# Online Development of Assistive Robot Behaviors for Collaborative Manipulation and Human-Robot Teamwork

**Bradley Hayes and Brian Scassellati**
Yale University
Computer Science Department
New Haven, CT 06511

## Abstract

Collaborative robots that operate in the same immediate environment as human workers have the potential to improve their co-workers' efficiency and quality of work. In this paper we present a taxonomy of assistive behavior types alongside methods that enable a robot to learn assistive behaviors from interactions with a human collaborator during live activity completion. We begin with a brief survey of the state of the art in human-robot collaboration. We proceed to focus on the challenges and issues surrounding the online development of assistive robot behaviors. Finally, we describe approaches for learning when and how to apply these behaviors, as well as for integrating them into a full end-to-end system utilizing techniques derived from the learning from demonstration, policy iteration, and task network communities.

## Introduction

Human-robot teaming has the potential to extend robotics well beyond their current, limited roles in factory automation. Much of modern robotics remains inapplicable in many domains where tasks are either too complex, beyond modern hardware limitations, too sensitive for non-human completion, or too flexible for static automation practices. In these situations human-robot teaming can be leveraged to improve the efficiency, quality-of-life, and safety of human workers. As a community, we desire to create collaborative robots that can provide assistance when useful, remove dull or undesirable responsibilities when possible, and assist with dangerous tasks when feasible. In particular, this paper focuses on collaboration between a lead worker and robotic assistant, complementing prior work that develops collaborative robots as peers (Gombolay et al. 2013; Knepper et al. 2013; Nikolaidis and Shah 2013).

We specifically look into the process of building a system capable of producing an effective robot assistant that learns from demonstration. To be effective, this assistant must be capable of learning to anticipate the parts or tool-related needs of the lead worker, while maintaining the flexibility to adapt to different worker preferences while maintaining value from prior training. In constructing such a system, we address challenges in state estimation, policy optimization, influencing human task execution, and implicit social communication. Through this system, we aim to cover various types of supported assistive behaviors, ranging from simple stabilization operations to the joint manipulation of unwieldy objects. The work presented within represents an ongoing research effort and is organized as a survey of challenges and proposed solutions within this shared workspace human-robot teaming domain. Evaluations of the system presented within this work are designed to be carried out on the Collaborative Workbench (figure 1), an instrumented workbench with two mounted Kuka YouBot lightweight manufacturing arms and tablet-based controller interface.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

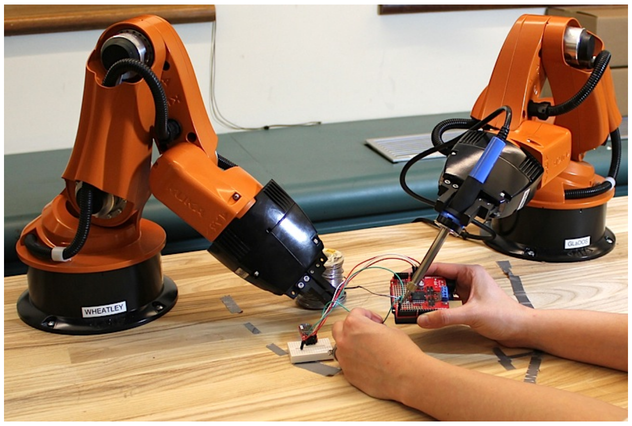

Figure 1: The Collaborative Workbench platform, designed for shared workspace human-robot teaming exercises.

## Background

Our work spans many active fields of robotics research, including Learning from Demonstration, Markov Decision Processes, and Reinforcement Learning. In this section, we define common terms and highlight relevant related work from the community.

### Learning from Demonstration

Learning from Demonstration (LfD) is a natural method for allowing novice operators to impart knowledge into a robot system (Chen and Zelinsky 2003). LfD is particularly useful due to its vast applicability, as demonstrations can be effectively provided via teleoperation (Ng et al. 2006; Sweeney and Grupen 2007), kinesthetic teaching (Akgun et al. 2012), or guided by human speech (Rybski et al. 2007) to achieve task or skill proficiency. LfD research has also been conducted to enable robots to learn from demonstrations performed by other robots (Konidaris et al. 2011), applying observations of other robotic agents to improve one's own performance.

A key benefit of LfD is that it enables novice users to easily program robot behaviors. Some modern systems even incorporate mechanisms allowing a user to shape and refine a robot's policy that had been generated from prior task demonstrations (Dillmann et al. 2000; Ogawara et al. 2002).

Especially relevant for inexperienced skill trainers, many LfD techniques are robust to the inconsistencies and suboptimality usually encountered from a human trainer. By attempting to learn from the perceived intention of a demonstration rather than its actual action sequence (Friedrich and Dillmann 1995), a learner is capable of extracting helpful knowledge from imperfect training data (Atkeson and Schaal 1997). Recent work has also provided novel means of segmenting observed action sequences to autonomously extract portable, goal-directed skills (Konidaris et al. 2010; Pastor et al. 2009).

### Markov Decision Processes

Markov Decision Processes (MDPs) provide a convenient and efficient representation with which to create flexible, arbitrarily complex options: closed-loop policies describing action sequences (Stolle and Precup 2002; Sutton et al. 1999). The temporal abstraction and generality of representation make MDPs a popular method of internally representing knowledge about actionable skills and tasks.

MDPs are represented by the 4-tuple $(S, A, R, P)$, which defines an environment-centric directed multigraph, with environmental knowledge encoded in vertices representing possible states $(s \in S)$. Directed edges between vertices (transitions) are labeled with an action $(a \in A)$. $R(s'|s, a)$ is the reward achieved by transitioning to state $s'$ from state $s$ by way of action $a$. $P(s'|s, a)$ is a transition probability function that indicates the probability of arriving in state $s'$ when executing action $a$ in state $s$.

In the MDP framework, agents acquire a policy $\pi$ that informs action selection, producing a state-action mapping function. In Semi-Markov Decision Processes (SMDPs) (Sutton et al. 1999), these actions can encapsulate arbitrary levels of complexity, typically in the form of a temporally abstracted, parameterized motor primitive. The generality and extensibility of this approach contributes to policy-based learning being a widely used method of skill representation (Kaelbling, Littman, and Moore 1996). With an environmental reward function, solving for an optimal action policy can often be accomplished autonomously and is only limited by the complexity of the problem's state representation.

### Partially Observable Markov Decision Processes

Partially Observable Markov Decision Processes (POMDPs) (Monahan 1982) are popular representations of MDPs that can be applied to real-world situations even when situational awareness is not perfect. In contrast to the 4-tuple $(S, A, R, P)$ describing an MDP, POMDPs are represented by the 6-tuple $(S, A, R, P, O, \Omega)$. $O$ represents a set of observations, members of which can be output from states in $S$. $\Omega$ describes the conditional probability of an observation given a state and action. Importantly, the precise state of the system within a POMDP is not known and must be reasoned about probabilistically given a series of observations.

In the context of an assistive robot collaborator, the problem of determining the intention of a human co-worker can be modeled as a POMDP built from a previously known task network. We utilize a POMDP since our assistive robot is able to exert a level of control over state transitions in addition to possessing the ability to test hypotheses regarding the current state of the task.

## Application Domain: Assistive Behaviors

In this work we predominantly consider scenarios in which a human is performing a construction or assembly task while sharing a workspace with an assistive robot. Our contribution centers on the development of a means of learning different types of assistive behaviors and when to apply them. To maximize relevance to the broader robotics community, we apply these behaviors within the context of an existing, popular task representation, namely MDPs.

We classify assistive behaviors as generally belonging to one of five identified high-level categories. While this is by no means an exhaustive list of assistive action types, the close proximity, shared workspace environment we utilize makes these classifications particularly relevant. The remainder of the paper focuses on robot actions in the areas of: materials stabilization, materials retrieval, collaborative object manipulation, enhancing awareness, and task progression guidance.

**Materials Stabilization** This behavior class involves having the robot hold an object maintaining a particular pose. The inherent goal of such an action is to position and orient an object such that the primary worker can more easily perform the target task. Examples of this include holding fabrics together to allow a human to sew them more easily, orienting a wire and PCB in such a way that they may be soldered together, or maintaining the position of a piece of furniture so a component may be pressed into place.

**Materials Retrieval** The retrieval class of behaviors involves changing the available workspace resources such that a worker needs to exert less effort or cognitive load to find necessary parts or tools. Actions in this class can range from non-intrusive methods such as pointing, to minimally intrusive actions such as placing objects conveniently nearby, to directly interactive handover actions.

**Collaborative Object Manipulation** For particularly unwieldy materials or in instances where precision is especially important, a robot assistant may render help in the form of moving objects in tandem with another worker. Two particularly salient use cases for these behaviors are the placement and orientation of heavy or large objects and the guidance of tools to effect other materials. In the case of the latter, one can imagine situations that may arise in complex tasks where motion constraints on particular tools will greatly reduce the difficulty of a subtask. A concrete example of this may be an assistive robot enforcing a planar constraint while a human manipulates a jointly held sanding tool or enforcing a depth constraint when etching a fragile surface with a dremel tool.

**Enhancing Awareness** There are behaviors that do not directly manipulate the environment, yet may also be of exceptional assistance to a worker. One example of an awareness-enhancing behavior would be a robot manipulating a camera to provide an alternative, more helpful perspective to a worker. Other, less complicated actions in this category may include using a flashlight to illuminate areas inside a part being worked on, or directing a laser pointer to indicate relevant through-hole paths on a PCB that is being wired.

**Task Progression Guidance** An assistive robot may also perform actions that inherently guide the progression of a task towards a particular action sequence. Unlike the other four classifications of assistive behavior, influencing the ordering of subtask completion involves reducing the cost (in terms of time or effort) of a collaborator pursuing certain actions over others. One minimal example where this behavior can be applied is a drilling task, where it may be more efficient for a worker to place all of the screws in pilot holes before drilling them all, rather than placing and drilling them one at a time. An assistive robot can manipulate the course of action by performing materials retrieval actions giving the worker screws rather than the drill, or even by moving the drill further from the active work area until the screws are placed.

## Approach

In this section, we describe techniques for integrating assistive actions with complex tasks. We begin by outlining a method for associating assistive behaviors with task completion status and active goals of the lead worker. Once an association can be made between these assistive behaviors and the task at hand, we describe a means of learning assistive behaviors from demonstration. We proceed to describe an approach allowing for the refinement of these behaviors, while allowing for the potential of personalization on the level of individual workers. Finally, we briefly discuss implementation strategies for the aforementioned action classes.

### Merging Task State with Assistive Behaviors

Mutually compatible, shared task comprehension is a critical element to the success of any human-robot team (Nikolaidis and Shah 2013). In our work, we represent tasks as directed graphs, encoding both environment states and the actions that transition the world between these states. Differing from the standard SMDP formulation where vertices represent environment states and edges are labeled with action

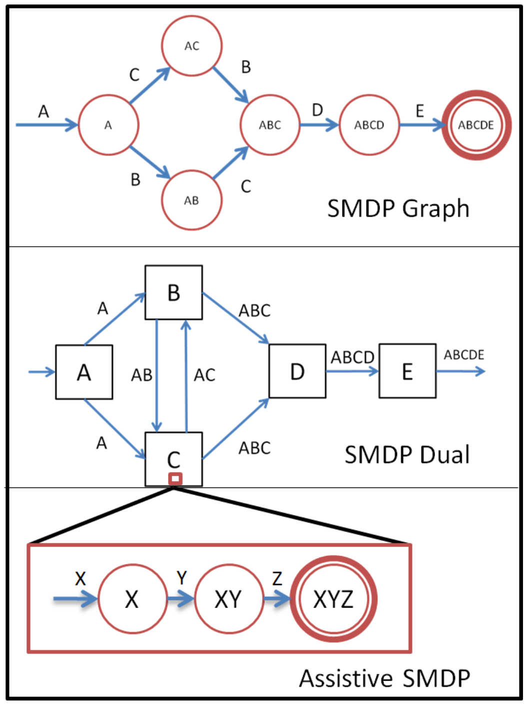

Figure 2: SMDP (top) and its corresponding dual (middle) with associated assistive behavior SMDP (bottom). In the SMDP graph, environment states are encoded in vertices and edges are labeled with their corresponding actions. In the dual graph, skills are represented as vertices and edges are labeled with environmental constraints that prevent agents from accessing its destination vertex. In this example, the assistive behavior SMDP can be acted upon by a partner to faciliate action C.

choices, we build our task network from its dual. By representing agent actions as vertices and labeling edges with the required prerequisite environment state feature vector(s) for transition, an action-centric task representation is achieved. From the perspective of a robot collaborator, this provides a model of the activities and goals that a human principal worker is desiring to accomplish.

When trying to model the underlying structure in complex task networks, traditional MDP graphs' environment-centric representations do not immediately convey features that facilitate discerning ordering constraints or isolating action-oriented goals. To mitigate these issues, we augment the environment-space SMDP graph via a transformation function to produce a constraint network on action orderings from its dual, resulting in a graph constructed around known skill primitives rather than environment states. We refer to this graph as the SMDP-dual (figure 2-middle), where vertices represent actions and eligible transitions are determined by prerequisite environmental states.

By converting the task graph into an action-centric representation, a robot teammate now has a behavioral model of its collaborator within which it can associate assistive behaviors. The active state of the task can then be inferred by existing techniques (Hawkins et al. 2013) based on state-related observations relating to part positioning, tool use, and workspace utilization. If each vertex of this graph is intuitively considered to represent a subtask goal for the human (e.g., "drill holes in part 3"), an assistive robot can use this context to decide how best to render help. In this paper, we describe a process for learning, applying, and refining these assistive behaviors in the context of such a task graph. This is accomplished by learning a distinct SMDP for assistive actions that can be performed at a given stage in the target task, which can then parameterize dynamic motor primitives (Schaal 2006) belonging to the classes of action described previously.

## Learning Assistive Behaviors

We consider two main challenges in the context of learning assistive behaviors: how does a robot acquire these skills, and how can a robot account for individual user preferences in execution style or timing? We address the first challenge with a skill training algorithm based on the state of the art in Learning from Demonstration. To handle issues related to the second challenge, we incorporate a reinforcement learning signal into the behavior selection mechanism.

**Skill Acquisition** Learning initial models of the desired behaviors can be incredibly challenging without explicit instruction. To quickly achieve a moderately successful policy of assistive behaviors, we leverage the human's presence to provide explicit training to the robot for each desired assistive skill. For our application, this means using kinesthetic teaching (Akgun et al. 2012) to learn a corresponding SMDP or parameterization for each assistive behavior, mapping it to the current task state in the POMDP (figure 2-bottom) describing the task being completed by the lead worker.

During the skill training phase, feature selection becomes an important consideration for action reproduction. At the time of training, it may be unclear which objects are relevant to the assistive behavior being taught. Further, once one is confident of the relevant scene entities, it may not be straightforward which features relevant to that entity should be recorded: distance to an object may be important in some situations, whereas the relative orientation of the robot's gripper to the entity may be important in others. To solve this ambiguity, we provide the trainer a mechanism by which they may select the relevant features of the skill manually from a curated list of possibilities. In future work, we are interested in leveraging feature-specific statistical regularities across training demonstrations to progress this feature selection process without user intervention.

**Skill Refinement** Once an agent has initial models of the skills to be executed, users will invariably find the need to refine them to adapt to changes in work conditions or changing personal preferences. In our work, we treat this as a supervised learning problem in the immediate term, and a reinforcement learning problem in the longer term.

In the short term, an agent that sub-optimally performs an assistive behavior requires immediate correction. In the absence of a predetermined, real-time computable objective function, these adjustments cannot be made in a convenient or effective manner. As such, we again leverage the presence of a human trainer to provide corrective feedback, allowing the user to kinesthetically correct the position or redemonstrate the desired trajectory of the robot, providing a distinct training example as compared to what may have been initially trained.

More specifically, if a user engages in a corrective behavior for a given action, we modify the behavior's underlying SMDP by adding new training data. At each step through the corrected demonstration, new states, actions, and transitions are added to the SMDP. To accommodate separate sources of reward, we modify the reward function $R(s'|s,a)$ such that instead of returning a scalar value, it returns a set of (label, scalar) tuples representing 'layers' of reward. This allows for flexibility in considering which sources of reward to integrate into a decision or policy. For example, in the refinement scenario each state-action transition is given an explicit 'trainer' reward to only be associated with the worker that demonstrated it. Unless otherwise stated, we consider the reward of a state transition $R(s'|s,a) = \sum_{label} R_{label}(s'|s,a)$.

Over longer timespans, we are able to use objective task execution-level data to measure the effectiveness of assistive behaviors. This is accomplished by measuring subtask completion durations as well as overall task completion durations. As we describe in the next section, once an assistive behavior is known and it is associated with a subtask in the lead worker's POMDP, the system is faced with the challenge of selecting the best or most relevant method of executing it from its training samples.

## Performing Assistive Behaviors

Perhaps one of the most difficult challenges facing an assistive robot is determining when and how best to perform assistive behaviors. This requires a knowledge of the lead worker's intentions, goals, and preferences. At any given task state, a robot assistant must be capable of either influencing or detecting the next desired task state. This allows for the robot to facilitate the transition between the two, which typically involves modifying the environment in such a way as to make it easier for the lead worker to execute some manner of motor primitive.

**Choosing Actions** In our assistive behavior framework, we utilize a set of user-taught, parameterized, assistive behavior motor primitives that are associated with state-action transitions in the overall task POMDP. As previously mentioned, we not only learn these individual motor primitives, but also orderings for their execution at any task state through a separate "assistive behavior" SMDP (see figure 2-bottom) we refer to as $M$. Thus, $M = (M_S, M_A, M_R, M_P)$, where $M_S$ represents the states of $M$, $M_A$ represents the action set of $M$, $M_R$ the transition reward function of $M$, and $M_P$ the transition probabilites in $M$. The set of actions $M_A$ is comprised entirely of the aforementioned user-taught skills. We use previously gathered task completion timespan information to inform the reward function $M_R$, which is in turn used to select the best possible policy through $M$ via a standard policy iteration method.

**Policy Optimization** Throughout task execution, we are concerned with optimizing two tightly related policies. At a high level, we wish to optimize the lead worker's POMDP traversal policy, while at a lower level we wish to optimize the assistive behavior policy called at each task step. For each individual subtask $s$ in the POMDP-dual, we seek to find a policy $\pi$ for the set of associated assistive motor primitives in the SMDP $M$ that minimizes time spent in $s$ (referenced hereafter as $M_\pi$).

After each successful subtask execution, the sequence of assistive primitives has its transition rewards adjusted inversely proportional to the amount of time spent in $s$. More formally, for each transition between assistive primitives covered during execution the reward function is updated as:

$$R_{\text{duration}}(s'|s,a) = \alpha R_{\text{duration}}(s'|s,a) + (1-\alpha)(-d(s))$$

where $d(s)$ represents the duration spent in $s$ and $\alpha \in [0,1]$ is a learning factor.

We seek to simultaneously optimize over observed task-level policies ($T_\pi \in T_\Pi$) and assistive behavior policies ($M_\pi \in M_\Pi$) given a task POMDP $T = (T_S, T_A, T_R, T_P, T_O, T_\Omega)$. To do this, we choose a policy for each state-associated assistive SMDP, creating an overall assistive policy $A_\pi = \{M_\pi^1, M_\pi^2, ..., M_\pi^{|T_S|}\}$ that dictates how best to help for each possible state in $T_S$. Given a transition duration estimator $d(s', s|s, s' \in T_S, A_\pi)$ that returns the expected transition time from $s \rightarrow s'$ in $T$ under assistive behavior policy $A_\Pi$, we choose a policy set $A_\pi$ that maximizes:

$$\sum_{T_\pi \in T_\Pi} \sum_{s \in T_S} \sum_{s' \in T_S} T_P(s', s|A_\pi, T_\pi) * P(T_\pi) * -d(s', s|A_\Pi)$$

This function optimizes for choosing the best assistive action sequences given the system's experience with prior observations of the task being completed. It accomplishes this by minimizing the time spent in states likely to be reached, given knowledge of past task completions ($T_\pi \in T_\Pi$) weighted by the likelihood of the lead worker choosing that policy ($P(T_\pi)$).

**Personalization of Behaviors** Within each action of $M_A$, the acting agent must select an instantiation to perform. For an action $a$, this is done by choosing a policy for its SMDP with the best expected outcome as measured by its reward function conditioned upon the current lead worker. In other words, instead of choosing a policy determined by the values from $R(s'|s,a)$, we instead look at $R(s'|s,a,\text{leader\_id})$.

User personalization should be considered at the individual demonstration level of granularity in addition to longer-term objective measures. When determining the rewards to associate with selecting each potential instantiation (policy) of a trained skill, an artificial incentive (reward layer) is added to those behaviors trained by the current lead worker. This is based on a loose optimality assumption that behaviors trained by an individual are those considered most helpful for that individual. In spite of this, the strength of this personalization incentive should be tempered by multiplying it with a decay function as the number of training examples for a given skill increases. We incorporate this to trend towards more objectively optimal solutions (which may originate from any worker/set of workers), discounting the value of an individual's personal solutions as confidence in the overall proficiency of the skill increases. In our ongoing work, we intend to test user response to variants of this action parameterization bias.

**Rejected Actions** It is also possible for the lead worker to outright reject a particular assistive behavior due to personal preference. Some examples of this include ignoring the assistant when it attempts to perform a tool retrieval handover, taking away a material that the assistant was stabilizing, pushing the assistant's arm away when it attempts to perform collaborative object manipulations, or dismissing an assistant positioning a flashlight to offer greater workpiece visibility. Just as it is possible to use subtask and task duration to evaluate the impact of particular assistive actions on the overall task progress, similar logic can be applied to determine if a behavior is being rejected.

By associating expected durations with transitions within an individual motor primitive's SMDP, a measure of whether reasonable progress is occurring can be made. If the progression time between states within an assistive motor primitive exceeds the bounds for reasonable progress, the system can assert with increasing confidence that its behavior is not desired. Once recognized, the rejection of this skill $a \in M_A$ may be reflected in $M$ by adding a rejection reward layer to the transitions that include executing $a$ from the current environment state.

**Executing Behaviors** Despite kinesthetically learning an SMDP for each assistive motor primitive, most assistive behaviors we mention are likely best represented as dynamic motor primitives (Schaal 2006), merely informed by the training data in order to determine a valid parameterization. In the case of materials stabilization actions, the training data is used to extract meaningful object pose and relative position requirements. A linear program can then be used to find a kinematic solution for the robot such that these learned feature constraints are honored. In the case of materials retrieval, the salient parameters to be learned are object selection (e.g., a glass of water), grasping orientation (e.g., pick up from the side), and object orientation constraints in transit (e.g., a glass of water should be upright). A standard motion planner (Sucan, Moll, and Kavraki 2012; Zucker et al. 2013) can then be employed to solve for a valid trajectory subject to the learned constraints.

Collaborative manipulation behaviors are the most complex type of assistive action to execute, as they require constant adjustment due to dynamic input from an uncontrolled entity. We view this as a special case of materials stabilization that changes over time. Training data can be used to extract axial or planar constraints by attempting to fit the demonstrated object trajectories to a line or plane. If such constraints are found, they are simply added to the linear program that would be used for stabilization tasks. Modern solvers allow for real-time solutions both to this program and its solutions' corresponding inverse kinematic poses giving the responsiveness required to react to the perturbations caused by the manipulation partner.

## Conclusion

We have presented an overview of ongoing work concerned with building a system to enable an assistive robot collaborator. We discuss various classifications of assistive actions for shared workspace interaction, as well as approaches to learning assistive skills from demonstration, accepting refinement through experience in live task execution, and optimizing these behaviors over time. In future work, we anticipate completing human-robot interaction studies evaluating individual components of the proposed system to better understand the expectations humans have of an assistive robot collaborator.

## Acknowledgments

This work is supported by Office of Naval Research grant #N00014-12-1-0822 and NSF grants 1139078 and 1117801.

## References

Akgun, B.; Cakmak, M.; Jiang, K.; and Thomaz, A. L. 2012. Keyframe-based learning from demonstration. *International Journal of Social Robotics*.

Atkeson, C., and Schaal, S. 1997. Robot learning from demonstration. In *International Conference on Machine Learning*, 11–73.

Chen, J., and Zelinsky, A. 2003. Programing by demonstration: Coping with suboptimal teaching actions. *The International Journal of Robotics Research* 22(5):299–319.

Dillmann, R.; Rogalla, O.; Ehrenmann, M.; Zollner, R.; and Bordegoni, M. 2000. Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm. In *Robotics Research International Symposium*, volume 9, 229–238.

Friedrich, H., and Dillmann, R. 1995. Robot programming based on a single demonstration and user intentions. In *3rd European workshop on learning robots at ECML*, volume 95. Citeseer.

Gombolay, M. C.; Wilcox, R. J.; Diaz, A.; Yu, F.; and Shah, J. A. 2013. Towards successful coordination of human and robotic work using automated scheduling tools: An initial pilot study. In *Proc. Robotics: Science and Systems (RSS) Human-Robot Collaboration Workshop (HRC)*.

Hawkins, K. P.; Bansal, S.; Vo, N.; and Bobick, A. F. 2013. Modeling structured activity to support human-robot collaboration in the presence of task and sensor uncertainty. In *Intelligent Robots and Systems (IROS), Workshop on Cognitive Robotics Systems*.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4.

Knepper, R. A.; Layton, T.; Romanishin, J.; and Rus, D. 2013. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 855–862. IEEE.

Konidaris, G.; Kuindersma, S.; Barto, A.; and Grupen, R. 2010. Constructing skill trees for reinforcement learning agents from demonstration trajectories. *Advances in neural information processing systems* 23:1162–1170.

Konidaris, G.; Kuindersma, S.; Grupen, R. A.; and Barto, A. G. 2011. Autonomous skill acquisition on a mobile manipulator. In *Twenty-Fifth Conference on Artificial Intelligence*, 1468–1473.

Monahan, G. E. 1982. State of the arta survey of partially observable markov decision processes: Theory, models, and algorithms. *Management Science* 28(1):1–16.

Ng, A. Y.; Coates, A.; Diel, M.; Ganapathi, V.; Schulte, J.; Tse, B.; Berger, E.; and Liang, E. 2006. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*. Springer. 363–372.

Nikolaidis, S., and Shah, J. 2013. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, 33–40. IEEE Press.

Ogawara, K.; Takamatsu, J.; Kimura, H.; and Ikeuchi, K. 2002. Generation of a task model by integrating multiple observations of human demonstrations. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, 1545–1550. IEEE.

Pastor, P.; Hoffmann, H.; Asfour, T.; and Schaal, S. 2009. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 763–768. IEEE.

Rybski, P.; Yoon, K.; Stolarz, J.; and Veloso, M. 2007. Interactive robot task training through dialog and demonstration. In *Human-Robot Interaction (HRI) 2007*, 49–56. IEEE.

Schaal, S. 2006. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*. Springer. 261–280.

Stolle, M., and Precup, D. 2002. Learning options in reinforcement learning. In *Abstraction, Reformulation, and Approximation*. Springer. 212–223.

Sucan, I. A.; Moll, M.; and Kavraki, E. 2012. The open motion planning library. *Robotics & Automation Magazine, IEEE* 19(4):72–82.

Sutton, R.; Precup, D.; Singh, S.; et al. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1):181–211.

Sweeney, J., and Grupen, R. 2007. A model of shared grasp affordances from demonstration. In *Humanoid Robots, 2007 7th IEEE-RAS International Conference on*, 27–35. IEEE.

Zucker, M.; Ratliff, N.; Dragan, A. D.; Pivtoraiko, M.; Klingensmith, M.; Dellin, C. M.; Bagnell, J. A.; and Srinivasa, S. S. 2013. Chomp: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research* 32(9-10):1164–1193.