtually embodied one) has significant positive effects such as increased compliance [149], better learning gains [150], improved social facilitation [151], and higher perceived social presence [152]. However, physical robots still present a variety of problems, such as limited battery life, need for mechanical maintenance, wear and tear, and physical limitations in the range of behaviors displayed. Such problems are particularly significant for robots intended as social companions or as collaborative robots aiming to provide assistance since they are expected to be constantly present and available for interaction. Virtual embodiments are more robust, allow more complex interaction design and greater portability. An agent that can co-exist across multiple embodiments can be helpful in obtaining the benefits of both types of embodiment and mitigating their individual disadvantages.

Entity migration is defined as "the process by which an agent moves between embodiments" [153]. This concept has been studied in Human-Computer Interaction and HRI as an important paradigm allowing for the extension of a particular embodiment's abilities and limits. Migration has a huge potential to benefit applications seeking to employ social companions providing assistance to people in a multitude of domains, such as supervision, companionship, coaching, motivation, and so on. Migration can also strongly benefit robots that are meant to collaborate with users in industrial settings, where the aim is to provide humans with supportive behaviors throughout the execution of a physical task. This is due to the fact that each mentioned application has its own requirements for the length, duration, frequency, and form of interactions. Switching between a physical and virtual

embodiment can greatly help meet the necessities dictated by the task at hand at any point during the interaction.

We present an investigation into entity migration within the context of a robot companion whose long-term intended purpose is to keep adolescents motivated to engage in daily physical activity. Such an application domain for robot companions (especially relevant to adolescents, for whom physical activity plays a key role in healthy growth and development [122] and who show severely low adherence to physical activity levels recommended by health guidelines [121]) stands to fully benefit from employing entity migration. Thus, providing constant support necessary for behavior change [123] can be achieved through continual access to a virtual embodiment, while employing persuasive strategies necessary for motivation [136] would gain from temporarily switching to a physical embodiment. We argue that the way in which we perform migration from one embodiment to the other can engender different perceptions of the agent in users, providing a valuable tool for achieving more positive perceptions.

We thus explore the effects of the triggering phase of migration on users' friendship and social presence perceptions of the agent. Such perceptions are important for the supportive behaviors provided by our agent to be effective in a future-intended long-term deployment or in an HRC context. These perceptions can be used to leverage humans' tendency to develop social relationships with the objects they interact with, tendency that is especially strong with respect to biological and artificial agents. Leveraging this tendency is important in guiding users towards perceiving the robot companion as more than a mere tool, rather as a social and persuasive agent (capable of changing their attitudes, behaviors, or opinions) [154].

In this work, we develop a proximity-based method of performing migration triggering (a more novel, sensor-based approach) and a control method (a button-based approach). We test the effects of our migration triggering methods during a single-session study centered around our physical activity motivation scenario. We hypothesize that:

H1: Users will perceive the agent as more of a friend in the proximity-based condition (PBC) than in the control condition (CC).

H2: Users will perceive the agent as more socially present in the PBC than in the CC.

The results support both H1 and H2, revealing that we can effect stronger friendship and social presence perceptions via migration triggering methods that rely on more innovative approaches. This suggests that exploring different migration aspects has the potential of bolstering positive rapport building in HRI.

6.2.1 Related Work

One of the first projects investigating the concept of "migration" was The Agent Chameleon Project [155]. This work addressed research questions including agent identity [156], migration architecture design [155], and agent change of form based on deliberative reasoning [157]. Other projects have investigated how to implement migration between two robots [158], how to implement migration of multiple companions who share the same embodiment [159], whether migrating the memory of the agent would affect users' perception of consistent agent identity [160], and higher engagement of children with blended reality characters [161]. The current work investigates the triggering phase of migration and its effects on users' perceptions of the robot, an aspect of migration not present in this body of work.

In a study designed to construct an emotional relationship between humans and interactive systems, participants' tendency to "help" an agent is higher when the agent first migrates into a physical robot than when it remains in a virtual avatar displayed on a laptop [162]. This study indicates the relevance of using a physical embodiment to establishing a positive emotional relationship. The enhancement of user-agent relationship is also stressed in a study in which participants interact with a personal agent on a mobile PC that migrates to a physical robot guiding them on a tour [163]. The system is designed to

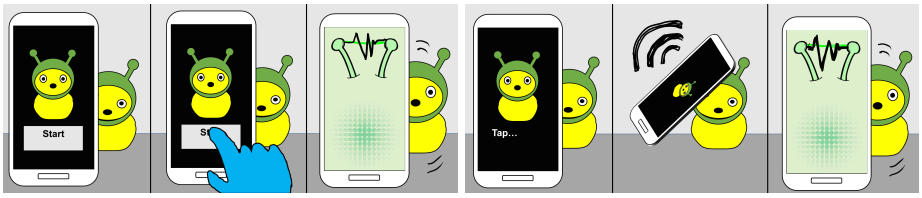
support the user gaining familiarity with the personal agent, but no aspects about users' perceptions of the agent are discussed. Finally, a study in which children interact with an artificial pet dinosaur that migrates between a virtual avatar on a smartphone and a physical embodiment finds that close to half of the participants perceive the two embodiments as corresponding to the same entity [153].

This work investigates an aspect that has not been previously explored in migration literature. The studies presented here do not tackle the triggering of migration and they either do not specify exactly how this is performed or they trigger migration through a button-based approach [153]. In this work, we focus on how changes in the triggering phase can affect how strongly users perceive the agent as a friend and as a socially present party in the interaction. Perceptions of friendship and social presence engendered by migration triggering also constitute a research direction that has not yet been investigated in this literature and represent an important step for gaining insights into how we can leverage migration to effect more positive perceptions of agents.

6.2.2 Methodology

Application Scenario

This study centers around the application scenario of a robot companion motivating adolescents to engage in daily physical activity. We present here a single-session study, representing the first interaction of the intended long-term deployment. The current study consists of the robot walking users through its back-story, while explaining the different motivational strategies it would employ during the longer study. The agent has the back-story of a "robot-alien" whose space ship broke down on Earth, and needs to gain "energy points" for repairing its ship and returning home. By exercising routinely, the user can transfer points to the robot. The transfer is mediated through a wristband device (which the robot can connect to) provided to users during the intended long-term deployment, measuring their daily physical activity level.



(a) Control condition (b) Proximity-based condition

Figure 6.4: Interaction design showing the agent's physical and virtual embodiments.

During the intended long-term study, the agent can employ four motivational strategies: cooperative persuasion, competitive persuasion, conveying information about physical activity via lessons and quizzes, and promoting self-reflection [136]. We sustain these strategies by providing users with a smartphone they can carry with them constantly, including when they do not have access to the physical robot. Participants also use the device during their short, daily interactions with the physically-embodied robot to communicate with it.

Forms of Embodiment

Physical. This embodiment is based on the Keepon robot, a non-mobile platform with four degrees of freedom [146]. We use a commercially available version named MyKeepon, modified to make programmable. This is a robust and simple platform allowing for a potential long-term use of our agent. This embodiment allows for a straightforward design of our 2-D virtual avatar as the robot has salient features that can be intuitively replicated, i.e. its shape and color. The green hat with the antennae add-on is reminiscent of an alien and is used to complement the agent's "robot-alien" back-story, as seen in Figure 6.4.

Virtual. This embodiment consists of a 2-D avatar with similar visual appearance to Keepon (so that users perceive they are interacting with the same entity across conditions). We display this avatar as part of an application installed on a smartphone that helps the user communicate with the agent. The application consists of an interface showing the avatar. When the user performs the migration triggering action, the interface

changes to display a set of antennae (reminiscent of the physical robot's hat) with a signal crossing between them. This animation always appears when the agent is talking, as suggested by the signal crossing between the two antennae. This sequence is depicted in Figure 6.4.

Migration Triggering Methods

Control condition (CC). This condition is depicted in Figure 6.4 (a) and represents a basic, button-based approach of performing migration. This method consists of the initial screen showing an image of the avatar together with a "Start" button beneath it, as can be observed in the first sub-image. The second sub-image highlights the action the user needs to accomplish to trigger the migration, namely pressing the button displayed under the virtual avatar. When the user does so, the avatar disappears from the screen with a fade-off animation and the physical robot wakes up and starts talking.

Proximity-based condition (PBC). This condition is depicted in Figure 6.4 (b) and represents our sensor-based approach. For this method, we use Near Field Communication (NFC). NFC is a technology that enables a device to communicate with another at a maximum distance of around 20 cm, and is widely used in commercially available smartphones. Triggering migration via NFC allows users to employ devices they are already familiar with for communicating with artificial agents. NFC has been used in gaming applications [164], but has not been previously employed in the context of HRI for migration.

We place the NFC tag on top of the robot's head, under its hat. Users see a similar initial screen as in the CC showing an image of the avatar, with a message underneath reading the text "Tap the robot's head with the back of the phone". When the user does so, the phone vibrates once and the avatar disappears by displaying a

swirling away animation and sound. This is represented in the second sub-image. The physical robot then displays the same "waking up" animation as in the CC and the interaction proceeds in the same manner thereafter. We decided to use a vibration effect when touching the robot. This is common in NFC-based applications as it gives users additional feedback that an action was triggered, something that is clear when pressing a button. We chose to change the animation from fade-off to twirling such that the agent seems to be disappearing into the robot's head. We did so to maximize the visualization of the PBC.

Participants

We conducted a study in local schools in Connecticut, USA. The study population consisted of early adolescents and adolescents aged 13-to-15, with an average age of 14. Participants were recruited with the help of the school staff and did not receive monetary compensation. The study consisted of $n = 26$ participants, 14 female and 12 male.

Procedure

We assigned participants randomly to one of our two conditions. We employed the CC for 13 participants and the PBC for the other 13. Each participant was given an assent form to read and sign prior to taking part. The participant was then seated at a table where the physical robot was already present yet still and silent. A researcher started by giving a brief overview of the project so that the participant would be aware of the physical activity motivation scenario.

Participants were not instructed on how to use the system, rather they were provided with a smartphone presenting a screen with our virtual avatar. The screen presented different instructions based on the triggering migration group participants were assigned to, with explanations to proceed. In the CC, participants saw a button labeled with the text "Start" underneath the virtual avatar, whereas in the PBC, participants read the text "Tap

the robot's head with the back of the phone" displayed underneath the virtual avatar.

The interaction proceeded as follows. Users were able to initiate the migration process from the virtual avatar to the physical robot at any point, as explained by the instructions for each condition. The migration occurred as described in Subsection 6.2.2. After the process was complete, participants interacted with the agent by using our custom dialogue interface. During this part, users were introduced to the story line of our scenario, as the agent talked about the importance of physical activity and different motivational strategies it would employ in a long-term scenario. An example of something the robot said is "On my home planet, we know a lot about physical activity", linking elements of its back-story to the physical activity topic. The agent asked six questions as part of the dialogue, and participants could choose between three simple entries to answer. I used the Google Speech Recognition API to allow users to verbally respond with one of three simple answer choices displayed on the smartphone. The different choices slightly affected the robot's responses so that participants were assured that the robot understood their answers. The script was otherwise fixed and straightforward. Overall, the interaction lasted approximately seven minutes.

After the interaction with the agent, participants filled out two questionnaires, and were interviewed by a researcher on the migration aspect they had just experienced.

6.2.3 Measures

During the study, we gathered both quantitative and qualitative data. First, we collected questionnaire data in the form of one-to-five Likert scales for both friendship and social presence. Second, we simultaneously recorded and wrote succinct transcriptions of users' answers to interview questions about migration.

The friendship and social presence measures we used in this work are the same as those presented in Section 6.1.

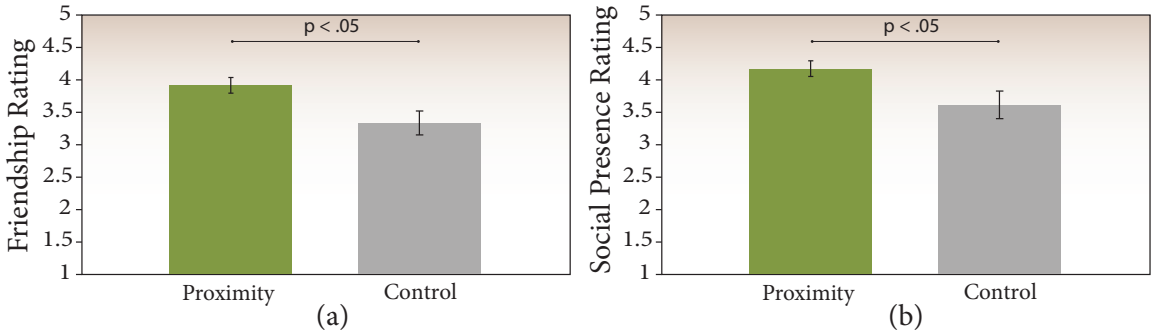


Figure 6.5: Participant ratings for the (a) friendship scale, (b) social presence scale. Error bars represent $\pm 1SE$.

6.2.4 Results

Here, we present the study findings and reports on both questionnaire and interview data.

Friendship Perceptions. We evaluated the internal consistency of the complete friendship scale of 30 items (highly reliable, Cronbach's $\alpha = .947$), as well as that of the six subscales (reliable and highly reliable values for each). We thus computed the score for the complete scale by averaging over all values, and scores for each of the six subscales by averaging over values within each. The Shapiro-Wilk test of normality suggested that normality of data was a reasonable assumption ($S - W = .97, df = 26, p = .707$), and so we employed an independent-samples t-test to compare friendship ratings between conditions.

Comparing the ratings for the agent between the two conditions yielded significant results for the complete friendship scale. Figure 6.5 (a) shows participants rated the agent significantly higher in the PBC ($M = 3.89, SD = 0.47$) than in the CC ($M = 3.35, SD = 0.74$), $t(24) = 2.24, p = .034$. This supports hypothesis H1 and emphasizes the effect of migration on how well users are able to relate with a social agent. Below, we look at the data more in-depth.

Given that we had a total of six subscales for each questionnaire, we performed the independent sample t-test using a Bonferroni-adjusted α level of $.008$ $(.05/6)$ for significance. The data yielded significant results for the intimacy subscale, with users rating the agent significantly higher in the PBC ($M = 3.63, SD = 0.58$) than in the CC

($M = 2.46, SD = 0.33$), $t(24) = 3.18, p = .005$ (t-value reported for unequal variances).

Social Presence Perceptions. We analyzed the social presence data in a similar way to the friendship data. We obtained highly reliable internal consistency for the complete scale (Cronbach's $\alpha = .874$), and reliable and highly reliable values for each of the six subscales. Scores for the complete scale and the subscales were computed in a similar fasion to the friendship scores. The Shapiro-Wilk test again suggested that normality of data was a reasonable assumption ($S - W = .97, df = 26, p = .560$), and so we employed an independent-samples t-test to compare social presence ratings between conditions.

Comparing the ratings for the agent between the two conditions yielded significant results for the complete scale. Figure 6.5 (b) highlights significantly higher ratings in the PBC ($M = 4.15, SD = 0.49$) than in the CC ($M = 3.60, SD = 0.81$), $t(24) = 2.08, p = .049$. This finding validates hypothesis H2 and supports the importance of investigating how migration effects social presence perceptions when creating an agent intended to engage with its users in a compelling way.

Interview Data. Alongside investigating the effects of migration triggering on users' friendship and social presence perceptions of the agent, we also conducted participant interviews. The goal was to gauge the impressions users had on the migration process, as well as whether they perceived having interacted with a single entity across conditions. Perceptions of interaction with a single entity are useful when employing migration in order to preserve the acquired positive rapport with the same agent (as perceived by users).

We asked users a range of questions including what they thought was happening when the avatar would disappear from the smartphone and the physical robot started to move, what they felt was being transferred between the two embodiments, and when they felt this transfer was happening. Based on the interview transcriptions, we grouped the answers in themes to gain a better understanding of whether users truly understood the concept of a single "entity" moving between the two embodiments. The main themes we obtained regarding how the agent was described are: "entity," "energy source," "soul," and "spirit."

Two interesting examples of responses include "I thought it was a spirit or an energy source because it was talking about inhabiting" and "The robot is a vessel for the entity." These answers show users' understanding that they were interacting with a single agent capable of migrating between different embodiments.

We also asked users the following question: "Did you think that the character in the mobile phone and the yellow robot in the physical world were the same entity?". This question was asked of all participants, regardless of the migration triggering group they were in. An overwhelming majority, 23 out of 26, of participants perceived they had interacted with a single entity during the study. When looking at the groups individually, we obtained the same results. 11 out of 13 participants in the CC and 12 out of 13 in the PBC said they perceived having interacted with a single entity.

Another finding from the interview data was the positive reaction to the migration aspect in terms of its perceived benefits. One participant noted "It is easier to carry around the phone than the actual robot," realizing how useful employing this technique would be in a long-term scenario such as the intended context of use for the robot companion.

6.2.5 Discussion and Conclusions

We have investigated entity migration between a physically- and virtually-embodied agent as an important paradigm for contexts requiring us to leverage advantages of both types of embodiment. Within the application context of physical activity motivation, we investigated how changes in the triggering phase of migration affect users' friendship and social presence perceptions of the agent.

This study reveals that participants rated the social agent significantly higher in the PBC than the CC for both friendship (H1) and social presence (H2). For the friendship measure in particular, we obtained a statistically significant difference for the intimacy subscale, with users rating the agent higher in terms of intimacy in the PBC than in the CC. The PBC employed a more physical trigger for migration through the use of an NFC

sensor requiring a more direct type of interaction. We speculate that for friendship, this trigger effected a more personal type of interaction causing users' perceptions of intimacy toward our agent to increase. For social presence, the proximity required by the method seems to have increased the feeling of "being with the agent", and thus engendered higher perceptions overall. These results show that we can use different aspects of the migration process to our advantage in order to engender higher perceptions of agents, which play an important role in building positive user-agent rapport. Exploring the various nuances of entity migration is thus an avenue well worth pursuing within the context of creating robot companions for application domains such as HRC, health, education, coaching, and beyond.

6.3 Designing Behaviors for Effective Interactions with Robots

In this work, we investigated the impact of endowing robots with mimicking capabilities during their interactions with humans. This work is based on [165] [4]. Behavioral mimicry - the imitation of gestures, postures, mannerisms, and other motor movements - is pervasive in human interactions [166]. Behavioral mimicry in particular is often unconscious and unintentional [167], [168]. For example, studies have shown participants mimicking confederates tapping their feet or touching their faces during interactions, even without realizing they were doing so [169]. Furthermore, research has also shown that participants found confederates more likable when the confederates mimicked the postures of the participants.

From a neurological standpoint, there exist certain parts of the brain responsible for this mimicry, known as mirror neurons or the "mirror system" in humans, first highlighted in macaques in 1992 [170]. Specifically, perceived actions can trigger the mirror system

4. A. Suman, R. Marvin, E. C. Grigore, H. Admoni, and B. Scassellati, "Prior behavior impacts human mimicry of robots", in Proceedings of the 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), New York, USA, 2016, August 26–31, pp. 1057–1062.

in humans [169]. Mimicry has been shown to create liking, empathy, and affiliation between interaction partners and act as the "social glue" that brings people together and bonds them [166], [171]. Also, research suggests that mimicry serves a prosocial communicative purpose [172], [166].

We find at least 4 reasons as to why mimicry should be studied in human-robot interaction. First, mimicry could be valuable in providing us with insight on how to design social robots and engineer their interactions with humans. Mimicry influences human-human interactions in a variety of ways, resulting in smoother interactions, smoother negotiations, more interpersonal trust, and more likability with their partners [173], [174], [175]. Designing our robots to better incorporate these considerations will make our robots better equipped for the types of interactions we envision them having in the future, especially as they move from functional spaces to social ones. Second, studying human-robot mimicry informs us about ingroup vs. outgroup divides between humans and robots and thus highlights how humans view robots as social partners. Also, knowing these divides gives us more information when we look to design our social robots with mimicry in mind. Psychology research tells us that there are restrictions on whom and when to mimic. For example, research has shown that people classify others they interact with as members of either an ingroup or an outgroup. When mimicry is performed by a member of someone's outgroup it can have negative effects on the interaction between those partners [176], [166], [177], [178]. Third, there are potential consequences that can arise from mimicking a robot, especially in a world where robots are closely integrated in our daily lives. For example, one can foresee possible concerns of a child or infant mimicking a robot in the home. This is particularly concerning if we have robots performing tasks that humans cannot or should not be performing. Fourth, understanding how human-robot interaction works for mimicry can act as a stepping stone to the understanding of the larger phenomenon of social contagion within human-robot interaction [166]. Social contagion is defined as mimicry with all aspects of social experience and not just motor behavior [166].

There has been work done showing that people who are mimicked by robots have found the robots more likable and their interactions with them smoother, more persuasive, and more positive [179], [166], [180]. On the other hand, very little work has centered around humans mimicking robots during human-robot interactions. Our work focuses specifically on whether or not humans actually mimic robots. This work is important because the effects we highlighted in the above paragraph apply not only to mimickees but also to mimickers [173], [174], [175]. The current study tests if humans mimic a robot doing a set of postures while interacting collaboratively during a task. The robot performs one of two behaviors and we measure the difference between how much a participant displays that behavior before and after the robot does it.

6.3.1 Related Work

There exists a rich body of literature that focuses on how humans mimic each other. One of the seminal papers in human-human mimicry, Chartrand & Bargh, shows that people do mimic one another during social interactions, and that mimickees find mimickers more likeable and have smoother and more positive interactions with them [169]. In particular, this study demonstrated mimicry in both directions, participants mimicking and participants being mimicked, while they described paintings alongside a confederate.

There has been work focusing on human-robot mimicry, particularly robots or virtual agents mimicking humans [179], [180]. For example, humans found a computer avatar more likable when it mimicked their head postures than when it did not [179], [166]. Work on embodied robots has shown difficulties with assessing human-robot interactions using a survey and the difficulties of capturing and mimicking behaviors between humans and robots [180]. There has also been preliminary support for more satisfactory interactions when facial expressions were mimicked by an ape-like robot, although the findings were in a pilot [181].

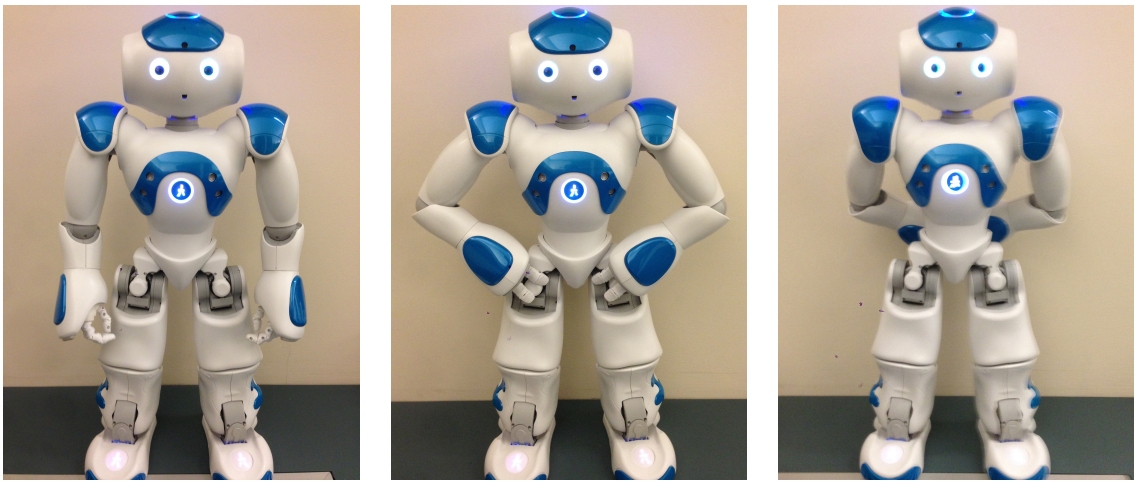
There is less work in the opposite direction, that of a human mimicking a robot. One

such study suggests a robot could elicit mimicry in humans [182]. It demonstrated through EEG that activation of the mirror neuron system in humans can occur through the perception of robot behavior, even without objects. This is significant as it informs us that the mirror neuron system and the perception-behavior link in humans is not uniquely limited to perceiving and reacting to human actions [182]. Other work found that humans can spontaneously match facial expressions of an embodied android present in the room. That study suggested that the salience of mimicry depended on how human-like the android presented is [183].

Overall, little work has been done with human-robot mimicry in general, particularly with humans mimicking robots. Our study draws from the setup of Chartrand & Bargh [169]. It focuses neither on human-human mimicry nor on robots mimicking humans. Rather it focuses on humans mimicking robots. Hofree's work broaches this area partially [183], but our study provides a more rigorous baseline for human mimicry of robots because our study focuses on a robot that minimizes human-like features and emotions. This allows us to test on a more basic set of behaviors, devoid of expressions or emotions such as anger or happiness. Our study also aims to minimize features that would increase the likelihood of mimicry such as goal to affiliate or pre-existing rapport, which have been shown to increase mimicry [166], [184], [185], [186], [187]. This makes our results very strong as they show mimicry without a catalyst. Finally, our study employs a robot that is humanoid but does not have a human-like face like the one in the Hofree study [183].

6.3.2 Methods

Our study borrows its experimental design from Chartrand & Bargh [169] but uses a robot in the role of the confederate. The participants were given the task of describing paintings together with the robot. This task is a collaborative and not competitive task because we did not want to confound the social interaction between participants and the Nao. The robot performs one of two behaviors while describing paintings. It either puts its hands on its



(a) idle (b) hips (c) back

Figure 6.6: Nao Robot Displaying Different Behaviors

hips or behind its back. We measure the time the participants put their hands on their hips or behind their backs both before and after the robot does so and we look for the difference between the two.

We initially ran a pilot study from which we observed there were two groups of people, those who spontaneously performed a behavior without the robot doing so and those who did not. In particular, we observed that those who did not spontaneously perform a behavior started doing so after seeing the robot display it and that those who did spontaneously perform a behavior started doing so less after seeing the robot display it. We thus define spontaneous exhibition of a behavior to be performing a specified behavior for any period of time prior to observing the robot perform said behavior.

Based on this, we hypothesized the following:

H1 People who do not spontaneously exhibit a behavior will perform that behavior more after seeing a robot perform it.

H2 People who do spontaneously exhibit a behavior will perform that behavior less after seeing a robot perform it.

For this experiment our robot platform was a Nao, a 58-cm tall humanoid robot, as

shown in figure 6.6(a), designed by Aldebaran on the Naoqi operating system [188]. Nao has 25 degrees of freedom, 2 cameras, 4 microphones, speakers, touch sensors, and an inertial measurement unit [188]. For the study, Nao's legs were employed for standing up, Nao's arms were used to assume 1 of the 2 aforementioned postures, Nao's touch senors were used to start a script of behaviors, and Nao's speakers were used to voice the painting descriptions. Python was used to program the Nao for the study.

The behaviors we chose for the Nao were hands behind back, as shown in figure 6.6(c), and hands on hips, as shown in figure 6.6(b). These behaviors were chosen because they were easily identifiable and were readily programmable on the Nao. We coded a participant putting hands on hips with 4 different designations and a participant putting hands behind back with 3 different designations. The different designations ensured we were able to cover all variations of the behaviors in the event they were displayed. For hands on hips the designations were two hands on hips, one hand on hip, hands in pockets, and hands in belt loops. We ultimately did not use hands in pockets or hands in belt loops because they did not accurately resemble the Nao's behavior. For hands behind back the designations were two hands behind back, one hand behind back, and hands in back pockets. We ultimately did not use hands in back pockets because it did not accurately resemble the Nao's behavior.

For our analysis we broke both behaviors into a strict and loose definition. For hands on hips, the strict definition matches the Nao's behavior of putting two hands on hips. The loose definition is a superset of this, with the participant exhibiting *at least* one hand on hip. For hands behind back, the strict definition matches the Nao's behavior of putting two hands behind back. The loose definition is a superset of this, with the participant exhibiting *at least* one hand behind back. Having a strict and loose interpretation allowed us to take into account participants who partially performed the behavior in our analysis.

Participants were first asked to fill out consent and video release forms. Participants were randomly assigned to a group that saw hands behind back or hands on hips during the first session (with the other behavior occurring in the second session). Participants

were brought in to a closed 420 cm x 300 cm room. The setup of the room can be seen in figure 6.7. Participants faced the Nao at a distance of 180 cm. The Nao stood on a platform raised 75 cm off the ground. Next to the Nao stood a 68 cm monitor which ran Python scripts for the Nao's behaviors through a local area network connected to the Nao's head and displayed a slideshow. The monitor was placed immediately adjacent to the Nao to ensure that participants could describe the paintings without missing the Nao perform its gestures. A camera was located at the back of the room and aimed at the participant's back. A camera was placed in the corner of the room facing the participant.

The participant was asked to read the first slide of directions while the experimenter turned on the cameras. The directions told the participant that the Nao would describe the painting for 1 minute and that the participant would then describe painting for 1 minute. The participant was told not to speak while the Nao was speaking. The participant was also told that he or she could repeat what the Nao said. Next, the experimenter started the slideshow and tapped the Nao's head sensor to start the Nao's Python scripts. The slides and scripts were synchronized. The Nao would turn its head to "see" the painting, turn back to the participant, and describe a painting (descriptions were pre-scripted) for 1 minute after which the participant was notified on-screen to do so as well. The Nao displayed no other idle behaviors while describing paintings or while the participant described paintings. This continued for 3 paintings total. At this point, the Nao performed the assigned behavior and maintained that posture for 3 more paintings while also turning its head to "see" the painting at the start of its turn. Thereby, half of the time (3 paintings worth) was before the Nao performed the behavior and half of the time was after the Nao performed the behavior. After 6 total paintings, the Nao returned to a crouch position and the experimenter replaced the Nao with a new one. The same process was repeated with 6 more paintings, except the other behavior was performed in session 2.

Employing two separate Naos came from the use of two different confederates in the Chartrand & Bargh study, which also had two different behaviors (touching the face and

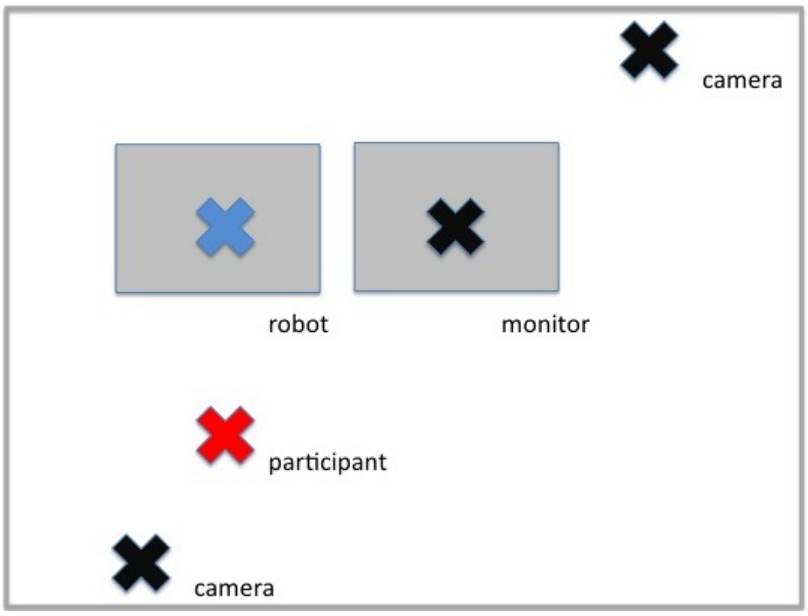


Figure 6.7: Experimental Setup of the Room

tapping the feet) [169]. The use of continuous postures (like hands on hips) rather than a discrete behavior (like tapping feet, which can be counted) was largely due to limitations of the robot, but we had little to reason to believe continuous postures would not be effective behaviors given their use in mimicry research [166], [189], [190].

The Nao's descriptions of the paintings were kept to be as simple as possible, with little to no emotion or interpretation [183]. The Nao also made no acknowledgement of the participants or their descriptions and the behaviors of hands behind back and hands on hips were chosen to minimize postural communication. This falls in line with our goals for understanding human mimicry of robots at its most basic level.

After both sessions were completed, the participant completed an online survey, received $5, and left.

Video of the participants were collected through a camera that captured the front of the participant and a camera that captured the back of the participant (this was necessary in order to validate any behaviors the participants portrayed behind their backs).

The videos were coded using ELAN 4.7.3. Both behaviors and variations of the behav-

iors were coded for. The coders were two of the researchers and they did not overlap on coding the videos. The videos were coded in their entirety, starting with the first minute and ending with the last minute of the video. Participants also filled out a survey comprising of Likert Scale questions on intelligence and likability, short answer questions on what they liked/noticed about the trial, and demographic questions.

Participants comprised of 47 undergraduates of which 43 were used in the final data analysis (4 participants experienced a technical problem, such as losing internet connection, during the trial). Participants were undergraduates recruited through direct appeal and fliers.

6.3.3 Results

The central question of this study is whether or not a robot can induce mimicry in humans. This experiment yielded quantitative results from video coding of the recordings of participants and from self-reporting through a survey of Likert scale and short-answer questions.

Statistical Methods Our determinations for statistical significance used the following guidelines and justifications. Given that the experiment was run as a within-participant study, a paired t-test was used. We used one-tailed t-tests because of our division of spontaneous and non-spontaneous participants based on their performance of a behavior being 0 or positive before the Nao performed the respective behavior. We did this because of the observations we were able to make as a result of our initial pilot, as mentioned in the Methods Section. Thereby, we expected the direction for the spontaneous group would be negative and the direction for the non-spontaneous group would be positive. This justifies our use of a one-tailed t-test.

For a given participant, the "before" period is defined as the time before the Nao performs the specified behavior and the "after" period is defined as the time after the Nao performs the specified behavior.

Behavior 1: Hands on Hips

For the **strict** definition of hands on hips, non-spontaneous participants performed the behavior more in the "after" period (M = 9280.71 ms, SD = 21273.90 ms) than in the "before" period (M = 0.00 ms, SD = 0.00 ms), $t(33) = 2.54$, $p = 0.008$.

For the **loose** definition of hands on hips, non-spontaneous participants performed the behavior more in the "after" period (M = 11592.13 ms, SD = 25920.62 ms) than in the "before" period (M = 0.00 ms, SD = 0.00 ms), $t(32) = 2.53$, $p = 0.008$.

The results for the non-spontaneous hands on hips behavior, both strict and loose, are shown in figure 6.8.

For the **strict** definition of hands on hips, spontaneous participants performed the behavior less in the "after" period (M = 42047.00 ms, SD = 59608.97 ms) than in the "before" period (M = 90381.00 ms, SD = 88705.54 ms), $t(8) = 2.29$, $p = 0.026$.

For the **loose** definition of hands on hips, spontaneous participants performed the behavior less in the "after" period (M = 39655.36 ms, SD = 64828.17 ms) than in the "before" period (M = 80540.36 ms, SD = 92746.96 ms), $t(10) = 2.11$, $p = 0.030$.

The results for the spontaneous hands on hips behavior, both strict and loose, are shown in figure 6.8.

Behavior 2: Hands Behind Back

For the **strict** definition of hands behind back, non-spontaneous participants performed the behavior more in the "after" period (M = 13420.10 ms, SD = 47423.54 ms) than in the "before" period (M = 0.00 ms, SD = 0.00 ms), $t(29) = 1.55$, $p = 0.066$.

For the **loose** definition of hands behind back, non-spontaneous participants performed the behavior more in the "after" period (M = 22292.48 ms, SD = 63531.61 ms) than in the "before" period (M = 0.00 ms, SD = 0.00 ms), $t(28) = 1.89$, $p = 0.035$.

The results for the non-spontaneous hands behind back behavior, both strict and loose, are shown in figure 6.9.

For the **strict** definition of hands behind back, spontaneous participants performed the behavior less in the "after" period (M = 49043.77 ms, SD = 99975.38 ms) than in the

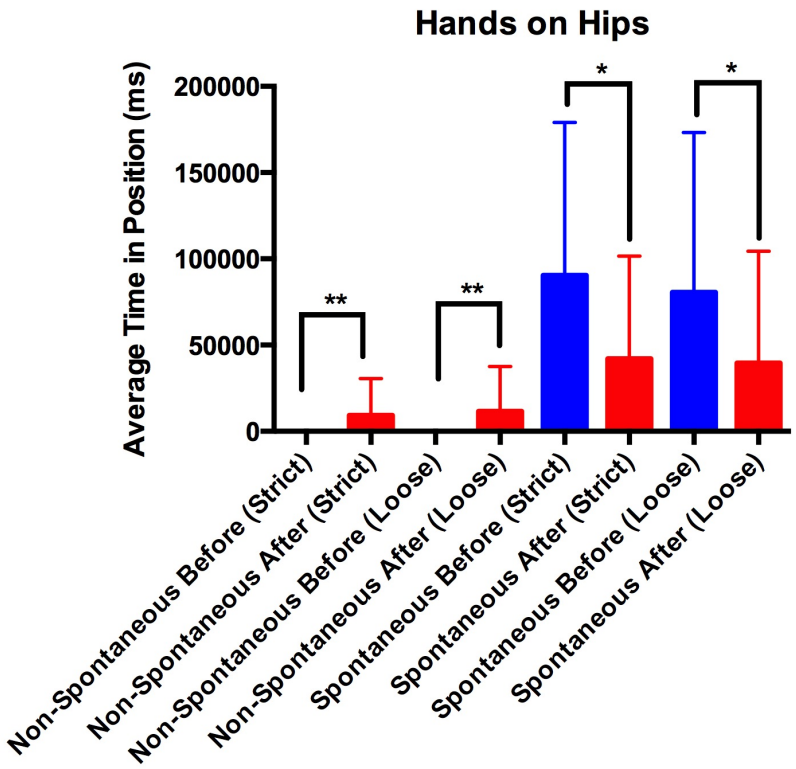


Figure 6.8: Average time participants perform hands on hips behavior before and after the Nao displayed hands on hips for both strict and loose definitions. * represents $p \leq 0.05$, ** represents $p \leq 0.01$

"before" period (M = 116665.08 ms, SD = 144992.65 ms), $t(12) = 1.91$, $p = 0.040$.

For the **loose** definition of hands behind back, spontaneous participants performed the behavior less in the "after" period (M = 73551.36 ms, SD = 116104.35 ms) than in the "before" period (M = 135802.93 ms, SD = 145636.49 ms), $t(13) = 1.76$, $p = 0.051$.

The results for the spontaneous hands behind back behavior, both strict and loose, are shown in figure 6.9.

Survey Results Beyond providing us insight for our inferences in our discussion, we found no correlation for likability, intelligence, gender, or race and the performance of behaviors before or after the Nao performed them.

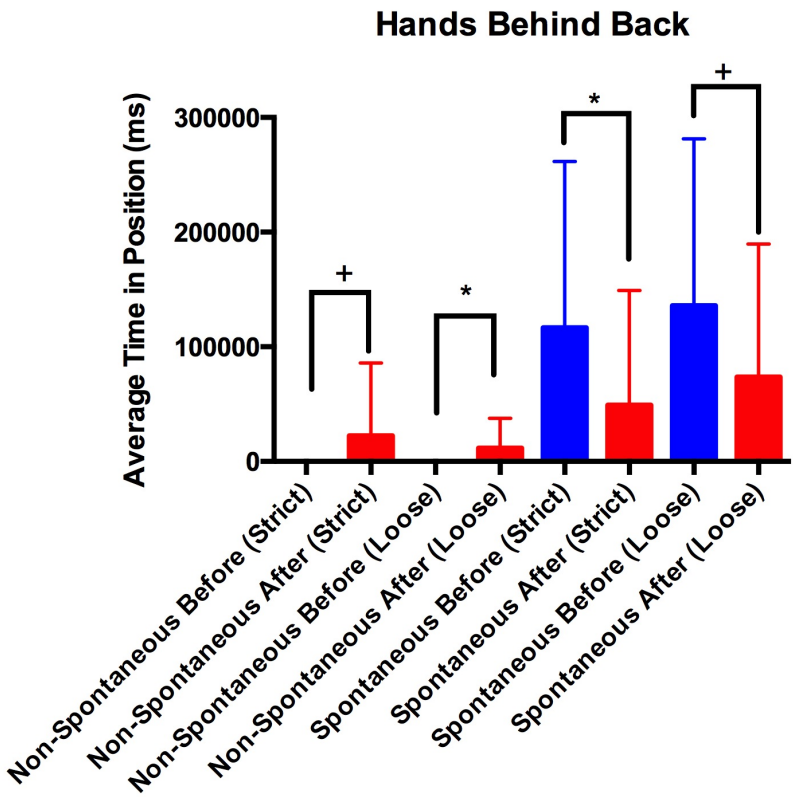


Figure 6.9: Average time participants perform hands behind back behavior before and after the Nao displayed hands behind back for both strict and loose definitions. + represents $p \leq 0.1$, * represents $p \leq 0.05$

6.3.4 Discussion

Our results yield two main findings about inducing mimicry in humans through robots that support both of our initial hypotheses.

- RESULT 1. *Humans who do not spontaneously demonstrate a behavior prior to observing a robot do so perform that behavior more after observing a robot perform it.*

- RESULT 2. *Humans who spontaneously demonstrate a behavior prior to observing a robot do so perform that behavior less after observing a robot perform it.* Our results support our first hypothesis H1. Our results support our second hypothesis

H2.

Interpretation of Mimicked Behaviors Our results support our first hypothesis H1. For both the strict and loose definitions of hands on hips, non-spontaneous participants significantly put their hands on their hips more after seeing the Nao do so. This presents very strong evidence for a robot's ability to induce mimicry. For the hands behind back behavior, the strict definition was only marginally significant while the loose definition was statistically significant. This makes sense as the loose definition focuses on the population who put neither hand behind the back, meaning no part of this group even partially performed the behavior spontaneously. Because the loose group focuses on people who in no way perform the behavior spontaneously, their potential for mimicry is maximized in the after period.

As far as why the strict definition of hands behind back was only marginally significant while the strict definition of hands on hips was statistically significant, the difference in significance can possibly be explained by the nature of the behaviors themselves. Hands on hips is a more stable behavior in that occurrences of it are longer in duration than the sometimes short burst of hands behind back. This is best captured by the sample size of strict hands on hips vs. strict hands behind back (34 vs. 30). Essentially, there are less participants who by chance or very quickly or casually perform a hands on hips behavior spontaneously. This is especially true with one hand behind back, which is why the loose definition helps to bring hands behind back into statistical significance (the loose definition separates the populations of spontaneous vs. non-spontaneous more thoroughly).

Spontaneous Performers Mimic Less Our results support our second hypothesis H2. Participants who spontaneously performed hands on hips or hands behind back prior to the Nao doing so actually performed that behavior *less* after the Nao did. While we cannot conclude why this would happen we can make inferences, especially with the help of our survey responses. Several responses noted spontaneous participants saying that when they saw the Nao first put its hands on its hips or put its hands behind its back, they thought the

Nao was mimicking their behavior. Such a thought process makes sense when we recall that the spontaneous participants had performed that behavior already at some point before ever seeing the Nao do so. This stands in contrast to several non-spontaneous participant responses who correctly identified the purpose of the study and did not confuse the direction of mimicry since they had never performed it up until that point. With this idea in hand that participants thought they were being mimicked, we can go back to psychology literature to look for explanations. Mimicry research in psychology has shown that mimicry can lead to socially cold feelings or the feeling that something is "off" [191]. In particular, an inappropriate amount of mimicry arouses suspicion in the party being mimicked [192], [193], [194], [195]. This can possibly explain why there is a decrease in performance of hands on hips or hands behind back for spontaneous participants. Another possible explanation is that humans view the Nao as a member of a social outgroup. It is possible that viewing the Nao as a member of an outgroup makes participants who think the Nao is mimicking them feel that the mimicry is inappropriate [177].

There are several caveats to consider with the theory that participants had a negative reaction to being mimicked. Previous studies showed positive human reactions to being mimicked by a digital avatar when head posture was mimicked [179], which seems contradictory to our possible explanation. In that study, however, the mimicking was done on a delay and participants were not aware of the mimicry [179]. In our study, participants saw what they thought was an explicit attempt at mimicry. The explicit attempt fits more appropriately with the psychology research that discusses problems with "inappropriate" amounts of mimicry.

Within the surprising finding for spontaneous performers of the behavior is that the loose definition saw smaller decreases than the strict definition. For hands behind back, the loose definition had such a small decrease that it even moved the loose definition for spontaneous participants from statistically significant to marginally significant. This can possibly be explained by the fact that our cutoff for spontaneous vs. non-spontaneous was

0. This means that participants who barely performed the behaviors before the Nao did, even for only 1000 milliseconds, were counted in the spontaneous group. These participants had such a low "before" time that they could easily have a higher "after" time. This could happen just by chance (particularly for hands behind back which had more small bursts of the behavior) or because the participants who performed the behavior for small periods in the "before" period did not take the Nao performing the behavior as an attempt at mimicry, silencing the concern of inappropriate mimicry. Fundamentally, this problem is more pronounced given the small sample size for the spontaneous group.

6.3.5 Conclusion

Understanding mimicry in human-robot interaction is an essential piece of designing social robots. In particular, mimicry needs to be actively considered in design paradigms and when analyzing the social impact of a robot. In light of this, we ran a study to observe the ability of a robot to induce mimicry in humans. Participants described paintings while interacting with a robot that assumed the posture of either hands on hips or hands behind back. Based on an initial pilot, we identified two groups of people within our sample: those who spontaneously exhibited hands on hips or hands behind back and those who did not. Our study produced two findings. People who did not spontaneously exhibit a behavior, mimicked the robot doing the behavior. People who spontaneously exhibited a behavior, performed the behavior less after observing the robot did the behavior. We discussed possible explanations for our results and their caveats in the previous section.

Implications In light of our first finding we can conclude that robots can induce mimicry in humans. This opens many new research possibilities and questions. Does the salience of mimicry in human-robot interaction move in the same patterns as it does in humans? For example, does having a goal to affiliate or similarity between partners induce greater mimicry in human-robot interaction as it does in humans [166]? Our second finding also raises concerns about building mimicry into human-robot interaction, especially in terms

of having robots mimic humans. Perhaps there are certain prerequisites that must be fulfilled in a human-robot partnership before mimicry is considered appropriate. Lastly, this study raises issues for design of robots and their impact on humans around them. Just as we are wary of how other humans may mimic our actions, we must be wary of how robots may induce humans to mimic tasks, particularly if they are tasks we design robots to do specifically because we don't want humans doing them.

Looking back at the 4 reasons we presented in the Introduction Section for why mimicry should be studied in human-robot interaction, our implications for our results can be linked to each of them.

First, we can conclude that robots can elicit mimicry and as such this is a feature that could be possibly be used in designing social robots in such a way that their interactions with humans become smoother and more positive and engender more trust for the human involved.

Second, our results showing the decrease in behavior displayed for spontaneous participants suggests that there are situations in which it might not be favorable for robots to mimic humans. This could cause problems in interactions between humans and robots, making the humans feel socially cold or uncomfortable.

Third, our finding that robots can induce mimicry in humans suggests that designers need to be careful when considering scenarios where robots will be interacting with individuals, such as children, who are easily influenced. They also need to be cautious when designing robots who fulfill tasks that are specifically dangerous to humans, as other humans may mimic the robots without realizing the consequences.

Fourth, seeing mimicry being induced in humans by robots suggests that there is room for broader social contagion between humans and robots. While it is unclear what the boundaries of this might be, mimicry in motor tasks is the first hurdle in showing social contagion in human-robot interaction.

6.4 Application of Social Collaboration Tools: Conveying Information Effectively during Interactions with Robots

This work presents an application of a robot that exhibits tools like those investigated in this chapter during an extended interaction, which presents evidence for relationship-building. We investigated whether we can engender learning and higher engagement during interactions with a robot acting as a peer during interactions. This work is based on [120] [5]. This work describes an extended (6-session) interaction between an ethnically and geographically diverse group of 26 first-grade children and the DragonBot robot in the context of learning about healthy food choices. We find that children demonstrate a high level of enjoyment when interacting with the robot, and a statistically significant increase in engagement with the system over the duration of the interaction. We also find evidence of relationship-building between the child and robot, and encouraging trends towards child learning. These results are promising for the use of socially assistive robotic technologies for long-term one-on-one educational interventions for younger children.

Children love imaginary play, and such play can make learning more engaging and effective [196, 197]. There is also much need for individualized support in a classroom, allowing each child to progress at their own speed using the learning strategies most appropriate for that child. Child-friendly social robots have the potential to provide this individualized support through imaginary play, allowing children to engage with their lessons in a tangible and active way. We take a Socially Assistive Robotics (SAR) approach to teaching nutrition to 1st-grade children, with the goal of promoting positive habits and behavior change through human-robot interaction (HRI). In order to move towards the kind of long-term deployment that would be necessary to see major gains in learning (consider

5. E. Short, K. Swift Spong, J. Greczek, A. Ramachandran, A. Litoiu, E. C. Grigore, D. Feil-Seifer, S. Shuster, J. J. Lee, S. Huang, et al., "How to train your dragonbot: socially assistive robots for teaching children about nutrition through play", in Proceedings of the 23rd IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), IEEE, 2014, August 25–29, pp. 924–929.

that a single topic might be covered in a school setting for many weeks), we first evaluate the feasibility of this approach, including measuring children's engagement with the SAR system over time and evaluating whether it has the potential to facilitate nutrition information learning. We present a semi-autonomous SAR system, and a three-week biweekly evaluation study with a diverse group of 26 5-8 year old children.

6.4.1 Related Work

Childhood obesity has tripled in the United States over the past 4 decades [198]. Obesity among children and adolescents has been shown not only to lead increase risk of being overweight in adulthood [199], but also diseases later in life, including high cholesterol and triglycerides, hypertension, and type 2 diabetes [200]. Educating children about healthy food and beverage choices, and motivating them to make healthier choices, can help to lower rates of obesity [201]. Technological interventions in particular lend themselves to the broad replication and personalization that will be necessary to combat what has proven to be a challenging problem [202]. Several technology-based nutrition interventions for children have been developed, using smartphones, computers, and video games (see [203] or [204] for a review). While these interventions make use of technologies that are widely available, there is some evidence that HRI systems promote learning more than screen-based technologies. Leyzberg et al. showed that the physical presence of a robot can increase cognitive learning gains [150], while in children, Movellan et al. [205] demonstrated that a social robot could be used to teach young children new words, despite other results suggesting that young children do not learn language from pre-recorded human speech [206]. Other work in HRI focuses on the use of robotic teachers for english language learning [207, 208] and teaching children to play chess [209]. In adults, Kidd and Breazeal [210] show that SAR systems have the potential to improve eating and exercise beyond a paper- or computer based intervention.

Our work aims to use SAR to leverage children's excitement about both pretend play

and technology, to provide an affordable, accessible, and personalized means of delivering nutrition education and coaching. In this paper we focus on teaching first-grade children (primarily 6-7 years old, although several participants were 5 or 8 years old at the time of the study). Children this age do not have significant control over their food choices, allowing us to instill healthy habits before unhealthy ones become ingrained.

6.4.2 Methodology

In the sections that follow, we describe the design of a SAR-based nutrition education intervention, as well as the study used to test the effects of the initial version of the intervention. Our study addresses five research questions, as follows: Q1: Do children enjoy interacting with the SAR system? Q2: Are children able to maintain engagement with a SAR system over time? Q3: Are children able to build a relationship with the SAR system over time? Q4: What is the impact of the SAR system on child learning of nutrition information? Q5: What is the relationship between the child's temperament and their interaction with the SAR system?

Intervention Design

While most nutrition education interventions are $12-16$ weeks long, most SAR systems are used in much shorter term interactions (often as little as one session). In order to move towards a longer intervention, we developed a $3-$ week long, twice-weekly intervention for first grade children, with a within-subjects design. Each of the six intervention sessions consists of an approximately $5-$ to $10-$minute long one-on-one interaction between the child and the robot. We used a Wizard-of-Oz interaction and monitoring design, with a tele-operator providing dialogue selection and some perceptional capabilities, with pre-scripted dialogue behaviors (including both speech and movement), and autonomous control of the overall interaction flow. Each session consisted of two parts, introduction and food selection game, as described in section III-D. In the first session of each week, the robot acted

Figure 6.10: Intervention room setup

as an expert, giving feedback on food choices one-by-one, while in the second session of the week the child and robot collaborated toward making healthy choices together.

Robot and Experimental Setup

The intervention centered around an interaction with the DragonBot robot [211], a dragon-like squash-and-stretch robot with five degrees of freedom (see Figure 6.10, center-left). The robot is covered with a plush skin designed in collaboration with an expert puppeteer. The skin includes posable arms and tail, as well as removable wings in four sizes, allowing the robot's wings to "grow" from session to session. The robot is approximately 18 inches tall at its full height and can be seen in Figure 2. The interaction was conducted in a small parent-teacher conference room, allowing children to interact individually with the robot. The robot was set up on a table facing the child, with realistic artificial foods arranged around it; the arrangement of foods was kept constant across participants. In order to provide the richest possible dataset for future analysis, a Microsoft Kinect sensor, an HD camera, and USB camera are arranged as seen in Figure 6.11, in addition to the two laptops, the DragonBot base station (providing power to the robot), and set of speakers necessary

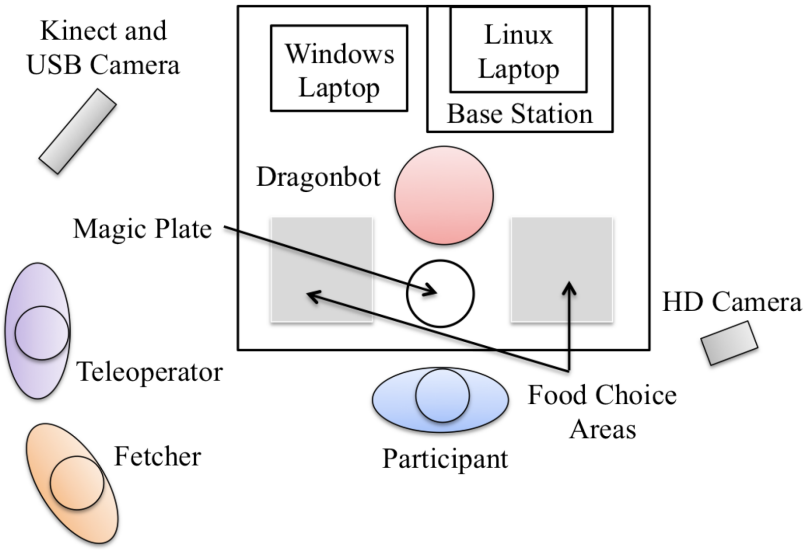


Figure 6.11: Diagram of the intervention setup

to run the robot. The intervention setup can be seen in Figure 6.10 (the cameras and Kinect are out of the frame).

Interaction Structure and Progression

The verbal interaction between the child and robot used pre-recorded speech following a script that was written with the assistance of a screenwriter experienced in children's television, and follows the story of Chili the DragonBot through several weeks of training and preparation for the big upcoming "dragon race." We employed child-centric storytelling techniques such as character development and backstory to create a richer interaction. We also increase the difficulty of the task over the three-week intervention, challenging the child within her or his zone of proximal development, using a socio constructivist approach [212, 213] to integrate the increasing difficulty and social interaction, maximizing the child's learning potential.

Over the course of a six-session, three-week experiment, we covered three nutrition topics, one per week: packing a lunch box (choosing whole grains and non-sugary drinks), choosing after-school snacks (avoiding nutritionally bankrupt junk food), and building bal-

anced meals (choosing whole grain breakfast cereals and colorful vegetables at dinner). Each topic had two sessions devoted to it; each session built upon the previous one, using a gradual increase in challenge of content. In the first session, the robot served as an expert, sampling foods offered to it one at a time and providing nutritional information as feedback. In the second session, the robot behaved in a more cooperative role, in order to provide feedback in a more challenging game where the child chooses several foods at one time. We refer to the first of these as the expert sessions (ES) and the second as the cooperative sessions (CS).

Interaction Segments

Before the beginning of the interaction, the participating child was brought from the classroom by an adult experimenter (not the teleoperator). This person, the ?fetcher?, remained in the room, sitting behind the child, instructed to answer as simply as possible any questions asked by the child, and to make sure that the child did not damage the robot. The child was invited to sit down, and told to wake up the robot. Once the child said something to the robot, the robot woke up and began the first segment, the introduction. The introduction included a welcome speech, relationship building through small talk, and finally backstory and character progression. During the CS, the second day on a given nutritional topic, there was a brief review of the nutritional curriculum from the previous session. The interaction then moved into the food choice segment. For the ES, there were two rounds of the food choice game, where the child was asked to choose a food, the robot ?tasted? the food, and then provided feedback. In the CS, the child was asked to select a food items until they chose a healthy item, receiving progressively more specific advice about how to improve the selections, based on an evaluation external to both robot and child (in this case, a "magic plate"), and following the theory of graded cueing, an occupational therapy technique [214]. For both session types, some vocabulary used in the feedback was explained with a backstory-type dialogue item.

Data Collection and Associated Measures

We recorded several types of electronic data during the intervention in order to build a rich dataset, including teleoperator selections, pre- and post-questionnaires (modified to be administered orally by an experimenter), audio- and video-data, and Kinect pose data. In addition to the technology mediated data collection, we collected data from four questionnaires widely used either in the robotics or child development research fields as measures of child personality, child evaluation, and child interaction.

The Child Behavior Questionnaire [215] (CBQ-S; Cronbach's = 0.87) contains a 4-point Likert-type scale and requires parents to rate various aspects of their child's behavior and personality (affect). It contains three subscales and provides an efficient child temperament measure for school aged children (ages 5- 12). The CBQ-S subscales are surgency (positive affect), effortful control (self-regulation) and negative affect. The following three were administered directly to the children, in an interview-type format to accommodate the children's limited cognitive and developmental abilities and to avoid biases associated with diversity in child reading abilities [216].

The Perceived Value Questionnaire is adapted from an evaluative questionnaire by Lombard [217] and also used in SAR related research by Kidd and Breazeal [218] (Cronbach's = 0.95). This questionnaire required child participants to rate their interaction with the robot using an 8-point Likert-type scale. The ?utility? and ?value? subscales of this questionnaire were administered after the first interaction that the child had with the robot, and then again after the final interaction with the robot at the culmination of the intervention.

The Social Presence Questionnaire was used to quantify the effectiveness of the robot's social capabilities (or social presence). The social presence of the robot was measured by an 8-point Likert-type scale using questionnaire items established from Jung and Lee [219], (Cronbach's = 0.82). We administered this questionnaire after the first intervention session and at the end of the intervention. Finally, the Adapted Companion Animal Bonding

Scale [220] asks the child to rate the various features of the robot including whether the robot is bad/good, loving/not loving, cuddly/not cuddly, and warm cold. This questionnaire was administered twice, first before the children interacted with the robot (but after a brief introduction to the robot), and again at the culmination of the intervention.

Hypotheses

Six research hypotheses were developed that are associated with the study's research questions:

- *H1:* Participants will have a positive reaction to the SAR system, that will increase over time.
- *H2:* Children will be more engaged with the robot over time, as measured by a decrease in their response time to the robot's verbal questions (a well-established proxy for engagement in the child development literature [221]).
- *H3:* Children will use more complex speech with the robot over time, as measured by mean length of utterance (MLU) and a qualitative categorization of their utterances.
- *H4:* Children's time to make a choice of food when prompted by the robot will decrease.
- *H5:* Children's performance on the nutrition task will improve over time, as measured by a choice indicating ratio.
- *H6:* Children with a positive affect and higher self-regulatory ability will have greater interaction with the robot.

Study Population

As we hope to design and evaluate systems that can be deployed to populations of children across a wide range of economic and social backgrounds, we collected data from

two highly diverse sites: a west coast site in an urban center and an east coast site that drew from primarily suburban households. We treated these samples as one cohort in our analysis to highlight the commonalities that would need to be utilized for a more robust long-term deployment. There were twenty-six participants in the study with age range of 5-8 (twenty-two 6-7-year-olds; two 5-year-olds; two 8- year-olds). Seventeen of the children were female (65%) and the remaining nine were male (35%). In terms of ethnicity, the sample was diverse and representative of the population in the areas in which the participants reside. The largest ethnic group represented was children of Hispanic descent (69%), 19% were children of African American descent and the remaining 6% were European American or had mixed ethnicities. Approximately 62% of the study participants reside in the western United States in a large urban city and the remaining study participants (38%) came from the eastern most region of the U.S. The participating children's parents' ages ranged between $20 - 49$, with 25% of the parents Fig. 3. Child evaluation of the robot in several categories in the $20 - 29$ year age range, 44% between $30 - 39$ years of age, and 31% between $40 - 49$ years of age. With regard to parental education, approximately 19% of the children's parents did not graduate from high school, 19% were high school graduates, 43% had completed some college, 19% of the parents had bachelor's degrees, and the remaining 6% had completed graduate or professional education.

6.4.3 Results

Evaluation and Perception of the Robot

In general, the participants in our study had positive perceptions and reactions to the socially assistive robot before, during, and and after the SAR intervention. The children's perception of the robot was high ($M = 7.58, SD = .76$, 8-pt. scale) pre-intervention and remained relatively constant through the culmination of the intervention ($M = 7.45, SD = .80$). Specific to the measures of robot evaluation, the children perceived the robot as useful ($M_{PRE} = 7.35, SD = 1.9; M_{POST} = 7.31, SD = 1.7$) and enjoyable ($M_{PRE} =$

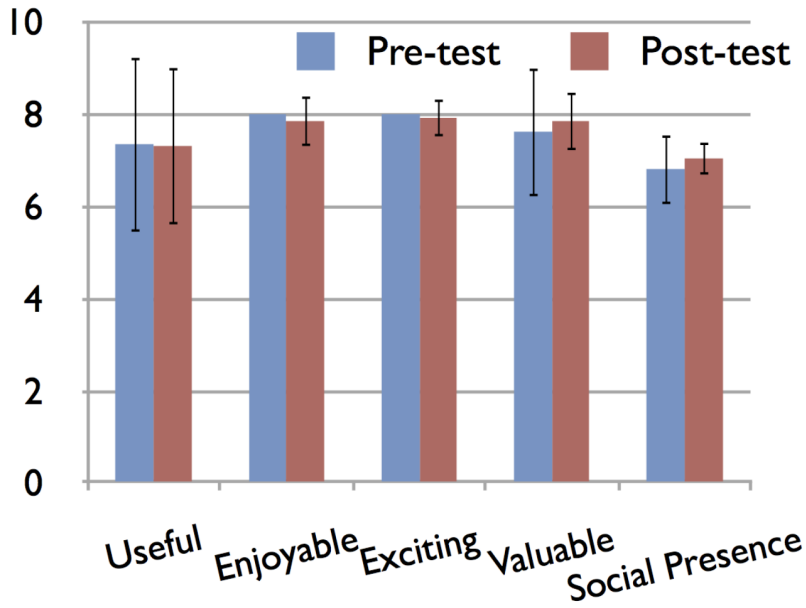


Figure 6.12: Child evaluation of the robot in several categories

$8, SD = 0; M_{POST} = 7.85, SD = .54$). They also rated it as exciting ($M_{PRE} = 8, SD = 0; M_{POST} = 7.92, SD = .40$), valuable ($M_{PRE} = 7.62, SD = 1.4; M_{POST} = 7.85, SD = .63$), as having strong social presence ($M_{PRE} = 6.81, SD = .76; M_{POST} = 7.04, SD = .43$), and as attractive ($M_{PRE} = 6.65, SD = 1.7; M_{POST} = 7.89, SD = .35$). We did not find significant differences between the pre-and post-intervention ratings, as seen in Figure 6.12, likely due to the extremely high positive evaluation. Thus the first part of *H1*, the positive perception of the robot, is supported, but not that there is an increase in this perception over the duration of the intervention.

Child-Robot Interaction

To determine the level of engagement between the robot and the child during the intervention, we calculated the children's mean response time (in seconds) when prompted by the robot's verbal questions. Although we found that child response times ebbed and flowed across intervention sessions, the comparative results of these calculations revealed that the mean child response time decreased from the first day of the intervention to the end of

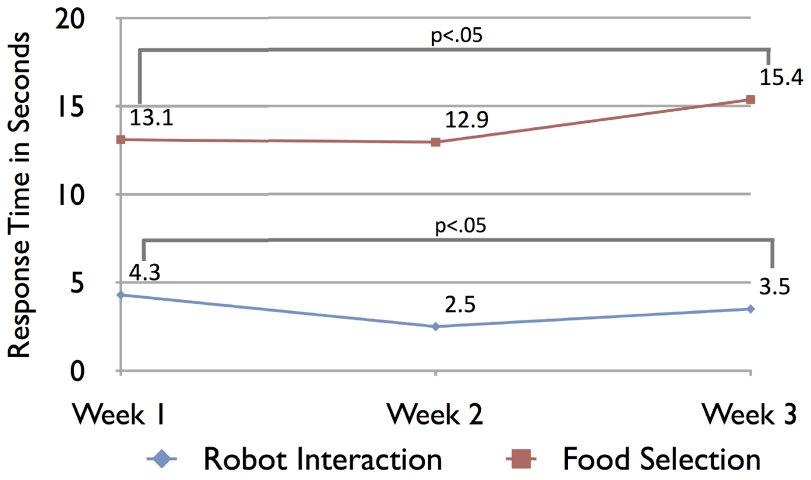


Figure 6.13: Child response times to robot conversational queries and food selection prompts

the intervention. The mean response time across children during the first intervention was 4.3 seconds and the mean response time(see Figure 6.13). Child response times to robot conversational queries and food selection prompts was 3.5 for the last session, indicating a $.8$ second mean decrease in response time. A paired t-test was performed to ascertain whether there was statistically significant response efficiency between the beginning and culminating weeks of the intervention; we use this pre-post comparison of means throughout the analysis in this paper, because the limited length of the intervention renders mid-intervention measurements and predictive analyses non-significant (see [222]). The change response efficiency was significant, $M = 1.35, SD = 3.13, N = 26, t(25) = 2.206$, two-tailed, $p < .05$ (see Figure 6.13). A Cohen's d statistic was also computed to measure this effect size ($Cohen'sd = .57$), which indicates a moderate intervention effect. This supports *H2*, that children's engagement increases over time.

Conversation as a Means of Interaction

We were also interested in the level of conversation that the child had with the robot. To demonstrate this, we calculated each participating child's mean length of utterance (MLU). The MLU is a widely utilized proxy for speech production by researchers and practitioners

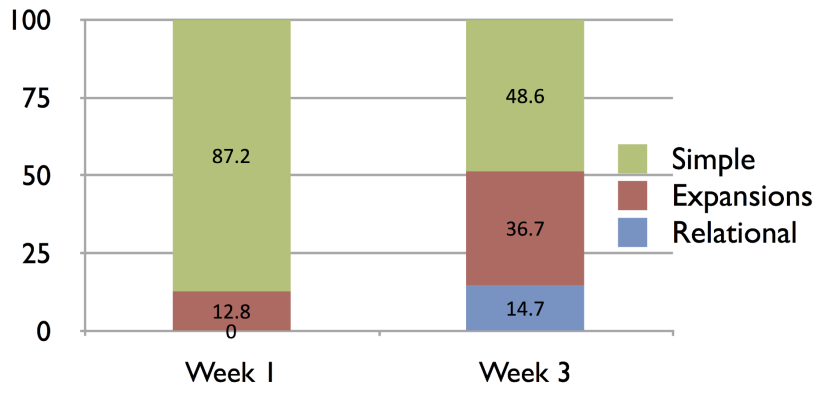


Figure 6.14: Response categories over time in child-robot interaction

in the speech and language fields. The mean length of utterance for participants across all weeks of the intervention was $28.41(SD = 24.57)$. There were successive changes in response words to the robot over time. There was a 2.29 mean increase in utterance length from the start to end of the intervention period, in accordance with *H3*, however this trend did not reach statistical significance.

In order to explore differences in the types of utterances employed by the children, we employed an empirical analysis of the spoken language transcripts. We analyzed the content of the 137 verbal utterances employed by the participants, categorized these interactions and quantified them (via frequency distribution) to measure changes in types and frequency of interaction over time. The categories we used to identify patterns in the transcripts included: (a) simple responses to prompts, which included one-to-three word responses to robot prompts (e.g., yes, no, huh, um, okay), (b) expansions that included responses with details (e.g., it's healthy, I like it, a magic plate!), and (c) relational responses, which demonstrate evidence that the child was beginning to relate to the robot (e.g., he's hungry? you said you didn't like it). While there was great variability in Figure 6.14. Response categories over time in child robot interaction response types across children, we noted robust changes in types of interactions over time. For example, one child began his interaction with "yeah" and "maybe," and proceeded to recall what the robot had said in a previous session and said "You said you don't like mashed potatoes!", demonstrating both

relational speech and, given the tone of the interaction, humor with the robot. We computed the percentage of each category by child and then computed an average percentage per week, per category for the study sample. Figure 6.14 illustrates these changes between week one and three of the intervention. Thus we find overall some weak support for H3, that children use more complex speech with the robot.

SAR in the Context of Learning

Similar to our measurement of child-robot interaction, we calculated the mean time that children in the intervention took to make food choices in response to the robot's prompting. The mean response time increased from 13.11 seconds in the first session to 15.36 seconds in the final intervention session, a 2.25 second increase. A paired t-test was performed to ascertain whether the child's food selection time became more efficient (measured by decreases in response times). The mean response difference for food selection ($M = 2.24, SD = 5.26, N = 26$), ($t(25) = 2.174$, two-tailed, $p < .05$), $Cohen'sd = .45$ providing evidence of the effect size of this change (see Figure 6.13). This is in direct contradiction to *H4*, however, the child task expectations increased in difficulty each week. Accordingly, we interpreted this result as an indication that as the tasks in the interventions became successively more challenging, the children took additional time to make thoughtful selections. As an indicator of whether the child participants had learned how to make healthy food selections, we computed the ratio of poor to healthy food selection for each child as a way to normalize choice quality across sessions and facilitate comparison. We found a trend towards improved choices, with a decrease the the mean poor choice ratio from $.46$ to $.39$, indicating that the children may have begun to make healthier food selections between the first and last intervention sessions, weakly supporting *H5*, however this change was not statistically significant.

Temperament, Interaction, and Learning

As a final comparative measurement, we wished to determine if child temperament and associated behavior influenced child interaction with the robot. Therefore, we used CBQ-S as a comparative metric for our other research results. The mean child self-regulation score on the CBQ-S was $3.08(SD = .53)$, the mean child positive affect level was $2.95(SD = .38)$, and the mean negative child affect was scored as $2.40(SD = .34)$ for our study sample. Specifically, using this measure, we wished to determine whether child temperament predicted child interaction time, child food selection time, or food ratio (described above). Accordingly, each of these variables was selected as dependent variables in multiple regression analysis, with Bonferroni error correction. A step-wise regression was selected using our theoretical perspectives as a means of determining the order in which independent variables would be loaded into the model. Accordingly, socio-demographic variables were loaded into the model first, followed by child temperament variables. These analyses revealed that positive child affect was moderately predictive of food ratio (serving as a proxy for learning), $r^2 = .359, F(3, 14) = 6.15, p < .05$, supporting the first part of *H6*, that children with a positive affect will have greater learning agains. Child-robot interaction and food selection did not contribute to the model and therefore are not predicted by child temperament (thus leaving *H6* unsupported), however child temperament (which contributed to the model) may predict child learning.

Discussion

The goal of this study was to examine the feasibility of a SAR-based intervention for teaching children about nutrition. We wished to examine the children's evaluation of, and engagement with, the robot, changes in their verbal interaction with the robot over time, the learning effects of the SAR system, and the effect of child temperament on interaction and outcomes. We find that children rate the robot highly positively across all measures used, and retain a positive perception of the robot after a three-week intervention. We find

that over the course of the intervention, children respond more quickly to the robot's verbal queries, suggesting that they not only maintain engagement with the SAR system over time, but likely become more comfortable with the system, perhaps even building rapport with the robot character. While we do not find changes in their MLU, we do find that they use more complex speech with the robot over time, suggesting that the social presence of the robot encouraged child-robot relationship building. We do not find a relationship between between child temperament and social interaction with the robot (*H6*), but this may mean that children with diverse temperaments can interact equally well with the SAR system.

In terms of the educational goal of the intervention, we find limited evidence that the children in the study learned about nutrition over the intervention (*H5*). We find that positive affect is moderately predictive of healthier food choices (our proxy for learning), which is consistent with the literature on education. Finally, our results indicate that children take longer to make food selections over time, contrary to *H4*, however we have modest evidence that children choose healthier foods over time, suggesting that the increase in time may be indicative of greater thoughtfulness in their responses.

In terms of limitations, while this study examines a relatively large, diverse group of children over many sessions, the sample size and intervention length are still limited relative to typical nutrition interventions. Because this work is designed to examine the changes in the interaction over time, we use a within-subjects design and do not include a control group. In future work with a more primary focus on learning, we would include such a control.

6.4.4 Conclusions

In summary, our results provide a promising basis for future SAR interventions for teaching young children, in particular for the task of nutrition education. We demonstrate that children have an extremely positive reaction to the robot, even after several weeks of periodic interaction with the SAR system. Children were able to sustain engagement with

the robot character, and perhaps even build a relationship with the robot, a result that is promising for future, longer, interventions. While we do not find strong learning gains over this intervention, we expect that such longer interventions would show greater changes in knowledge. We are therefore encouraged to extend this work to longer interventions as we move towards a more autonomous SAR system.

Chapter 7

Conclusion

In this dissertation, we tackle the challenge of developing adaptive robots for Human-Robot Collaboration (HRC) that aim to provide assistance to a human worker throughout the progression of a task. To do so, we focus on robots that do not aim to complete a task on their own, autonomously. Rather, we focus on developing robots that aim to provide assistance that is tailored to individual human workers' preferences. This allows both the robot and the human to focus on the types of actions they are best suited for, and helps bridge the gap between the capabilities a robot would need in order to autonomously execute complex tasks, which require both a high degree of knowledge about the task and a high dexterity level in terms of manipulating objects.

Developing robots that assist, however, does not come without its challenges. Doing so requires developing robust techniques for learning about object and human motion and how this translates to actions performed by the human, learning about the structure of the task, learning about users' individual preferences with respect to what support the robot should offer, and learning about different actions the robot vs. the human should be performing. Furthermore, all of these problems need to be tackled as part of a scenario where user-provided demonstrations or labels are expensive to obtain since we they require the user to either interact with the robot or take time to label demonstrations.

7.1 Summary of Contributions

To address the challenges reiterated above, we summarize the contributions we made throughout this thesis below.

- *We tackled the difficulty of learning useful information about what happens throughout the execution of a complex task in HRC:* **We learned action primitives (building blocks of motion) that a human worker executes throughout a physical task. We presented a framework that discovers whether a coarser or a finer-grained level is better suited for a primitive given the task at hand, coining this concept the *granularity level* of a primitive.**

- *We tackled scenarios where it is feasible to obtain a large pool of data consisting only of task trajectories executed by the human worker, with no information about preferences for robot assistance:* **We created a model of personalized supportive behavior preferences for different users that can be trained from as few as three task trajectories labeled by a user, built atop of a single, cross-users model of the task. This system achieves on par with human-level performance.**

- *We tackled scenarios where it is difficult to obtain large amount of data and thus desire a model-free approach:* **We introduced *MAB-RL* and *HMAB-RL*, a multi-agent based reinforcement learning paradigm and for HRC and its extension to hierarchical reinforcement learning, where we consider both the robot and the human worker as agents in the environment, but where we do not control the person's actions throughout the task. We compared algorithms based on these paradigms that employ different ways of synchronizing the robot and human's action during each time step, including an algorithm that utilizes *communicative* actions. We presented results that show the robot can perform on par with human-level performance from training sessions composed of 40 episodes, with varying episode lengths.**

- *We explored the social dimension of collaboration:* **We performed a number of studies that presented results supporting different behaviors we can endow collaborative robots with in order to keep their users more engaged and motivated throughout the task.**

7.2 Future Work

Throughout this thesis, we have tackled learning useful information for our HRC scenario starting at the low-level and exploring what we can learn from raw data, and moving to the high-level and focusing on building both model-based and model-free techniques and paradigms for the robot to be able to make high-level decisions. Two of the many extensions to this work that we believe to be of high importance moving forward include building a system that links and integrates the work at the different levels that we presented herein, and working to improve the models at the high-level in order to reduce training time and facilitate transfer learning.

7.2.1 Integrating low- and high-level algorithms

In order to have a robot that can be placed in a manufacturing setting and trusted to collaborate with a human worker without constant supervision, creating a system that can work autonomously (not in terms of completing the task on its own, but in terms of operating within the bounds of offering assistance to a human) is crucial. Connecting algorithms and paradigms like the ones we introduce in this dissertation is non-trivial and potentially requires novel explorations into how to best achieve this.

Another research direction that would be extremely powerful in this context would be to create an end-to-end system, capable of applying the paradigms and algorithms presented in Chapter 4 and in Chapter 5 directly on raw data. Such systems indeed exist and have already been used in robotics in the form of deep reinforcement learning, but have so far only

been applied to more constrained tasks like manipulation. Creating novel ways of applying such algorithms, or developing new algorithms that might be better suited to applying the types of models that need to work at the high-level is a promising and exciting research direction.

7.2.2 Improving modeling techniques for facilitating better learning and transfer learning

In this dissertation, we presented both model-based and model-free paradigms for HRC, each best suited for different scenarios depending on how much, as well as what kind of data is available for training. Developing these techniques further would allow us to come up with model-free methods that are not as data hungry and that would not require us to build a model of the task. In this category, we believe exploring more paradigms in hierarchical reinforcement learning, including methods for us to learn macro-actions directly from the data without hand-specifying them, would allow re-use of higher and higher level actions, and would also facilitate transfer learning. In this case, transfer learning would be useful for making use of the knowledge the robot has acquired for a assembling a chair, for example, in order to assemble a table that might require similar sub-tasks.

Another direction is to develop model-based techniques, where we focus on learning better models of the task, and focus on developing less data hungry techniques for this part, so that we can easily build personalized models for the users' preferences atop. This is the method we employ in **??**, which can be extended and combined with techniques from model-based reinforcement learning, for example.

To conclude the work presented in this dissertation, we are excited to advance the vision of bringing robots closer to being able to operate in environments where they can seamlessly interact with humans, and adapt to their environments in a robust manner.

Appendices

Appendix A

Exploring Useful Supportive Behaviors: Modeling Skill Duration for Collaborative Tasks

In this work, we investigated an approach for autonomously predicting the amount of time required for different agents to complete actions. This work is based on [223] [1]. A system's ability to assess its (and others') skill level regarding the performance of different tasks is essential to achieving efficient scheduling and collaboration. In this appendix, we present an observation-driven modeling approach allowing an agent to autonomously predict the amount of time required for different agents to complete actions. This approach utilizes insights and observations from the developmental psychology and operations research communities to accurately develop agent-personalized skill proficiency models. We demonstrate our model by evaluating its performance at estimating agent performance in a set of common assembly tasks. Our evaluation measures knowledge-transfer via novel

1. B. Hayes, E. C. Grigore, A. Litoiu, A. Ramachandran, and B. Scassellati, "A developmentally inspired transfer learning approach for predicting skill durations", in Proceedings of the 4th Joint IEEE International Conferences on Development and Learning and Epigenetic Robotics (ICDL-Epirob), IEEE, 2014, October 13–16, pp. 181–186.

task introduction, as well as extrapolation by predicting future performance given previous experience.

A.1 Introduction

Collaborative activities are strongly present in people's day-to-day lives. Although people commonly successfully engage in different types of collaborative activities and thus understand how to coordinate their actions to achieve a common goal, there is no consensus on how adults come to have this complex understanding [224]. An essential part of this understanding is derived from evaluating one's own proficiency at performing different tasks — not only to reach a common goal, but to reach it as efficiently as possible.

Prior work has confirmed that adults are good at evaluating other people in a variety of social contexts. Research shows that adults achieve this by analyzing both the behavior and physical features of people they interact with [225, 226]. Investigations of mechanisms employed for self-evaluation, however, constitute an entirely different line of research.

Adults do not make perfect assessments [227] and can easily overestimate or underestimate their capabilities [228] because of potential damage to an individual's self-esteem [229], difficulty in processing available information [230], or uncertainty related to evaluating outcomes [231]. However, they do have a clear and full understanding of the concept of ability and self-efficacy. The process through which adults achieve this understanding is being extensively studied in developmental psychology research. It is believed that young children, preschoolers and kindergartners, do not have a clear notion of ability as an internal quality [232].

Changes in the way children think about ability have been observed to occur between kindergartners and 7-8 year olds, and between 7-8 and 10-12 year olds [232]. Qualitative changes begin at 7-8 years old, when children become increasingly interested in performance and ability. This is indicated by the act of making comparisons with peers, with

respect to friendship formation, classroom norms, and academic achievement. The notion of ability starts becoming its own domain for children at this age, whereas younger children can sometimes mix up intelligence with conduct, likability, or social behavior [233, 234]. More importantly, children start making more accurate predictions about their own capacities. Younger children tend to suffer from positive biases or "wishful thinking" [235], overestimating their abilities in a range of domains, such as school achievement [236, 237], running [238], or peer group dominance [239]. Research has shown that at the age of 7-8 years, children begin to adjust expectations, become more accurate in their self-perceptions, and get closer to their teachers' ratings [232, 240, 241].

The second stage of changes happens between 7-8 and 10-12 year old children. At this time, children's reasoning skills develop further, as they start being able to differentiate between the concepts of ability and effort [232, 242, 243]. Children at this stage begin interpreting ability as a capacity rather than a set of skills and knowledge [232, 244]. Although the development of their reasoning skills helps increase self-evaluation accuracy during this stage [245, 246], some children continue their trend of becoming more pessimistic and thus sometimes underestimate their ability [237].

In this work we present a skill estimation system inspired by these developmental shifts in self-evaluation. Similar to the transition experienced between kindergartners and 7-8 year olds, where comparisons between tasks and between others become prevalent, our system utilizes transfer learning strategies that allow for the use of comparable, indirectly related experiences to improve estimation proficiency. By exploiting these experiences, our system is similarly able to converge on more accurate estimations of ability (as measured by predicting task durations).

The second developmental transition, where the notion of ability separates from the skills themselves, is represented through components of the presented model that augment heuristic-driven predictions (e.g., casual or generic estimates akin to "It should take someone about 10 minutes") of expected performance with evaluations of personal abil-

ity. These components model agents' proficiencies with the tools or motor skills required for a task, allowing for an accurate personal interpretation of impersonal, or crowd-based knowledge. Our work is particularly motivated by scheduling and multi-agent planning scenarios, where providing tighter bounds for worker-subtask pairings can directly result in improved time management and more efficient planner solutions.

A.2 Related Work

Our work utilizes techniques drawn from the Transfer Learning (TL) and Learning from Demonstration (LfD) communities. TL is a well-studied area within both robotics and psychology, leveraging the idea that one can generalize across topics as well as within them [247]. This idea has been applied within several domains, including reinforcement learning [248], cognitive architectures [249, 250], machine learning [251, 252], and planning [253, 254].

Transfer Learning-based techniques have also been used for improving performance on supervised learning tasks in situations where adequate labeled data are not readily available [255]. In other work, a conceptual framework is presented to capture the process of transferring skills and knowledge from one task to another [256]. Relevant work in observation-driven skill acquisition and evaluation includes the construction of parameterized skills from experience [257], the construction of skill trees from demonstration trajectories [258], and building of hierarchical collections of skills for intrinsically motivated reinforcement learning paradigms [259].

A.3 Domain

In this work we address the problem of estimating agent skill proficiency, in terms of generating estimations of the amount of time required for their completion. In particular, we build models that accurately estimate the amount of time required for an agent to perform

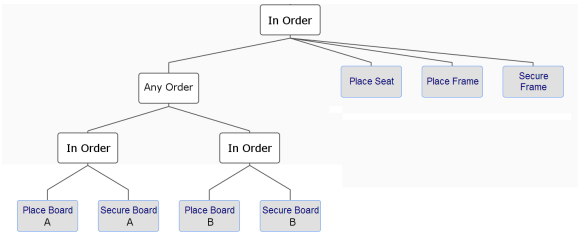


Figure A.1: The hierarchical task network used for assembling the Lätt chair, indicating the ordering constraints and goal of each step. Actions were coded in terms of the goals represented by each leaf node.

Figure A.2: Pieces of furniture that were used in assembly tasks. (Left) IKEA Lätt table and chairs. (Right) IKEA Kallax shelving unit.

subtasks obtained from decompositions of complex tasks (Fig. A.1). We chose a manufacturing domain for our evaluation, using IKEA furniture construction as a task class representative of well specified, multi-step, complex assembly work.

We model construction times for subtasks within the assembly of three different pieces of IKEA furniture: a chair, a table, and a shelving unit (Fig. A.2). Instructions for these pieces were segmented into seven, seven, and nine subtasks, respectively. These tasks were chosen due to their partial overlap in terms of tools required, motor skill similarity, and for their parallels to many real-world tasks.

A.4 Approach

We approach the problem of skill proficiency estimation as one of agent modeling. By analyzing video recordings of humans performing assembly tasks, we were able to extract relevant features such as execution duration, tools required, and motor skills used for each subtask. Once the recordings were annotated, we were able to look for correlations in the data indicating potential areas for transfer learning, with the goal of making previously

unrelated training data applicable. An agent simulator was then built based on the human task performance data. This simulator, capable of generating agents with a multitude of proficiencies and learning curves based on the observed human data, allowed for the proposed modeling technique to be applied to a much wider and more difficult range of worker types. Finally, we evaluate our system's performance both on its ability to estimate performance for previously untrained tasks and on its ability to extrapolate agent performance for already known tasks.

A.4.1 Data Collection

We recorded four participants performing multiple assemblies of three different types of IKEA furniture. Participants were given instructions for each task in the form of a hierarchical task network (the "task network"), providing both goal information and ordering constraints to the worker (Fig. A.1). These task networks were generated from a Semi-Markov Decision Process-compatible encoding of the instruction manuals that were provided. Each action in the task's SMDP was annotated with metadata including any tools required (e.g., "hex wrench", "hammer") in addition to a keyphrase describing the motor skill required (e.g., "peg in hole", "slide in place", "secure 2 bolts"). Skills and subtasks were coded at a granularity such that an annotation indicating a required tool or motor skill meant that the skill or subtask was dominated by the use of that tool or motor skill.

Each participant performed between 5 and 10 assemblies of each piece of furniture in succession. For each subtask or action represented in the task network, the duration of time the worker spent was annotated from the video recording. Each demonstration resulted in an execution trace of the task that indicates the order of subtasks completed and their durations. These execution traces, in the form of (agent name, subtask name, start time, end time)-tuples, serve as input for the model learning step.

A.4.2 Feature Analysis and Data Synthesis

A more in-depth analysis of recorded data provided more detailed features for each subtask. The data showed a clear experience curve as participants made procedural improvements over successive demonstrations, some nearly halving their execution time over the course of their trials. As such, we investigated the magnitude of change and rate of convergence to a stable execution proficiency to model this curve for each skill. Other features we examined included the amount of variance present once a stable execution strategy was achieved, as well as the range of bias of these values as measured across agents.

Additionally, we found correlations within agents between durations for subtasks that shared annotations either in terms of required tools or motor skill description. This is supported by the intuition that for any set of skills dominated by a particular action, such as the use of the hex wrench in "Secure Board A" and "Secure Frame" (Fig. A.1), an agent's performance is tied to its ability to use the tool.

Using these data, we developed generic agent templates that could generate a multitude of agents, simulating a range of proficiencies mimicking trends seen in the real-world data.

A.4.3 Evaluation Criteria

Our approach is evaluated based on its performance in two important use cases: adaptation to new tasks and extrapolation on known tasks. These metrics are measured using the average error on a per-skill basis across tasks and agents. The model is evaluated based on its ability to predict the time taken for each execution the agent performs of each skill, incorporating effects due to procedural improvement and variance.

In the case of novel task introduction, we evaluate the model's ability to use knowledge transfer. This is accomplished by evaluating predictions of an agent's performance exclusively on a new, untrained task, while varying the amount of training provided to non-target tasks. Additionally, we investigate effects that arise when modulating the number of non-target tasks provided. In the case of extrapolation-based evaluation, we evaluate the

model's ability to predict future performance across all tasks, given a variable amount of training data.

A.5 Agent Proficiency Modeling

Based on our analysis of the annotated video data, we predictively model an agent's skill performance based on four types of features: a heuristic prior for a skill's expected duration ('E'), a skill's experience curve ('xp'), an agent's tool proficiencies ('tp'), and an agent's motor skill proficiencies ('mp'). The heuristic prior used for each skill represents an expected duration (analogous to a common-knowledge estimate), based on the mean of observed execution times from the human-recorded data. This value is agent-agnostic, and does not directly represent any empirically derived knowledge from evaluated agents. A skill's experience curve represents the efficiency gains made from procedural improvements that occur over the course of practicing a skill or task. Tool and motor skill proficiencies are indicators of an agent's proficiency with particular tools or types of manipulation. Each feature is represented as a corrective scaling factor to be applied against the expected duration of the skill (E), with default values of 1.0 in the case of the experience curve, and 0 in the case of tool or motor proficiencies.

We combine these features by predicting a skill's execution duration with the following formula, given an agent a, skill s, expected duration of s from the heuristic prior $E(s)$, and demonstration number x:

$$\text{estimate}(a, s, x) = E(s) * xp(s, x) * (1 + tp(a, s) + mp(a, s)) \tag{A.1}$$

A.5.1 Experience Curve

The experience curve of a subtask represents the effects of transitioning from a novice to an experienced practitioner. This is represented by the changes in expected duration that

occur over the course of gaining familiarity with practice. These functions, also known as learning curves, have been studied extensively within operations research [260, 261, 262] as they are important in many real-world planning scenarios.

It is important to learn a subtask's experience curve for predicting the completion duration of a skill. It is equally important to learn for the purpose of correcting experience-related effects out of observed data, allowing for the isolation of other factors that may be contributing to a subtask's execution duration. Beneficially, doing so enables one to include early demonstrations in subtask analysis and feature extraction, as these initial samples will no longer be dominated by effects from procedural improvements. We compute this value across agents, independent of factors attributable to individual agent variation when possible, such as tool or motor skill proficiency, with the intention of isolating effects intrinsic to the particular skill or subtask's progression as familiarity increases.

As the experience curve can be heavily influenced by agent-specific factors, we attempt to remove effects attributable to each agent's tool and motor proficiency effects as a pre-processing step. A subtask's experience curve is calculated by partitioning observed executions into groups by demonstration index, independent of the performing agent. For each indexed demonstration (e.g., 1st, 2nd, ..., nth), we compute the ratios of the observed durations to the expected duration (given by the baseline prior) of the subtask. With these values, we perform a power fit to obtain an estimate for how experience changes the expected duration with practice.

For a pool of agents A, skill or subtask s, heuristic function E(s) returning the expected duration of s, and function d(agent, skill, iteration) returning the observed duration of a particular iteration of a skill by an agent, we compute the parameters (α, β) of the experience curve. We begin by constructing the set of coordinates $(x, y) \in P$, where x is the demonstration index (given X total demonstrations) and y is the execution duration ratio

with tool and motor proficiency effects removed.

$$P_s = \{(i, \frac{d(a,s,i)}{(1 + tp(s,a) + mp(s,a)) * E(s)}) | \forall a \in A, i \in [1, X]\}$$

$$\beta_s = \frac{|P_s| * \sum_{p \in P_s} ln(p_x) ln(p_y) - \sum_{p \in P_s} ln(p_x) * \sum_{p \in P_s} ln(p_y)}{|P_s| * \sum_{p \in P_s} ln(p_x)^2 - (\sum_{p \in P_s} ln(p_x))^2}$$

$$\alpha_s = \frac{\sum_{p \in P_s} ln(p_y) - \beta_s * \sum_{p \in P_s} ln(p_x)}{|P_s|}$$

With these parameters, we obtain the experience curve modifier value for iteration x:

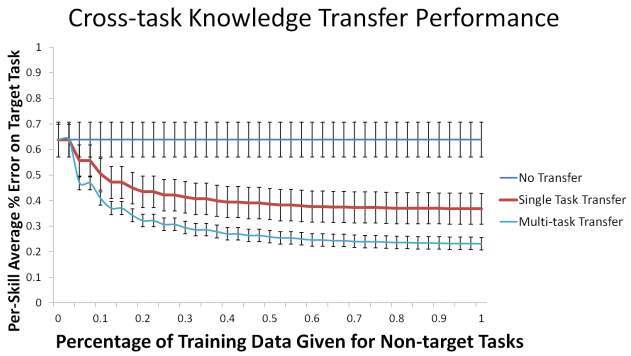
$$xp(s,x) = e^{\alpha_s} x^{\beta_s}$$

A.5.2 Tool and Motor Proficiency

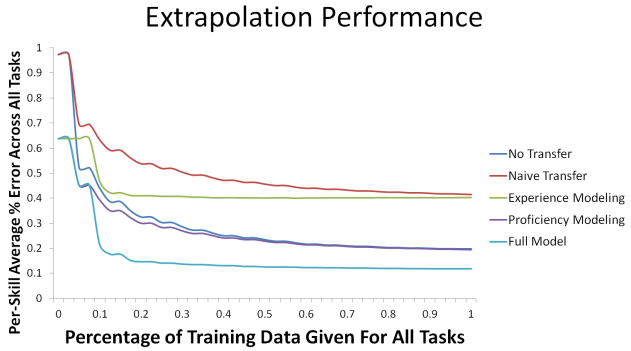
Analysis of the annotated video data revealed correlations between durations of subtasks with similarly annotated tool requirements or motor skill descriptions, suggesting that these could be substantial factors in a skill's execution duration. For some subtasks, this can be explained as the usage of a tool (e.g., how fast one can turn a screwdriver) dominating how long a subtask takes to be completed. For other subtasks, the type of motor skill involved can just as easily be a major factor, such as how well someone performs at peg-in-hole tasks or how adeptly they can manipulate a cumbersome type of object.

To measure these proficiencies, we isolate the observed execution times from any effects arising from an agent's learning curve. In practice this is imperfect, as agents may improve motor skills or tool proficiency over time concurrent with the task. In our data, these abilities were not observed to change in a perceptible way when contrasted with the procedural improvements attributed to the experience curve.

The calculation of an agent's proficiency for a given tool involves examining performance across all skills that utilize the target tool. To isolate the effects of tool use from other contributing factors, we ensure no effects from procedural improvements (as repre-



(a) Performance graph for estimating completion time of novel, untrained tasks using cross-task knowledge transfer.



(b) Predictor performance for estimating future completion time of tasks with prior experience.

Figure A.3: Evaluation results of the presented skill proficiency assessment predictor.

sented in an experience curve) are being mistakenly attributed to tool proficiency. Our approach accounts for this by limiting sampling whenever possible to demonstrations after the agent's performance curve has leveled off (where the magnitude of $\frac{\delta}{\delta x} xp(s, x)$ is small). Unlike experience curves that are calculated cross-agent, tool and motor proficiency values are calculated strictly within-agent as a personalization feature.

Given an agent (a), a set of skills or subtasks that agent has performed $(s \in S)$, their expected durations $(E(s))$, and the set of demonstrations for a given skill s by agent a $(D_{a,s})$, we can compute the agent's tool proficiency.

First, we define a function r that returns the average ratio of observed to expected

execution durations for a given agent and skill.

$$r(a,s) = \frac{1}{|D_{a,s}|} \sum_{d \in D_{a,s}} \frac{d}{E(s)}$$

Using this function, we can compute the weights to apply to each skill when assessing tool proficiency. In tool proficiency assessment, these weights are meant to assign more importance to skills that comprise a larger percentage of the target tool class and comprise a smaller percentage of their motor class. To do this, we assign weights inversely proportional to the distance between the mean of an individual skill's duration ratios and the mean of all skills utilizing the same tool. We also utilize the distance between a skill's duration ratio and the mean of all skills in its motor skill class. Intuitively, in measuring tool proficiency, this weighting scheme attributes more importance to the skill if it is either a poor representative of its motor skill class or if it is a good representative of its tool class.

We define T_s to be the set of all skills sharing the same required tool as skill s and M_s to be the set of all skills that share the same motor skill keywords as s. For an agent a and skill s, we compute the following weighting:

$$\omega_{a,s} = 1 - \frac{|r(a,s) - \sum_{t \in T_s} \frac{r(a,t)}{|T_s| * E(t)}|}{|r(a,s) - \sum_{t \in T_s} \frac{r(a,t)}{|T_s| * E(t)}| + |r(a,s) - \sum_{m \in M_s} \frac{r(a,m)}{|M_s| * E(m)}|}$$

Finally, we must compute the proportion of credit (γ) for any observed effect to assign to the tool proficiency (as opposed to a motor skill proficiency that may also be represented in the improvement).

This credit assignment is performed to proportionately distribute performance changes across all involved model features.

$$\gamma(s) = \frac{1}{2} * \left[\frac{|T_s \cap M_s|}{|T_s|} + 1 - \frac{|T_s \cap M_s|}{|M_s|} \right]$$

Combining these equations, we compute the tool proficiency of a given agent a for a

skill s:

$$tp(a,s) = \frac{1}{\sum_{t \in T_s} \omega_{a,t}} \sum_{t \in T_s} \gamma(t) * (r(a,t) - 1) * \omega_{a,t}$$

To calculate an agent's motor skill proficiency, we utilize the same process as for computing tool proficiency, replacing instances of T_s with M_s, and vice versa.

A.6 Evaluation and Discussion

We evaluate our model by testing its ability to predict execution durations for specific agents on a variety of tasks in the assembly domain. Our dataset included 20 simulated agents performing 25 demonstrations of each piece of furniture. This resulted in 75 total trials per agent and a total of 1500 demonstrations. We measured model performance in two scenarios: cross-task knowledge transfer and within-task extrapolation. Performance is measured across the entire collected data set of 1500 demonstrations, measuring the predictor's ability to generate accurate estimates of future results in addition to maintaining accurate representations of its history. Error is presented as $\frac{|\text{estimated} - \text{actual}|}{\text{actual}}$ for each skill execution in the data set.

A.6.1 Cross-Task Transfer Learning

In the cross-task knowledge transfer scenario, we define the 'target' task to be a novel, previously unseen task. Our predictor is then trained with execution data from all tasks except the target task. Performance is thus measured as the average, per-skill duration estimation error on the target task. We repeat this process using leave-one-out validation for all tasks, reporting the average, per-skill duration estimation error across each target task (Fig. **??**). The cross-task performance indicates the ability of the predictor to perform within-agent knowledge transfer.

In the no transfer case, there was no agent- or task-specific data to use, so the heuristic

prior is evaluated instead. In the single-task transfer case, the predictor was given a single task's worth of training data to estimate performance on the target task. In the multi-task transfer case, all non-target tasks were provided as training. Intuitively, as more tasks are added to the training set, more insights are available to boost cross-task performance.

Using solely the generic heuristic prior for each skill in the target task results in a substantial performance estimation error for our generated agent pool ($M = 0.638, \sigma = 0.136$). For a typical IKEA assembly task lasting 600 seconds, this means schedules based on the heuristic prior would typically be incorrect by approximately 382 seconds. Training the predictor with demonstrations of tasks sharing some tool requirements and motor skills dramatically increases performance. When trained with one other task, the prediction error reduces by over a third ($M = .368, \sigma = 0.06$). Providing training data on a second task further improves prediction results, resulting in a 24% error when fully trained ($M = 0.236, \sigma = 0.024$). In the 10 minute IKEA task example, this amounts to a prediction accuracy improvement of 4 minutes over a non-personalized approach.

These results are not meant to be representative of all tasks or skills, and may vary widely depending on the agents, tasks, tools, and motor skills being evaluated. However, they are encouraging and supportive of our work's premise: that building models of agent proficiencies that leverage transfer learning can provide valuable improvements over less personalized methods in planning domains.

A.6.2 Extrapolation-based Estimation

In the extrapolation scenario, we train our predictor on uniform amounts of execution data for each task. The model is then tested on its ability to estimate the future performance of each agent for each skill represented across each task. As in the cross-task transfer evaluation, performance is measured as the mean per-skill duration estimation error (Fig. **??**). Intuitively, this measures the predictor's ability to estimate future performance given a limited amount of direct target task observations.

We evaluate our predictor's ability to extrapolate on past performance by measuring performance across five methods: no transfer, naive transfer, experience modeling, proficiency modeling, and using our full model. In the no transfer case, agent performance is assumed to be the average of all demonstrations by that agent observed up to that point. The naive transfer predictor uses the average skill duration as computed across all agents as its estimate. The experience curve-based estimator fits a power curve to the unmodified observed data across all agents, using it as its predictor. Proficiency modeling utilizes the within-agent tool and motor skill proficiency estimation as a modifier on the heuristic prior. Finally, the full model combines the experience curve with tool and motor skill proficiency modeling (Eq. A.1). As studied within developmental literature, our model too improves its performance by leveraging behaviors noted in 7-8 year olds (cross-agent comparisons) and 10-12 year olds (cross-task transfer).

Our results strongly suggest that variations in agent performance rule out sole reliance on cross-agent transfer techniques for this scenario. In isolation, cross-agent methods take longer to converge and do so to less optimal solutions than their within-agent counterparts, with naive transfer and experience curve strategies peaking at approximately 41% error. Within-agent, cross-task methods do considerably better, with the proficiency modeling and no-transfer methods converging to approximately 20% error. Our full model, combining developmentally inspired aspects of cross- and within-agent knowledge transfer, performs noticeably better, rapidly converging to approximately 13% error.

As with the cross-task results, these findings are not meant to suggest a universal performance guarantee for any particular class of task or skill. These results support the premise that transfer learning is a promising approach to skill estimation, allowing for improved accuracy when considering multi-agent scenarios.

A.7 Conclusion

In this work we present a novel, developmentally inspired approach to estimating task performance. We use transfer learning techniques to generalize knowledge across tasks and across agents, allowing for more accurate estimates of skill execution durations with less training data. Our approach shows promising results for planning systems, particularly within the chosen manufacturing domain. These improvements are shown present both in cases of adapting and applying existing knowledge to novel tasks and extrapolating future performance from past experience.

Appendix B

Maintaining Motivation during Interactions with Collaborative Robots

Alongside physical assistance during tasks we envision in a collaborative scenario such as those encountered in the manufacturing industry, HRC and HRI include a wide range of collaborative scenarios during which we might wish a robot to provide assistance to the user throughout the interaction. These types of scenarios include tasks where the user and the robot are working towards the same goal, with the ultimate desire for the robot to help the user be more effective at accomplishing this goal.

This research aims to develop an adaptive human-robot interaction system that works with users over long periods of time to achieve a common goal that is beneficial to the user. This work is based on [118] [1]. The particular scenario I focus on is that of a robot companion interacting with adolescents, helping them succeed at achieving daily physical activity goals. To develop such a system, I propose a method of modeling the user's motivational state and employing this model in order to adapt motivational strategies best suited for each user. The proposed system uses both physical activity data obtained from

1. E. C. Grigore, "Modeling motivational states through interpreting physical activity data for adaptive robot companions", in Proceedings of the 23rd International Conference on User Modelling, Adaptation and Personalization (UMAP), Dublin, Ireland: Springer, 2015, June 29–July 3, pp. 379–384.

wearable sensors (such as wristband devices) and information acquired by the robot from its interaction partners.

B.0.1 Problem and Motivation

The benefits of physical activity are well known, as research shows that daily physical activity has wide-ranging benefits, from improving cognitive and academic performance to helping with bone development and health [263]. My work seeks to sustain these benefits with robot home companions through personalized, data-driven coaching.

The U. S. Department of Health and Human Services comprehensive guidelines on physical activity for individuals ages 6 and older state that children and adolescents should aim to accumulate at least 60 minutes of moderate- or vigorous-intensity aerobic physical activity on most days of the week, preferably daily [264]. Current evidence shows that levels of physical activity among youth remain low, and that levels of physical activity decline dramatically during adolescence [121]. These data show the importance of developing methods to keep adolescents on track to achieving daily-recommended levels of physical activity. With this in mind, I seek to develop a human-robot interaction system that motivates adolescents to engage in physical activity on a daily basis.

B.0.2 Background and Related Work

There exists a great deal of work concerning health and well-being applications, such as commercial systems that use motivational strategies to help users stay engaged in physical activity, or that support goal-setting and reflection for the user [265], [266]. Other prior work explores tracking of long term goals along with factors that have an important role in goal-setting theory [267].

In the field of human-computer interaction (HCI), wearable sensors are being used in the development of persuasive technologies that motivate healthy behavior. Work in this area is wide, ranging from activity recognition systems that employ this information to keep

users engaged in physical activity [268] to how such applications should be evaluated [269].

Social robots have also been used to promote and keep users engaged in physical activity within the field of human-robot interaction (HRI). Work in this area touches on investigating the role of praise and relational discourse [270], as well as on how to create a long-term interaction for help with weight loss [271].

The user model I am building employs an ontology-based approach. Ontologies are defined as explicit accounts of shared understanding in a given subject area, and bolster communication between users, experts, and technology systems [272]. Since ontologies are extremely useful in providing a formalism that specifies and clarifies concepts in a domain and the relationships among them, their value for health applications has been recognized by different lines of research. Such research includes the investigation of computational models of dialogue trying to simulate a human health counselor [273] and that of computerized behavioral protocols that help individuals improve their behaviors [274].

Although the work presented above is wide-ranging, the current paper aims to link all the components important in maintaining physical activity motivation into one system by leveraging human-robot interaction techniques together with the benefits of wearable sensors.

B.0.3 Research Questions and Intended Main Contributions

The key research questions I am investigating within this context are as follows. *How can I develop an ontology-based user model of people's motivational states by employing long-term human-robot interaction to gather information needed for the model? How can I then use this interpretation of the user's motivational state in order to create an adaptive robot companion that chooses appropriate motivational strategies to help the user stay on track to achieving daily physical activity goals?* The contributions expected from answering these two research questions consist of the following.

The first contribution represents the ontology-based user model, validated via expert

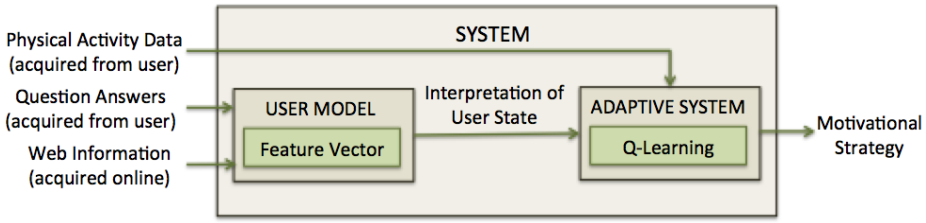


Figure B.1: System Diagram

interviews. The model makes use of long-term human-robot interaction by acquiring data from the user about his or her motivational state each interaction, and also acquires contextual information about external factors from online sources (e.g. weather). The second contribution represents an adaptive algorithm based on Q-Learning that uses both the output of the user model and the physical activity data collected from the wristband device. Its output consists of choosing the most appropriate motivational strategy the robot companion employs for a user each day.

Such a system would lie at the intersection of HRI, HCI, user modeling, and applied machine learning. Although there are a multitude of applications related to well-being and physical activity, there is no system to date that ties together continuous remote monitoring of user physical activity obtained from wearable sensors with an assistive robot for daily, long-term interactions.

B.0.4 Methodology and Design of the System

The robot platform chosen for this research is Keepon, a non-mobile platform with four degrees of freedom, designed to interact with children [146]. The adolescent wears a wristband device that keeps track of the number of steps taken throughout the day. He or she interacts with a robot once daily, both directly and via a phone application. The application presents an avatar for the robot with simple, 2D animations.

The system is depicted in Figure B.1 above. When the adolescent interacts with the robot, the back-story of the robot unfolds in order to keep the user engaged in the long-term interaction. The robot asks the user a series of questions meant to obtain information about his or her motivational state, as well as acquires external information from online

sources (e.g. weather information). This data is fed into the user model. The adaptive system then takes the user model output, together with the physical activity data from the wearable sensor, and outputs an appropriate motivational strategy for the user. This strategy is used to shape the new physical activity goal the adolescent needs to accomplish the next day.

B.0.5 User Model

The ontology my model relies upon needs to represent core concepts related to creating a profile of a user and keeping track of how this changes over time. Thus, I have initially performed extensive literature review in order to identify the key elements influencing the achievement of a goal. These are factors long studied in goal setting theory [275], social cognitive theory [276], and theory of planned behavior [277], and are introduced below. Since these key elements are part of the ontology-based model, the creation and validation of this model does not rely solely on literature review, but it is also heavily based on expert interviews.

As shown in Figure B.1, the inputs to the user model are dichotomized into two categories. The first represents the user's answers to the robot's questions about how the participant felt while trying to accomplish a goal. These questions regard key elements influencing the achievement of a goal, namely *socio-structural factors* (composed of *facilitators* and *impediments*), *social pressure*, *self-efficacy*, and *attitude toward behavior*. They are phrased in an easy to understand manner, avoiding academic and arcane terms. The second input category represents external information acquired by the system online about factors that might have influenced the achievement of a goal, e.g. daily weather, school schedule changes. This information is fed into the socio-structural factors, into either the facilitators or impediments feature. All of these core factors make up the feature vector for a user, which contains numerical values based on the user's responses during the interaction with the robot.

The output of the user model is an interpretation of the user's state. It represents the state of the world (the state the user is in) at each time step (each day). The numerical value of each feature is mapped onto one of three discrete categories representing intensity levels, namely low, neutral, and high. This mapping is applied to reduce the state space size and produce a more intuitive interpretation of the user's state. For example, a value of 7 on a 1-to-7 scale for self-efficacy would be assigned to the "high" level. The interpretation thus consists of a feature vector with intensity levels for each feature.

To validate the user model and create an ontology that links the core elements modeling a user to motivational strategies employed by experts (e.g. health and physical activity counselors, physical education teachers, personal trainers), I utilize a formal interview process. This aims to validate and identify missing main concepts in the domain of physical exercise motivation and the relationships among them. The interview is structured in two parts, as follows.

The first part asks the expert to list (1) important factors for keeping students engaged in physical activity (*corresponding to the main concepts used in the ontology*), (2) techniques used by experts to identify the students' motivational state (*used to create a mapping from users' answers to the state space*), (3) information about students that helps the expert interpret the students' success or failure (*used to validate the user model's output*), and (4) strategies and techniques employed by the expert to keep students engaged in physical activity (*corresponding to motivational strategies*).

The second part presents the expert with the same categories, but with given answers (drawn from literature). The expert is asked to indicate how much he or she agrees with the specific answer, using a Likert scale. This part aims to check if experts indeed use concepts identified in literature as key factors, acquire new key factors used in practice, and acquire the basis for creating an ontology linking the user model to the motivational strategies used by experts.

B.0.6 Adaptive System

My adaptive human-robot interaction system chooses a motivational strategy based on the user model's interpretation of "why" and "how" the user was (un)successful at his or her previous goal, i.e. the interpretation of the user's state. This is possible since the motivational strategies identified from literature (such as cooperative and competitive strategies, making the user aware of the importance of engaging in physical activity, autonomy support, structure, and involvement) are associated with factors present in the user model, e.g. "Intrinsic motivation is associated with the desire to master a task and the satisfaction that comes with that, whereas extrinsic motivation refers to completing tasks solely as an ego boost, be it beating peers in a competition, or receiving praise from a parent, teacher or colleague" [278].

My system adapts to an individual user by following a Q-learning approach. In the current work, the actions the "agent" (in our case the robot) can take are the different motivational strategies mentioned above, $a \in \{m_1, m_2, m_3, ...\}$, which are to be finalized after concluding expert interviews. The states of the world are the interpretations of a user's state (the output of the user model). Thus, a state is a feature vector representing the interpretation of the user's state, and is defined as $intrp_t = [f_1\ f_2\ f_3\ f_4\ f_5]$, where the f_is represent the features discussed above that take values associated with intensity levels. The reward for a particular state is the difference between the number of steps taken by the user and the number of steps set as the goal, $r_t = \#steps_{taken} - \#steps_{goal}$.

At time step t, the algorithm observes the current state $intrp_t$, chooses an action a_t among the motivational strategies based on an ϵ-greedy policy, takes the action, observes the reward r_t as well as the new state $intrp_{t+1}$, and then updates the Q-value for the state using the observed reward and the maximum reward possible for the next state. The update is performed based on the following formula: $Q(intrp_t, a_t) = Q(intrp_t, a_t) + \alpha[r_t + \gamma \max_{a_{t+1}} Q(intrp_{t+1}, a_{t+1}) - Q(intrp_t, a_t)]$, where α and γ are both set between 0 and 1 and specify how quickly learning occurs and how much future rewards matter, respectively.

The algorithm will thus work toward finding an optimal policy, in order to maximize the expected return, i.e. positive reward or small values for negative rewards.

In Section 6.1 and Section 6.2, I present two studies exploring this idea of maintaining engagement and engendering positive impressions of the robot in collaborative interactions that are intended for the long-term. These valuable tools can readily be employed by robots in HRC contexts, where we envision robots aiding humans throughout their daily tasks, be it in manufacturing settings like factories, or in our homes.

Bibliography

[1] Andrea Bauer, Dirk Wollherr, and Martin Buss. Human–robot collaboration: a survey. *International Journal of Humanoid Robotics*, 5(01):47–66, 2008.

[2] Oliver Kroemer, Christian Daniel, Gerhard Neumann, Herke Van Hoof, and Jan Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1503–1510. IEEE, 2015.

[3] Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *Proceedings of the 29th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.

[4] Justin Fu, Sergey Levine, and Pieter 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*, pages 4019–4026. IEEE, 2016.

[5] Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

[6] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[7] Koichi Ogawara, Jun Takamatsu, Hiroshi Kimura, and Katsushi Ikeuchi. Modeling manipulation interactions by hidden markov models. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 2, pages 1096–1101. IEEE, 2002.

[8] Keni Bernardin, Koichi Ogawara, Katsushi Ikeuchi, and Ruediger Dillmann. A sensor fusion approach for recognizing continuous human grasping sequences using hidden markov models. *IEEE Transactions on Robotics*, 21(1):47–57, 2005.

[9] Jie Yang and Yangsheng Xu. Hidden markov model for gesture recognition. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Robotics Inst, 1994.

[10] Emily B Fox, Michael C Hughes, Erik B Sudderth, Michael I Jordan, et al. Joint modeling of multiple time series via the beta process with application to motion capture segmentation. *The Annals of Applied Statistics*, 8(3):1281–1313, 2014.

[11] Maria Fox, Malik Ghallab, Guillaume Infantes, and Derek Long. Robot introspection through learned hidden markov models. *Artificial Intelligence*, 170(2):59–113, 2006.

[12] Qiuming Zhu. Hidden markov model for dynamic obstacle avoidance of mobile robot navigation. *IEEE Transactions on Robotics and Automation*, 7(3):390–397, 1991.

[13] Aleksandar Vakanski, Iraj Mantegh, Andrew Irish, and Farrokh 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)*, 42(4):1039–1052, 2012.

[14] Sylvain Calinon, Florent D'Halluin, Eric L Sauser, Darwin G Caldwell, and Aude 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*, 17(2), 2010.

[15] Monica N Nicolescu and Maja J Mataric. Learning and interacting in human-robot domains. *IEEE Transactions on Systems, man, and Cybernetics-part A: Systems and Humans*, 31(5):419–430, 2001.

[16] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[17] Mitsuo Kawato, Yoji Uno, Michiaki Isobe, and Ryoji Suzuki. Hierarchical neural network model for voluntary movement with application to robotics. *IEEE Control Systems Magazine*, 8(2):8–15, 1988.

[18] Eric W Aboaf, Christopher G Atkeson, and David J Reinkensmeyer. Task-level robot learning. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 1309–1310. IEEE, 1988.

[19] Jo-Anne Ting, Mrinal Kalakrishnan, Sethu Vijayakumar, and Stefan Schaal. Bayesian kernel shaping for learning control. In *Advances in neural information processing systems*, pages 1673–1680, 2009.

[20] Oliver Herbort, Martin Butz, and Gerulf Pedersen. The sure_reach model for motor learning and control of a redundant arm: from modeling human behavior to applications in robotics. *From motor learning to interaction learning in robots*, pages 85–106, 2010.

[21] René Felix Reinhart and Jochen J Steil. Attractor-based computation with reservoirs for online learning of inverse kinematics. In *ESANN*, pages 257–262, 2009.

[22] Mrinal Kalakrishnan, Jonas Buchli, Peter Pastor, and Stefan Schaal. Learning locomotion over rough terrain using terrain templates. In *Intelligent Robots and Systems,*

2009. IROS 2009. IEEE/RSJ International Conference on, pages 167–172. IEEE, 2009.

[23] Anelia Angelova, Larry Matthies, Daniel Helmick, and Pietro Perona. Learning and prediction of slip from visual information. *Journal of Field Robotics*, 24(3):205–231, 2007.

[24] Christopher G Atkeson and Jun Morimoto. Nonparametric representation of policies and value functions: A trajectory-based approach. In *Advances in neural information processing systems*, pages 1643–1650, 2003.

[25] Jun Morimoto, Garth Zeglin, and Christopher G Atkeson. Minimax differential dynamic programming: Application to a biped walking robot. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1927–1932. IEEE, 2003.

[26] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Advances in neural information processing systems*, pages 1–8, 2007.

[27] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[28] Duy Nguyen-Tuong and Jan R Peters. Incremental sparsification for real-time online model learning. In *International Conference on Artificial Intelligence and Statistics*, pages 557–564, 2010.

[29] Jan Peters and Stefan Schaal. Learning to control in operational space. *The International Journal of Robotics Research*, 27(2):197–212, 2008.

[30] Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008.

[31] Camille Salaün, Vincent Padois, and Olivier Sigaud. Control of redundant robots using learned models: an operational space control approach. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 878–885. IEEE, 2009.

[32] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[33] Todd Hester, Michael Quinlan, and Peter Stone. Generalized model learning for reinforcement learning on a humanoid robot. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2369–2374. IEEE, 2010.

[34] Georgios Theocharous, Khashayar Rohanimanesh, and Sridhar Maharlevan. Learning hierarchical observable markov decision process models for robot navigation. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 1, pages 511–516. IEEE, 2001.

[35] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2015.

[36] Kenji Doya, Kazuyuki Samejima, Ken-ichi Katagiri, and Mitsuo Kawato. Multiple model-based reinforcement learning. *Neural computation*, 14(6):1347–1369, 2002.

[37] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

[38] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[39] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. In *Innovations in Multi-Agent Systems and Applications-1*, pages 183–221. Springer, 2010.

[40] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.

[41] Ronald Edward Parr. *Hierarchical control and learning for Markov decision processes*. PhD thesis, Citeseer, 1998.

[42] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.(JAIR)*, 13:227–303, 2000.

[43] Mohammad Ghavamzadeh, Sridhar Mahadevan, and Rajbala Makar. Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 13(2):197–229, 2006.

[44] Rajbala Makar, Sridhar Mahadevan, and Mohammad Ghavamzadeh. Hierarchical multi-agent reinforcement learning. In *Proceedings of the fifth international conference on Autonomous agents*, pages 246–253. ACM, 2001.

[45] Hong Lin. *Architectural design of multi-agent systems: technologies and techniques: technologies and techniques*. IGI Global, 2007.

[46] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

[47] Michael L Littman. Friend-or-foe q-learning in general-sum games. In *ICML*, volume 1, pages 322–328, 2001.

[48] Peter Stone and Manuela Veloso. Team-partitioned, opaque-transition reinforcement learning. In *Proceedings of the third annual conference on Autonomous Agents*, pages 206–212. ACM, 1999.

[49] Maja J Matarić. Reinforcement learning in the multi-robot domain. In *Robot colonies*, pages 73–83. Springer, 1997.

[50] Nhan Nguyen-Duc-Thanh, Sungyoung Lee, and Donghan Kim. Two-stage hidden markov model in gesture recognition for human robot interaction. *International Journal of Advanced Robotic Systems*, 9(2):39, 2012.

[51] Geir E Hovland, Pavan Sikka, and Brenan J McCarragher. Skill acquisition from human demonstration using a hidden markov model. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 3, pages 2706–2711. Ieee, 1996.

[52] Yangsheng Xu and Jie Yang. Towards human-robot coordination: skill modeling and transferring via hidden markov model. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 2, pages 1906–1911. IEEE, 1995.

[53] Richard Kelley, Alireza Tavakkoli, Christopher King, Monica Nicolescu, Mircea Nicolescu, and George 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*, pages 367–374. ACM, 2008.

[54] Uri Kartoun, Helman Stern, and Yael Edan. A human-robot collaborative reinforcement learning algorithm. *Journal of Intelligent & Robotic Systems*, 60(2):217–239, 2010.

[55] Andrea Lockerd Thomaz and Cynthia Breazeal. Reinforcement learning with human teachers: Evidence of feedback and guidance with implications for learning performance. In *AAAI*, volume 6, pages 1000–1005, 2006.

[56] Tetsuya Ogata, Noritaka Masago, Shigeki Sugano, and Jun Tani. Interactive learning in human-robot collaboration. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 162–167. IEEE, 2003.

[57] Jim Mainprice and Dmitry Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 299–306. IEEE, 2013.

[58] Jim Mainprice, Rafi Hayne, and Dmitry 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*, pages 885–892. IEEE, 2015.

[59] Chien-Ming Huang and Bilge Mutlu. Anticipatory robot control for efficient human-robot collaboration. In *Human-Robot Interaction (HRI), 2016 11th ACM/IEEE International Conference on*, pages 83–90. IEEE, 2016.

[60] Guy Hoffman and Cynthia Breazeal. Cost-based anticipatory action selection for human–robot fluency. *IEEE transactions on robotics*, 23(5):952–961, 2007.

[61] Guy Hoffman and Cynthia Breazeal. Collaboration in human-robot teams. In *Proc. of the AIAA 1st Intelligent Systems Technical Conference, Chicago, IL, USA*, 2004.

[62] Cynthia Breazeal, Guy Hoffman, and Andrea 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*, pages 1030–1037. IEEE Computer Society, 2004.

[63] Stefanos Nikolaidis, Ramya Ramakrishnan, Keren Gu, and Julie 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*, pages 189–196. ACM, 2015.

[64] Stefanos Nikolaidis, Przemyslaw Lasota, Gregory Rossano, Carlos Martinez, Thomas Fuhlbrigge, and Julie Shah. Human-robot collaboration in manufacturing: Quantitative evaluation of predictable, convergent joint action. In *Robotics (isr), 2013 44th international symposium on*, pages 1–6. IEEE, 2013.

[65] Bradley Hayes and Brian Scassellati. Effective robot teammate behaviors for supporting sequential manipulation tasks. In *IROS*, pages 6374–6380, 2015.

[66] Bradley Hayes. *Supportive Behaviors for Human-Robot Teaming.* PhD thesis, Yale University, 2015.

[67] Bradley Hayes and Brian Scassellati. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5469–5476. IEEE, 2016.

[68] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Readings in speech recognition*, pages 267–296. Elsevier, 1990.

[69] Elena Corina Grigore and Brian Scassellati. Discovering Action Primitive Granularity from Human Motion for Human-Robot Collaboration. In *Robotics: Science and Systems (RSS)*, 2017, July 12–16.

[70] Stefan Schaal. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pages 261–280. Springer, 2006.

[71] Austin Tate. Generating project networks. In *Proceedings of the 5th international joint conference on Artificial intelligence-Volume 2*, pages 888–893. Morgan Kaufmann Publishers Inc., 1977.

[72] Michael C Hughes, Emily Fox, and Erik B Sudderth. Effective split-merge monte carlo methods for nonparametric models of sequential data. In *Advances in Neural Information Processing Systems*, pages 1295–1303, 2012.

[73] Qiang Yang. Formalizing planning knowledge for hierarchical planning. *Computational intelligence*, 6(1):12–24, 1990.

[74] Negin Nejati, Pat Langley, and Tolga Konik. Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning*, pages 665–672. ACM, 2006.

[75] Anahita Mohseni-Kabir, Charles Rich, Sonia Chernova, Candace L Sidner, and Daniel Miller. Interactive hierarchical task learning from a single demonstration. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 205–212. ACM, 2015.

[76] George Konidaris, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

[77] Dana Kulić, Christian Ott, Dongheui Lee, Junichi Ishikawa, and Yoshihiko Nakamura. Incremental learning of full body motion primitives and their sequencing through human motion observation. *The International Journal of Robotics Research*, page 0278364911426178, 2011.

[78] Jesse Butterfield, Sarah Osentoski, Graylin Jay, and Odest Chadwicke Jenkins. Learning from demonstration using a multi-valued function regressor for time-series

data. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 328–333. IEEE, 2010.

[79] Volker Krüger, Dennis Herzog, Sanmohan Baby, Aleš Ude, and Danica Kragic. Learning actions from observations. *IEEE Robotics Automation Magazine*, 17(2):3—43, 2010.

[80] Yi Li, Cornelia Fermuller, Yiannis Aloimonos, and Hui Ji. Learning shift-invariant sparse representation of actions. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2630–2637. IEEE, 2010.

[81] Yifan Shi, Yan Huang, David Minnen, Aaron Bobick, and Irfan Essa. Propagation networks for recognition of partially ordered sequential action. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–862. IEEE, 2004.

[82] Odest Chadwicke Jenkins and Maja J Mataric. Deriving action and behavior primitives from human motion data. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2551–2556. IEEE, 2002.

[83] Jernej Barbič, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K Hodgins, and Nancy S Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface 2004*, pages 185–194. Canadian Human-Computer Communications Society, 2004.

[84] Sven Hellbach, Julian P Eggert, Edgar Körner, and Horst-Michael Gross. Basis decomposition of motion trajectories using spatio-temporal nmf. In *International Conference on Artificial Neural Networks*, pages 804–814. Springer, 2009.

[85] Olivier Mangin and Pierre-Yves Oudeyer. Learning to recognize parallel combinations of human motion primitives with linguistic descriptions using non-negative ma-

trix factorization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3268–3275. IEEE, 2012.

[86] Thomas B Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer vision and image understanding*, 81(3):231–268, 2001.

[87] Yang Yang, Imran Saleemi, and Mubarak Shah. Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1635–1648, 2013.

[88] Odest Chadwicke Jenkins and Maja J Mataric. Automated modularization of human motion into actions and behaviors. 2002.

[89] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andre S Barreto. Constructing skill trees for reinforcement learning agents from demonstration trajectories. In *Advances in neural information processing systems*, pages 1162–1170, 2010.

[90] Daniel H Grollman and Odest Chadwicke Jenkins. Incremental learning of subtasks from unsegmented demonstration. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 261–266. IEEE, 2010.

[91] Silvia Chiappa and Jan R Peters. Movement extraction by detecting dynamics switches and repetitions. In *Advances in neural information processing systems*, pages 388–396, 2010.

[92] Scott Niekum, Sarah Osentoski, George Konidaris, Sachin Chitta, Bhaskara Marthi, and Andrew G Barto. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

[93] PhaseSpace. Phasespace motion capture, 2013. Available at http://www.phasespace.com/.

[94] Michael Hughes. Nonparametric bayesian inference for sequential data., 2012. Available at http://michaelchughes.github.io/NPBayesHMM/.

[95] Romain Thibaux and Michael I Jordan. Hierarchical Beta Processes and the Indian Buffet Process. In *AISTATS*, volume 2, pages 564–571, 2007.

[96] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[97] Donald J Berndt and James Clifford. Using Dynamic Time Warping to Find Patterns in Time Series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[98] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[99] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270. ACM, 2012.

[100] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[101] Elena Corina Grigore, Olivier Mangin, Alessandro Roncone, and Brian Scassellati. Predicting Supportive Behaviors for Human-Robot Collaboration. In *Proceedings of*

the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2018, July 10–15. Extended Abstract. To Appear.

[102] Alessandro Roncone, Olivier Mangin, and Brian Scassellati. Transparent role assignment and task allocation in human robot collaboration. *Robotics and Automation (ICRA), IEEE International Conference on*, pages 1014–1021, May 2017.

[103] Olivier Mangin, Alessandro Roncone, and Brian Scassellati. How to be helpful? implementing supportive behaviors for human-robot collaboration, 2017.

[104] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[105] S. Zeylikman, S. Widder, A. Roncone, O. Mangin, and B. Scassellati. The HRC model set for human-robot collaboration research. 2017.

[106] Elena Corina Grigore, Alessandro Roncone, Olivier Mangin, and Brian Scassellati. Preference-Based Assistance Prediction for Human-Robot Collaboration Tasks. 2018. In submission.

[107] Elena Corina Grigore and Brian Scassellati. Hierarchical Multi-Agent Reinforcement Learning through Communicative Actions for Human-Robot Collaboration. In *Proceedings of the Future of Interactive Learning Machines (FILM) Workshop at the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, 2016, December 5–10. Full paper.

[108] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

[109] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008*, 2008.

[110] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[111] Ronald A Howard. *Semi-Markov and decision processes*. Wiley, 1971.

[112] Martin L Puterman. Markov decision processes: Discrete dynamic stochastic programming. *New York, NY: John Wiley. doi*, 10:9780470316887, 1994.

[113] Mohammad Ghavamzadeh. *Hierarchical reinforcement learning in continuous state and multi-agent environments*. PhD thesis, University of Massachusetts Amherst, 2005.

[114] Khashayar Rohanimanesh and Sridhar Mahadevan. Learning to take concurrent actions. In *Advances in neural information processing systems*, pages 1619–1626, 2002.

[115] WW Szczechla, SA Connell, JA Filar, and OJ Vrieze. On the puiseux series expansion of the limit discount equation of stochastic games. *SIAM journal on control and optimization*, 35(3):860–875, 1997.

[116] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, University of Cambridge, 1989.

[117] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.

[118] Elena Corina Grigore. Modeling Motivational States Through Interpreting Physical Activity Data for Adaptive Robot Companions. In *Proceedings of the 23rd International Conference on User Modelling, Adaptation and Personalization (UMAP)*, pages 379–384. Springer, 2015, June 29–July 3.

[119] Marcel Heerink et al. Enjoyment intention to use and actual use of a conversational robot by elderly people. In *Proceedings of the 3rd ACM/IEEE International Conference on Human Robot Interaction*, pages 113–120, New York, USA, 2008. ACM.

[120] Elaine Short, Katelyn Swift-Spong, Jillian Greczek, Aditi Ramachandran, Alexandru Litoiu, Elena Corina Grigore, David Feil-Seifer, Samuel Shuster, Jin Joo Lee, Shaobo Huang, et al. How to Train Your Dragonbot: Socially Assistive Robots for Teaching Children about Nutrition through Play. In *Proceedings of the 23rd IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 924–929. IEEE, 2014, August 25–29.

[121] Physical activity guidelines for americans midcourse report: Strategies to increase physical activity among youth. Technical report, U. S. Department of Health and Human Services, 2013.

[122] Colin Boreham and Chris Riddoch. The physical activity, fitness and health of children. *Journal of sports sciences*, 19(12):915–929, 2001.

[123] Bess H Marcus and LeighAnn Forsyth. *Motivating people to be physically active*. Human Kinetics, 2003.

[124] Matteo Gerosa et al. A review of asr technologies for children's speech. In *Proceedings of the 2nd Workshop on Child, Computer and Interaction*. ACM, 2009.

[125] Jon Barker, Ricard Marxer, Emmanuel Vincent, and Shinji Watanabe. The third-chime'speech separation and recognition challenge: Dataset, task and baselines. In *2015 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU 2015)*, 2015.

[126] Paul Baxter, Rachel Wood, and Tony Belpaeme. A touchscreen-based 'sandtray' to facilitate, mediate and contextualise human-robot social interaction. In *Proceed-*

ings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI '12, pages 105–106, New York, NY, USA, 2012. ACM.

[127] Pierre-Yves Oudeyer, Pierre Rouanet, and David Filliat. An integrated system for teaching new visually grounded words to a robot for non-expert users using a mobile device. In *IEEE-RAS International Conference on Humanoid Robots*, Tsukuba, Japan, 2009.

[128] Ericsson. Ericsson mobility report. February 2016.

[129] Andrey Kudryavstev. Automatic speech recognition services comparison, 2016. Available at `http://blog.griddynamics.com/2016/01/automatic-speech-recognition-services.html`.

[130] Javi F Gorostiza et al. Multimodal human-robot interaction framework for a personal robot. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pages 39–44. IEEE, 2006.

[131] Dennis Perzanowski, Alan C Schultz, William Adams, Elaine Marsh, and Magda Bugajska. Building a multimodal human-robot interface. *Intelligent Systems, IEEE*, 16(1):16–21, 2001.

[132] Paul E. Rybski et al. Interactive robot task training through dialog and demonstration. In *Proceedings of the ACM/IEEE International Conference on Human-robot Interaction*, pages 49–56, New York, USA, 2007. ACM.

[133] David B Roe, Jay G Wilpon, et al. *Voice communication between humans and machines*. National Academies Press, 1994.

[134] Lingyun Qiu and Izak Benbasat. Online consumer trust and live help interfaces: The effects of text-to-speech voice and three-dimensional avatars. *International journal of human-computer interaction*, 19(1):75–94, 2005.

[135] Lingyun Qiu and Izak Benbasat. An investigation into the effects of text-to-speech voice and 3d avatars on the perception of presence and flow of live help in electronic commerce. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(4):329–355, 2005.

[136] Elena Corina Grigore. Modeling motivational states through interpreting physical activity data for adaptive robot companions. In *User Modeling, Adaptation and Personalization*, pages 379–384. Springer, 2015.

[137] MJ Mendelson and F Aboud. Mcgill friendship questionnaire—respondent's affection (mfq-ra). *Measurement Instrument Database for the Social Science*, 2012.

[138] Chad Harms and Frank Biocca. Internal consistency and reliability of the networked minds measure of social presence. 2004.

[139] Linda Tickle-Degnen and Robert Rosenthal. The nature of rapport and its nonverbal correlates. *Psychological inquiry*, 1(4):285–293, 1990.

[140] Iolanda Leite et al. "Why can't we be friends?" an empathic game companion for long-term interaction. In *Intelligent Virtual Agents*, Lecture Notes in Computer Science, pages 315–321. Springer Berlin Heidelberg, Sept 2010.

[141] John Short et al. The social psychology of telecommunications. 1976.

[142] André Pereira, Rui Prada, and Ana Paiva. Socially present board game opponents. In *Advances in Computer Entertainment*, pages 101–116. Springer, 2012.

[143] Paul Skalski and Ron Tamborini. The role of social presence in interactive agent-based persuasion. *Media psychology*, 10(3):385–413, 2007.

[144] Sigurdur O Adalgeirsson and Cynthia Breazeal. Mebot: a robotic platform for socially embodied presence. In *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 15–22. IEEE Press, 2010.

[145] Mel Slater. Place illusion and plausibility can lead to realistic behaviour in immersive virtual environments. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 364(1535):3549–3557, 2009.

[146] Hideki Kozima, Marek P Michalowski, and Cocoro Nakagawa. Keepon. *International Journal of Social Robotics*, 1(1):3–18, 2009.

[147] Google Inc. Android speech, 2014. Available at `http://developer.android.com/reference/android/speech/package-summary.html`.

[148] Elena Corina Grigore, Andre Pereira, Jie Jessica Yang, Ian Zhou, David Wang, and Brian Scassellati. Comparing Ways to Trigger Migration between a Robot and a Virtually Embodied Character. In *Proceedings of the 8th International Conference on Social Robotics (ICSR)*, pages 839–849. **Best student paper finalist.** Springer, 2016, November 1–3.

[149] Wilma Bainbridge et al. The effect of presence on human-robot interaction. In *Int'l Symposium on Robot and Human Interactive Communication*, pages 701–706. IEEE, 2008.

[150] Daniel Leyzberg, Samuel Spaulding, Mariya Toneva, and Brian Scassellati. The physical presence of a robot tutor increases cognitive learning gains. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society. Austin, TX: Cognitive Science Society*, 2012.

[151] Christoph Bartneck. Interacting with an embodied emotional character. In *Proceedings of the 2003 international conference on Designing pleasurable products and interfaces*, pages 55–60. ACM, 2003.

[152] André Pereira, Rui Prada, and Ana Paiva. Improving social presence in human-agent interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1449–1458. ACM, 2014.

[153] Paulo F Gomes, Alberto Sardinha, Elena Márquez Segura, Henriette Cramer, and Ana Paiva. Migration between two embodiments of an artificial pet. *International Journal of Humanoid Robotics*, 11(01):1450001, 2014.

[154] Kerstin Dautenhahn. Roles and functions of robots in human society: implications from research in autism therapy. *Robotica*, 21(04):443–452, 2003.

[155] Brian R Duffy, Gregory MP O'Hare, Alan N Martin, John F Bradley, and Bianca Schön. Agent chameleons: Agent minds and bodies. In *Computer Animation and Social Agents, 2003. 16th International Conference on*, pages 118–125. IEEE, 2003.

[156] Alan Martin, Gregory MP O'hare, Brian R Duffy, Bianca Schön, and John F Bradley. Maintaining the identity of dynamically embodied agents. In *Intelligent Virtual Agents*, pages 454–465. Springer, 2005.

[157] Alan Martin, Greg MP O'Hare, Bianca Schön, John F Bradley, and Brian R Duffy. Intentional embodied agents. In *18th International Conference on Computer Animation and Social Agents (CASA 2005), Hong Kong, China, October 17-25, 2005*, 2005.

[158] Krzysztof Arent and Bogdan Kreczmer. Identity of a companion, migrating between robots without common communication modalities: Initial results of vhri study. In *Methods and Models in Automation and Robotics (MMAR), 2013 18th International Conference on*, pages 109–114. IEEE, 2013.

[159] Krzysztof Arent, Bogdan Kreczmer, and Lukasz Malek. Identity of socially interactive robotic twins: initial results of vhri study. In *Methods and Models in Automa-*

tion and Robotics (MMAR), 2011 16th International Conference on, pages 381–386. IEEE, 2011.

[160] Ruth Aylett, Michael Kriegel, Iain Wallace, Elena Márquez Segura, Johanna Mecurio, Stina Nylander, and Patricia Vargas. Do i remember you? memory and identity in multiple embodiments. In *RO-MAN, 2013 IEEE*, pages 143–148. IEEE, 2013.

[161] David Robert et al. Blended reality characters. In *Int'l Conference on Human-Robot Interaction*, pages 359–366. ACM, 2012.

[162] Kohei Ogawa and Tetsuo Ono. Itaco: Constructing an emotional relationship between human and robot. In *Robot and Human Interactive Communication, 2008. RO-MAN 2008. The 17th IEEE International Symposium on*, pages 35–40. IEEE, 2008.

[163] Michita Imai, Tetsuo Ono, and Tameyuki Etani. Agent migration: communications between a human and robot. In *Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE Int'l Conference on*, volume 4, pages 1044–1048. IEEE, 1999.

[164] Joanna Lumsden. Emerging perspectives on the design, use, and evaluation of mobile and handheld devices. 2015.

[165] Apurv Suman, Rebecca Marvin, Elena Corina Grigore, Henny Admoni, and Brian Scassellati. Prior Behavior Impacts Human Mimicry of Robots. In *Proceedings of the 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1057–1062, 2016, August 26–31.

[166] Tanya L Chartrand and Jessica L Lakin. The antecedents and consequences of human behavioral mimicry. *Annual review of psychology*, 64:285–308, 2013.

[167] Tanya L Chartrand. The role of conscious awareness in consumer behavior. *Journal of Consumer Psychology*, 15(3):203–210, 2005.

[168] Katherine White and Jennifer J Argo. When imitation doesn't flatter: The role of consumer distinctiveness in responses to mimicry. *Journal of Consumer Research*, 38(4):667–680, 2011.

[169] Tanya L Chartrand and John A Bargh. The chameleon effect: The perception–behavior link and social interaction. *Journal of personality and social psychology*, 76(6):893, 1999.

[170] Temma Ehrenfeld. Reflections on mirror neurons. *Association for Psychological Science Observer*, 24(3), 2011.

[171] Jessica L Lakin, Valerie E Jefferis, Clara Michelle Cheng, and Tanya L Chartrand. The chameleon effect as social glue: Evidence for the evolutionary significance of nonconscious mimicry. *Journal of nonverbal behavior*, 27(3):145–162, 2003.

[172] Janet B Bavelas, Alex Black, Charles R Lemery, and Jennifer Mullett. " i show how you feel": Motor mimicry as a communicative act. *Journal of Personality and Social Psychology*, 50(2):322, 1986.

[173] William W Maddux, Elizabeth Mullen, and Adam D Galinsky. Chameleons bake bigger pies and take bigger pieces: Strategic behavioral mimicry facilitates negotiation outcomes. *Journal of Experimental Social Psychology*, 44(2):461–468, 2008.

[174] Mariëlle Stel and Roos Vonk. Mimicry in social interaction: Benefits for mimickers, mimickees, and their interaction. *British Journal of Psychology*, 101(2):311–323, 2010.

[175] Roderick I Swaab, William W Maddux, and Marwan Sinaceur. Early words that work: When and how virtual linguistic mimicry facilitates negotiation outcomes. *Journal of Experimental Social Psychology*, 47(3):616–621, 2011.

[176] Patrick Bourgeois and Ursula Hess. The impact of social context on mimicry. *Biological psychology*, 77(3):343–352, 2008.

[177] Liam C Kavanagh, Christopher L Suhler, Patricia S Churchland, and Piotr Winkielman. When it's an error to mirror the surprising reputational costs of mimicry. *Psychological science*, 2011.

[178] Yanelia Yabar, Lucy Johnston, Lynden Miles, and Victoria Peace. Implicit behavioral mimicry: Investigating the impact of group membership. *Journal of Nonverbal Behavior*, 30(3):97–113, 2006.

[179] Jeremy N Bailenson and Nick Yee. Digital chameleons automatic assimilation of nonverbal gestures in immersive virtual environments. *Psychological science*, 16(10):814–819, 2005.

[180] Laurel D Riek, Philip C Paul, and Peter Robinson. When my robot smiles at me: Enabling human-robot rapport via real-time head gesture mimicry. *Journal on Multimodal User Interfaces*, 3(1-2):99–108, 2010.

[181] Laurel D Riek and Peter Robinson. Real-time empathy: Facial mimicry on a robot. In *Workshop on Affective Interaction in Natural Environments (AFFINE) at the International ACM Conference on Multimodal Interfaces (ICMI 08). ACM*, 2008.

[182] Lindsay M Oberman, Joseph P McCleery, Vilayanur S Ramachandran, and Jaime A Pineda. Eeg evidence for mirror neuron activity during the observation of human and robot actions: Toward an analysis of the human qualities of interactive robots. *Neurocomputing*, 70(13):2194–2203, 2007.

[183] Galit Hofree, Paul Ruvolo, Marian Stewart Bartlett, and Piotr Winkielman. Bridging the mechanical and the human mind: Spontaneous mimicry of a physically present android. *PloS one*, 9(7):e99934, 2014.

[184] Meghann Drury. The effects of shared opinions on nonverbal mimicry. 2006.

[185] Jessica L Lakin and Tanya L Chartrand. Using nonconscious behavioral mimicry to create affiliation and rapport. *Psychological science*, 14(4):334–339, 2003.

[186] Katja U Likowski, Andreas Mühlberger, Beate Seibt, Paul Pauli, and Peter Weyers. Modulation of facial mimicry by attitudes. *Journal of Experimental Social Psychology*, 44(4):1065–1072, 2008.

[187] Daniel N McIntosh, Aimee Reichmann-Decker, Piotr Winkielman, and Julia L Wilbarger. When the social mirror breaks: deficits in automatic, but not voluntary, mimicry of emotional facial expressions in autism. *Developmental science*, 9(3):295–302, 2006.

[188] *Nao Documentation*.

[189] Banty Tia, Arnaud Saimpont, Christos Paizis, France Mourey, Luciano Fadiga, and Thierry Pozzo. Does observation of postural imbalance induce a postural reaction? *PloS one*, 6(3):e17799, 2011.

[190] Larissa Z Tiedens and Alison R Fragale. Power moves: complementarity in dominant and submissive nonverbal behavior. *Journal of personality and social psychology*, 84(3):558, 2003.

[191] Nicolas Guéguen, Angélique Martin, and Sébastien Meineri. Mimicry and helping behavior: an evaluation of mimicry on explicit helping request. *The Journal of social psychology*, 151(1):1–4, 2011.

[192] John A Bargh and Idit Shalev. The substitutability of physical and social warmth in daily life. *Emotion*, 12(1):154, 2012.

[193] N Pontus Leander, Tanya L Chartrand, and John A Bargh. You give me the chills embodied reactions to inappropriate amounts of behavioral mimicry. *Psychological science*, 23(7):772–779, 2012.

[194] Mariëlle Stel, Jim Blascovich, Cade McCall, Jessanne Mastop, Rick B Van Baaren, and Roos Vonk. Mimicking disliked others: Effects of a priori liking on the mimicry-liking link. *European Journal of Social Psychology*, 40(5):867–880, 2010.

[195] Chen-Bo Zhong and Geoffrey J Leonardelli. Cold and lonely does social exclusion literally feel cold? *Psychological Science*, 19(9):838–842, 2008.

[196] Jason F Jent, Larissa N Niec, and Sarah E Baker. Play and interpersonal processes. *Play in clinical practice: evidence-based approaches. Guilford Press, New York Google Scholar*, 2011.

[197] Jerome L Singer and Mawiyah A Lythcott. Fostering school achievement and creativity through sociodramatic play in the classroom. *Children?s play: The roots of reading*, 77:93, 2004.

[198] Cynthia L Ogden, Margaret D Carroll, Brian K Kit, and Katherine M Flegal. Prevalence of obesity and trends in body mass index among us children and adolescents, 1999-2010. *Jama*, 307(5):483–490, 2012.

[199] Amika S Singh, Chris Mulder, Jos WR Twisk, Willem Van Mechelen, and Mai JM Chinapaw. Tracking of childhood overweight into adulthood: a systematic review of the literature. *Obesity reviews*, 9(5):474–488, 2008.

[200] David S Freedman, William H Dietz, Sathanur R Srinivasan, and Gerald S Berenson. The relation of overweight to cardiovascular risk factors among children and adolescents: the bogalusa heart study. *Pediatrics*, 103(6):1175–1182, 1999.

[201] Donna Spruijt-Metz. Etiology, treatment, and prevention of obesity in childhood and adolescence: A decade in review. *Journal of research on Adolescence*, 21(1):129–152, 2011.

[202] Eleanor B Tate, Donna Spruijt-Metz, Gillian O'Reilly, Maryalice Jordan-Marsh, Marientina Gotsis, Mary Ann Pentz, and Genevieve F Dunton. mhealth approaches to child obesity prevention: successes, unique challenges, and next directions. *Translational behavioral medicine*, 3(4):406–415, 2013.

[203] MD Hingle, L Macias-Navarro, A Rezaimalek, and SB Going. The use of technology to promote nutrition and physical activity behavior change in youth: A review. *The Research Dietetic Practice Group Digest*, pages 1–10, 2013.

[204] Kimberly Hieftje, E Jennifer Edelman, Deepa R Camenga, and Lynn E Fiellin. Electronic media–based health interventions promoting behavior change in youth: a systematic review. *JAMA pediatrics*, 167(6):574–580, 2013.

[205] Javier Movellan, Micah Eckhardt, Marjo Virnes, and Angelica Rodriguez. Sociable robot improves toddler vocabulary skills. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 307–308. ACM, 2009.

[206] Patricia K Kuhl, Feng-Ming Tsao, and Huei-Mei Liu. Foreign-language experience in infancy: Effects of short-term exposure and social interaction on phonetic learning. *Proceedings of the National Academy of Sciences*, 100(15):9096–9101, 2003.

[207] Fumihide Tanaka and Shizuko Matsuzoe. Children teach a care-receiving robot to promote their learning: Field experiments in a classroom for vocabulary learning. *Journal of Human-Robot Interaction*, 1(1):78–95, 2012.

[208] Sangseok Yun, Jongju Shin, Daijin Kim, Chang Gu Kim, Munsang Kim, and Mun-Taek Choi. Engkey: tele-education robot. In *International Conference on Social Robotics*, pages 142–152. Springer, 2011.

[209] Iolanda Leite, Rui Henriques, Carlos Martinho, and Ana Paiva. Sensors in the wild: Exploring electrodermal activity in child-robot interaction. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 41–48. IEEE Press, 2013.

[210] Cory D Kidd and Cynthia Breazeal. Robots at home: Understanding long-term human-robot interaction. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 3230–3235. IEEE, 2008.

[211] Adam Adam Michael Setapen. *Creating robotic characters for long-term interaction*. PhD thesis, Massachusetts Institute of Technology, 2012.

[212] Norma González, Luis C Moll, and Cathy Amanti. *Funds of knowledge: Theorizing practices in households, communities, and classrooms*. Routledge, 2006.

[213] Jeanne Ellis Ormrod. Educational psychology, developing learners. 4th. *Upper Saddle River, NJ: Merrill Prentice Hall*, 2003.

[214] Carolina L Bottari, Clément Dassa, Constant M Rainville, and Élisabeth Dutil. The iadl profile: Development, content validity, intra-and interrater agreement. *Canadian Journal of Occupational Therapy*, 77(2):90–100, 2010.

[215] Mary K Rothbart, Stephan A Ahadi, Karen L Hershey, and Phillip Fisher. Investigations of temperament at three to seven years: The children's behavior questionnaire. *Child development*, 72(5):1394–1408, 2001.

[216] Mark Wilson. *Constructing measures: An item response modeling approach*. Routledge, 2004.

[217] Matthew Lombard, Theresa B Ditton, Daliza Crane, Bill Davis, Gisela Gil-Egui, Karl Horvath, Jessica Rossman, and S Park. Measuring presence: A literature-based approach to the development of a standardized paper-and-pencil instrument. In *Third international workshop on presence, delft, the netherlands*, volume 240, pages 2–4, 2000.

[218] Cory D Kidd and Cynthia Breazeal. Effect of a robot on user perceptions. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3559–3564. IEEE, 2004.

[219] Younbo Jung et al. Effects of physical embodiment on social presence of social robots. *Proceedings of PRESENCE*, pages 80–87, 2004.

[220] Robert H Poresky, Charles Hendrix, Jacob E Mosier, and Marvin L Samuelson. The companion animal bonding scale: Internal reliability and construct validity. *Psychological Reports*, 60(3):743–746, 1987.

[221] Amy C Thomason and Karen M La Paro. Measuring the quality of teacher–child interactions in toddler child care. *Early Education and Development*, 20(2):285–304, 2009.

[222] Elazar J Pedhazur and Liora Pedhazur Schmelkin. *Measurement, design, and analysis: An integrated approach*. Psychology Press, 2013.

[223] Barry Hayes, Elena Corina Grigore, Alexandru Litoiu, Aditi Ramachandran, and Brian Scassellati. A developmentally inspired transfer learning approach for predicting skill durations. In *Development and Learning and Epigenetic Robotics (ICDL-Epirob), 2014 Joint IEEE International Conferences on*, pages 181–186. IEEE, 2014.

[224] Annette ME Henderson and Amanda L Woodward. "Let's work together": What do infants understand about collaborative goals? *Cognition*, 121(1):12–21, 2011.

[225] J Kiley Hamlin, Karen Wynn, and Paul Bloom. Social evaluation by preverbal infants. *Nature*, 450(7169):557–9, November 2007.

[226] Alexander Todorov, Anesu N Mandisodza, Amir Goren, and Crystal C Hall. Inferences of competence from faces predict election outcomes. *Science*, 308(5728):1623–1626, 2005.

[227] Ruth Butler. The effects of mastery and competitive conditions on self-assessment at different ages. *Child Development*, 61(1):201–210, 1990.

[228] Albert Bandura. Self-efficacy: toward a unifying theory of behavioral change. *Psychological review*, 84(2):191, 1977.

[229] William L Mihal and Janet L Graumenz. An assessment of the accuracy of self-assessment for career decision making. *Journal of Vocational Behavior*, 25(2):245–253, 1984.

[230] William K Estes. The cognitive side of probability learning. *Psychological Review*, 83(1):37, 1976.

[231] James C Diggory and JC Diggory. *Self-evaluation: Concepts and studies*. Wiley New York, 1966.

[232] Carol S Dweck. The development of ability conceptions. In Allan Wigfield, Jacquelynne S Eccles, Ulrich Schiefele, Robert W Roeser, and Pamela Davis-Kean, editors, *Development of achievement motivation*. Wiley Online Library, 2007.

[233] Gail D Heyman, Carol S Dweck, and Kathleen M Cain. Young children's vulnerability to self-blame and helplessness: Relationship to beliefs about goodness. *Child development*, 63(2):401–415, 1992.

[234] Deborah J Stipek and Denise H Daniels. Children's use of dispositional attributions in predicting the performance and behavior of classmates. *Journal of Applied Developmental Psychology*, 11(1):13–28, 1990.

[235] Deborah J Stipek. Young children's performance expectations: Logical analysis or wishful thinking. *Advances in motivation and achievement*, 3:33–56, 1984.

[236] Yohanan Eshel and Zev Klein. Development of academic self-concept of lower-class and middle-class primary school children. *Journal of Educational Psychology*, 73(2):287, 1981.

[237] Joyce F Benenson and Carol S Dweck. The development of trait explanations and self-evaluations in the academic and social domains. *Child Development*, pages 1179–1187, 1986.

[238] William N Morris and Donald Nemcek Jr. The development of social comparison motivation among preschoolers: Evidence of a stepwise progression. *Merrill-Palmer Quarterly (1982)*, pages 413–425, 1982.

[239] Murray S Edelman and Donald R Omark. Dominance hierarchies in young children. *Social Science Information/sur les sciences sociales*, 1973.

[240] Allan Wigfield, Jacquelynne S Eccles, Kwang Suk Yoon, Rena D Harold, Amy JA Arbreton, Carol Freedman-Doan, and Phyllis C Blumenfeld. Change in children's competence beliefs and subjective task values across the elementary school years: A 3-year study. *Journal of Educational Psychology*, 89(3):451, 1997.

[241] John G Nicholls. The development of the concepts of effort and ability, perception of academic attainment, and the understanding that difficult tasks require more ability. *Child development*, pages 800–814, 1978.

[242] John G Nicholls and Arden T Miller. Reasoning about the ability of self and others: A developmental study. *Child development*, pages 1990–1999, 1984.

[243] William S Rholes, Melinda Jones, and Cindi Wade. Children's understanding of personal dispositions and its relationship to behavior. *Journal of Experimental Child Psychology*, 45(1):1–17, 1988.

[244] Kathleen M Cain and Carol S Dweck. The development of children's conceptions of intelligence: A theoretical framework. *Advances in the psychology of human intelligence*, 5:47–82, 1989.

[245] Phyllis C Blumenfeld, Paul R Pintrich, and V Lee Hamilton. Children's concepts of ability, effort, and conduct. *American Educational Research Journal*, 23(1):95–104, 1986.

[246] John G Nicholls, Michael Patashnick, and Gwendolyn Mettetal. Conceptions of ability and intelligence. *Child Development*, pages 636–645, 1986.

[247] Robert S Woodworth and EL Thorndike. The influence of improvement in one mental function upon the efficiency of other functions. (i). *Psychological review*, 8(3):247, 1901.

[248] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *The Journal of Machine Learning Research*, 10:1633–1685, 2009.

[249] John E Laird, Paul S Rosenbloom, and Allen Newell. Chunking in soar: The anatomy of a general learning mechanism. *Machine learning*, 1(1):11–46, 1986.

[250] Dongkyu Choi, Tolgo Konik, Negin Nejati, Chunki Park, and Pat Langley. Structural transfer of cognitive skills. In *Proceedings of the eighth international conference on cognitive modeling*, pages 115–120, 2007.

[251] Jonathan Baxter. A bayesian/information theoretic model of learning to learn via multiple task sampling. *Machine Learning*, 28(1):7–39, 1997.

[252] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? *Advances in neural information processing systems*, pages 640–646, 1996.

[253] Okhtay Ilghami, Hector Munoz-Avila, Dana S Nau, and David W Aha. Learning approximate preconditions for methods in hierarchical plans. In *Proceedings of the 22nd international conference on Machine learning*, pages 337–344. ACM, 2005.

[254] Alan Fern, Sungwook Yoon, and Robert Givan. Approximate policy iteration with a policy language bias. *Advances in neural information processing systems*, 16(3):847–854, 2004.

[255] Rajat Raina, Andrew Y Ng, and Daphne Koller. Constructing informative priors using transfer learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 713–720. ACM, 2006.

[256] George D Konidaris. A framework for transfer in reinforcement learning. In *ICML-06 Workshop on Structural Knowledge Transfer for Machine Learning*, 2006.

[257] Bruno Da Silva, George Konidaris, and Andrew Barto. Learning parameterized skills. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[258] George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3):360–375, 2012.

[259] Andrew G Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning (ICDL 2004)*, pages 112–19, 2004.

[260] Allen Newell and Paul S Rosenbloom. Mechanisms of skill acquisition and the law of practice. *Cognitive skills and their acquisition*, 1, 1981.

[261] Willard I Zangwill and Paul B Kantor. Toward a theory of continuous improvement and the learning curve. *Management Science*, 44(7):910–920, 1998.

[262] Timothy L Smunt and Charles A Watts. Improving operations planning with learning curves: overcoming the pitfalls of 'messy'shop floor data. *Journal of Operations Management*, 21(1):93–107, 2003.

[263] François Trudeau and Roy J Shephard. Physical education, school physical activity, school sports and academic performance. *Int'l Journal of Behavioral Nutrition and Physical Activity*, 5(1):10, 2008.

[264] 2008 physical activity guidelines for americans. Technical report, U. S. Department of Health and Human Services, 2008.

[265] Lifetick. Lifetick - goal setting the way it should be, 2014. Available at `https://www.lifetick.com/`.

[266] HealthMonth. Healthmonth - live healthier, for fun!, 2014. Available at `https://www.healthmonth.com/`.

[267] Debjanee Barua, Judy Kay, Bob Kummerfeld, and Cécile Paris. Modelling long term goals. In *User Modeling, Adaptation, and Personalization*, pages 1–12. Springer, 2014.

[268] Sunny Consolvo, David W McDonald, Tammy Toscos, Mike Y Chen, Jon Froehlich, Beverly Harrison, Predrag Klasnja, Anthony LaMarca, Louis LeGrand, Ryan Libby, et al. Activity sensing in the wild: a field trial of ubifit garden. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1797–1806. ACM, 2008.

[269] Predrag Klasnja, Sunny Consolvo, and Wanda Pratt. How to evaluate technologies for health behavior change in hci research. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3063–3072. ACM, 2011.

[270] Juan Fasola and Maja J Matarić. Socially assistive robot exercise coach: Motivating older adults to engage in physical exercise. In *Experimental Robotics*, pages 463–479. Springer, 2013.

[271] Cory David Kidd. Designing for long-term human-robot interaction and application to weight loss. 2008.

[272] Mike Uschold and Michael Gruninger. Ontologies: Principles, methods and applications. *The knowledge engineering review*, 11(02):93–136, 1996.

[273] Timothy W Bickmore, Daniel Schulman, and Candace L Sidner. A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology. *Journal of biomedical informatics*, 44(2):183–197, 2011.

[274] Leslie Lenert, Gregory J Norman, Mark Mailhot, and Kevin Patrick. A framework for modeling health behavior protocols and their linkage to behavioral theory. *Journal of biomedical informatics*, 38(4):270–280, 2005.

[275] Edwin A Locke and Gary P Latham. Building a practically useful theory of goal setting and task motivation: A 35-year odyssey. *American psychologist*, 57(9):705, 2002.

[276] Albert Bandura. Social cognitive theory of self-regulation. *Organizational behavior and human decision processes*, 50(2):248–287, 1991.

[277] Icek Ajzen. The theory of planned behavior. *Organizational behavior and human decision processes*, 50(2):179–211, 1991.

[278] John G Nicholls. Achievement motivation: Conceptions of ability, subjective experience, task choice, and performance. *Psychological review*, 91(3):328, 1984.