ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-863

Adaptive Symbiotic Collaboration for Targeted Complex Manipulation Tasks

Rui Silva[1] and **Francisco S. Melo**[2] and **Manuela Veloso**[3]

Abstract.

This paper addresses the problem of human-robot collaboration in the context of manipulation tasks. In particular, we focus on tasks where a robot must perform some complex manipulation that is successfully completed only upon reaching some target pose provided by a human user. We propose an approach in which the robot explicitly reasons about its ability to complete the task and proactively requests the assistance of the human teammate when necessary. Our approach effectively trades-off the benefits arising from the human assistance with the cost of disturbing the user. We also propose an adaptation mechanism that enables the robot to adjust its behavior to the particular manner by which the human user responds to the requests made by the robot. We test our approach in a simple illustrative scenario and in two real interaction scenarios involving the Baxter robot.

1 Introduction

In this paper we address the general problem of human-robot collaboration, where a human user and a robot work together towards the successful completion of some predefined task. We are particularly interested in tasks where the robot is required to perform a complex motion involving the manipulation of an object which will act as the "interface" supporting the interaction with the human user. Examples of the tasks we envision include assisting the human user in dressing a piece of clothing [5, 8] or jointly preparing a drink [4].

Our contribution is aimed towards three distinguishing aspects of the class of scenarios described:

- The task is usually hard to model, which poses difficulties in the application of standard motion planners. To deal with this challenge, we adopt a motion representation that is particularly suited for *learning from demonstration* [1]. Explicitly teaching the robot the desired motion circumvents the need for explicit planning. At the same time, it provides a natural and intuitive interface for human users to program the robot to execute new tasks [18].
- Although the particular motion that the robot must perform to complete the desired task depends on the human user (for example, the dressing motion depends on the pose of the human user; pouring a drink into a cup depends on where the user places the cup), the *general shape of the trajectory* is approximately the same. Our approach relies on the notion of *motion primitives* [7], which represent idealized motions to perform a given task. Motion primitives provide a flexible way of representing the desired trajectory while, at the same time, modulating the trajectory to adapt to changing task parameters (e.g., depending on the human user).
- Finally, when interacting with the robot, the human user seldom takes into consideration the motion limitations that the robot has. For example, in the dressing assistance task, the user may assume a pose that blocks the robot's motion; or, in the drink pouring scenario, the human user may place the cup outside of the robot's workspace. We follow recent work on *symbiotic autonomy* [16], where the robot acknowledges and explicitly reasons about its own limitations, and relies on the human assistance to overcome such limitations. In this process, the robot takes into consideration the burden imposed upon the human user, trading it off with the benefits obtained from the human intervention.

Summarizing, our contributions are threefold. First, we contribute a novel decision-theoretic framework enabling a robot to reason explicitly about its own limitations during task execution. Second, we exploit our framework in the context of *symbiotic autonomy* [16], allowing the the robot to consider the potential benefit of enrolling human assistance during task execution against the associated cost. Third, we combine our framework with an online adaptive mechanism that uses the outcome of the successive interactions between the robot and the user to refine the model of the user's behavior and improve the quality of the interaction.

1.1 Related work

The topic of cooperative object manipulation has been widely studied in the literature, where the standard example involves joint lifting a heavy object [20]. In such collaborative tasks, the human user typically plays the role of the "leader", guiding the execution of the task, and the robot adapts its execution to that of the leader. In such interaction paradigm, the robot must often *predict* the motion of the human and act accordingly, and several approaches have been proposed in which collaborative manipulation relies on the prediction of human motion [2, 10–12].

Recent work has, to some extent, shifted to the robot part of the responsibility and initiative in completing the task. In this line of work, the motion executed by the robot is planned so that it can easily be interpreted/predicted by the human user [4, 6, 13], thus facilitating the (implicit) coordination of the two elements. For example, Dragan et al. [4] investigate how the shape of the trajectory by the robot impacts task efficiency: trajectories that are more *legible* allow the human user to more effectively interact with the robot than trajectories that are more *functional*. A related idea is pursued in [22], where, in this case, the best motion is learned.

[1] INESC-ID and Instituto Superior Técnico, University of Lisbon and Carnegie Mellon University, e-mail: rui.teixeira.silva@tecnico.ulisboa.pt
[2] INESC-ID and Instituto Superior Técnico, University of Lisbon, e-mail: fmelo@inesc-id.pt
[3] Carnegie Mellon University, e-mail: veloso@cs.cmu.edu

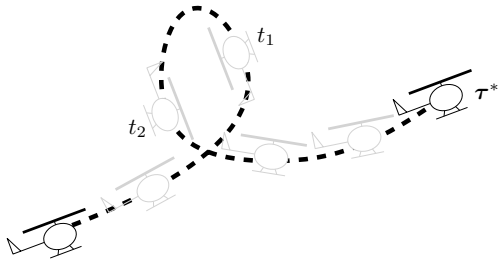


Figure 1. Illustration of a possible trajectory to perform a back flip. At some instant t_1, the tail of the helicopter will stand below the cabin; afterwards, at some time instant $t_2 > t_1$, the tail will pass over the cabin.

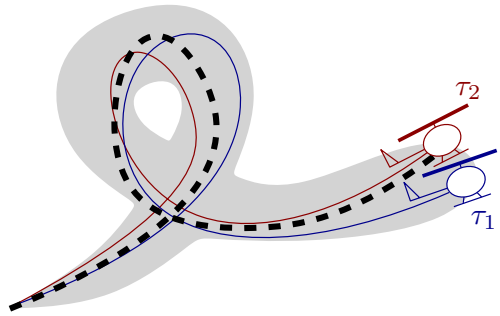


Figure 2. A ProMP that represents a helicopter back flip may assign positive probability to trajectories fulfilling the desired task (corresponding to the shaded area). Trajectories closer to the desired trajectory, herein represented as the dashed line, are assigned larger probability, which in this case could mean that $p_{\text{flip}}(\tau_2) > p_{\text{flip}}(\tau_1)$.

Another line of work closely related with our own work investigates *adjustable autonomy* in human-robot and human-agent teams [3, 17, 19]. For example, Scerri et al. [17] introduce the notion of *transfer-of-control*, where agents interacting with humans adopt a decision-theoretic framework to reason about when to handle the task control to the human users. Sellner et al. [19] investigate the impact of sliding autonomy in the performance of a multi-robot team.

Also relevant for our work is recent research in *impromptu teams* or *ad hoc teams* [14, 21]. Ad hoc teamwork seeks to develop strategies that enable an agent to rapidly infer the strategy of its teammates and act accordingly, towards the joint completion of some common task. Although most work in this area addresses this problem from a high-level, decision-theoretic perspective, the challenges faced in this topic of research bare a close resemblance to those found in collaborative human-robot manipulation: the "ad hoc agent" must infer the goal of the teammate and adapt its actions accordingly.

In this paper we propose an approach in which the robot takes upon itself the initiative of solving the task, explicitly requesting the assistance of the human teammate when necessary. This approach, in which the robot plays an active role in the interaction, can be seen as an instantiation of the concept of *symbiotic autonomy* [16], recently introduced and explored in the context of robot navigation. We propose an approach in which the robot is able to autonomously trade-off the cost associated with "disturbing" the human user with the potential benefit arising from the human intervention. In our approach we use *probabilistic motion primitives*, recently introduced by Paraschos et al. [15] as a flexible representation for robot motions and explored in the context of interaction in [9].

2 Probabilistic Motion Primitives

In our approach we use *probabilistic motion primitives* to represent the motion that the robot should perform during task execution.

Probabilistic motion primitives (ProMPs) were introduced in [15] as a way to represent in a flexible way the motion necessary to complete some well-defined task. Consider, for example, a helicopter for which we want to represent the motion associated with performing a back flip. There are multiple ways by which a back flip can be performed, all of which, however, share several distinctive features. For example, independently of how the back flip is performed, at some point in time the tail of the helicopter will stand below the cabin, after which the tail will move above the cabin (see Fig. 1).

A ProMP can be seen as a representation of an "idealized trajectory" $\boldsymbol{\tau}^*$ for the intended motion, where the different ways by which the task can be addressed are seen as *perturbations* of this idealized trajectory. Therefore, a ProMP takes the form of a probability distribution over the space of trajectories. Trajectories "closer" to the idealized trajectory are assigned a larger probability, while trajectories that are further away are assigned smaller probability. The probability of a given trajectory describes how likely it is for an agent to perform such trajectory when performing the desired task.

Returning to our previous helicopter example, a ProMP to represent a back flip could assign a positive probability to trajectories in which, at some point in time, the helicopter tail moves from below the cabin to above the cabin, assigning larger probability to those closer to the desired trajectory. Figure 2 illustrates this idea: the shaded area visually represents the "space of trajectories" that are assigned positive probability. Moreover, in this illustration the trajectory τ_2 is close to the desired trajectory (the dashed line), which would imply that the ProMP would assign larger probability to $\boldsymbol{\tau}_2$ than to $\boldsymbol{\tau}_1$.

We represent a trajectory as a sequence

$$\boldsymbol{\tau} = \{\boldsymbol{y}(0), \boldsymbol{y}(1), \ldots, \boldsymbol{y}(T)\},$$

where T denotes the trajectory length and $\boldsymbol{y}(t)$ is the pose of the robot at time-step t. We write $\mathcal{T}$ to denote the space of trajectories and refer to a ProMP M as some distribution p_M over $\mathcal{T}$, where $p_M(\boldsymbol{\tau})$ denotes the probability of trajectory $\boldsymbol{\tau} \in \mathcal{T}$ in the context of the desired task, as seen above.

For representational purposes, and following [15], we assume that the "idealized" trajectory associated with a ProMP can be constructed as the linear combination of a set of well-defined *trajectory features*, $\boldsymbol{\phi}_k, k = 1, \ldots, K$; the trajectories pertaining to the ProMP thus take the general form

$$\boldsymbol{y}(t) = \sum_{k=1}^{K} \boldsymbol{\phi}_k(t) w_k + \varepsilon(t) = \boldsymbol{\Phi}^\top(t)\boldsymbol{w} + \varepsilon(t), \qquad (1)$$

where $\{\varepsilon(t), t = 1, \ldots, T\}$ is a noise sequence and $\boldsymbol{w} \in \mathbb{R}^K$ is a vector of parameters, w_k representing the weight of feature $\boldsymbol{\phi}_k$ in the trajectory. This simplifying assumption establishes a correspondence between the space of trajectories $\mathcal{T}$ and the space of parameters, and allows the probability distribution over trajectories that represents the ProMP to be expressed as a distribution over parameters, which is easier to represent and manipulate. Therefore, one can easily express operations involving the motion primitive in terms of familiar concepts and operations from probability theory. For example,

- Prior knowledge regarding the task or preference over the trajectories that best accomplish it can be expressed in the form of a prior

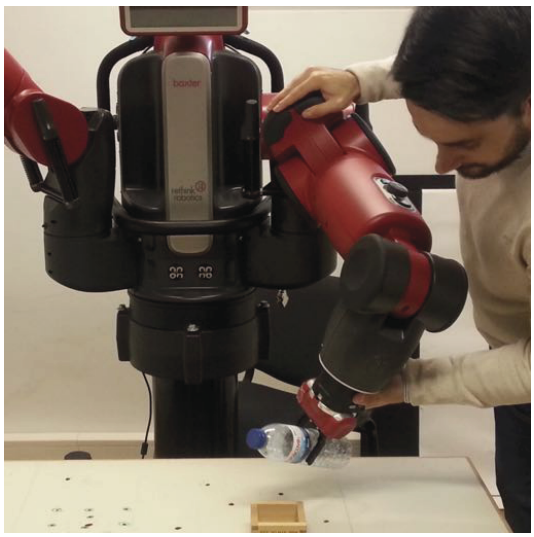


Figure 3. Kinesthetic teaching. In the image, the human teacher shows the robot the necessary motion to pour liquid into a cup.

distribution p_M over $\mathbb{R}^K$.

- Sample demonstrations of the desired trajectory provided by an expert can be integrated into the ProMP by a standard Bayesian update. For example, letting $\mathcal{D} = \{\boldsymbol{\tau}_1, \ldots, \boldsymbol{\tau}_N\}$ denote a dataset containing several (independent) trajectories demonstrating the desired motion, we can *learn* from such demonstrations simply by updating the ProMP distribution to the posterior

$$p_M(\boldsymbol{w} \mid \mathcal{D}) \propto p(\mathcal{D} \mid \boldsymbol{w}) p_M(\boldsymbol{w}) \tag{2}$$
$$= \prod_{n=1}^{N} p(\boldsymbol{\tau}_n \mid \boldsymbol{w}) p_M(\boldsymbol{w}),$$

  where $p(\boldsymbol{\tau}_n \mid \boldsymbol{w})$ represents the likelihood of observing a trajectory $\boldsymbol{\tau}_n$ when the idealized trajectory is represented by the parameter vector $\boldsymbol{w}$.
- Similarly, modulating the trajectory to reach a certain target pose $\boldsymbol{y}^*$ at time-step t can easily be achieved by computing

$$p_{\text{reach}}(\boldsymbol{w} \mid \boldsymbol{y}(t) = \boldsymbol{y}^*) \propto p(\boldsymbol{y}(t) = \boldsymbol{y}^* \mid \boldsymbol{w}) p_M(\boldsymbol{w}) \tag{3}$$

  where, once again, $p(\boldsymbol{y}(t) = \boldsymbol{y}^* \mid \boldsymbol{w})$ represents the likelihood of attaining pose $\boldsymbol{y}^*$ at time-step t when the idealized trajectory is represented by the parameter vector $\boldsymbol{w}$.

In the next section we describe the class of problems addressed in the paper, where a robot must perform a complex motion towards a target provided by a human user. We contribute a novel approach that enables the robot to reason about asking the human user for assistance, explicitly weighting the potential improvement in (task) performance that may result from such assistance against the cost of disturbing the human user. Our approach is designed to leverage the representational power of ProMPs while, at the same time, enable the robot to adjust to the particular user by learning how the latter responds to the requests made by the robot.

3 Collaborative Manipulation

We focus on tasks where a robot must perform some potentially complex manipulation that is successfully completed only upon reaching some target pose provided by a human user. Examples of such scenarios include placing an object in a container held by the user, or assisting a human to put on some piece of clothing. In both examples, the robot must perform a complex motion that culminates with reaching a target pose—the position of the container or a part of the human body.

In all our examples we start by building a ProMP M from a set of trajectories acquired by kinesthetic demonstration: a human expert guides the robot along the whole motion from the initial position to the target position (see Fig. 3). We adopt a fully Bayesian approach, where the demonstrated trajectories are used to compute a posterior distribution p_M over parameters, as suggested by the ProMP manipulations discussed in Section 2.

Following [15] we use a parametric representation for the distribution p_M, and denote by $\boldsymbol{\theta}$ the parameters of the distribution. We can rewrite the ProMP update equation for a single trajectory $\boldsymbol{\tau}$ to bring forth the role of the distribution parameters to yield

$$p_M(\boldsymbol{w} \mid \boldsymbol{\tau}, \boldsymbol{\theta}) = \frac{1}{\eta} p(\boldsymbol{\tau} \mid \boldsymbol{w}) p_M(\boldsymbol{w} \mid \boldsymbol{\theta}),$$

where η is a normalizing constant. The likelihood $p(\boldsymbol{\tau} \mid \boldsymbol{w})$ depends directly on the noise sequence $\{\varepsilon(t), t = 1, \ldots, T\}$ in (1). For simplicity, we take $\{\varepsilon(t), t = 1, \ldots, T\}$ to be white Gaussian noise with zero mean and known covariance $\boldsymbol{\Sigma}_\varepsilon$. Therefore, and since the sequence $\{\varepsilon(t), t = 1, \ldots, T\}$ is uncorrelated in virtue of our whiteness assumption, the likelihood $p(\boldsymbol{\tau} \mid \boldsymbol{w})$ can be written as

$$\begin{aligned} p(\boldsymbol{\tau} \mid \boldsymbol{w}) &= \prod_{t=1}^{T} p(\boldsymbol{y}(t) \mid \boldsymbol{w}) \\ &= \prod_{t=1}^{T} \mathsf{Normal}\left(\boldsymbol{y}(t) \mid \boldsymbol{\Phi}^\top(t)\boldsymbol{w}, \boldsymbol{\Sigma}_\varepsilon\right), \end{aligned}$$

and taking a Gaussian prior over $\mathcal{W}$, we have, for each trajectory $\boldsymbol{\tau}$,

$$\begin{aligned} &p_M(\boldsymbol{w} \mid \boldsymbol{\tau}, \boldsymbol{\theta}) \\ &= \frac{1}{\eta} \prod_{t=1}^{T} \mathsf{Normal}\left(\boldsymbol{y}(t) \mid \boldsymbol{\Phi}^\top(t)\boldsymbol{w}, \boldsymbol{\Sigma}_\varepsilon\right) \mathsf{Normal}\left(\boldsymbol{w} \mid \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right), \end{aligned}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\}$.

3.1 Task execution

Given a ProMP M, learned from a set of demonstrated trajectories, and a target pose $\boldsymbol{y}^*$, we can now modulate the trajectories of the ProMP M using (3) which, in light of our Gaussian assumption, translates into the standard updates for $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$,

$$\begin{aligned} \boldsymbol{\mu}_w^{\text{new}} &= \boldsymbol{\mu}_w + \mathbf{K}(\boldsymbol{y}^* - \boldsymbol{\Phi}^\top(T)\boldsymbol{\mu}_w) \\ \boldsymbol{\Sigma}_w^{\text{new}} &= (\boldsymbol{I} - \mathbf{K}\boldsymbol{\Phi}^\top(T))\boldsymbol{\Sigma}_w, \end{aligned}$$

where the matrix $\mathbf{K}$ is given by

$$\mathbf{K} = \boldsymbol{\Sigma}_w \boldsymbol{\Phi}(T) \mathbf{S}^{-1}$$

and

$$\mathbf{S} = \boldsymbol{\Phi}^\top(T) \boldsymbol{\Sigma}_w \boldsymbol{\Phi}(T) + \boldsymbol{\Sigma}_\varepsilon.$$

The updated Gaussian distribution, with parameters $\boldsymbol{\theta}^{\text{new}} = \{\boldsymbol{\mu}_w^{\text{new}}, \boldsymbol{\Sigma}_w^{\text{new}}\}$, can be used to obtain a trajectory that leads the robot to the new target position.

However, for targets $\boldsymbol{y}^*$ too distant from those observed in the demonstrations, the modulation may yield trajectories quite different from the desired one that may actually extend beyond the workspace of the robot. We introduce a *cost function*, $c_E : \mathcal{T} \to \mathbb{R}$, providing the robot with a quantitative measure of the quality of any trajectory and a way to explicitly reason about its ability to perform it.

We write $c_E(\boldsymbol{\tau})$ to denote the *execution cost* associated with a trajectory $\boldsymbol{\tau} \in \mathcal{T}$ in light of the target task. The cost function c_E can account for task success, safe execution, robustness, or any other performance criteria that the task designer specifies. Then, given a target pose $\boldsymbol{y}^*$ and a ProMP M, we can compute the *expected execution cost* of M given $\boldsymbol{y}^*$ as

$$C_E(\boldsymbol{y}^*) = \mathbb{E}_\tau \left[c_E(\boldsymbol{\tau})\right] \triangleq \int c_E(\boldsymbol{\tau}) p(\boldsymbol{\tau} \mid \boldsymbol{y}^*, \boldsymbol{\theta}) d\boldsymbol{\tau}, \qquad (4)$$

where, as before, $\boldsymbol{\theta}$ represents the parameters of the ProMP distribution p_M. For the Gaussian setup considered, we can further break down the computation in (4). Abusing somewhat our notation, let $c_E(\boldsymbol{w})$ denote the (expected) execution cost associated with the parameter vector $\boldsymbol{w}$, i.e.,

$$\begin{aligned} c_E(\boldsymbol{w}) &= \int c_E(\boldsymbol{\tau}) p(\boldsymbol{\tau} \mid \boldsymbol{w}) d\boldsymbol{\tau} \\ &= \int c_E(\boldsymbol{\Phi}^\top \boldsymbol{w} + \boldsymbol{\varepsilon}) p(\boldsymbol{\varepsilon}) d\boldsymbol{\varepsilon} \end{aligned}$$

where we wrote $\boldsymbol{\varepsilon}$ to denote the noise sequence. Then

$$\begin{aligned} C_E(\boldsymbol{y}^*) &= \int c_E(\boldsymbol{w}) p(\boldsymbol{w} \mid \boldsymbol{y}^*, \boldsymbol{\theta}) d\boldsymbol{w} \\ &= \int c_E(\boldsymbol{w}) \mathsf{Normal}\left(\boldsymbol{w} \mid \boldsymbol{\mu}_w^{\text{new}}, \boldsymbol{\Sigma}_w^{\text{new}}\right) d\boldsymbol{w}. \end{aligned}$$

3.2 Active collaborative manipulation

A natural possibility to address the difficulties posed by targets placed in inconvenient locations (in the sense discussed above) is to acknowledge such difficulties and request the human user for assistance in moving the target to a more convenient location. Suppose then that the robot has available a finite set of instructions, $\mathcal{A} = \{a_0, a_1, \ldots, a_J\}$, that it can voice to the human user. Instructions $a_1, \ldots, a_J$ correspond to requests to the human user to move the target in a specific manner, while a_0 corresponds to the *null instruction* (request nothing from the human user). For example, in the scenario of Fig. 3 where the robot must pour a liquid in a cup, a possible request could be to place the cup closer to the robot.

We associate with each instruction $a \in \mathcal{A}$ a *request cost*, $c_R(a)$, that quantifies the burden imposed upon the user in executing the action associated with instruction a, and should be designed relatively to $c_E(\boldsymbol{\tau})$. Thus, assuming a $c_E(\boldsymbol{\tau})$ in $[0, 1]$, $c_R(a) = 0.1$ indicates that action a adds an additional 10% burden. Additionally, associated with each instruction $a \in \mathcal{A}$ and each target position $\boldsymbol{y}$, we consider a *transition model* that describes (probabilistically) the outcome of the user's action in terms of the target position. In particular, we write $p_{\text{target}}(\boldsymbol{y}' \mid \boldsymbol{y}, a)$ to denote the probability that, after requesting a, the target moves from $\boldsymbol{y}$ to $\boldsymbol{y}'$. For action a_0, $p_{\text{target}}(\boldsymbol{y}' \mid \boldsymbol{y}, a_0) = \delta(\boldsymbol{y}', \boldsymbol{y})$.

With the elements above, the total cost incurred by the robot upon performing request a at the target position $\boldsymbol{y}$ and then moving to reach the target in the resulting position is given by

$$\begin{aligned} C_M(a) &= c_R(a) + \mathbb{E}_y \left[C_E(\boldsymbol{y}') \mid \boldsymbol{y}, a\right] \\ &= c_R(a) + \int_{\boldsymbol{y}'} C_E(\boldsymbol{y}') p_{\text{target}}(\boldsymbol{y}' \mid \boldsymbol{y}, a) d\boldsymbol{y}'. \end{aligned}$$

Assuming that the instruction set $\mathcal{A}$ always includes the null instruction a_0 (with a cost of $c_R(a_0) = 0$), the robot will select action

$$a^* = \arg\min_{a \in \mathcal{A}} C_M(a),$$

thus pro-actively requesting the user's assistance in the reaching task whenever the benefit from the user's aid effectively surpasses the burden imposed upon that user. The interplay between the two components of the cost C_M, namely c_E and c_R, determines in which circumstances it pays off to rely on human assistance: when c_R is small it is generally better to request, while for large c_R it is generally better to execute without human assistance.

3.3 An illustrative example

We now illustrate the framework for active collaborative manipulation introduced above in a simple 2D example. We used kinesthetic teaching to collect a number of projected trajectories, depicted in Fig. 4,[4] and computed the associated ProMP parameters $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$.

In this example scenario, we consider a simple execution cost that penalizes deviations from the known target area—the shaded region in Fig. 4(a). In fact, targets away from those reached in the demonstrations lead to trajectories with shapes significantly different from those demonstrated, as seen in Fig. 4(b). Therefore, in this example scenario, given a trajectory $\boldsymbol{\tau} = \{\boldsymbol{y}(0), \ldots, \boldsymbol{y}(T)\}$ we use

$$\begin{aligned} c_E(\boldsymbol{\tau}) &= K_E\big(1 - p_M(\boldsymbol{y}(T) \mid \boldsymbol{\theta})\big) \\ &= K_E \left(1 - \int p(\boldsymbol{y}(T) \mid \boldsymbol{w}) p_M(\boldsymbol{w} \mid \boldsymbol{\theta}) d\boldsymbol{w}\right), \end{aligned}$$

where

$$\begin{aligned} p(\boldsymbol{y}(T) \mid \boldsymbol{w}) &= \mathsf{Normal}\left(\boldsymbol{y}(T) - \boldsymbol{\Phi}^\top(T)\boldsymbol{w} \mid 0, \boldsymbol{\Sigma}_\varepsilon\right), \\ p_M(\boldsymbol{w} \mid \boldsymbol{\theta}) &= \mathsf{Normal}\left(\boldsymbol{w} \mid \boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w\right), \end{aligned}$$

and K_E is a constant.

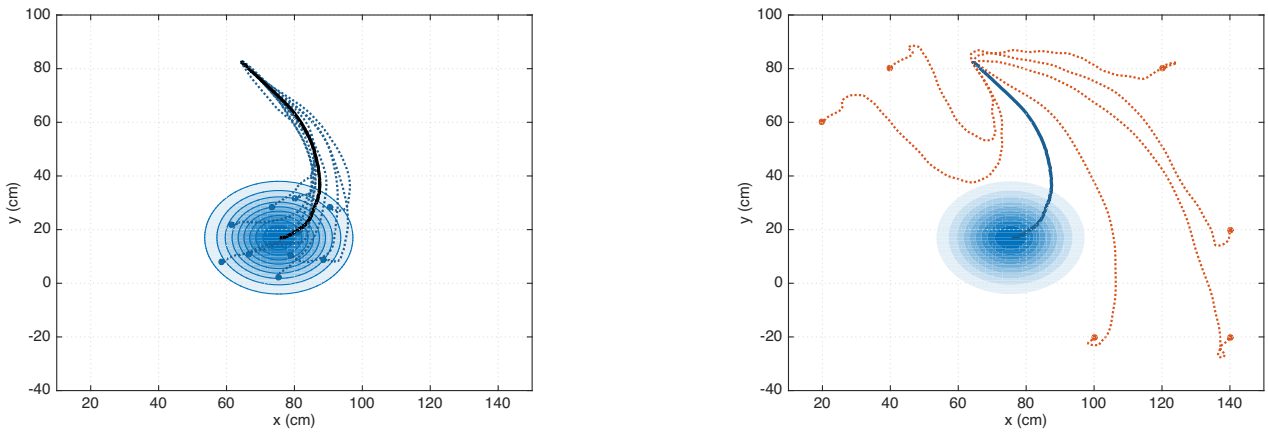
We consider an instruction set $\mathcal{A} = \{\emptyset, U, D, L, R\}$, corresponding to the null request ($\emptyset$) and requests to move the target up (U), down (D), left (L) and right (R), and a constant request cost $c_R(a) = K_R, a \in \mathcal{A} \backslash \{\emptyset\}$. Figure 5 compares, for different values of K_E and K_R the average performance obtained by (i) always trying to execute the trajectory most likely to lead to the target, without ever requesting for assistance (Always execute); (ii) always requesting assistance before executing (Always request); and (iii) using our approach.[5]

Several aspects are worth noting in Fig. 5. First of all, as expected, the total cost grows linearly with both K_E and K_R, as seen in the curves corresponding to the approach that always executes and the approach that always requests, respectively.

A second point worth noting is that, also as seen in Section 3.2, for small K_R, it is generally better to request, while for large K_R it is always better to execute without human assistance.

[4] The trajectories were collected using the Baxter robot and the positions of the end effector were then projected in a 2D plane.

[5] The results in Fig. 5 were obtained by sampling $1,000$ targets uniformly at random from the robot's workspace, and averaging the performance of the different approaches in these $1,000$ points. Human intervention after an instruction by the robot leads to an average displacement of the target of $\sim$20cm in the requested direction. Finally, we considered that there was no execution noise.



(a) (b)

Figure 4. (a) 2D trajectories collected through kinesthetic teaching (dotted lines). The solid line represents the average trajectory, while the contours represent the distribution of targets estimated from the demonstrated trajectories. (b) Trajectories obtained by modulating the ProMP to targets away from the "known area", represented by the contour. The solid line represents the "mean trajectory".

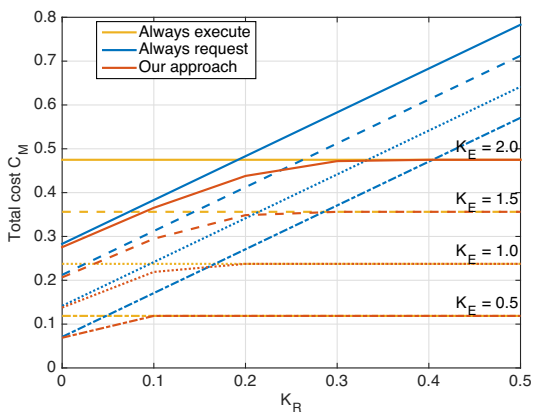


Figure 5. Performance of three approaches to the manipulation task that rely on human assistance at different levels. See text for details.

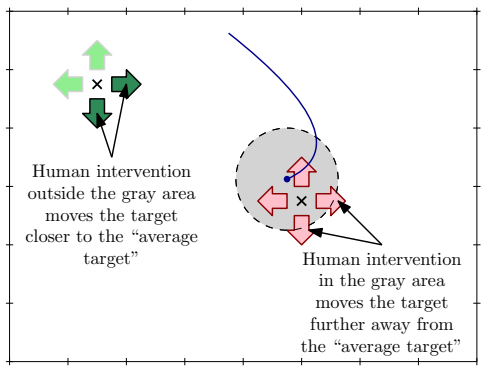


Figure 6. Diagram illustrating the impact of human intervention in different regions of the robot's workspace.

Finally, we note that our approach naturally outperforms the other two naïve approaches, striking the correct trade-off between the costs and benefits of requesting human assistance, the difference being largest in those situations where the latter two perform similarly. This observation can be explained by noting that, except in a "comfort region" around the "average target"—in which any human action moves the target away—human intervention may always be selected so as to bring the target closer to the comfort region. When the two naïve approaches perform similarly means that the larger execution cost incurred by the approach that always executes in areas far from the comfort region compensates the request cost incurred by the approach that always requests around the comfort zone. Our approach, nevertheless, takes the best of the two and is thus able to attain significant improvements with respect to the other two approaches.

4 Adapting to the human user

According our approach the robot selects the action

$$a^* = \arg\min_{a \in \mathcal{A}} C_M(a),$$

where the cost C_M is given by

$$C_M(a) = c_R(a) + \mathbb{E}_y \left[C_E(\boldsymbol{y}') \mid \boldsymbol{y}, a \right].$$

As pointed out before, the cost C_M expresses the trade-off between the costs and benefits of recruiting human assistance in the execution of the desired task. However, computing C_M requires knowledge of the transition probabilities $p_{\text{target}}(\boldsymbol{y}' \mid \boldsymbol{y}, a)$, which essentially corresponds to a model of how the human user moves the target in response to the robot's request. Such model can be learned from data obtained from people moving the target. However, there is still the possibility that, for one particular user, the model is incorrect—for example if the user has some physical limitation, it may not be able to perform some of the actions requested by the robot.

To illustrate the potential impact of a wrong model in the performance of the robot, we return to the example from Section 3.3 and determine what happens when the transition model $p_{\text{target}}(\boldsymbol{y}' \mid \boldsymbol{y}, a)$ is incorrect. In particular, let us suppose that a particular user cannot perform vertical motions. We set $K_R = 0.17$ and $K_E = 2$ and draw $50,000$ target poses uniformly at random in the robot's workspace.

The results are summarized in Table 1.

Table 1. Performance of different approaches with correct and incorrect models.

Approach	Avg. cost
Always execute	0.4764
Always request (incorrect model)	0.5184
Always request (correct model)	0.5023
Our approach (incorrect model)	0.4748
Our approach (correct model)	0.4405

The results in Table 1 show that, in the example, the approach that never requests assistance from the human user is generally superior to the approach that always requests. It is also worth noting that the performance of the approach that always requests does not significantly change by using an incorrect model. Both approaches always request human assistance, and the only difference lies on the selection of the instructions to be voiced. The difference in performance is explained mostly by those situations in which the best action is a request to move the target horizontally but the approach with the invalid model incorrectly requests the user to move the target vertically, and vice-versa.

The results also show that even with an incorrect model our approach is still able to outperform (even if by just a small margin) the naïve approaches. Finally, the results confirm that an incorrect model indeed impacts the performance of the robot—when using the correct model, our approach presents a significant advantage over all other approaches.

To address the possibility of an incorrect transition model, we introduce an adaptation mechanism that, after each action requested from the human user, updates the model to more faithfully translate the most recent evidence. In particular, we adopt a parameterized representation for the transition probabilities p_{target} and write $p_{\text{target}}(\boldsymbol{y}' \mid \boldsymbol{y}, \boldsymbol{\alpha})$ to denote the transition probabilities corresponding to the parameter $\boldsymbol{\alpha}$. We associate a parameter vector $\boldsymbol{\alpha}_a$ with each action $a \in \mathcal{A}$, implying that

$$p_{\text{target}}(\boldsymbol{y}' \mid \boldsymbol{y}, a) \equiv p_{\text{target}}(\boldsymbol{y}' \mid \boldsymbol{y}, \boldsymbol{\alpha}_a).$$

Then, by updating the parameters $\boldsymbol{\alpha}_a$ associated with each action $a \in \mathcal{A}$ the robot is able to incrementally adapt to the specific human user that it is currently interacting with.

We conclude by revisiting the example of Section 3.3 to illustrate the impact of adaptation in our approach.

4.1 Illustrative example revisited

We adopt a simple parametric model for the transition probabilities p_{target}. Namely, when the robot voices an instruction $a \in \mathcal{A} \setminus \{a_0\}$,

$$\boldsymbol{y}' = \boldsymbol{y} + \boldsymbol{\alpha}_a + \nu,$$

where $\boldsymbol{\alpha}_a$ is a displacement vector associated with action a and ν is a zero-mean Gaussian disturbance. Equivalently,

$$p_{\text{target}}(\boldsymbol{y}' \mid \boldsymbol{y}, \boldsymbol{\alpha}_a) = \mathsf{Normal}\left(\boldsymbol{y}' - (\boldsymbol{y} + \boldsymbol{\alpha}_a), \boldsymbol{\Sigma}_\nu\right),$$

where $\boldsymbol{\Sigma}_\nu$ is the variance of the disturbance. The update of each parameter vector $\boldsymbol{\alpha}_a$ can be done more or less aggressively, depending on how much weight we want to assign to the robot's experience.

We set $K_R = 0.17$ and $K_E = 2$ and draw $5,000$ target poses uniformly at random in the robot's workspace. After every 250 steps we update the parameters $\boldsymbol{\alpha}_a, a \in \mathcal{A}$, in a total of 20 update steps. Figure 7 depicts the obtained results, including the performance of the five approaches featured in Table 1, as well as the learning approach. As is clear from the plot, the learning approach initially exhibits a performance similar to our approach with an incorrect model. However, as more information from the user becomes available, the performance slowly converges towards that of a correct model.

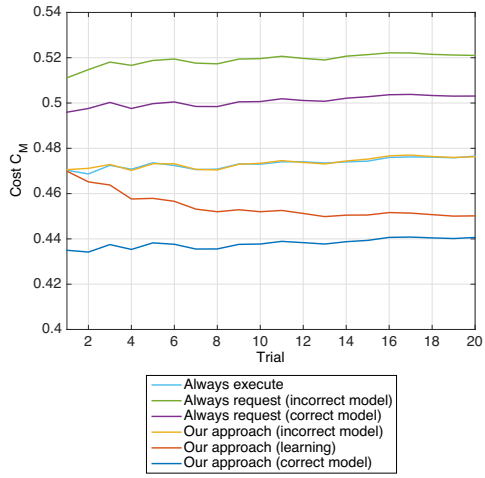


Figure 7. Average learning performance in the example of Section 3.3.

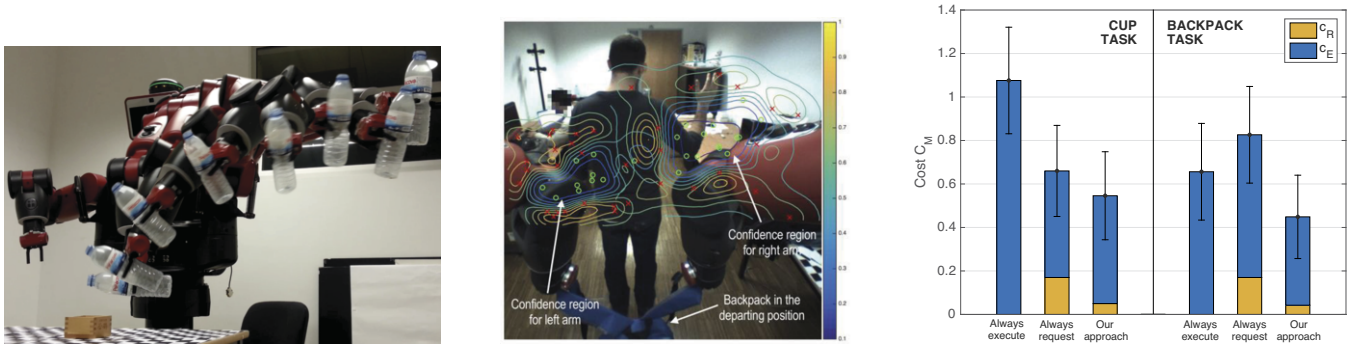
5 Experimental results

We illustrate the application of our framework in two real-world tasks involving the Baxter robot. In both tasks, Baxter posed the requests to the human user through voice commands. In the first task, the robot pours a bottle to a cup placed on a table. The robot was provided with five demonstrations of the intended motion through kinesthetic teaching (as demonstrated in Fig. 3), with the cup placed in different positions in the table.[6] These trajectories were used to compute a ProMP as described in the illustrative example of Section 3. At execution time, we use homography to determine the position of the cup in the table, and modulate the ProMP as in (3) to obtain the trajectory for that target. Figure 8(a) illustrates the execution of a pouring motion. If the bottleneck ends over the cup, the motion is considered a success and the robot incurs a small execution cost (depending on the distance to the center of the cup). If the bottleneck does not end over the cup, the motion is considered a failure, and the robot incurs a large penalty. For decision-making, a simple model predicting the probability of failure is used to compute the execution cost.

Note that, even if this task in itself is not very complex, the limited number of demonstrations prevents the robot from confidently generalizing the motion to the whole table, allowing us to study the effectiveness of our approach in handling the robot's limitations.

In the second task, the robot helps the user to put on a backpack. This task is significantly more complex, comprising two motions involving both of Baxter's arms (in a total of 14 DoF) and requiring some compliance from the user. The robot was provided with four demonstrations of the intended motion, with the user placed in different positions in front of the robot. We used as targets the hands of

[6] Trajectories were stored in joint space.



(a) (b) (c)

Figure 8. (a) Example of a pouring motion executed by Baxter. (b) Contour lines of the execution cost function c_E for the backpack task, superimposed on the robot's view. (c) Comparison of the average performance of the three approaches. The results portrayed are averages over 17 trials (cup) and 16 trials (backpack) for each method. Error bars depict the standard error.

the user. At execution time, we again use homography to determine the position of the hands of the user in a vertical plane. The execution cost merely measures the success or failure of the motion. For decision-making, we use an SVM trained to predict success/failure given the target position to estimate the execution cost. Figure 8(b) shows the contours of the cost function for the backpack task.

We compare the performance of three approaches: (i) executing without ever requesting user assistance (Always execute); (ii) executing after always requesting user assistance (Always request); and (iii) ask human assistance when convenient (Our approach). Each approach was evaluated for a number of trials in each of the two tasks. In each trial, the robot is allowed to interact with a human user, where the target (cup or user) is placed in random positions in front of the robot. The robot proceeds as prescribed by the approach under consideration, and we evaluate the success of the corresponding motion. The results are summarized in Fig. 8(c).

Analyzing the results in detail, we note first that, as expected, the Always execute approach incurs no request cost. The Always request approach, on the other hand, incurs the largest request cost. However, overall, it is significantly better than the approach that always executes, which indicates that the request cost incurred by the former when requesting unnecessary user assistance still compensates the execution cost incurred by the latter with targets that it cannot safely reach. Our approach is the most cost-effective of all three, incurring less execution cost, request cost and overall cost than any of the other approaches. The results show that our approach successfully weights the robot's limitations, enlisting user assistance when the latter can be of use, thus establishing a form of symbiotic autonomy where both user and robot assist one another for mutual benefit.

We illustrate in Figures 9(a)-9(c) the performance of the different approaches for different positions of the cup. Several interesting observations are in order. First of all, considering Fig. 9(a), it is possible to observe that the robot is able to successfully complete the tasks when the cup is in the vicinity of the area demonstrated (marked as the shaded region). When the cup was placed in a position relatively distant from that area, the modulation process is less accurate, leading the robot to "miss" the cup and fail the task.

Considering now Fig. 9(b), we note that by requesting the user to move the cup, a significant number of cup positions that the robot missed in the Always execute approach can now be successfully corrected into positions that the robot can successfully reach. However, for those positions that the robot was already able to reach, the request to the user is unnecessary, leading the robot to incur an unnecessary request cost.

Finally, considering Fig. 9(c) it is clear that the robot requests only the user's assistance in those positions that it cannot successfully reach. Some of these remain out of the reach of the robot, but a significant number of them (as already observed in the Always request approach) can now successfully be reached. These results illustrate that our approach is, indeed, able to successfully trade-off the cost of requesting the assistance from the human user and the potential benefit arising from such assistance. The results illustrated in Fig. 9 are summarized (in terms of cost) in the plot of Fig. 8(c). As expected, the approach Always execute incurs no request cost. Conversely, the average request cost for Always request is $K_R = 0.17$, since all trials incur exactly this cost. Our approach incurs smaller cost both in terms of execution and in terms of request.

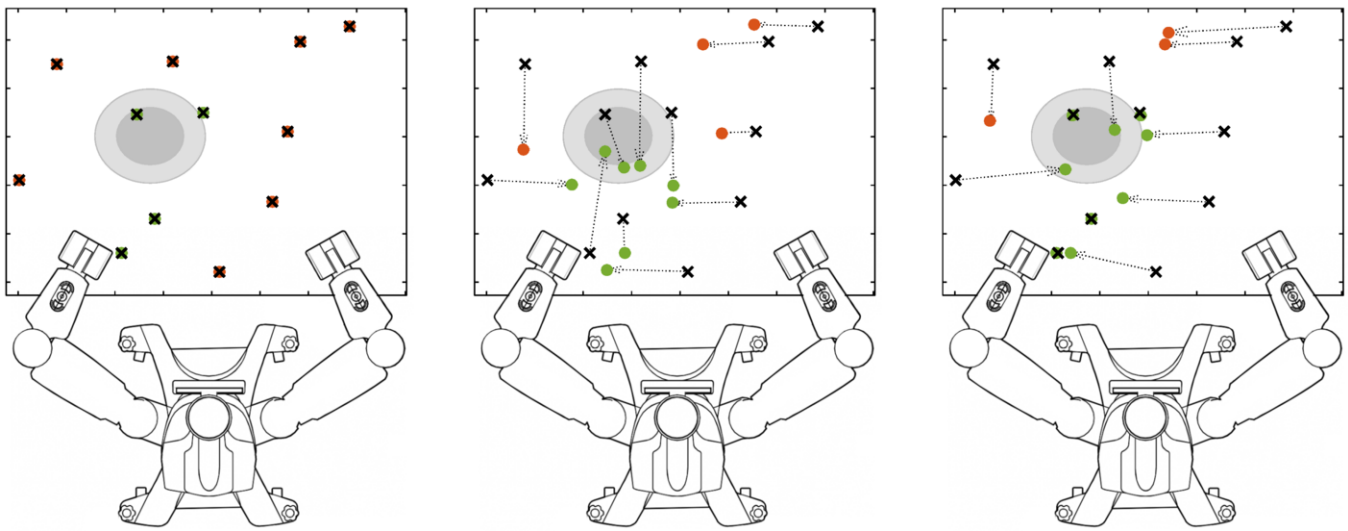
6 Conclusions

In this paper we presented a novel approach that explores the concept of *symbiotic autonomy* in the context of collaborative manipulation tasks. Our approach enables a robot to reason explicitly about its limitation when executing a complex manipulation task with respect to a target provided by a human user, and reason about when the cost of disturbing the human user and requesting its assistance is compensated by the benefits arising from such assistance. Our approach relies on a model of the human responses to the robot's request; when discrepancies are detected, our approach successively adjusts the model to a particular user, as the number of interactions increases. Our results show that our approach is effectively able to perform the desired trade-off, as well as adapt to gross errors in the user's model.

Our work also opens interesting directions for future work. Our current decision process is myopic: it selects the best option at each time-step disregarding the potential future effect of the actions. However, by using a more evolved decision-theoretic approach, it is possible to further optimize the (long-term) performance of the robot. In fact, given the Markovian nature of the target displacement model used, it should be possible to formulate the decision problem of the agent using a decision-theoretic model such as a Markov decision process, for which solution techniques are readily available.

Acknowledgements

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/



(a) Always execute. (b) Always request. (c) Our approach.

Figure 9. Visual representation of some of the tests performed with the Baxter robot in the pouring task. The gray area corresponds to the "comfort zone", i.e., the area covered by the demonstrated trajectories. The black crosses ("×") represent some of the positions where the cup was placed, and red and green dots represent whether that particular motion was successful or not (we consider a success whenever the bottle stops on top of the cup). For the approaches involving human motion, the dotted arrows represent the motion performed by the human user upon request.

50021/2013 and the Carnegie Mellon Portugal Program and its Information and Communications Technologies Institute, under project CMUP-ERI/HCI/0051/2013. The first author acknowledges the PhD grant SFRH/BD/113695/2015.

References

[1] B. Argall, S. Chernova, and M. Veloso, 'A survey of robot learning from demonstration', *Robotics and Auton. Systems*, **57**(5), 469–483, (2009).
[2] B. Corteville, E. Aertbelien, H. Bruyninckx, J. De Schutter, and H. Van Brussel, 'Human-inspired robot assistant for fast point-to-point movements', in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 3639–3644, (2007).
[3] M. Dias, B. Kannan, B. Browning, E. Jones, B. Argall, M. Dias, M. Zinck, M. Veloso, and A. Stentz, 'Sliding autonomy for peer-to-peer human-robot teams', in *Proc. 10th Int. Conf. Intelligent Autonomous Systems*, pp. 332–341, (2008).
[4] A. Dragan, S. Bauman, J. Forlizzi, and S. Srinivasa, 'Effects of robot motion on human-robot collaboration', in *Proc. 10th ACM/IEEE Int. Conf. Human-Robot Interaction*, pp. 51–58, (2015).
[5] Y. Gao, H. Chang, and Y. Demiris, 'User modelling for personalised dressing assistance by humanoid robots', in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 1840–1845, (2015).
[6] M. Gielniak, C.K. Liu, and A. Thomaz, 'Generating human-like motion for robots', *Int. J. Robotics Research*, **32**(11), 1275–1301, (2013).
[7] A. Ijspeert, J. Nakanishi, and S. Schaal, 'Learning attractor landscapes for learning motor primitives', in *Adv. Neural Information Processing Systems*, volume 15, pp. 1547–1554, (2003).
[8] S. Klee, B. Ferreira, R. Silva, J. Costeira, F. Melo, and M. Veloso, 'Personalized assistance for dressing users', in *Proc. 7th Int. Conf. Social Robots*, pp. 359–369, (2015).
[9] G. Maeda, M. Ewerton, R. Lioutikov, H. Ben Amor, J. Peters, and G. Neumann, 'Learning interaction for collaborative tasks with probabilistic movement primitives', in *Proc. 14th IEEE-RAS Int. Conf. Humanoid Robots*, pp. 527–534, (2014).
[10] Y. Maeda, T. Hara, and T. Arai, 'Human-robot cooperative manipulation with motion estimation', in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 2240–2245, (2001).
[11] J. Mainprice and D. Berenson, 'Human-robot collaborative manipulation planning using early prediction of human motion', in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 299–306, (2013).
[12] J. Mainprice, R. Hayne, and D. Berenson, 'Predicting human reaching motion in collaborative tasks using inverse optimal control and iterative re-planning', in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 885–892, (2015).
[13] J. Mainprice, E.A. Sisbot, L. Jaillet, J. Cortés, R. Alami, and T. Siméon, 'Planning human-aware motions using a sampling-based costmap planner', in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 5012–5017, (2011).
[14] F. Melo and A. Sardinha, 'Ad hoc teamwork by learning teammates' task', *Autonomous Agents and Multi-Agent Systems*, **30**(2), 175–219, (2016).
[15] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, 'Probabilistic movement primitives', in *Adv. Neural Information Proc. Systems*, volume 26, pp. 2616–2624, (2013).
[16] S. Rosenthal, J. Biswas, and M. Veloso, 'An effective personal mobile robot agent through symbiotic human-robot interaction', in *Proc. 9th Int. Joint Conf. Autonomous Agents and Multiagent Systems*, pp. 915–922, (2010).
[17] P. Scerri, D. Pynadath, and M. Tambe, 'Towards adjustable autonomy for the real world', *J. Artificial Intelligence Res.*, **1**, 2–50, (2003).
[18] S. Schaal, 'Is imitation learning the route to humanoid robots?', *Trends in Cognitive Sciences*, **3**(6), 233–242, (1999).
[19] B. Sellner, F. Heger, L. Hiatt, R. Simmons, and S. Singh, 'Coordinated multiagent teams and sliding autonomy for large-scale assembly', *Proc. IEEE*, **94**(7), 1425–1444, (2006).
[20] W. Sheng, A. Thobbi, and Y. Gu, 'An integrated framework for human-robot collaborative manipulation', *IEEE Trans. Cybernetics*, **45**(10), 2030–2041, (2015).
[21] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein, 'Ad hoc autonomous agent teams: Collaboration without pre-coordination', in *Proc. 24th AAAI Conf. Artificial Intelligence*, pp. 1504–1509, (2010).
[22] F. Stulp, J. Grizou, B. Busch, and M. Lopes, 'Facilitating intention prediction for humans by optimizing robot motions', in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 1249–1255, (2015).