2015 IEEE International Conference on Robotics and Automation (ICRA)
Washington State Convention Center
Seattle, Washington, May 26-30, 2015

Learning Symbolic Representations of Actions from Human Demonstrations

Seyed Reza Ahmadzadeh[1,2] , Ali Paikan[2], Fulvio Mastrogiovanni[3],
Lorenzo Natale[2], Petar Kormushev[1] and Darwin G. Caldwell[1]

Abstract— **In this paper, a robot learning approach is proposed which integrates Visuospatial Skill Learning, Imitation Learning, and conventional planning methods. In our approach, the sensorimotor skills (i.e., actions) are learned through a learning from demonstration strategy. The sequence of performed actions is learned through demonstrations using Visuospatial Skill Learning. A standard action-level planner is used to represent a symbolic description of the skill, which allows the system to represent the skill in a discrete, symbolic form. The Visuospatial Skill Learning module identifies the underlying constraints of the task and extracts symbolic predicates (i.e., action preconditions and effects), thereby updating the planner representation while the skills are being learned. Therefore the planner maintains a generalized representation of each skill as a reusable action, which can be planned and performed independently during the learning phase. Preliminary experimental results on the iCub robot are presented.**

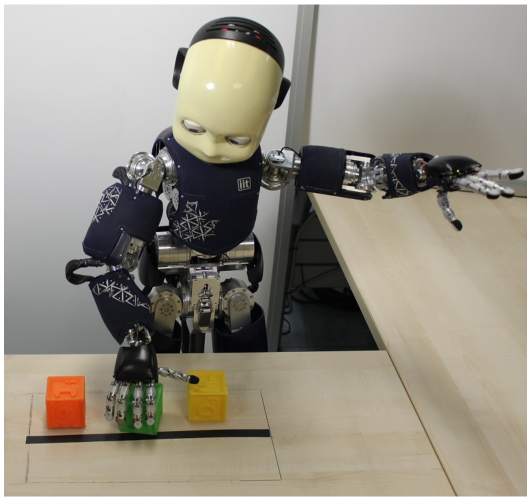


Fig. 1: The iCub robot interacting with objects.

I. INTRODUCTION

Robot learning approaches based on demonstrations are motivated by the fact that imitation enables humans to learn new skills. As a result, many skill learning approaches have been developed during the past decade that benefit from human demonstrations [1]. Such approaches enable non-experts to interact with robots and teach them new skills efficiently with a few demonstrations.

These approaches can be categorized in two main subclasses: trajectory-based and goal-based approaches. The former group focuses on recording and re-generating trajectories [1], [2] or forces [3] to emulate the demonstrated skill. For instance, in [4] a humanoid robot learns to play air hockey by learning primitives. In many cases, however, it is not the trajectory that is of central importance, but the goal of the task (e.g., solving a jigsaw puzzle). Learning every single trajectory in such tasks actually increases the complexity of the learning process unnecessarily [5]. In order to address this drawback, several goal-based approaches have been proposed [6–8]. For instance, a number of symbolic learning approaches exist that focus on goal configuration rather than action execution [8]. However, in order to ground the symbols, they comprise many steps inherently, namely segmentation, clustering, object recognition, structure recognition, symbol generation, syntactic task modeling, motion grammar, rule generation, etc. Another drawback of such approaches is that they require a significant amount of *a priori* knowledge to be manually engineered into the system [5], [7]. An alternative to motor skill learning approaches are visual learning approaches [9–11]. These approaches are based on observations acquired from human demonstrations and using human-like visuospatial skills to replicate the specified task [11–13]. Visuospatial skill is the capability to visually perceive the spatial relationship among objects. In our previous work [11], we have proposed a robot learning approach based on visuospatial skill learning in humans for solving tasks consisting of object reconfigurations. Visuospatial Skill Learning (VSL) enables a robot to identify the spatial relationship between objects and learn a sequence of actions for achieving the goal of the task. VSL is capable of learning and generalizing different skills such as object reconfiguration, classification, and turn-taking interactions from a single demonstration [11], [13].

One of the shortcomings of VSL is that the primitive actions such as pick, place, and reach have to be manually programmed into the robot. For instance, in [11], [13] a simple trajectory generation module was devised that is able to produce trajectories for pick and place. This issue can be addressed by combining VSL with a trajectory-based learning approach such as imitation learning. The obtained approach can learn the primitive actions of the task using

[1] Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova, Italy. {reza.ahmadzadeh, petar.kormushev, darwin.caldwell} at iit.it
[2] iCub Facility, Istituto Italiano di Tecnologia, via Morego, 30, 16163 Genova, Italy. {ali.paikan, lorenzo.natale} at iit.it
[3] Department of Informatics, Bioengineering, Robotics and Systems Engineering, University of Genova, via Opera Pia 13, 16145 Genova, Italy. fulvio.mastrogiovanni@unige.it

conventional Learning from Demonstration strategy while it learns the policy (e.g., sequence of actions), preconditions and effects of the actions using VSL.

Although VSL has its own internal planner, in order to utilize the advantages of standard symbolic planners, additionally in this work, we replaced the internal planner with an already available action-level symbolic planner, namely SGPlan [14]. In order to ground the actions, the symbolic actions are defined in the planner and VSL maps identified preconditions and effects in a formalism suitable to be used at the symbolic level.

In this paper, we propose a new robot learning approach by integrating a sensorimotor learning framework (imitation learning), a visual learning approach (VSL), and a symbolic-level action representation and planning layer. We evaluate the capabilities and performance of our framework using real-world experiments with an iCub humanoid robot.

II. Related Work

A number of approaches have been proposed for teaching skills to the robots at the symbolic and planning level based on demonstrations. The work by Kuniyoshi and Inaba [9] is one of the first papers to learn task-execution plans from demonstrations. The method learns a plan for an object manipulation task by observing the movements of the tutor's hand, using visual information. The objects, however, were known to the robot in advance, and the primitive actions were pre-programmed into the robot. The method decomposes the demonstrated task into functional and symbolic units and makes a hierarchical task plan. If the task is not learned correctly there is no way to improve it.

The method proposed in [15], extracts knowledge about a task from sequence of actions. The task is setting a table using pick-and-place actions. The robot extracts the proper primitive actions and the constraints of the task from demonstrations. In [16], the learned motor skills from demonstrations are encoded into a non-linear differential equation that can reproduce those skills afterwards. In order to provide primitive actions with semantics, the concept of Object-Action Complexes [17] is utilized. However, the resulting symbolic actions are never used in a high-level planner.

Niekum *et al.*, claim that, the most natural way to demonstrate a task is by performing it continuously from start to finish [5]. In their proposed trajectory-based approach, a Beta-Process Autoregressive Hidden Markov Model is employed to segment and recognize an unstructured demonstration (i.e., continuous trajectory). The approach is combined with Dynamic Movement Primitives (DMP) to address the problem of task representation and generalization. A coordinate frame centered on each known object is defined and the objects are detected using either stereo vision or visual fiducials. Our approach is independent from the trajectories performed by the tutor, in that we identify and extract the underlying constraints of the task. Finally, they build a skill library including the learned skills. However, the robot needs a new demonstration to distinguish a skill from those present in the library. A framework for manipulation tasks has been introduced in [18]. The proposed framework includes two main layers: the former optimizes the parameters of the motor primitives, whereas the latter relies on pre- and post-conditions to model actions. In contrast to our approach, the pre- and the post-conditions are manually engineered into the system. Aksoy *et al.*, showed how symbolic representations can be linked to the trajectory level using spatiotemporal relations between objects and hands in the workspace [19].

The most similar work to our approach is that by Ekvall *et al.*, [20]. In their work, a task planning approach is used in combination with robot learning from demonstration. The robot generates states and identifies constraints of a task incrementally according to the order of the action execution. Imitation learning is employed to teach the task to the robot. The constraints are identified in two ways: either the tutor instructs the system or the constraints are considered by merging of multiple observations. Differently from our approach, the objects are first modeled geometrically and a set of SIFT features for each object is extracted in off-line mode and used during the learning phase. They build a planning scenario that is bounded to the objects considered in the learning phase, whereas we define actions whose validity is not limited to the actual learning scenario.

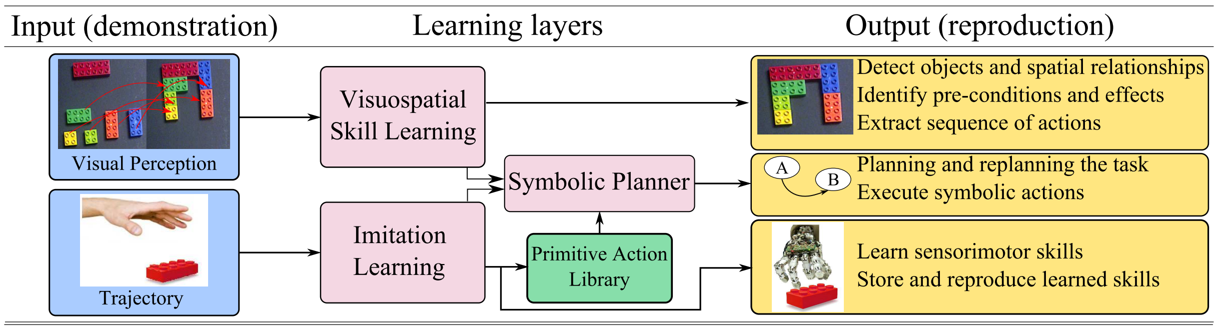
III. Overview

The proposed learning approach consists of three layers:

- *Imitation Learning (IL)*, which is suitable for teaching trajectory-based skills (i.e. actions) to the robot [1]. For instance, pouring and reaching.
- *Visuospatial Skill Learning (VSL)*, which is a goal-based approach based on visual perception [11]. It captures the spatial relationship between an object and its surrounding objects and extracts a sequence of actions to reach the goal of the task.
- *Symbolic Planning (SP)*, which uses a symbolic representation of actions, to plan or replan the execution of the task.

In our framework, IL is employed to teach sensorimotor skills to the robot (e.g., pull and push). The learned skills are stored in a primitive action library, which is accessible by the planner. VSL captures the object's context for each demonstrated action. This context is the basis of the visuospatial representation and encodes implicitly the relative positioning of the object with respect to multiple other objects simultaneously. By capturing the spatial relationship among objects, VSL extracts a sequence of actions to reach the goal of the task. In addition, it is capable of identifying the preconditions and effects for each action.

VSL is internally equipped with a simple planner which has adequate capabilities as shown in [11] and [13]. In order to employ the advantages of symbolic-level action planners, PDDL 3.0 compliant planner is integrated with IL and VSL to represent a general purpose representation of a planning domain. In our system, VSL identifies action preconditions and effects to modify their formal PDDL definition (i.e., the planning domain). Once formal action



Input (demonstration)
Learning layers
Output (reproduction)
Visual Perception
Trajectory
Visuospatial Skill Learning
Imitation Learning
Symbolic Planner
Primitive Action Library
Detect objects and spatial relationships
Identify pre-conditions and effects
Extract sequence of actions
A
B
Planning and replanning the task
Execute symbolic actions
Learn sensorimotor skills
Store and reproduce learned skills

Fig. 2: A flow diagram illustrating the proposed approach consisting of three main layers. The robot learns to reproduce primitive actions through imitation learning. It also learns the sequence of actions and identifies the constraints of the task using VSL. The symbolic planner is employed to solve new symbolic tasks and execute the plans.

definitions are obtained, it is possible to exploit the planning domain to reason upon scenarios that are not strictly related to what has been learned, e.g., including different object configuration, a varied number of objects or different objects at all. One of the main advantages of the proposed approach is that it extracts and utilizes multi-modal information from demonstrations including both the sensorimotor information and visual perception, which is used to build symbolic representation structures for actions. A high-level flow diagram of the proposed approach including the aforementioned layers together with their input and output is depicted in Figure 2.

IV. Methodology

The robot learns the primitive trajectory-based actions through IL. The layer requires a set of demonstrations for each action, from which a set of attractors are learned and used to reproduce the action. The learned actions are stored in a primitive action library. In addition, the robot learns the sequence of actions through VSL using visual perception. The robot records a set of visual observations during the demonstration phase. It can then identify and reproduce the sequence of actions to reach the desired goal of the demonstrated task using spatial relationships among the objects [11]. As shown in Figure 2, both IL and VSL observe similar demonstrations (not necessarily at the same time) but utilize different types of data. IL uses sensorimotor information (i.e., trajectories), whereas VSL uses visual data. VSL also extracts the preconditions and effects of each action, thereby updating the PDDL description of each learned action, as well as (if required) the demonstrated problem, including the strict sequence of shown actions. Obviously enough, if the knowledge about such a sequence is not considered in subsequent planning steps, the planner is free to choose any suitable course of actions to solve a given planning problem. Once a symbolic representation of actions is obtained, SP can be used to solve new problems in the learned domain, as well as reproduce plans learned using VSL.

A. *Learning Sensorimotor Skills by Imitation Learning*

IL enables robots to learn and reproduce trajectory-based skills from a set of demonstrations through kinesthetic teaching [21]. In our framework, IL is utilized to teach primitive actions (i.e., pull and push) to the robot. In particular, we utilize Dynamic Movement Primitives (DMP) which are designed for modeling attractor behaviors of autonomous nonlinear dynamical systems [22].

In order to create a pattern for each action, multiple desired trajectories in terms of position, velocity and acceleration have to be demonstrated and recorded by a human operator, in the form of a vector $[y_{demo}(t), \dot{y}_{demo}(t), \ddot{y}_{demo}(t)]$, where $t \in [1, ..., P]$. A controller converts desired position, velocity, and acceleration, y, $\dot{y}$,and $\ddot{y}$, into motor commands. DMP employs a damped spring model that can be modulated with nonlinear terms such that it achieves a desired attractor behavior. The transformation system which represents dynamics of a damped spring system is as follows:

$$\begin{aligned} \tau\dot{z} &= \alpha_z(\beta_z(g-y)-z)+f+C_f \\ \tau\dot{y} &= z \end{aligned} \tag{1}$$

where y and z denote the position and the velocity respectively, τ is a time constant and α_z and β_z are positive constants. With $\beta_z = \alpha_z/4$, y monotonically converges toward the goal g. f is the forcing term; C_f is an application dependent coupling term. Trajectories are recorded independently of time. Instead, the canonical system, is defined as:

$$\tau\dot{x} = -\alpha_x x + C_c \tag{2}$$

where α_x is a constant and C_c is an application dependent coupling term. x is a phase variable, where $x = 1$ indicates the start of the time and the x close to zero means that the goal g has been achieved. Starting from some arbitrarily chosen initial state x_0 such as $x_0 = 1$ the state x converges monotonically to zero. In addition, f, in (1) is chosen as follows:

$$f(x) = \frac{\sum_{i=1}^{N}\psi_i(x)\omega_i}{\sum_{i=1}^{N}\psi_i(x)}x(g-y_0) \tag{3}$$

where $\psi_i(x)$ $i = 1,\ldots,N$ are fixed exponential basis functions, $\psi_i(x) = \exp(-h_i(x-c_i)^2)$, where σ_i and c_i are the width and centers of the basis functions respectively. w_i denote weights and y_0 is the initial state $y_0 = y(t = 0)$. The

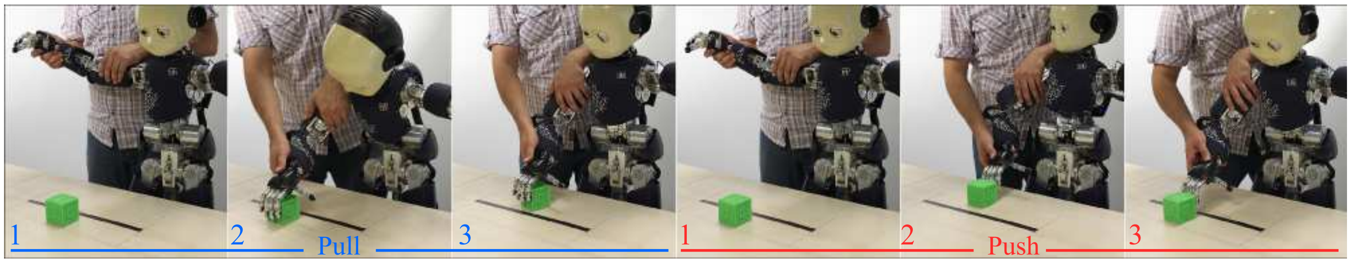


Fig. 3: The tutor is teaching the primitive actions including pull and push to the robot using kinesthetic teaching (For more details, see Section IV-A).

parameter g is the target coinciding the end of the movement $g = y_{demo}(t = t_{final})$, $y_0 = y_{demo}(t = 0)$. The parameter τ must be adjusted to the duration of the demonstration. The learning process of the parameters w_i is accomplished with a locally weighted regression method, because it is very fast and each kernel learns independent of others [22].

In the considered scenario, two primitive actions, namely pull and push are used. For executing the push action on an object, firstly, the robot needs to reach a location just in front of the object; Whereas, for executing the pulling action the robot needs to reach a location just beyond the object. As the first step, the tutor demonstrates each primitive action multiple times while holding and moving the robot's hand. Using the recorded set of trajectories and applying DMP, the robot learns a set of attractors by which it can reproduce the action starting from a different pose towards a target. We decomposed the *pull* action into *reach after* and *move backward* sub-actions and also decomposed the *push* action into *reach before* and *move forward* sub-actions. The recorded sets of demonstrations for both *reach before* and *reach after* sub-actions are depicted as black curves in Figure 4. Each red trajectory in Figure 4 illustrates a reproduction. In both Figures the goal, (i.e., the object), is shown in green. Finally, the learned primitive actions are stored in the primitive action library. Later, the planner connects the symbolic actions to the library of actions. The *move forward* and *move backward* sub-actions are straight line trajectories that can be either learned or implemented directly by a trajectory generation module.

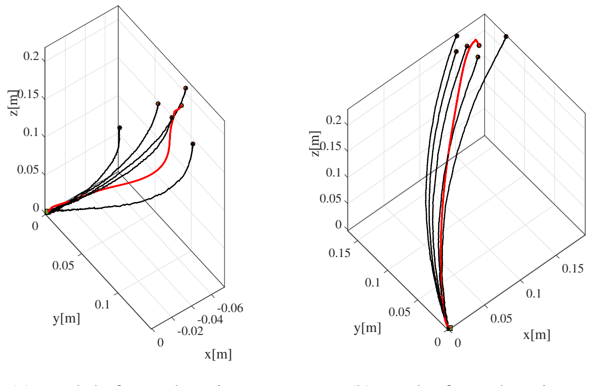


Fig. 4: The recorded set of demonstrations for two sub-actions are shown in black. The reproduced trajectories from an arbitrary initial position towards the target (the green square) are shown in red (For more details, see Section IV-A).

B. Learning Action Sequences by Visuospatial Skill Learning

At this stage, a library of primitive actions has been built. The robot learns the sequence of actions required to achieve the goal of the demonstrated task through VSL [11]. It also identifies the spatial relationship between objects by observing demonstrations.

The basic terms which are used to describe VSL consist of, *World*: the workspace of the robot which is observable by the robot. *Frame*: a bounding box which defines a cuboid in 3D space or a rectangle in 2D space. *Observation*: the captured context of the *world* from a predefined viewpoint using a specific *frame*. An *observation* can be a 2D image or a cloud of 3D points. *Pre-action observation*: An *observation* which is captured just before the action is executed. The robot searches for preconditions in the *pre-action observations* before selecting and executing an action. *Post-action observation*: An *observation* which is captured just after the action is executed. The robot perceives the effects of the executed actions in the *post-action observations*.

Formally, we define a process of VSL as a tuple $\mathcal{V} = \{\mathcal{W}, \mathcal{O}, \mathcal{F}, \mathcal{A}, \mathcal{C}, \Pi, \phi\}$, where $\mathcal{W} \in \Re^{m\times n}$ is a matrix which represents the context of the *world* including the workspace and all objects. $\mathcal{W}_D$ and $\mathcal{W}_R$ indicate the *world* during the demonstration and reproduction phases respectively; $\mathcal{O}$ is a set of *observation* dictionaries $\mathcal{O} = \{\mathcal{O}^{pre}, \mathcal{O}^{post}\}$; $\mathcal{O}^{pre}$ and $\mathcal{O}^{post}$ are *observation* dictionaries comprising a sequence of *pre-action* and *post-action observations* respectively. $\mathcal{F} \in \Re^{m\times n}$ is an *observation frame* which is used for capturing the *observations*. $\mathcal{A}$ is a set of primitive actions defined in the learning task (e.g., pick). $\mathcal{C}$ is a set of *constraint* dictionaries $\mathcal{C} = \{\mathcal{C}^{pre}, \mathcal{C}^{post}\}$; $\mathcal{C}^{pre}$ and $\mathcal{C}^{post}$ are *constraint* dictionaries comprising a sequence of *pre-action*, and *post-action constraints* respectively. Π is a policy or an ordered action sequence extracted from demonstrations. ϕ is a vector containing extracted features from *observations* (e.g., SIFT features). Pseudo-code of VSL is given in Algorithm 1.

At the beginning of the demonstration, the objects are randomly placed in the *world* ($\mathcal{W}_D$). In case that, any absolute landmark is being used in the demonstration (e.g., borderline in our experiments), the robot should be able to

```
Input  : $\{\mathcal{W}, \mathcal{F}, \mathcal{A}\}$
Output : $\{\mathcal{O}, \mathcal{P}, \Pi, \mathcal{C}, \mathcal{B}, \phi\}$
1  Initialization: detect absolute landmarks if defined
2  $l$=detectBorderline($\mathcal{W}$)
3  $i = 1$ , $j = 1$
   // Part I : Demonstration
4  for each operation do
5      $\mathcal{O}_i^{pre}$ = getPreActionObs($\mathcal{W}_D, \mathcal{F}_D$)
6      $\mathcal{O}_i^{post}$ = getPostActionObs($\mathcal{W}_D, \mathcal{F}_D$)
7      $[\mathcal{B}_i, \mathcal{P}_i^{pre}, \mathcal{P}_i^{post}, \phi_i]$ = getObject($\mathcal{O}_i^{pre}, \mathcal{O}_i^{post}$)
8      $[\mathcal{C}_i^{pre}, \mathcal{C}_i^{post}]$ = getConstraint($\mathcal{B}_i, \mathcal{P}_i^{pre}, \mathcal{P}_i^{post}, l$)
9      $\Pi_i$ = getAction($\mathcal{A}, \mathcal{C}_i^{pre}, \mathcal{C}_i^{post}$)
10     $i = i + 1$
11 end
   // Part II : Reproduction (this part is
      used in absence of symbolic planner.)
12 for $j = 1$ to $i$ do
13     $\mathcal{P}_j^{*pre}$ = findBestMatch($\mathcal{W}_R, \mathcal{O}_j^{pre}, \phi_j, \mathcal{C}_j^{pre}, l$)
14     $\mathcal{P}_j^{*post}$ = findBestMatch($\mathcal{W}_R, \mathcal{O}_j^{post}, \phi_j, \mathcal{C}_j^{post}, l$)
15     executeAction($\mathcal{P}_j^{*pre}, \mathcal{P}_j^{*post}, \Pi_j$)
16 end
```

Algorithm 1: Pseudo-code for VSL.

detect it in the *world* (line 2). During the demonstration, VSL captures one *pre-action observation* ($\mathcal{O}^{pre}$) and one *post-action observation* ($\mathcal{O}^{post}$) for each operation executed by the tutor using the specified *frame* ($\mathcal{F}_D$) (lines 5,6). For each detected object in the *world*, VSL creates a symbolic representation ($\mathcal{B}$) and extracts a feature vector (ϕ). The symbolic object is used by the symbolic planner and the extracted features are used by VSL for detecting the object during the reproduction phase. VSL also extracts the pose of the object before and after action execution ($\mathcal{P}^{pre}, \mathcal{P}^{post}$). The pose vectors together with the detected landmark (l) are used to identify preconditions and effects of the executed action through spatial reasoning (line 8). These predicates are then utilized to identify the executed action from the action set $\mathcal{A}$ (line 9). The sequence of identified actions are then stored in a policy vector Π. In this framework, the symbolic planner, utilizes the output of VSL to reproduce the task properly. Furthermore, the second part of the algorithm is able to execute the learned sequence of actions independently [11], [13]. In such case, VSL observes the new *world* ($\mathcal{W}_R$) in which the objects are replaced randomly. Comparing the recorded *pre-* and *post-action observations*, VSL detects the best matches for each object and executes the actions from the learned policy.

C. Generalization of Learned Actions as Symbolic Action Models

In order to use the learned primitive actions for general-purpose task planning, the actions need to be represented as a symbolic, discrete, scenario-independent form. To this aim, we have defined a symbolic representation for both the pull and push actions based on the PDDL 3.0 formalism. However, it is necessary to learn preconditions and effects for each action in the primitive action library. Instead of manually defining them, we exploit the sensorimotor information in the skill learning process to derive symbolic predicates. So, we first let the robot perceive the initial workspace in which one object is placed in `far`. Then we ask the robot to execute the pull action, and after the action is finished, the robot perceives the workspace again. The *pre-action* and *post-action observations* are shown in Figure 5. The robot uses VSL to extract the preconditions and effects of the pull action. We repeat the same steps for the push action. The domain file in the PDDL is updated automatically by applying this procedure. The extracted preconditions and effects for the pull and push actions are reported in table I.

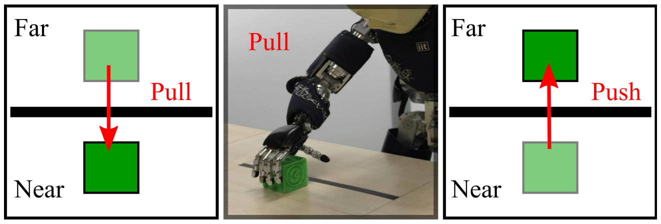


Fig. 5: By extracting the preconditions and effects of the primitive actions while executing them, the robot generalizes the learned sensorimotor skills as symbolic actions (For more details, see Section IV-C).

TABLE I: The preconditions and effects extracted by VSL and written to the planner's domain file (For more details, see Section IV-C).

Action	push	pull
precondition	`¬ (far ?b)`	`(far ?b)`
effect	`(far ?b)`	`¬ (far ?b)`

V. Implementation

After the VSL model has been identified, it is possible to generalize each action as well as the demonstrated action sequences, using a PDDL 3.0 compliant formalism. It is noteworthy that in the present work we do not bootstrap symbolic knowledge from scratch. On the contrary, starting from the knowledge of the performed action types during the VSL demonstration, we update symbolic-level knowledge with two kinds of constraints. The former class includes constraints (in the form of PDDL predicates) related to preconditions and effects. This is done by mapping the elements of observation dictionaries O^{pre} and O^{post} to relevant predicates. In the considered scenario, one predicate is enough to characterize push and pull actions, namely `(far ?b)`, where `far` is the predicate name and `?b` is a variable that can be grounded with a specific object that can be pushed and pulled. Specifically, `push(?b)` is an action that makes the predicate truth value switching from `¬(far ?b)` to `(far ?b)`, whereas the opposite holds for `pull(?b)`. Visual information about the location of objects in the *World* is mapped to the proper truth value of the `(far ?b)` predicate. The execution of a `push` demonstration on a green cube (as processed by VSL) allows the system to understand that before the action ¬`(far` B_{green}`)` holds, after the action `(far` B_{green}`)` holds, and the two predicates contribute to the `:precondition` and `:effect` lists of a `push` action defined in PDDL.

The second class includes constraints (in the form of PDDL predicates) related to the trajectory of the plan that is executed in the demonstration. This can be done analyzing the sequence of actions as represented in the VSL model (i.e., the implicit chain defined by predicates in O^{post} and O^{pre} observation dictionaries). In PDDL 3.0, it is possible to define additional constraints, which implicitly limit the search space when the planner attempts to find a solution to the planning problem. In particular, given a specific planning domain (in the form of push and pull actions, as obtained by reasoning on the VSL model), the `problem` can be constrained to force the planner to follow a specific trajectory in the solution space. One possible constraint is to impose, for instance, the sequence of satisfiability of predicate truth values. As an example, let us assume the VSL model represents the fact that B_{green} must be always pushed after B_{blue}. According to our formal definition, this implies that the predicate `(far` B_{green}`)` must hold sometime after the predicate `(far` B_{blue}`)` holds. To encode this constraint in PDDL formalism, it is sufficient to use the constraint `(somewhere-after (far` B_{green}`)` `(far` B_{blue}`))`. If the constraint is used in the planning problem, the planner attempts to find a plan where this sequence is preserved, thereby executing `push(`B_{green}`)` strictly after `push(`B_{green}`)`. Finally, it is noteworthy that if we remove such a constraint, the planner is free to choose any suitable sequence of actions to be executed (independently of the demonstration), provided that the goal state is reached.

VI. Experimental Setup

In order to evaluate the capabilities and performance of our approach, a set of real-world experiments is performed with an iCub humanoid robot. As shown in Figure 1, the robot is standing in front of a tabletop including some polyhedral objects. All the objects have equivalent size but different colors and textures. Firstly, the tutor teaches the primitive actions to the robot using imitation learning. The primitive actions include push and pull movements. The tutor performs a set of demonstrations for each primitive action by holding and moving the robot's hand while the robot is perceiving the workspace (see Figure 3). Using the recorded set of demonstrations, the robot learns to reproduce each action from different initial position of the hand towards a desired object. The learned primitive actions are stored in the primitive action library.

To introduce the concept of `far` and `near` in our experiments, a black line, which is a borderline, is placed in the middle of the robot's workspace. The robot can detect the line using conventional edge detection and thresholding techniques. In addition, the size of the *frames* ($\mathcal{F}_D, \mathcal{F}_R$) are defined equal to the size of the *world*.

During the demonstration phase, for each operation, the tutor removes the object from the *world*, and brings it back after an *observation* has been captured. Figure 6 shows some instances of image processing in VSL. Since image processing is not the focus of this paper, a number of simple techniques have been used for this purpose. In our previous research, the object detection is done using background subtraction technique and it has been shown that the object identification process can be done using 2D and 3D match finding techniques (e.g., SIFT) [11], [13]. In this paper, however, we utilize blob detection and filtering by color on plane, considering the fact that the iCub robot is capable of detecting the height of the table at the beginning of the experiment by touching it. To detect the object which is moved and the place that the object has been moved to, background subtraction is used. For both cases, the center of the detected object and the center of the detected area are estimated and are shown with a red marker in Figure 6.

Furthermore, a simple (boolean) spatial reasoning has been applied to the VSL algorithm that utilizes the detected pose of the object and the detected borderline, and then decides that the object is `far` or `near`. However, to extract more sophisticated constraints, qualitative spatial reasoning languages such as region connection calculus [23] can be employed.

The video accompanying this paper shows the execution of the tasks and is available online at [24].

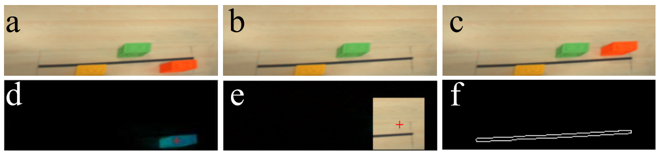


Fig. 6: (a) and (b) are rectified pre-action and post-action observations; (c) is a rectified pre-action observation for the next operation; (d) is the result of background subtraction between (a) and (b); (e) is the result of background subtraction between (b) and (c); (f) is the result of detecting the borderline for spatial reasoning (For more details, see Section VI).

VII. Experimental Validation

A. A Simple VSL Task

In order to evaluate the obtained connection between the symbolic actions and their equivalent sensorimotor skills, we conducted the following experiment. Initially, there are three objects on the table. The tutor provides the robot with a sequence of actions depicted in Figure 7. The robot perceives the demonstration, records the observations and extracts the sequence of actions using VSL. In the reproduction phase, the objects are reshuffled in the workspace. For the reproduction of the task, instead of the generic SP, the internal planner of VSL is exploited. Starting from a different initial configuration of the objects in the workspace, VSL is capable of extracting the ordered sequence of actions and reproduce the same task effortlessly. As shown in Figure 8, the robot achieves the goal of the task using VSL. This experiment shows that the robot can relate the learned primitive actions including push and pull, with their preconditions.

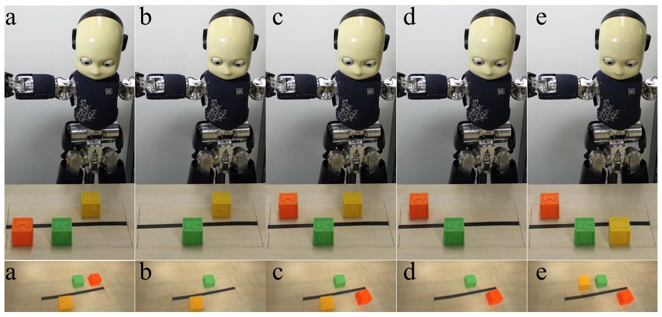


Fig. 7: The set of demonstrations by the tutor for a simple VSL task. The bottom row shows the workspace from the robot's point of view (For more details, see Section VII-A).

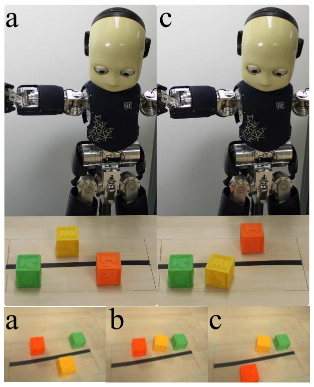


Fig. 8: Starting from a new configuration, the robot reproduces the learned task using VSL. (a) and (c) show initial and final configurations. The bottom row shows the workspace from the robot's point of view (For more details, see Section VII-A).

B. Keeping All Objects Near

In the previous experiment, we have shown that the internal planner of VSL is capable of reproducing the task. However, SP is more capable in generalizing the learned skill. In this experiment, we want to teach the robot a simple game including an implicit rule: *keep all objects near*. For each object, two properties are defined: color and position. SP provides our approach with the capability of generalizing over the number and even shape of the objects. In order to utilize this capability, the tutor performs a set of demonstrations to eliminate the effect of the color implying that any object independent of its color should not be `far` from the robot. Three sets of demonstrations are performed by the tutor which are shown in Figure 9. Each demonstration starts with a different initial configuration of the objects. The performed demonstrations imply the following rules:

$$\begin{aligned} r_1^1 &: \langle B_{green} \rightarrow \texttt{near}\rangle \text{ IF } \langle B_{orange}\ is\ \texttt{far}\rangle \\ r_1^2 &: \langle B_{orange} \rightarrow \texttt{near}\rangle \text{ IF } \langle B_{green}\ is\ \texttt{near}\rangle \\ r_2^1 &: \langle B_{green} \rightarrow \texttt{near}\rangle \text{ IF } \langle B_{orange}\ is\ \texttt{near}\rangle \\ r_3^1 &: \langle B_{orange} \rightarrow \texttt{near}\rangle \text{ IF } \langle B_{green}\ is\ \texttt{far}\rangle \end{aligned} \tag{4}$$

where r_i^j indicates the j^{th} rule extracted from the i^{th} demonstration. The first demonstration includes two rules r_1^1, r_1^2 because two actions are executed. It can be seen that the right parts of r_1^1 and r_2^1 and the right parts of r_1^2 and r_3^1 eliminate each other. Also the color attribute on the left side of the rules eliminates the effect of the color on action execution. During the demonstration, for each object, the robot extracts the spatial relationships between the objects and the borderline, and decides if the object is `far` or `near` (i.e., `not far`). It then applies the extracted precondition of $r_1^1 : \langle B_? \rightarrow \texttt{near}\rangle$ to that object. Afterwards, the robot is capable of performing the learned skill on a various number of objects and even unknown objects with different colors.

C. Pull All Objects in a Given Order

In the previous experiment, the robot learned that all objects must be kept close, without giving importance to the order of actions. In this experiment, we include the color attribute for each object to give priority to the sequence of actions to be performed. For instance, consider the ordered set $\langle orange, green, yellow\rangle$ in which the orange cube and the yellow cube are the most and least important cubes, respectively. This means that we want the orange cube to be moved before the green one, and the green cube before the yellow one. The robot has to learn that the most important cube has to be operated first. Besides, learning a sequence of actions in which the order is important, is one of the main capabilities of VSL. Therefore, if the task was demonstrated once and the reproduction part was done separately by VSL (and not by SP) the robot would learn the task using one single demonstration. In addition, VSL provides the robot with the capability of generalizing the task over the initial configuration of the objects.

Furthermore, it is possible to encode such an ordering also in a planning domain, at the cost of adding extra constraints (in the form of predicate) to the definition of the planning problem. In this case the planner should include two extra predicates, namely `(somewhere-after (far` B_{green}`) (far` B_{orange}`))` and `(somewhere-after (far` B_{yellow}`) (far` B_{green}`))`. The inclusion of these predicates must be managed at an extra-logic level, e.g., by extracting the sequence of actions using VSL and inserting the corresponding constrains in the planner problem definition. In this case, there is no need to have more demonstrations, but expressing the order of sequence as preconditions may be done automatically.

VIII. CONCLUSIONS

We have proposed a robot learning approach by integrating Imitation Learning (IL), Visuospatial Skill Learning (VSL),

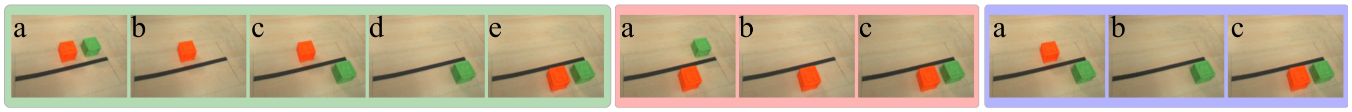


Fig. 9: Three different sets of demonstrations devised by the tutor for implying the rules of the game in the second experiment (For more details, see Section VII-B).

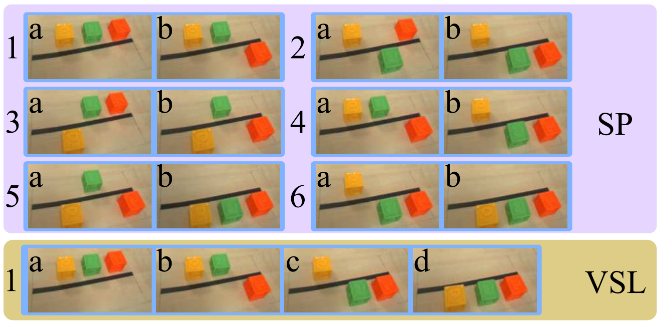


Fig. 10: In the third experiment, using SP the robot needs six sets of demonstrations for each of which the initial (a) and final (b) configurations of the objects are shown. While using VSL the robot requires a single demonstration for which the sequence of actions are shown (a-d) (For more details, see Section VII-C).

and conventional planning methods. Our approach is capable of learning sequential tasks by observing demonstrations. The primitive actions which are trajectory-based skills are learned using IL. VSL learns the sequence of operations through visual perception. VSL not only extracts the spatial relationships among objects and learns the sequence of operations, but also identifies the underlying constraints of a task and extracts symbolic predicates. A standard action-level planner has been utilized to represent a symbolic description of the skill, which allows the system to generalize the skill into a symbolic form. In the performed experiments we have shown that although the symbolic planner provides the proposed approach with more generalization capabilities, when the order of actions is important, the planner can still benefit from this feature of VSL.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, vol. 25, no. 5, pp. 581–603, 2011.

[3] K. Kronander and A. Billard, "Online learning of varying stiffness through physical human-robot interaction," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1842–1849.

[4] D. C. Bentivegna, C. G. Atkeson, A. UDE, and G. Cheng, "Learning to act from observation and practice," *International Journal of Humanoid Robotics*, vol. 1, no. 04, pp. 585–611, 2004.

[5] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 5239–5246.

[6] D. Verma and R. Rao, "Goal-based imitation as probabilistic inference over graphical models," *Advances in neural information processing systems*, vol. 18, pp. 1393–1400, 2005.

[7] N. Dantam, I. Essa, and M. Stilman, "Linguistic transfer of human assembly tasks to robots," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 237–242.

[8] C. Chao, M. Cakmak, and A. L. Thomaz, "Towards grounding concepts for transfer in goal learning from demonstration," in *Development and Learning (ICDL), 2011 IEEE International Conference on*, vol. 2. IEEE, 2011, pp. 1–6.

[9] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observation of human performance," *Robotics and Automation, IEEE Transactions on*, vol. 10, no. 6, pp. 799–822, 1994.

[10] M. Lopes and J. Santos-Victor, "Visual learning by imitation with motor representations," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 35, no. 3, pp. 438–449, 2005.

[11] S. R. Ahmadzadeh, P. Kormushev, and D. G. Caldwell, "Visuospatial skill learning for object reconfiguration tasks," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 685–691.

[12] A. K. Pandey, M. Ali, M. Warnier, and R. Alami, "Towards multi-state visuo-spatial reasoning based proactive human-robot interaction," in *Advanced Robotics (ICAR), 2011 15th International Conference on*. IEEE, 2011, pp. 143–149.

[13] S. R. Ahmadzadeh, P. Kormushev, and D. G. Caldwell, "Interactive robot learning of visuospatial skills," in *Advanced Robotics (ICAR) 2013, 16th International Conference on*. IEEE, 2013, pp. 1–8.

[14] Y. Chen, C.-W. Hsu, and B. W. Wah, "Sgplan: Subgoal partitioning and resolution in planning," *Edelkamp et al.(Edelkamp, Hoffmann, Littman, & Younes, 2004)*, 2004.

[15] M. Pardowitz, R. Zollner, and R. Dillmann, "Incremental acquisition of task knowledge applying heuristic relevance estimation," in *Robotics and Automation (ICRA), 2006 IEEE International Conference on*. IEEE, 2006, pp. 3011–3016.

[16] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal, "Learning and generalization of motor skills by learning from demonstration," in *Robotics and Automation, 2009. (ICRA). IEEE International Conference on*. IEEE, 2009, pp. 763–768.

[17] N. Kruger, J. Piater, F. Worgotter, C. Geib, R. Petrick, M. Steedman, T. Asfour, D. Kraft, B. Hommel, A. Agostini, *et al.*, "A formal definition of object-action complexes and examples at different levels of the processing hierarchy," *Computer and Information Science*, pp. 1–39, 2009.

[18] O. Kroemer and J. Peters, "A flexible hybrid framework for modeling complex manipulation tasks," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1856–1861.

[19] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, "Learning the semantics of object–action relations by observation," *The International Journal of Robotics Research*, pp. 1–29, 2011.

[20] S. Ekvall and D. Kragic, "Robot learning from demonstration: a task-level planning approach," *International Journal of Advanced Robotic Systems*, vol. 5, no. 3, pp. 223–234, 2008.

[21] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2001, pp. 752–757.

[22] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[23] D. A. Randell, Z. Cui, and A. G. Cohn, "A spatial logic based on regions and connection." *Knowledge Representation and Reasoning*, vol. 92, pp. 165–176, 1992.

[24] Video, "Video accompanying this paper, available online," http://kormushev.com/goto/ICRA-2015_Reza, 2014.