Autonomous Robots
https://doi.org/10.1007/s10514-018-9706-9

Kinesthetic teaching and attentional supervision of structured tasks in human–robot interaction

Riccardo Caccavale[1] · **Matteo Saveriano**[2] · **Alberto Finzi**[1] · **Dongheui Lee**[2,3]

Received: 31 December 2016 / Accepted: 29 January 2018

Abstract
We present a framework that allows a robot manipulator to learn how to execute structured tasks from human demonstrations. The proposed system combines physical human–robot interaction with attentional supervision in order to support kinesthetic teaching, incremental learning, and cooperative execution of hierarchically structured tasks. In the proposed framework, the human demonstration is automatically segmented into basic movements, which are related to a task structure by an attentional system that supervises the overall interaction. The attentional system permits to track the human demonstration at different levels of abstraction and supports implicit non-verbal communication both during the teaching and the execution phase. Attention manipulation mechanisms (e.g. object and verbal cueing) can be exploited by the teacher to facilitate the learning process. On the other hand, the attentional system permits flexible and cooperative task execution. The paper describes the overall system architecture and details how cooperative tasks are learned and executed. The proposed approach is evaluated in a human–robot co-working scenario, showing that the robot is effectively able to rapidly learn and flexibly execute structured tasks.

Keywords Multimodal human–robot interaction · Attentional supervision · Learning from demonstration · Intuitive kinesthetic teaching

1 Introduction

The integration of robotic devices in human populated environments requires the ability of the robot to continuously learn novel tasks and to adapt their execution to the human intentions and behaviors. In a human-dwelled environment, indeed, the robot will be asked to execute incrementally complex activities both autonomously or in cooperation with human co-workers. In this scenario, the interaction with the human should be natural and fluent during both task execution and task learning. In this work, we propose a framework which allows natural human–robot interaction along with incremental teaching and autonomous or cooperative execution of structured tasks.

A structured task, like preparing a certain recipe, can be hierarchically decomposed in different subtasks involving

Riccardo Caccavale and Matteo Saveriano are co-first authors.

This is one of several papers published in *Autonomous Robots* comprising the "Special Issue on Learning for Human-Robot Collaboration".

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10514-018-9706-9) contains supplementary material, which is available to authorized users.

✉ Matteo Saveriano
matteo.saveriano@tum.de

Riccardo Caccavale
riccardo.caccavale@unina.it

Alberto Finzi
alberto.finzi@unina.it

Dongheui Lee
dhlee@tum.de

[1] Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, Università di Napoli Federico II, via Claudio 21, 80125 Naples, Italy

[2] Human-Centered Assistive Robotics (HCR), Technical University of Munich, Karlstrasse 45, 80333 Munich, Germany

[3] Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Münchener Str. 20, 82234 Weßling, Germany

Published online: 19 February 2018

multiple primitive actions and manipulated objects. Actions have to be performed in a coherent manner, meaning that the actions have to be executed on certain objects with a particular order. For example, to pour water in a cup, the robot has to take the bottle, reach the cup, and then pour the liquid. In order to make a robot able to learn how to perform structured tasks and collaboratively execute them, our approach integrates multimodal interaction (Rossi et al. 2013), attentional supervision (Norman and Shallice 1986; Cooper and Shallice 2006; Caccavale and Finzi 2015, 2016), and kinesthetic teaching (Lee and Ott 2011; Saveriano et al. 2015). In our framework, the human operator can naturally interact with the robot using voice and physical guidance, while a supervisory attentional system (Norman and Shallice 1986; Cooper and Shallice 2006) continuously monitors and tracks the human–robot interactive activities during both training and execution sessions.

Attentional mechanisms that are suitable for human–robot task teaching have been explored in the robotic literature, mainly in the context of visual attention (Nagai 2009; Breazeal and Berlin 2008; Borji et al. 2010); in contrast, in this work we focus on attentional supervision and physical interaction. Namely, in course of a kinesthetic teaching session, the human can physically interact with the robot to demonstrate the execution of the actions, while the supervisory system is exploited to interpret the human guidance in the context of a structured task. In this setting, the supervisory attentional system supports implicit non-verbal communication and permits to track the human demonstration at different levels of abstraction.

More specifically, human demonstrations are automatically segmented into basic movements, exploiting contextual information (e.g. the relative distance between the robot and the objects to manipulate, explicit human commands, etc.). The generated primitives are simultaneously monitored by the attentional system, which relates them to the associated task structure exploiting top-down (task-based) and bottom-up (stimuli-driven) attentional mechanisms. These mechanisms enable also a natural interaction of the robot with the teacher, which can exploit attention manipulation (object and verbal cueing) to facilitate the learning process (Nicolescu and Mataric 2003). Notice also that in the proposed framework, action segmentation, annotation, and (task-based) contextual interpretation are one-shot and automatic, hence they do not require any manual post-processing of the collected data.

In summary, our contributions in this paper are the following:

- We present a framework that combines the benefits of kinesthetic teaching and attentional supervision to allow natural teaching by demonstration and flexible/collaborative execution of structured tasks.
- We propose an approach to action segmentation and annotation that simultaneously associates the generated segments to the task structure during a one-shot kinesthetic demonstration.
- We demonstrate the overall system providing empirical results to show the effectiveness of proposed approach in task teaching and execution.

The source code of the entire system—fully integrated with the Robot Operating System (ROS)—can be downloaded from https://github.com/matteosaveriano/task-teaching-and-supervision.

The rest of the paper is organized as follows. Section 2 presents and discusses related work. The proposed architecture for multimodal teaching/execution is detailed in Sect. 3. Section 4 describes how structured tasks are learned and executed using the proposed architecture. Experiments in a real word scenario are presented in Sect. 5. Finally, Sect. 6 states the conclusions and proposes further extensions.

2 Related work

Kinesthetic teaching is a natural and intuitive way to teach elementary robotic motions (Lee and Ott 2011; Saveriano et al. 2015). The goal of kinesthetic teaching is to physically guide the robot to show the desired behavior. In this setting, collected demonstrations are used to learn and reproduce the elementary motions. Current approaches for kinesthetic teaching are mainly concerned with learning and reproducing basic motion primitives, while our goal in this paper is to learn how to execute structured cooperative tasks from kinesthetic demonstrations. The works by Kulić et al. (2012) and Takano et al. (2016) focus on segmenting demonstrated movements in order to create a dictionary of basic motions, which can then be combined to generate more complex behaviors. These algorithms are effective in segmenting motion data into basic primitives, but they do not address the problem of extracting the associated execution constraints (e.g. an object firstly has to be grasped and then placed) from demonstrations. In contrast, our approach allows us to generate execution constraints (preconditions, postconditions, etc.) during the demonstration; these constraints are needed to monitor and flexible execute the learned structured tasks. Moreover, differently from Kulić et al. (2012) and Takano et al. (2016) we can learn goal-oriented activities involving interaction with the environment.

The problem of deciding the next motion to execute can also be treated as a classification problem. The approach in Pastor et al. (2012) uses nearest neighbor classification to determine the next action to execute. In Manschitz et al. (2015), a graph is used to represent transitions between elementary motions. A classifier associated with each node in

the graph determines when a transition occurs, i.e., when a motion is finished and the robot can execute the next one. These approaches permit to learn and reproduce complex robotic tasks from human demonstrations, however, differently from our approach, they do not consider the possibility of executing the learned tasks in a flexible manner or in cooperation with the human.

Alternative works have focused on the problem of learning high-level task representations from human observations (Argall et al. 2009). In Tenorth and Beetz (2013) and Zoliner et al. (2005), sequential constraints (like reaching an object and then grasping it) are used to determine a set of semantic rules that determine the sequence of actions to perform. Semantic rules are also used by Ramirez-Amaro et al. (2015) to learn, recognize and reproduce human activities from video sequences. Human activities are segmented following the popular approach by Fod et al. (2002), which suggests to segment an action stream looking at the zeros in the joint velocities. Velocities smaller than a given threshold value are considered as zero. The segmented activities are then matched with a set of pre-programmed motion primitives and executed by the robot. The problem of task learning from human activity observations is also faced by Dillmann (2004). Here, the human demonstration is used to generate a robot-independent task structure associated with robot-specific primitives. Aforementioned approaches are effective in learning the task structure from human observations, but motion primitives are assumed as given. Our approach is complementary, we assume that an abstract description of the task is available, while our goal is to learn both the motion primitives and their relations with the task structure. Other works in the context of human–robot collaboration are proposed in Magnanimo et al. (2014) and Koppula and Saxena (2015). In this case, collaborative activities are recognized in order to infer the future human actions. Human action anticipation is used by the robot to generate the right response to the human behavior (Koppula and Saxena 2015) in so enhancing the collaboration. These approaches consider the robot as an assistant, which is unable to autonomously execute the task. In this respect, our system permits to learn and execute both autonomous and cooperative tasks.

We propose a framework that enables incremental task teaching and cooperative task execution at different levels of abstraction. Moreover, we are interested in natural and smooth human–robot interaction that supports cooperative task execution and incremental adaptation. In this respect, related to our work, in Nicolescu and Mataric (2003) the teacher can use simple verbal cues to facilitate the learning process. In particular, the authors propose explicit verbal instructions to bias the learner's attention to relevant aspects of the demonstration, but an attentional framework is not deployed. Differently from this approach, we propose to deploy a supervisory attentional system that enables more complex attention-base interaction (verbal, non-verbal, explicit, implicit, etc.) during both the teaching and the execution phase. Social attentional mechanisms for non-verbal task teaching are proposed and investigated by Breazeal and Berlin (2008). In this case, the authors mainly focus on visual attention and gaze direction. In particular, they show the effectiveness of spatial scaffolding cues during interactive task demonstration. Visual attention mechanisms for robot learning are also proposed by Nagai (2009), Borji et al. (2010) and Belardinelli et al. (2007). In contrast to these works, we focus on executive attention and cognitive control mechanisms supporting kinesthetic task teaching. Supervisory attentional frameworks for robotic system have been proposed (Kawamura et al. 2007) considering also cooperative tasks execution (Caccavale and Finzi 2016; Caccavale et al. 2016), but not in a learning by demonstration context.

3 System architecture

The overall system architecture is depicted in Fig. 1. The human can interact with the robot in a multimodal manner with gestures, speech, and physical guidance during both task execution and kinesthetic teaching sessions. An attentional system supervises both the human and the robot activities (*Attentional Behavior-based System*) and manages high-level tasks monitoring and execution (*Attentional Executive System*). On the other hand, the *Robot Manager* is responsible for low-level task supervision, execution and learning. These components are detailed below.

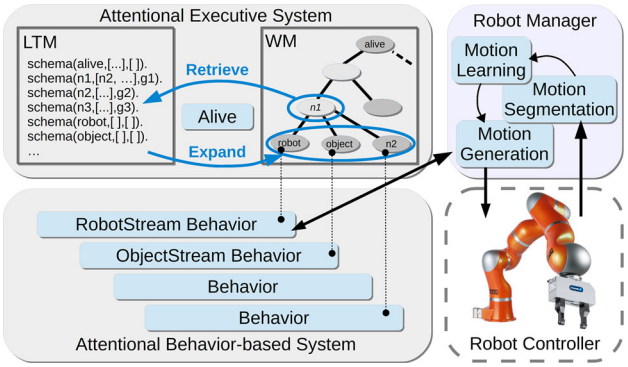


Fig. 1 The overall architecture for teaching and execution. The attentional system supervises task execution and learning, while the *Robot Manager* enables segmentation of the robot activities (*Motion Segmentation*), kinesthetic teaching, primitive action learning (*Motion Learning*) and execution (*Motion Generation*). The attentional system manages the execution of high-level tasks (*Attentional Executive System*) and low-level sensorimotor processes (*Attentional Behavior-based System*). The communication between the *Robot Manager* and the attentional system is managed by the *RobotStream* (robot motion data) and *ObjectStream* (perceived data from the RM to the attentional system)

3.1 Robot manager

The Robot Manager (RM) handles low-level aspects of the human–robot interaction and it is responsible for a correct task execution. In particular, the RM is responsible for: (i) smooth transition between teaching and execution modes; (ii) segmentation of the human demonstration into basic actions; (iii) scene monitoring (objects classification and tracking); and (iv) robot state monitoring (robot-object distance, motion primitives learned or executed). Task teaching is performed by means of kinesthetic teaching (Lee and Ott 2011). In this work, we use the gravity compensation control to make the robot ideally weightless for an easy and safe physical guidance. High level tasks are segmented into a set of point-to-point motions (reaching and manipulating objects). Segmented data are compactly represented as stable dynamical systems (DS), that we call motion primitives. The learned motion primitives are used to generate motor commands in the execution phase. Notice that stable DS are well-suited for point-to-point motion generation since they are guaranteed to converge towards a given target, and they can rapidly adapt to external perturbations, like changes in the initial/target location and unforeseen obstacles (Saveriano and Lee 2013, 2014). Learned DS generate the reference trajectories that the robot tracks using a Cartesian impedance control with high stiffness gains (2000 N/m for the position and 200 Nm/rad for the orientation). Hence, the RM has two control modes: a gravity compensation control for the teaching phase, and a Cartesian impedance control for the execution. These two control modes are enough to teach and execute structured tasks. Another control mode, such as the controller proposed by Lee and Ott (2011), can be eventually added to the RM in order to allow the task refinement during the execution.

3.2 Attentional system

The attentional system provides the cognitive control mechanisms needed to flexibly orchestrate the execution of complex tasks and to monitor the human activities. Following a supervisory attentional system (SAS) approach (Norman and Shallice 1986; Cooper and Shallice 2006), we propose a framework where interactive action execution and learning are supported by attentional mechanisms. In a SAS framework, the executive control depends on two main mechanisms: contention scheduling and supervisory attention. The first one allows to reactively select and regulate routinized activities depending on bottom-up perceptual stimuli and internal drives; the second one is a higher-level process that drives the system towards task-oriented behaviors through attentional regulations. In our human–robot interaction setting, the attentional system exploits hierarchical task representations to supervise and regulate the robot actions, while interacting with the human.

More specifically, we rely on the system proposed by Caccavale and Finzi (2015) and Caccavale and Finzi (2016). This framework is endowed with a Long Term Memory (LTM) and a Working Memory (WM) (see Attentional Executive System in Fig. 1). These components are detailed below.

Long term memory The LTM contains the procedural knowledge available to the system, that is, the actual robotic behavioral repertory that includes the abstract descriptions of the tasks the robotic system can perform [an instance can be found in Eq. (4)]. More specifically, each task is hierarchically defined in the LTM by a set of predicates of the form **schema**(m, l, p), where m is the name of the task, l is a list of m_i subtasks associated with enabling conditions r_i (releasers), i.e. $l{=}\langle(m_1, r_1), \ldots, (m_n, r_n)\rangle$, while p represents a post-condition used to check the accomplishment of the task. Notice that these task definitions are exploited to be retrieved, allocated, and instantiated in the WM for execution. For this purpose, we introduce a special process, called *alive* that continuously updates the WM by allocating and deallocating a hierarchy of behaviors that implements the corresponding task schemata in the LTM. For instance, in Fig. 2, the $add(Obj)$ schema is retrieved by *alive* (t_1 step in Fig. 2) and then instantiated and allocated in the WM as the behavior $add(water)$, which is ready for the execution (t_2 step in Fig. 2).

Working memory The Working Memory (WM) permits to temporarily maintain and manipulate the information needed to execute task-oriented activities; it represents the executive state of the system and collects the processes recruited and instantiated for task execution. In our framework, these processes are represented by an annotated rooted tree

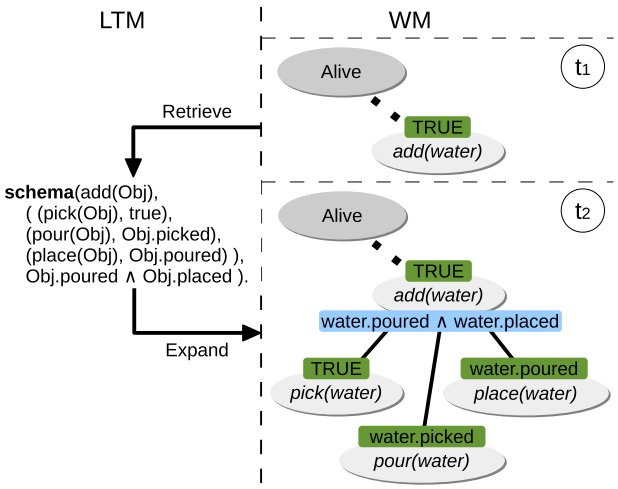


Fig. 2 Representation of the WM expansion process managed by *alive*. When the new node $add(water)$ is allocated in WM the associated schema is selected from LTM (retrieve phase) and exploited to decompose the node in WM by adding and instantiating the new abstract or concrete sub-nodes mentioned in the schema (expand phase)

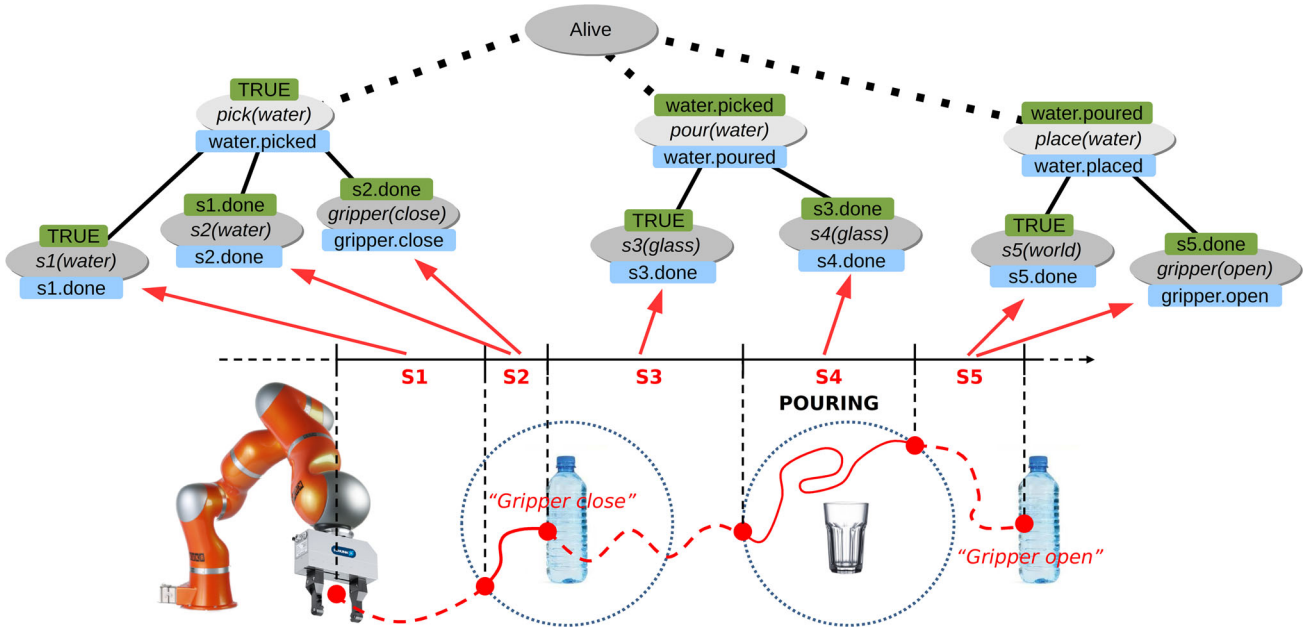


Fig. 3 Action segmentation and hierarchical task decomposition during kinesthetic teaching of a pouring task. The robot has to pick-up the bottle (*pick(water)*), reach the glass, pour the water (*pour(water)*) and place the bottle (*place(water)*). The Robot Manager (down) performs action segmentation ($S_1, S_2, \ldots, S_5$) and learns the associated motion primitives, while the attentional system (up) connects the generated segments and primitives to the task structure ($s1(water)$, $s2(water)$, and $gripper(close)$ connected to $pick(water)$; $s3(glass)$ and $s4(glass)$ connected to $pour(water)$, etc.). The green and blue labels represent, respectively, releasers and post-conditions

$T = (r, B, E)$, whose nodes in B represent allocated processes/behaviors, the root $r \in B$ is the *alive* process, which bootstraps and manages the WM, while the edges E represent parental relations among sub-processes/sub-behaviors. These nodes can be either *concrete*, representing real sensorimotor processes, or *abstract*, which are for complex behaviors to be hierarchically decomposed according with the associated schemata in LTM. An example of the hierarchical behaviors generated for the execution of the $add(water)$ is illustrated Fig. 2 (bottom, right), while the ones for the pouring task can be found in Fig. 3. In these illustrations, for each node, green labels represent $releasers$ (enabling-conditions), while blue labels are for a post-condition ($goal$ conditions) exploited to monitor the accomplishment of goal-oriented activities. More specifically, each node b in WM is represented by a 5-tuple $(m_b, q_b, p_b, x_b, \mu_b)$, where m_b is the name of the allocated task, q_b and p_b represent the releaser and post-condition respectively, x_b is the set of sub-behaviors generated by m_b, while μ_b is a value that represents the current attentional state of the behavior (see magnitude below). Here, m_b, q_b, p_b, x_b are instances of the associated schema in the LTM. Indeed, each node b in WM, is generated by the *alive* process that allocates a **schema**(m, l, p) with a variable binding that instantiates m in m_b, p in p_b and the list l in the list of sub-behaviors x_b and the associated releasers. For instance, in Fig. 3, the $pour(Obj)$ task is instantiated by the argument $water$ for the variable Obj, hence task, releaser, post-condition and sub-behaviors are also instantiated by $water$. This process is analogous to the one introduced for HTN planning (Nau et al. 2003). Each concrete behavior accesses sensory data σ_b, affects control variables c_b and updates a set of state variables V representing the current state of the overall system. For instance, in Fig. 3, when the $water.picked$ boolean variable is set to true, the *pick* task is accomplished while the next *pour* task is enabled to be executed. In this case, the state variable $water.picked$ is updated by the concrete behavior *gripper(close)* when the grasped object is the water.

Attentional regulation In line with (Norman and Shallice 1986), we assume that each node in the WM is also endowed with an activation value regulated by attentional mechanisms. This value is affected by top-down and bottom-up attentional processes. In our framework, the activation value of concrete behaviors is a frequency that represents the resolution at which a sensorimotor process is monitored and controlled. More specifically, in concrete behaviors, the activation value is bottom-up regulated by a monitoring function $g(\sigma_b, \varepsilon_b) = \lambda_b$, which depends on behavior-specific stimuli σ_b and the behavioral state variables ε_b (subset of the state variables V). In this work, analogously to (Caccavale and Finzi 2016), we consider the distance of targets as an estimation of behavioral accessibility, hence σ_b is here directly associated to the minimal distance of the target for the behavior. In particular, we assume that the activation period $\lambda_b \in [\lambda^{min}, \lambda^{max}]$ is bottom-up regulated by the following

saturating (and increasing) linear function:

$$\lambda_b = g(\sigma_b, \varepsilon_b) = \begin{cases} \lambda^{min} & \text{if} \quad \sigma \leq r^{min} \\ \lambda^{max} & \text{if} \quad \sigma \geq r^{max} \\ \alpha \cdot \sigma + \beta & \text{otherwise} \end{cases} \tag{1}$$

specified by two parameters r^{min} and r^{max}, with $\alpha = (\lambda^{max} - \lambda^{min})/(r^{max} - r^{min})$ and $\beta = \lambda^{min} - \alpha \cdot r^{min}$ used to describe the linear increase of g for σ in the interval $[r^{min}, r^{max}]$. Notice that in this formulation, we assume a direct bottom-up influence of the σ_b stimuli, hence ε_b here is not exploited, however, more complex regulations can be introduced (see for example Broqure et al. 2014). This bottom-up regulation is then top-down modulated by a magnitude value μ_b that summarizes the overall influence of the WM on the behavioral attentional state. In concrete behaviors, top-down and bottom-up influences are then combined in an *emphasis* value $e_b = \mu_b/\lambda_b$ representing the actual activation frequency for the behavior b. The absence of a top-down influence is represented by $\mu_b = 1$. Whenever a magnitude change happens for a node in the WM, this update is inherited by all its descendants. In addition, we assume that when a behavior is accomplished, the magnitude of the parent node is increased by a constant value k, which is then propagated towards its active successors. In this setting, the magnitude of a generic behavior is then given by $\mu_b = \mu_f + kn$, where μ_f is the parent magnitude and n is the number of accomplished sub-behaviors. This mechanism facilitates active behaviors representing the continuation of accomplished subtasks, in so inducing a smooth drive towards task accomplishment with an associated reduction of task switching.

Conflict regulation The behavioral activation level is then exploited to regulate behavioral competitions and conflicts. Indeed, multiple tasks can be allocated in the WM at the same time, therefore several behaviors can compete for the execution generating conflicts and impasses (Botvinick et al. 2001). Contentions among alternative concrete behaviors are solved exploiting the attentional activation: following a winner-takes-all approach, the behaviors associated with the higher emphasis are selected with the exclusive access to mutually exclusive resources. More specifically, in our framework this mechanism occurs when multiple concrete behaviors simultaneously try to access and update a mutually exclusive control variable c. In this case, given the set $C(c)$ of competitors for the c variable, the system selects the most emphasized concrete behavior:

$$b_{win} = \underset{b \in C(c)}{\text{argmax}}\, e_b. \tag{2}$$

The selected behavior b_{win} can then modify the variable c with the exclusive access. As already stated above, once a behavior is accomplished, the upward propagation of magnitude described above permits to facilitate task-related behaviors in conflicts, in so orienting the system towards task continuation and accomplishment. This task-oriented facilitation mechanism can be enhanced or reduced by tuning the k parameter.

4 Kinesthetic teaching of structured tasks

The proposed framework supports human–robot interaction during both task demonstration and task execution. In order to enable natural interaction and incremental task learning, the system can anytime switch between teaching and execution. The teaching phase can start from the human or the robot initiative. In the first case, the human can explicitly switch to a demonstration session through a command (either vocal and/or gestural) and directly show the execution of a task. Otherwise, in the second case, the robot can wait for the human assistance when not able to execute an activity. This happens when a task under execution is not linked to concrete sensorimotor behaviors. In this case, the system waits for the user guidance in order to learn how to perform the missing subtasks.

During the teaching phase, the human can physically guide the robot in order to demonstrate the correct execution of the task. This kinesthetic teaching session is supervised by the attentional system, which has to connect the segmented training motions to the related tasks and subtasks. The attentional system tracks and monitors both the human and the robot task execution. This way, the low-level robotic actions taught by the user through kinesthetic teaching are labeled by the higher level tasks/subtasks managed by the attentional system. For instance, Fig. 3 illustrates the action segmentation of a water-pouring task along with the associated hierarchical task decomposition. During the teaching mode, the RM provides action segmentation and motion primitive learning, while the attentional system monitors the subtasks to be fulfilled (*pick(water)*, *pour(water)* and *place(water)*). When a new segment is recognized by the system (S_1, S_2, ..., S_5), a new node in the tree is generated ($S_1(water)$, $S_2(water)$, ..., $S_5(word)$, $gripper(open)$) and linked to the most emphasized subtask.

During the demonstration, the human can also facilitate the learning process by providing additional verbal cues to the robot (such as object handling, vocal commands, etc.). These cues can affect the attentional state of the robot, hence they can influence task/subtask contentions and segments associations. Moreover, the human can always inspect the result of a training session by invoking the repetition of learned tasks and subtasks. Indeed, if the learned activities are not satisfactory, task demonstrations can be repeated.

Notice that, in a collaborative task execution setting, specific subtasks can be directly assigned to the human or executed either by the human or the robot. For instance, a pouring task may also include an explicit *open(bottle)* subtask that requires human manipulation. If the *open(bottle)* subtask is executed by the human during the teaching, then it will be assigned to the human. Hence, during the execution phase, the robotic system will wait for the human assistance in order to successfully execute the task. The overall learning process is further detailed in the rest of the section.

4.1 Action segmentation

The demonstrated task has to be segmented into elementary movements. An effective segmentation strategy has to be fast enough to work in real-time, consistent across different demonstrations of the same task, and complete, meaning that the generated segments represent the entire task.

In this work, we propose a simple and effective segmentation mechanism, which is based on object proximity and explicit human commands. Following the approach by Wächter et al. (2013), each object in the environment is associated with a proximity area, i.e., a sphere of radius r around each object (we set $r = 120\,\text{mm}$). When the end-effector of the robot enters or leaves the proximity area of an object, a new segment is generated. Analogously, when a human command (e.g. open/close gripper) is executed a new low-level action is created. The attentional system can then automatically connect the generated action segments to the task structure (see Fig. 3), while the Robot Manager uses the robot's trajectories to learn a motion primitive for each action segment. Human commands are also included in the task structure, in order to control the gripper when the robot executes the task. We distinguish between two classes of actions:

- Near-Object Action (NOA): the action is segmented inside the proximity area of an object. In this case, we exploit Dynamic Movement Primitives (DMP) (Ijspeert et al. 2013) to compute a robust approximation of the observed trajectory in order to accurately reproduce the motion.
- Far-Object Action (FOA): the action is segmented out of the proximity area of any object. In this case, only the end-point of the observed trajectory is considered. The action is then reproduced with a point-to-point motion, generated with a linear dynamical system. This way, the robot reaches the proximity area always with the same pose, and executes NOA actions starting from a state which is consistent with the demonstration (see Sect. 4.2).

The proposed segmentation mechanism allows the system to reproduce complex actions involving two or more objects.

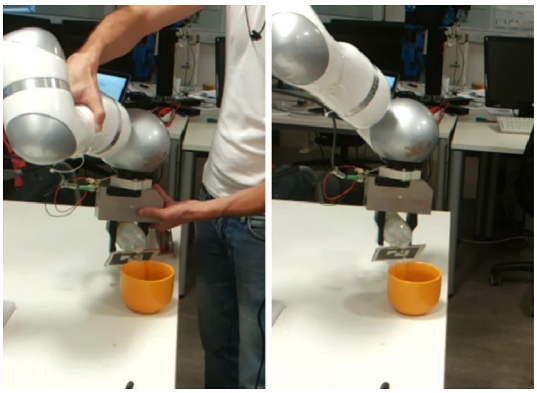


Fig. 4 Teaching and execution of the *pouring* action (NOA). In the teaching phase (left) the user drives the robot near the cup and pours water, while in the execution phase (right) the robot reproduces the movement

For example, the pouring action (NOA) illustrated in Fig. 4 has been trained with high accuracy and associated with the $pour(water)$ primitive behavior within the abstract task of pouring.

The segmentation strategy requires the robot-object distances. Possible failures may occur if the objects are not properly tracked, for instance if the teacher hides the object to manipulate during the teaching. Failures may also occur if the robot enters in the proximity area of multiple objects simultaneously and each of these can be associated with the generated segment. This occurrence can be prevented by properly choosing the radius of the proximity area r. Finally, the segmentation strategy may generate unnecessary segments if the teacher guides the robot inside/outside the proximity area of different objects without grasping them. Even in this case, the learned task can be correctly executed although the robot performs unnecessary motions like the human demonstrator. This undesirable behavior can be prevented by instructing the teacher to directly guide the robot towards the object to use.

4.2 Learning motion primitives

The described segmentation approach transforms the human demonstration into a set of basic actions with associated target poses. In order to reproduce the actions on a real robot, we encode the segmented data into stable dynamical systems and refer to these systems as motion primitives. In this work, motion primitives are learned from demonstrations using Dynamic Movement Primitives (Ijspeert et al. 2013). DMP encode a motion primitive into a second order, non-linear dynamical system (Park et al. 2008)

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{3a}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{K}(\boldsymbol{g} - \boldsymbol{p}) - \boldsymbol{D}\boldsymbol{v} - \boldsymbol{K}(\boldsymbol{g} - \boldsymbol{p}_0)s + \boldsymbol{K}\boldsymbol{f}(s) \tag{3b}$$

$$\dot{s} = -\gamma s \tag{3c}$$

where $\boldsymbol{p}$ is the robot position/orientation, $\boldsymbol{v}$ its linear/angular velocity, $\boldsymbol{g}$ the desired (goal) position, $\boldsymbol{p}_0$ is the initial position of the robot, $\boldsymbol{K}$ and $\boldsymbol{D}$ are positive definite gain matrices. The nonlinear forcing term $\boldsymbol{f}(\cdot)$ reshapes the linear dynamics to follow the demonstrated trajectory. The forcing term is deactivated by the clock signal $s \rightarrow 0$ to guarantee convergence to $\boldsymbol{g}$. The scalar gain $\gamma > 0$ determines how fast $s \rightarrow 0$. Recalling that, in practice, $s = 0$ after $5/\gamma$ seconds, we set each γ equal to 5 over the duration of the NOA action.

DMP have several properties which make them well-suited for our approach. First, the forcing term is learned using a single demonstration. Hence, the user does not have to repeat the same action multiple times. Second, DMP guarantee convergence towards the target from any initial position. Third, stable dynamical systems are robust to changes in the target position and can be eventually combined with reactive collision avoidance strategies to generate converging and collision-free paths (Saveriano et al. 2017).

The full DMP structure in Eq. (3a)–(3c) is exploited to learn and retrieve Near-Object-Actions, e.g., the nonlinear part of motion of a pouring action. For Far-Object-Actions, instead, we only consider the linear part of the DMP and neglect the terms $-\boldsymbol{K}(\boldsymbol{g} - \boldsymbol{p}_0)s + \boldsymbol{K}\boldsymbol{f}(s)$ in Eq. (3c). This way, the robot executes complex actions always from the same initial state, preventing the problem of the excessive magnification of trajectories generated from different initial states (Ijspeert et al. 2013). In order to reproduce the demonstrated motion when the objects are placed at different locations, goal poses are referred to a frame attached to the specific object. At run time, the current goal is first referred to the robot frame (located at the base of the robot) and then passed together with the robot pose to the dynamical system that generates the motion. It is worth noticing that the combination of DMP and the proposed segmentation strategy permits to learn motion primitives without any off-line data processing. In particular, the target position for each action, as well as the demonstrated trajectory, are automatically provided by the segmentation approach and then used to learn the action, without further human intervention or data post-processing.

4.3 Task learning

In this section, we illustrate how the generated segments are connected to high-level task structures. This process is managed by the attentional system while monitoring the human demonstration. We assume that a description of the task structure is already provided in the LTM (see Eq. (4) for an example), while the learning problem is to produce an updated LTM where all the associations between subtasks and segments are represented. We call *open subtasks* the schemata of LTM that represent concrete behaviors [as $subtask(take, Obj)$ in Eq. (4)] but are not associated to segments (i.e. that cannot be executed by the robot). In this setting, starting from an initial LTM^0 that contains a set of n open subtasks, our learning process produces an updated LTM^l where the open subtasks in LTM^0 are further decomposed by the m generated segments, each associated with a motion primitive in the Robot Manager. When a teaching phase starts, the abstract behavior representing the task to be demonstrated is allocated in the WM and then hierarchically decomposed by the *alive* process (see Sect. 3.2). This way, a behavioral tree T_{task} is generated in the WM that contains a set of open subtasks $O = \{sub_1, \ldots, sub_n\}$ to be linked to the segments produced by the RM. In order to describe this process, we consider the demonstration of a water-pouring task (see Fig. 5). This task is hierarchically decomposed in the *take-water* and *pour-water* subtasks (frame $t1$), which are denoted in the LTM by the following schemata:

$$\begin{aligned}
&\textbf{schema}(add(Obj),\\
&\quad \langle (subtask(take, Obj), hand.free),\\
&\quad (subtask(pour, Obj), Obj.taken) \rangle,\\
&\quad Obj.used)\\
&\textbf{schema}(subtask(take, Obj), \langle\,\rangle, Obj.taken)\\
&\textbf{schema}(subtask(pour, Obj), \langle\,\rangle, Obj.used)
\end{aligned} \tag{4}$$

Here, the pick-and-pour task can be instantiated with different objects (Obj). The subtask $take$ is enabled when the hand is free (releaser $hand.free$) and associated with the $Obj.taken$ post-condition, while the $pour$ subtask is enabled when the object is taken (releaser $Obj.taken$) and related to the $Obj.used$ post-condition.

In order to be executed, the $add(Obj)$ has to be instantiated and allocated in the WM. However, the two subtasks ($pour$ and $take$) are not linked to concrete sensorimotor processes, which are automatically generated during the kinesthetic teaching process. Since each subtask is implemented by a concrete WM node, it is associated with an activation level, which is (bottom-up) affected by the proximity of the objects in the scene [see Eq. (1)] and (top-down) modulated by the overall tasks allocated and enabled in the WM. Therefore, during the human demonstration, the attentional system enhances the activations of the subtasks which are accessible (i.e. closer to the associated target objects) and task relevant (i.e. top-down stimulated through the task structure). These activation values are then used to link the concrete subtasks to the generated segments (and to the associated motion primitives in the Robot Manager), as described in Algorithm 1.

In particular, when a new segment is created by the Robot Manager (line 2), all the enabled open sub-tasks of

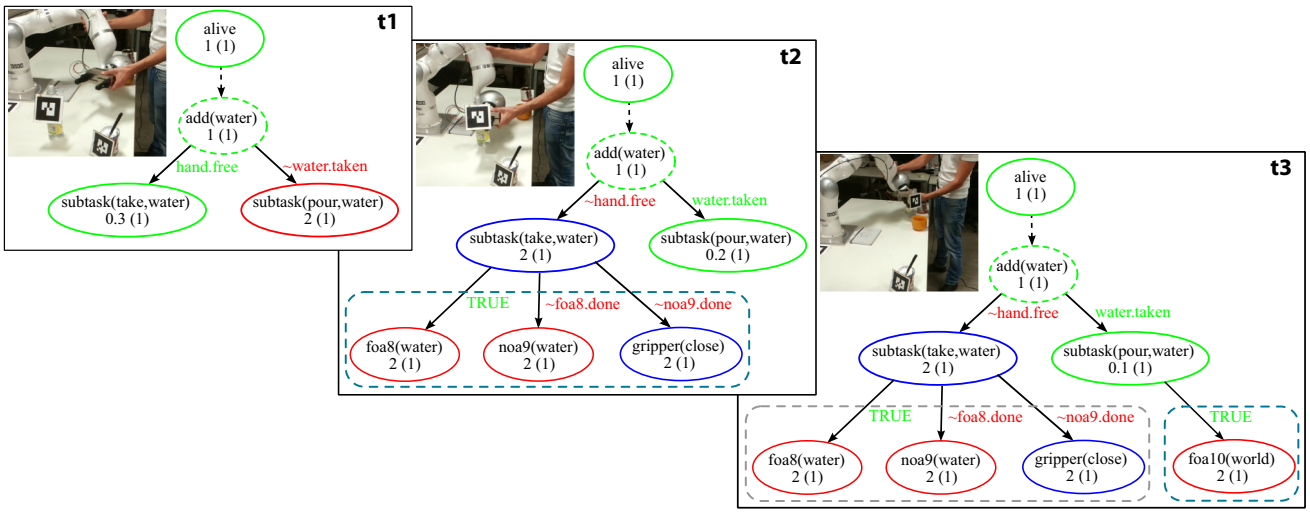


Fig. 5 Representation of the WM update during the *pouring* task. The system starts from a simple structure for the $add(water)$ task ($t1$). During the user demonstration new segments are added to the *take-water* subtask ($t2$) along with their releaser (labels on the arrows). When the new *pour-water* subtask is selected ($t3$) a new FOA is linked with a *true* releaser. Here, green and red ovals represent enabled and disabled behaviors (satisfied and unsatisfied releasers), blue ovals are for accomplished behaviors (satisfied postconditions), dotted ovals are for abstract behaviors. For each behavioral node, the values outside/inside brackets are for the inverse of emphasis $1/e_b$ (i.e. activation period) and magnitude μ_b (top-down influence) respectively

Algorithm 1 Allocation of a new segment in the task hierarchy.

1: **while** true **do**
2: **if** a new segment seg from RM exists **then**
3: get winner $sub_{win} \leftarrow \underset{sub_i \in O}{\text{argmax}}\, e_{sub_i}$
4: **if** sub_{win} is a new subtask and seg is FOA **then**
5: set releaser $q \leftarrow true$
6: **else**
7: set releaser $q \leftarrow p_{prev}$
8: **end if**
9: set post-condition $p \leftarrow seg.done$
10: add behavior $(seg, q, p, \emptyset, \mu)$ to sub_{win} in WM
11: add new **schema**$(seg, \langle\,\rangle, p)$ in LTM
12: add $\langle seg, q \rangle$ to sub-task list of sub_{win} in LTM
13: **end if**
14: **end while**

the WM compete to add the segment as a new child node (line 3). In our framework, this competition is managed by a winner-takes-all approach where the most emphasized subtask acquires the new segment [see Eq. (2)]. When a new segment is generated, we have to define its releaser and post-condition (lines 4–9). The releaser is always enabled ($true$) if a FOA segment is added to a subtask with no other child nodes (lines 4, 5). Otherwise, the execution of the segments has to be chained, hence the post-conditions of the previous segment is exploited as the releaser of the current one (lines 6, 7). The post-condition of each segment is then set to $seg.done$ (line 9). This symbolic post-condition is associated with a sub-symbolic constraint used to check whether the associated motion has been actually executed by the robot. If a segment is associated with a motion primitive, sub-symbolic constraints are directly provided by the RM (e.g. target zones for the end-effector). Instead, predefined commands (e.g. open/close gripper) are directly associated with predefined sub-symbolic conditions (e.g. constraints on the gripper state). When a new segment is generated, a corresponding new behavior is allocated in WM as a child node of the winning open subtask (line 10) (see also Fig. 5, frame $t2$, where the linked segments are indicated by the dotted line) and the LTM is updated accordingly (lines 11, 12). Notice that the chaining constraint is introduced for segments belonging to the same subtask or if the subtask starts with a NOA segment, which requires the fixed starting point provided by the previous segment. On the other hand, if the new subtask starts with a FOA segment, we can keep it decoupled from the previous subtask, in so enabling reusability and flexible execution of the associated subtask. Indeed, at the execution time, all the enabled segments of the WM compete to acquire the control of the robotic platform. Hence, multiple independent tasks and subtasks can be executed in a flexible manner, diverging from the sequence illustrated during the demonstration. The overall method is exemplified in Fig. 5. Once the user drives the robot towards the bottle and grasps it, the system generates 3 new segments: $foa8(water)$ when the robot enters the objects area, $noa9(water)$ and *gripper(close)* when the bottle is reached and grasped. These segments are attached to the *take-water* subtask, which is the only one available in this context, while the associated enabling conditions are needed to ensure the

sequence of the segments (i.e. $noa9$ executed after $foa8$, and $gripper(close)$ after $noa9$). Afterwards, when the robot is driven towards the cup, the novel segment $foa10(world)$ is generated and linked to the *pour-water* subtask which becomes active after the bottle grasping. In this case, the motion between the bottle and the cup represents a FOA and the new generated segment is associated with a $true$ releaser (i.e. always enabled).

5 Experimental results

In this section, we propose some experiments to show that the proposed approach can be effectively applied for (i) incremental learning and execution of structured tasks, (ii) execution of learned tasks in cooperation with the human, and (iii) reuse of acquired knowledge in different contexts. To this end, we consider two typical tasks of a kitchen scenario; namely prepare coffee and prepare tea. The robot is a KUKA LWR IV+ (Bischoff et al. 2010), equipped with a WSG50 2-fingers gripper. As illustrated in Fig. 6, objects are recognized and tracked using markers and a RGB-D camera as in Garrido-Jurado et al. (2014). The marker close to the robot base is used to compute the coordinate transformation between the camera frame and the robot base. Due to possible marker occlusions during the teaching, we estimate the robot-camera transformation and the pose of the cup at the beginning of each experiment and keep them constant during the execution. All the other objects, instead, are continuously tracked at 30 Hz. The user initiates a kinesthetic teaching session via the speech command *teach*. The teaching session is terminated by the speech command *done*. The user can interrupt/restart the execution of a learned task using the speech commands *stop/repeat*. Graduate students in robotics and automation participated to the experiments as teachers. The parameters used in our system are listed in Table 1.

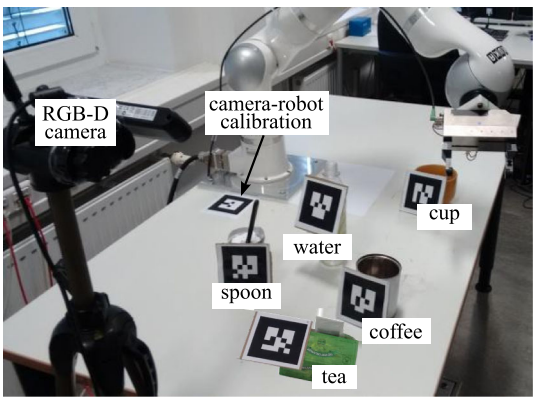


Fig. 6 Experimental setup

Table 1 Parameters used in the experimental evaluation

Parameter	Meaning	Value
Robot Manager		
r	Radius of the proximity area	0.12 m
$\mathbf{K} = \text{diag}(\mathbf{k_1}, \ldots, \mathbf{k_6})$	DMP stiffness gains	$k_i = 70.0$
$\boldsymbol{D} = \text{diag}(d_1, \ldots, d_6)$	DMP damping gains	$d_i = 2\sqrt{k_i}$
Attentional System		
λ^{max}	Max behavior period	1 s
λ^{min}	Min behavior period	0.1 s
r^{max}	Max object distance	2 m
r^{min}	Min object distance	0 m
k	Magnitude increment	1

5.1 Pouring a drink

In the first experiment we teach the robot how to pour water in a cup. The pouring task consists of two subtasks: *take-water* and *pour-water* (see Fig. 5, t1). During the teaching process, the teacher has to simply guide the robot towards the task execution, providing sparse speech commands (*open*/*close*) to control the gripper. In Fig. 7, we illustrate the WM state after a one-shot teaching session. At the end of the demonstration, we can find nine generated segments, which are linked to the associated subtasks. These new elements are also associated with preconditions, effects, and activation values. As detailed in Algorithm 1, these generated elements are also stored in the LTM to be re-used in future scenarios. Once learned, the task can be executed. In this situation, the attentional system first selects the subtask *take-water*, which is enabled when the robot has no object in its gripper (*hand.free*). The segments linked to the same subtask are

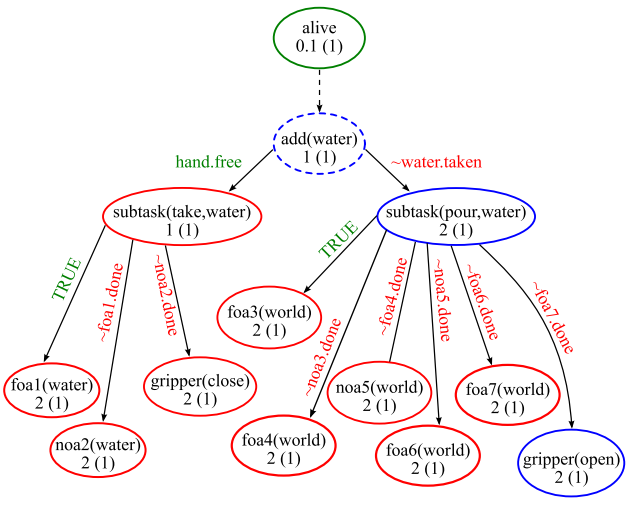


Fig. 7 The WM state after the pouring task demonstration. Nine generated segments are linked to the associated subtasks

Table 2 Results for ten repetitions of the pouring a drink task

take-water	*pour-water*	*add(water)*
Teaching time (s) (mean ± std)		
20.1 ± 1.2	30.3 ± 0.8	50.4 ± 2.0
Execution time (s) (mean ± std)		
31.0 ± 1.5	46.5 ± 0.8	77.5 ± 2.3
Success rate		
1	1	1

executed in the order shown during the demonstration. For example, in order to perform the *take-water* subtask, the robot executes *foa1(water)*, *noa2(water)* and then *gripper(close)*.

In order to quantitatively evaluate the effectiveness of the proposed approach, we measured teaching and execution times over ten repetitions of the task. Moreover, in order to show the robustness of our approach with respect to the initial conditions, we performed ten repetitions of the task with the bottle placed at random positions, measuring the *success rate* as the number of correct executions over the total executions. A trial is considered successful if the robot grasps the bottle and pours the water within the cup.

As shown in Table 2, teaching this relatively complex task requires approximately 50 s. Moreover, the task was successfully executed in all the ten trials (success rate equal to 1). These results show that the proposed framework allows to transfer novel skills to a robotic device in a quite fast, natural, and effective manner. Indeed, for each session, the task is illustrated through a one-shot kinesthetic teaching demonstration, associated with sporadic verbal commands only used to open or close the gripper. Notice also that the execution time for the pouring task (77.5 s, on average) appears here slightly longer than the time needed to demonstrate the task (50.4 s, on average). This slower execution does not depend on the attentional system, which can effectively monitor and select the robotic tasks and actions. Instead, it mainly depends on the convergence time of the dynamical systems used to generate motor commands (see Sect. 4.2). A possible way to reduce the execution time is to perform each action at a predefined speed ϵ, i.e., by generating a velocity command with $\dot{\boldsymbol{p}} = \epsilon \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}$ instead of Eq. (3a).

5.2 Prepare coffee: task learning and autonomous execution

This experiment shows how a complex, structured task is learned and executed using the proposed framework. We consider the task of preparing a coffee, in which the robot has to: (i) pour the water in the cup, (ii) add the coffee powder, and (iii) mix water and coffee powder with a spoon. Before learning, the WM only contains the three behaviors *add(water)*, *add(coffee)* and *use(spoon)* without any link to motion primitives, as illustrated in Fig. 8. The action primitives and segments are automatically added during the kinesthetic teaching and then used to reproduce the task. Note that the order of execution of *add(water)* and *add(coffee)* is not relevant for task learning and execution, therefore, they are both enabled when the task starts. In this case, task selection only depends on the attentional regulations. In Fig. 9, we show teaching and execution snapshots of *add(water)*, *add(coffee)* and *use(spoon)*, each associated with the WM state obtained at the end of a learning session. Here, the user can directly teach the overall *prepare coffee* task and then execute it, otherwise the task can be step by step demonstrated and executed (see the *prepare tea* task in Sect. 5.4).

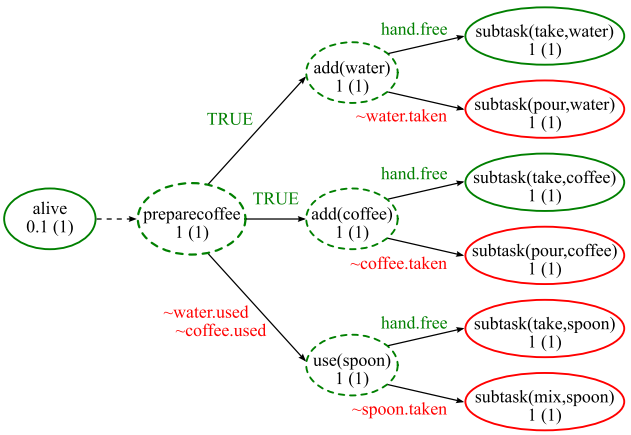


Fig. 8 The WM before learning how to prepare a coffee. The task *preparecoffee* has three child nodes, namely *add(water)*, *add(coffe)* and *use(spoon)*. *add(water)* and *add(coffe)* can be executed in any order (*true* releaser), while *use(spoon)* requires that both the water and the coffee powder are added. Initially, both *subtask(take,water)* and *subtask(take,coffee)* are enabled, hence they compete for the initial segments

Similarly to the previous experiment, we measured teaching and training time, as well as, the success rate over ten task repetitions (with objects randomly placed). Results in Table 3 show that, on average, teaching the prepare coffee task takes less than 3 min, while executing the task takes about 3.7 min. Analogously to the previous experiment, the longer execution time mainly depends on the convergence time of the dynamical systems. Table 3 also shows training and execution times for each subtask. Looking at these results, we notice that the time to grasp an object is almost independent on the particular item. This is because, in our setup, objects are relatively close and they are grasped in a similar manner. We also notice that *take-spoon* takes always longer than other take actions. The reason is that the subtask *use(spoon)* is always executed at the end, and the robot has to cover a bigger distance to reach the spoon. Moreover, Table 3 shows that the task execution has less variability than the teaching. This means that, despite the user intro-

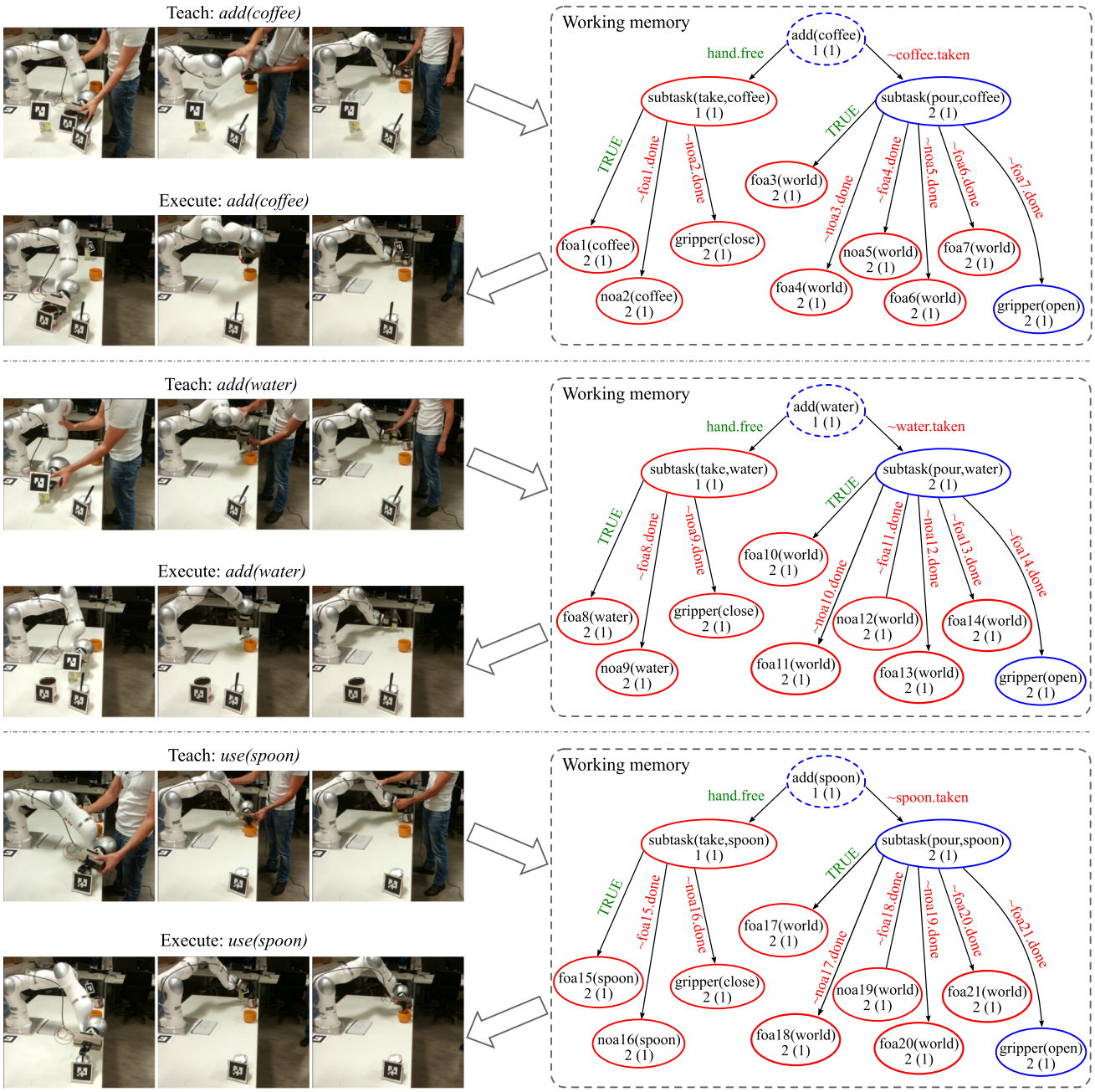


Fig. 9 The robot learns how to prepare a coffee. (Left panels) Snapshots of the kinesthetic teaching and autonomous task execution. (Right panels) Actions are automatically attached to the behaviors in the WM and used to reproduce the task

duces some variability across different demonstrations, the task execution time is relatively constrained. Several actions of the learned task are, in fact, linear point-to-point motions which are executed in similar times across different repetitions.

Also in this case, the task success rate is quite high (0.9) and only one failure occurs over ten trials. In the failed trial, the robot did not grasp the coffee jar sufficiently close to its center of mass, probably due to an error in the tracking system. Being the jar turned, the robot failed to add the coffee in the cup. Notice that, in the current implementation, we exploit a simple grasping strategy. A possible way to increase the robustness of the system is to use a multi-fingered robotic hand and perform a power grasp (Roa et al. 2012), or to exploit tactile sensing in order to detect and avoid the slipping (De Maria et al. 2015).

Table 3 Results for ten repetitions of the prepare coffee task

take-water	*pour-water*	*add(water)*	*take-coffee*	*pour-coffee*	*add(coffee)*	*take-spoon*	*mix-spoon*	*use(spoon)*	*prepareCoffee*
Teaching time (s) (mean ± std)									
20.0 ± 2.3	30.1 ± 1.4	50.1 ± 3.7	19.8 ± 2.6	37.2 ± 1.9	57.0 ± 4.5	21.3 ± 2.7	37.3 ± 1.9	58.6 ± 4.6	165.7 ± 12.8
Execution time (s) (mean ± std)									
28.4 ± 0.5	42.7 ± 0.3	71.1 ± 0.8	27.9 ± 0.7	47.7 ± 0.3	75.6 ± 1.0	29.8 ± 0.7	48.0 ± 0.4	77.8 ± 1.1	224.5 ± 2.9
Success rate									
1	1	1	1	0.9	0.9	1	1	1	0.9

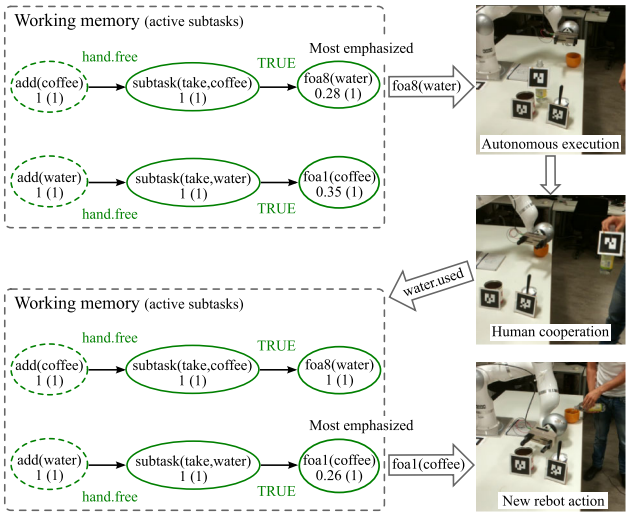


Fig. 10 Cooperative execution of the prepare coffee task. The user takes the bottle and pours the water while the robot is approaching the bottle. Notice that, before the human intervention the most emphasized action segment is *foa8(water)*. On the other hand, when the human performs the action, the robotic task execution is online adapted: the most emphasized action segment becomes *foa1(coffee)* and the robot takes the coffee jar

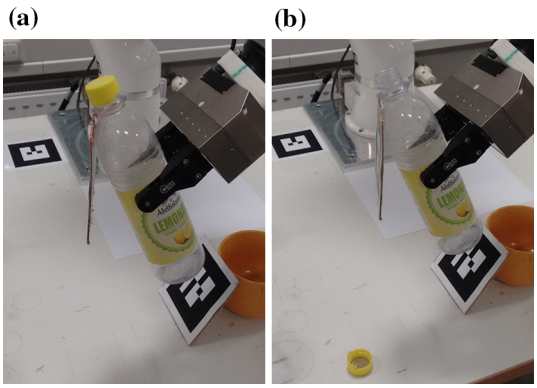


Fig. 11 **a** The bottle is closed and the robot will fail to pour the water. **b** Once the user has suspended the task and removed the cap, the robot can correctly execute it

5.3 Prepare coffee: cooperative task execution

The proposed framework permits a cooperative execution of the learned tasks. As a proof of concepts, we consider the coffee task described in the previous experiment and two cooperative scenarios. In the first case, the human helps the robot to fulfill the task by adding the water himself. To show the on-line capabilities of the attentional system, the user intentionally takes the bottle, while the robot is approaching it, (i.e., while it is executing the *foa8(water)* action in Fig. 10) and pours the water. In this situation, the system has to rapidly adapt task execution with respect to the human behavior. Since the water is not anymore available in the scene, the *add(water)* behavior becomes less attractive for the robot that starts to execute the *add(coffee)* (which is available and enabled in the WM). At the same time, the system can monitor the human behavior and assign the *add(water)* execution to the human. In this setting, for the sake of simplicity, the above assignment is explicitly communicated by the human through a vocal utterance. Notice, however, that more complex activity recognition methods can be deployed for the same purpose (Caccavale et al. 2014). We executed this cooperative task ten times, obtaining an average execution time of 149.8 ± 3.5 s. A comparison with the autonomous execution time in Table 3 allows us to conclude that the cooperative execution is beneficial in terms of execution time. In particular, we observe that the time needed for task adjustment does not have a significant impact on the total execution time.

Human–robot cooperation can be also exploited to overcome robot limitations. As a proof of concepts, we consider the case in Fig. 11a where the robot is pouring the water with the bottle closed. Even with a more sophisticated perception system able to recognize the cap, a single manipulator could not open the bottle and the pouring task would fail. In this case, the human intervention is essential to fulfill the task. In the proposed framework, during the execution phase, the user can suspend the task execution via the speech command *stop*, open the bottle (see Fig. 11b), and restart the execution (speech command *repeat*). Otherwise, we can explicitly introduce in the task structure a subtask *open(Obj)* which is directly assigned to the human manipulation and left unlinked

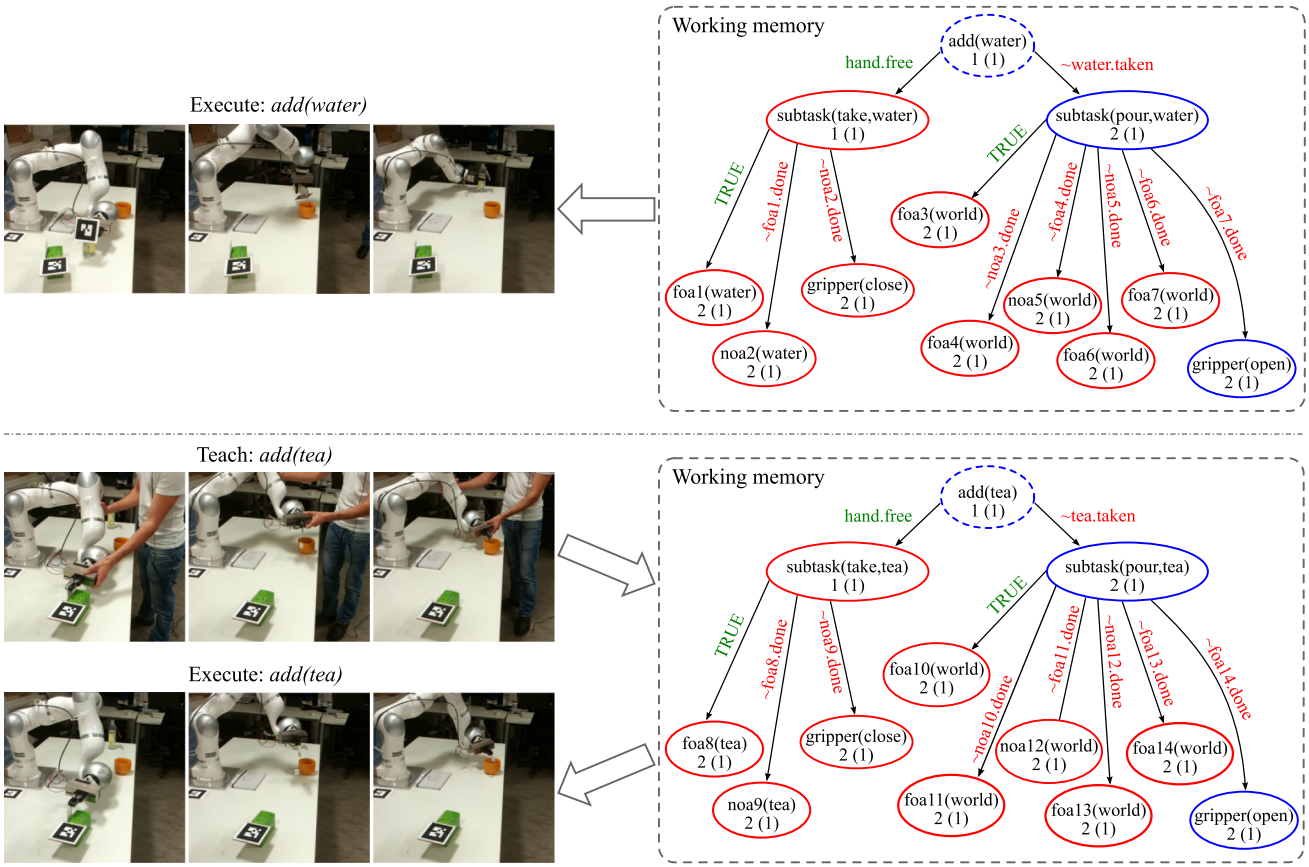


Fig. 12 The robot learns how to prepare a tea. (Top-left) The *add(water)* behavior has been already demonstrated for the *prepare coffee task* and can be reused in the *prepare tea task*. (Top-right) The WM state after the *add(water)* execution. (Bottom-left) The human can demonstrate the novel subtask through kinesthetic teaching, then the robot can autonomously execute the rest of the task (*add(tea)*). (Bottom-right) The WM state after the *add(tea)* demonstration

for the robot execution. This way, the robot is to wait for the human help or guidance in order to execute the task. During the teaching phase, this subtask can be executed by the human (under the attentional supervision), while the rest of the task can be again demonstrated through kinesthetic teaching. Analogously to the *close* subtask, for the sake of simplicity, the human may just verbally declare the end of his/her intervention. It is also worth noting that, in this cooperative setting, the operator can teach motion primitives in preparation of the human interventions. For instance, when the next subtask is a human manipulation (e.g. *open(water)*), the robot should provide the object (bottle of water) in a comfortable position for the operator, hence in the learning phase this subtask should be also demonstrated taking the human into account.

5.4 Prepare tea: task re-usage

In the last experiment, we show that the acquired knowledge can be re-used to speed-up the acquisition of novel tasks. We consider the task of preparing a tea, where the robot has to pour the water in the cup and add a tea bag. The *add(water)* behavior is the same behavior used to prepare the coffee and can be re-used in this novel scenario. In other words, the already learned behavior can be loaded from the long term memory and instantiated in the working memory, while the user has only to teach how to add the tea bag (see Fig. 12). Once allocated in the WM, all the enabled and linked subtasks (e.g. *add(water)*) can be executed until the open subtask (*add(tea)*) is selected. In this case the robot needs the human demonstration to learn how to complete the overall task. In order to assess the effectiveness of task re-usage, we run ten teaching sessions: in five runs the teacher has to demonstrate the entire task, while in the remaining five runs the operator only teaches the missing *add(tea)* behavior. In this second setting, the robot waits for the human assistance, whenever not able to execute a subtask. As reported in Table 4, in the tea scenario, task re-usage is effective and can reduce the teaching time of about 53%.

Table 4 Results for ten training trials of the prepare tea task

Teaching time (s) (mean ± std)			Task re-usage
add(water)	*add(tea)*	*prepareTea*	(yes / no)
50.1 ± 1.9	34.1 ± 1.0	84.2 ± 2.9	No
−	35.6 ± 1.4	35.6 ± 1.4	Yes

The symbol "−" indicates an already learned subtask

6 Conclusions and future work

We presented a framework that allows a robot manipulator to learn how to execute structured tasks from one-shot kinesthetic demonstrations. In this framework, a supervisory attentional system continuously monitors the human and the robot activities during both task execution and task learning. During kinesthetic teaching, the human demonstration is automatically segmented producing motion primitives, while the attentional system relates the generated segments to the task structure exploiting attentional regulations. The framework has been evaluated considering a robotic manipulator operating in a kitchen scenario. Obtained results show that the system allows the robot to quickly learn and robustly execute typical structured activities that involve object manipulation. Moreover, we have shown how the attentional supervision of both the user and the robot activities enables cooperative execution of the learned tasks with an associated reduction of the execution time. Finally, we illustrated how learned tasks/subtasks can be reused in the context of novel task, in so enabling the acquisition of incrementally complex capabilities.

The focus of this work was on learning and executing kinematic tasks; extending the framework to learn and execute force patterns will be the topic of our future research. Moreover, we plan to investigate more complex human interaction scenarios along with more sophisticated attentional cueing mechanisms during both task teaching and execution. Finally, we aim at assessing the effectiveness of the proposed framework with an extended user study involving inexpert users.

Acknowledgements The research leading to these results has been partially supported by the Technical University of Munich, International Graduate School of Science and Engineering, by Helmholtz Association, by RoMoLo project, MISE F/050277/01–02–X32, under EU-funded Actions for R&D, and by MARsHaL project founded by DIETI.

References

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, *57*(5), 469–483.

Belardinelli, A., Pirri, F., & Carbone, A. (2007). Bottom-up gaze shifts and fixations learning by imitation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *37*(2), 256–271.

Bischoff, R., Kurth, J., Schreiber, G., Koeppe, R., Albu-Schäffer, A., Beyer, A., Eiberger, O., Haddadin, S., Stemmer, A., Grunwald, G., & Hirzinger, G. (2010). The KUKA-DLR lightweight robot arm—A new reference platform for robotics research and manufacturing. In *International symposium on robotics* (pp. 1–8).

Borji, A., Ahmadabadi, M. N., Araabi, B. N., & Hamidi, M. (2010). Online learning of task-driven object-based visual attention control. *Image and Vision Computing*, *28*(7), 1130–1145.

Botvinick, M. M., Braver, T. S., Barch, D. M., Carter, C. S., & Cohen, J. D. (2001). Conflict monitoring and cognitive control. *Psychological Review*, *108*(3), 624.

Breazeal, C., & Berlin M. (2008). Spatial scaffolding for sociable robot learning. In *Proceedings of AAAI-2008* (pp. 1268–1273).

Broquère, X., Finzi, A., Mainprice, J., Rossi, S., Sidobre, D., & Staffa, M. (2014). An attentional approach to human robot interactive manipulation. *International Journal of Social Robotics*, *6*(4), 533–553.

Caccavale, R., Cacace, J., Fiore, M., Alami, R., & Finzi, A. (2016). Attentional supervision of human–robot collaborative plans. In *RO-MAN* (pp. 867–873).

Caccavale, R., & Finzi, A. (2015). Plan execution and attentional regulations for flexible human–robot interaction. In *Proceedings of SMC, 2015* (pp. 2453–2458).

Caccavale, R., & Finzi, A. (2016). Flexible task execution and attentional regulations in human-robot interaction. *IEEE Transactions on Cognitive and Developmental Systems*, *9*(1), 68–79.

Caccavale, R., Leone, E., Lucignano, L., Rossi, S., Staffa, M., & Finzi, A. (2014). Attentional regulations in a situated human–robot dialogue. In: *RO-MAN* (pp. 844–849).

Cooper, R. P., & Shallice, T. (2006). Hierarchical schemas and goals in the control of sequential behavior. *Psychological Review*, *113*(4), 887–916.

De Maria, G., Falco, P., Natale, C., & Pirozzi, S. (2015). Integrated force/tactile sensing: The enabling technology for slipping detection and avoidance. In *International Conference on Robotics and Automation* (pp. 3883–3889).

Dillmann, R. (2004). Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, *47*(2), 109–116.

Fod, A., Matarić, M. J., & Jenkins, O. C. (2002). Automated derivation of primitives for movement classification. *Autonomous Robots*, *12*(1), 39–54.

Garrido-Jurado, S., Muñoz Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, *47*(6), 2280–2292.

Ijspeert, A., Nakanishi, J., Pastor, P., Hoffmann, H., & Schaal, S. (2013). Dynamical Movement Primitives: Learning attractor models for motor behaviors. *Neural Computation*, *25*(2), 328–373.

Kawamura, K., Gordon, S. M., Erdemir, E., & Hall, J. (2007). Implementation of cognitive control for a humanoid robot. *International Journal of Humanoid Robotics*, *5*(4), 547–586.

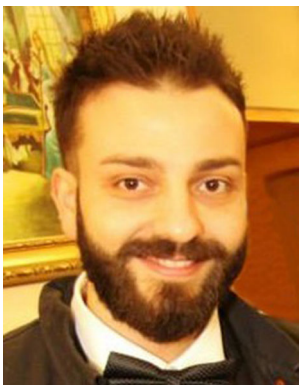
Koppula, H. S., & Saxena, A. (2015). Anticipating human activities using object affordances for reactive robotic response. *Transactions on Pattern Analysis and Machine Intelligence*, *38*(1), 14–29.

Kulić, D., Ott, C., Lee, D., Ishikawa, J., & Nakamura, Y. (2012). Incremental learning of full body motion primitives and their sequencing through human motion observation. *International Journal of Robotics Research*, *31*(3), 330–345.

Lee, D., & Ott, C. (2011). Incremental kinesthetic teaching of motion primitives using the motion refinement tube. *Autonomous Robots*, *31*(2), 115–131.

Magnanimo, V., Saveriano, M., Rossi, S., & Lee, D. (2014). A bayesian approach for task recognition and future human activity prediction. In *International symposium on robot and human interactive communication* (pp. 726–731).
Manschitz, S., Kober, J., Gienger, M., & Peters, J. (2015). Learning movement primitive attractor goals and sequential skills from kinesthetic demonstrations. *Robotics and Autonomous Systems*, *74*(Part A), 97–107.
Nagai, Y. (2009). From bottom-up visual attention to robot action learning. In *Proceedings of international conference on development and learning* (pp. 1–6).
Nau, D. S., Au, T., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., et al. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research (JAIR)*, *20*, 379–404.
Nicolescu, M. N., & Mataric, M. J. (2003). Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *Proceedings of the second international joint conference on autonomous agents and multiagent systems, ACM* (pp. 241–248).
Norman, D. A., & Shallice, T. (1986). Attention to action: Willed and automatic control of behavior. In R. J. Davidson, G. E. Schwartz, D. Shapiro (Eds.), *Consciousness and self-regulation: Advances in research and theory* (Vol. 4, pp. 1–18).
Park, D. H., Hoffmann, H., Pastor, P., & Schaal, S. (2008). Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields. In *International conference on humanoid robotics* (pp. 91–98).
Pastor, P., Kalakrishnan, M., Righetti, L., & Schaal, S. (2012). Towards associative skill memories. In *International conference on humanoid robots* (pp. 309–315).
Ramirez-Amaro, K., Beetz, M., & Cheng, G. (2015). Understanding the intention of human activities through semantic perception: Observation, understanding and execution on a humanoid robot. *Advanced Robotics*, *29*(5), 345–362.
Roa, MA., Argus, MJ., Leidner, D., Borst, C., & Hirzinger, G. (2012). Power grasp planning for anthropomorphic robot hands. In *International conference on robotics and automation* (pp. 563–569).
Rossi, S., Leone, E., Fiore, M., Finzi, A., & Cutugno, F. (2013). An extensible architecture for robust multimodal human–robot communication. In *Proceedings of IROS-2013* (pp. 2208–2213).
Saveriano, M., Lee, D (2013). Point cloud based dynamical system modulation for reactive avoidance of convex and concave obstacles. In *International conference on intelligent robots and systems* (pp. 5380–5387).
Saveriano, M., & Lee, D. (2014). Distance based dynamical system modulation for reactive avoidance of moving obstacles. In *Proceedings of ICRA-2014* (pp. 5618–5623).
Saveriano, M., An, S., & Lee, D. (2015). Incremental kinesthetic teaching of end-effector and null-space motion primitives. *ICRA*, *2015*, 3570–3575.
Saveriano, M., Hirt, F., & Lee, L. (2017). Human-aware motion reshaping using dynamical systems. *Pattern Recognition Letters*, *99*(11), 96–104.
Takano, W., & Nakamura, Y. (2016). Real-time unsupervised segmentation of human whole-body motion and its application to humanoid robot acquisition of motion symbols. *Robotics and Autonomous Systems*, *75*(Part B), 260–272.
Tenorth, M., & Beetz, M. (2013). Knowrob—A knowledge processing infrastructure for cognition-enabled robots. *International Journal of Robotics Research*, *32*(5), 566–590.
Wächter, M., Schulz, S., Asfour, T., Aksoy, E., Wörgötter, & Dillmann, R. (2013). Action sequence reproduction based on automatic segmentation and object-action complexes. In *International conference on humanoid robots* (pp. 189–195).
Zoliner, R., Pardowitz, M., Knoop, S., & Dillmann, R. (2005). Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration. In *International conference on robotics and automation* (pp. 1535–1540).

Riccardo Caccavale is currently a postdoctoral researcher at the Department of Electrical Engineering and Information Technology, University of Naples "Federico II" (Italy). His research interests are focused on Cognitive Robotics, Human–Robot Interaction, Cognitive Control. He is involved in research projects sponsored by the European Community.



Matteo Saveriano is a postdoctoral researcher at the German Aerospace Center (DLR). He received is B.Sc. (2008) and M.Sc. (2011) in Automatic Control Engineering from Università degli Studi di Napoli "Federico II", and a Ph.D. in Robotics from the Technical University of Munich (2017). His research interests include safe human-robot interaction, representation and classification of human activities, imitation learning and reactive collision avoidance.

Alberto Finzi is Assistant Professor at the Department of Electrical Engineering and Information Technology (DIETI), University of Naples "Federico II" (Italy). He received his Ph.D. degree in Computer Engineering from Sapienza University of Rome (Italy). His research interests include Cognitive Robotics, Human–Robot Interaction, Executive and Cognitive Control, Autonomous and Adaptive Systems, Planning and Scheduling Systems, Multi-agent Systems, Formal Methods for Autonomous Systems. He has been recently involved in several research projects sponsored by the EC (European Community), NASA (National Aeronautics and Space Administration), ESA (European Space Agency), ASI (Italian Space Agency), FWF (Austrian Science Fund), MIUR (Italian Ministry for University and Research), and private industries.

Dongheui Lee is Associate Professor of Human–centered Assistive Robotics at the TUM Department of Electrical and Computer Engineering. She is also director of a Human–centered assistive robotics group at the German Aerospace Center (DLR). Her research interests include human motion understanding, human robot interaction, machine learning in robotics, and assistive robotics. Prior to her appointment as Associate Professor, she was an Assistant Professor at TUM (2009–2017), Project Assistant Professor at the University of Tokyo (2007–2009), and a research scientist at the Korea Institute of Science and Technology (KIST) (2001–2004). After completing her B.S. (2001) and M.S. (2003) degrees in mechanical engineering at Kyung Hee University, Korea, she went on to obtain a Ph.D. degree from the department of Mechano–Informatics, University of Tokyo, Japan in 2007. She was awarded a Carl von Linde Fellowship at the TUM Institute for Advanced Study (2011) and a Helmholtz professorship prize (2015). She is coordinator of both the euRobotics Topic Group on physical Human Robot Interaction and of the TUM Center of Competence Robotics, Autonomy and Interaction.