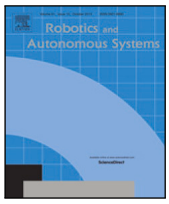
Robotics and Autonomous Systems 193 (2025) 105071

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



End-to-end action model learning from demonstration in collaborative robotics

Andrea Maria Zanchettin

Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza Leonardo da Vinci, 32, Milano, Italy

ARTICLE INFO

Keywords:
Collaborative robotics
Symbolic AI
Programming by demonstration
Action model learning

ABSTRACT

Access to advanced technology is crucial across all engineering disciplines. In the realm of industrial automation, collaborative robotics serves as a key solution, particularly for small or medium-sized enterprises facing frequent shifts in production demands.

This paper introduces a Symbolic Programming by Demonstration approach to efficiently configure and operate a collaborative robotics workstation. While motion profiles (i.e., the *how*) are taught through the commonly used lead-through programming method, the conditions to check before the execution of a motion and its impact on the environment (the *when* and *what*, respectively) are automatically derived using visual feedback. Differently from related works, the present methodology does not require a pre-compiled domain knowledge to encode the semantic characterisation of a demonstrated action (i.e., preconditions and effects).

An industrially-relevant use-case, consisting in a collaborative robotics assembly application, is introduced to validate the approach. Results show high success rates in interpreting and solving user-defined tasks (i.e., goals) as well as the capability of the method to generalise well in situations never seen during the acquired demonstrations.

1. Introduction

Action Model Learning (AML) [1] is a branch of symbolic Artificial Intelligence (AI) studying how to allow an agent to discover modular models of atomic actions that can be composed to form more complicated tasks. In a nutshell, considering a trace, or trajectory, i.e. a sequence of possibly partially observable states, the key problem of AML is to model the actions that have been applied in between every state transition.

Results from the literature can be roughly arranged in terms of two major characteristics: the level of observability of the state (whether or not all the literals determining the state are accessible), and whether the dynamics of the system has to be considered deterministic (single outcome of an action) or stochastic (multiple possible outcomes according to a certain probabilistic distribution). All options have been somehow investigated in the literature. A method for AML in fully observable settings is reported in [2]. Partial observability of states has been handled, e.g., in [3]. The work from Aineto et al. [4] also documents the extreme case of minimal observability, i.e. when traces only comprise the initial and final states. Probabilistic models have been also intensively investigated both in fully- [5] and partially-observable [6,7] settings. Recently, researchers have started focusing on online (one-shot or few-shots) learning, [8], i.e. incrementally learning or generating action models along with the execution of a plan.

When applied to robotics, plans cannot be executed by the physical agent before actions are correctly modelled. Using Programming by Demonstration (PbD, [9]), physical human demonstrations directly executed on the robot can replace planning traces [10]. Alternatively, some authors have also investigated the use of virtual reality, [11,12]. Most of the available AML methods require a predefined set of predicates, i.e. a Domain Knowledge (DK), and cannot operate in the absence of a suitable DK. As a consequence, many works in robotics adopt well-engineered and application-dependent sets or libraries of predicates, see e.g. [13,14]. Many efforts have been spent by the research community to semantically describe the spatial relationship between objects so to increase the modelling capabilities. Results range from pure geometrical features [15] to more articulated relations inferred directly from sensing devices [16,17]. The design of a proper set of predicates that is possibly portable among different domains is anyhow a time-consuming and difficult task and therefore still constitutes an open research question.

This paper proposes a method for AML that does not require a predefined and application-dependent domain knowledge. Starting from the observation that many research works use vision to infer properties of the environments in terms of predicates, the key idea is to completely avoid the need for such a domain knowledge and investigate whether it is still possible to effectively learn action models.

E-mail address: andreamaria.zanchettin@polimi.it.

https://doi.org/10.1016/j.robot.2025.105071

Available online 4 June 2025

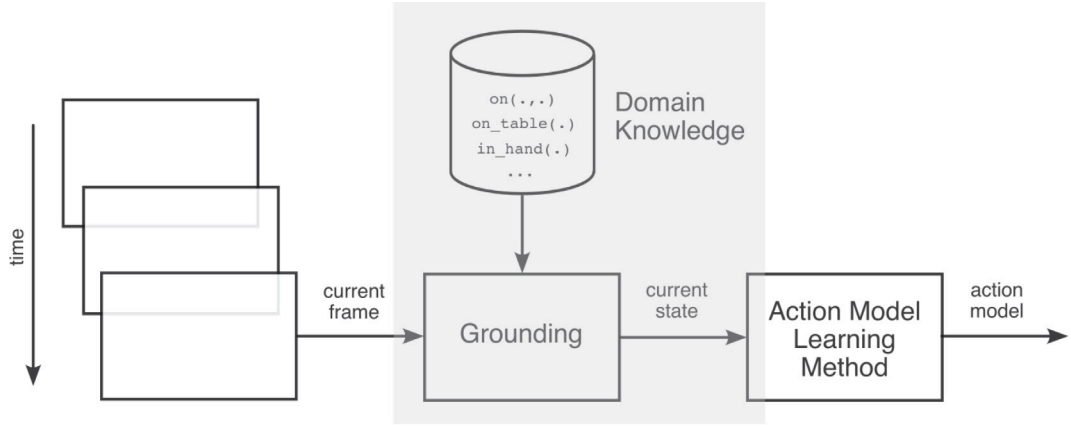


Fig. 1. Schematic approach of state-of-the-art methods for AML as applied to robotics: a Domain Knowledge has been pre-engineered and is composed by predicates. Vision is adopted to ground (variables are substituted with constants) such predicates and to form a set of atoms composing the current state which is eventually used (together with previous states) to learn the action model.

The remainder of this work is organised as follows. Background and relevant works are described in Section 2, together with a description of the problem this work is aiming to solve. Section 3 details the main contribution of the work, presenting how visual feedback can be directly used to extract action models without the need of a pre-defined domain knowledge. Possible limitations of the method are discussed in Section 5, whilst the outcome of validation experiments is presented in Section 4. Finally, concluding remarks are given in Section 6.

2. Background and related materials

The foundations of AML lie in first-order languages comprising finitely many predicates and constants, as well as infinitely many variables, [18]. A predicate is a parametrised proposition, which can be instantiated (grounded) by binding variables to proper constants forming an atom. For example, the predicate `on(x,y)` instantiated with `x=cup` and `y=tray` becomes `on(cup,tray)` and indicates whether or not the object (constant) `cup` is `on` the object (constant) `tray`. A set of grounded predicates (atoms) representing what is currently true encodes the state. A sequence of visited states finally forms a trace (or trajectory). The key problem in AML is to model the transitions from a state to another in terms of operators which are characterised by finitely many variables, finitely many predicates using (a portion of) such variables to identify preconditions, as well as finitely many predicates using (a portion of) such variables to identify (positive and negative) effects. Finally, a grounded operator forms an action.

Consider an action in the form $a = \langle \mathcal{P}, \mathcal{A}, D \rangle$, where $\mathcal{P}, \mathcal{A}, D$ are sets of atoms identifying preconditions, positive (add) and negative (delete) effects, respectively, and are such that $D \subseteq \mathcal{P}, \mathcal{A} \cap D = \emptyset$. Then, assuming s being the current state, provided that $\mathcal{P} \subseteq s$ (preconditions $\mathcal{P}$ of action a are *all* satisfied in current state s), the evolution of the system in terms of the state s' next to the execution of action a is given by

$$s' = (s \setminus D) \cup \mathcal{A} \tag{1}$$

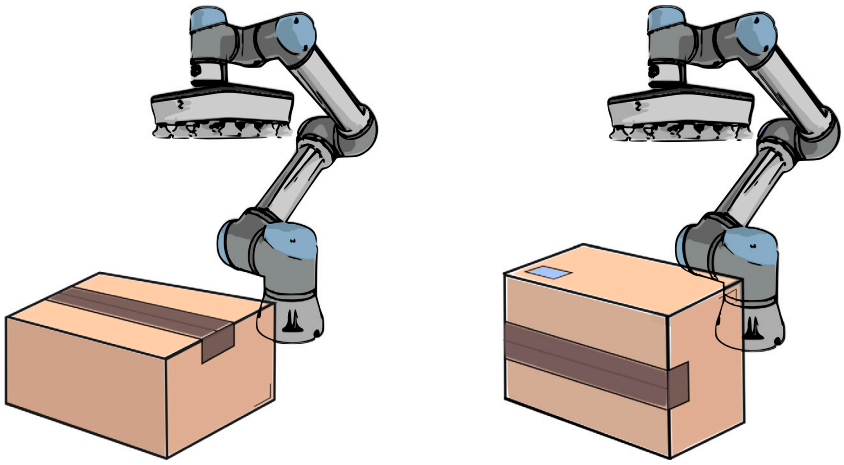
Planning is finally associated to the problem of computing the set of actions to be applied to reach a goal state s_{goal} from an initial state s_{init}, [18].

The first attempts in symbolic-level representations of actions from demonstrated robotic tasks are dating back to the mid 2000's. The generation of a proper encoding of the state as well as the discovery of constraints on the demonstrated actions has been proposed in [19]. Since then, the majority of works assumed a certain vocabulary of predicates (the Domain Knowledge) to encode the state to be given. Fig. 1 tries to summarise the concept behind the majority of state-of-the-art approaches.

Abdo et al. [20], as well as more recently Liang et al. [21] and Diehl et al. [11], adopted hand-crafted predicates somehow borrowed from the *BlocksWorld* domain. Their grounding along the demonstration is used to identify preconditions and effects. Zeng at al. [22] proposed to extract information regarding the initial and the goal state from RGB-D images to feed a planner. The work from Mao et al. [23] proposes a TAsk and Motion Planning (TAMP) framework to learn classifiers (grounding methods for predicates) from observation. Other recent works have introduced additional granularity in the definition of an action by means of Behaviour Trees (BTs), [24,25]. The use of BT has been recently extended to hybrid position/force-controlled demonstrations, [26].

As noticed by Konidaris et al. in [27], in most of the research works in robotics and AI, symbolic vocabulary, i.e. the domain knowledge, is given to the robot by a designer, see again Fig. 1. To avoid the need for a pre-engineered vocabulary, the authors propose to learn classifiers of sensing data for the automatic encoding of the state. The work from Xie et al. [28] is also based on the assumption that the effects of an action can be extracted in terms of what changes between visual observations. These action-dependent changes, named Visual Rewrite Rules (VRRs), probably borrowed from Graphical Rewrite Rules proposed in [29], are then used to describe the action at semantic level.

More recently, Deep Learning (DL) techniques have been exploited to characterise actions at semantic level. For example, in Ma et al. [30] demonstrations are used to train a visual predictive model to infer the visual representation of the environment after the execution of an action (i.e. its effects). These predictive models are used for planning in the execution phase. A similar approach, but performed in a simulated environment where the robot can learn from synthetic demonstrations, has been proposed in [31]. Similarly, Bagatella et al. [32] proposed to learn a Reinforcement Learning (RL) policy directly encoding in a lower-dimensional space images immediately before and after the execution of an action. Finally, inspired by [33], Ahmetoglu et al. [34] suggested to train an encoder–encoder network with a Boolean latent space which encodes the expected outcome of an action given a certain initial state. Lastly, along with the increasing diffusion of Vision Language Models (VLMs), several research teams are addressing the problem of learning planning domains and/or problems with the use of VLMs. For example, Huang et al. [35] propose to retrieve a plan given textual instruction and a point cloud of the scene. Athalye, Kumar and co-workers [36] as well as Liang et al. [37] recently proposed two similar neurosymbolic learning frameworks allowing an agent to propose and ground predicates from visual information with the support of a pre-trained VLM. Finally, Han et al. [38] present a framework based on VMLs to iterative learn predicates with explicit human feedback.



(a) Before the demonstration (b) After the demonstration

Fig. 2. Example of initial and final states of a demonstrated action consisting in tilting a carton box (to make the logo of the shipping company visible).

In the view of the discussion above, this paper attempts to tackle the AML problem in robotic manipulation starting from physical demonstrations:

- without the need for a pre-compiled Domain Knowledge, or the collection of an application-specific training set;
- from visual observations;
- from scratch and in a one-shot fashion.

In a nutshell, the method proposed in this paper is capable to derive, from demonstration, both the symbolic description and the domain of the demonstrated actions, without any prior information and relying on sensing data only. The human will be responsible for identifying and breaking down atomic demonstrations, which a planner will later use as operators. Alongside the physical demonstration, the robotic agent will learn a model of each action by visually analysing the changes in the scene.

With reference again to Fig. 1, the key contribution is to eliminate the operations highlighted in shaded grey and derive online actions model directly from visual observations of the workspace. Applied to manufacturing, the method is particularly suited for setting-up or reallocating a collaborative robotics applications, without the need for the complex engineering task of (re)designing the domain knowledge.

The idea is indeed quite similar to the ones in [27,28]. On the other hand, the former does not provide a framework from one-shot learning, the latter has been applied to train an agent to play Sokoban and MiniGrid and has never be extended to physical agents. In turn, the major drawback of approaches based on DL and latent binary variables is mainly related to selection of the dimension of the latent space which is highly application-dependent, [33,34]. In addition, latent variables encoding the state of the environment lacks of interpretability as their relation to spatial dimensions is difficult to extract. Huang et al. [35] assume a fixed set of predicates to learn action models from, hence limiting the applicability of the method to the describing capability of vocabulary of predicates. Han et al. [38] attempt to learn predicates from visual and textual inputs. The domain produced by the VLM requires human intervention in the form of textual feedback for the refinement of the model. Finally, the two works [36,37] adopt pre-trained VLMs to invent and ground semantically meaningful predicates without human intervention. While surely promising, they are both verified in simulated robotic domains, only.

3. Method for action model learning

In this Section, the method developed in this work to handle the AML problem is described. The method is split into two phases: the teaching part requires a human demonstrator to train the robot using the available lead-through programming, [39–41]. Action models are automatically learnt from these demonstrations using visual features. Once the teaching phase is completed, the robotic system is capable of autonomously plan actions to reach a goal state specified by a user. As already introduced in Section 2, the two functionalities (teaching and execution) are uniquely based on visual features acquired from a vision system and do not require any kind of pre-compiled predicate vocabulary or Domain Knowledge. As a matter of fact, it is the AML method that automatically selects, from visual feedbacks, what is relevant for the task and compiles the set of predicates to describe the demonstrated actions and encode the corresponding states.

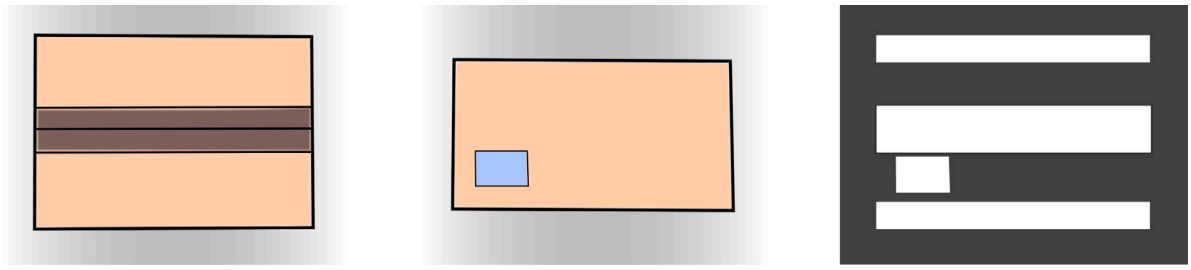
3.1. Teaching phase

Consider, as an illustrative example, the situations depicted in Fig. 2, where Fig. 2(a) and Fig. 2(b) sketch the states of the workspace before and after the demonstration, respectively. As one can notice, the demonstration is intended to teach the robot how to tilt a carton box. Following the best-practice, a domain expert would have first to design the set of application-dependent predicates to describe all possible scenes and states. In the given example, the condition before the demonstration, see Fig. 2(a) would be, e.g., rendered as $s = \{\texttt{box\_present}, \texttt{tape\_on\_top}\}$. In turn, the situation immediately after, i.e. the one in Fig. 2(b), would be identified by $s' = \{\texttt{box\_present}, \texttt{logo\_on\_top}\}$. Given the domain predicates, using the approach described in [10], a robotic programmer would automatically obtain a model of the tilting action in the following terms[1]:

```
; tilting a carton box (to make logo visible)
(:action action_000
 :precondition (
  and (box_present) (tape_on_top)
 )
 :effect (
  and (logo_on_top) (not (tape_on_top))
 )
)
```

meaning that the demonstrated action can be executed whenever a carton box is present in front of the robot with the tape on the top side, and the corresponding effect is to have the box rotated so to

[1] Without any lack of generality, PDDL [42] is used here and in the following.



(a) Before the demonstration (b) After the demonstration (c) Change mask

Fig. 3. Example of visual encoding of the states before and after the tilting action, and change mask.

have the logo of the shipping company visible. In the given example, predicates `box_present`, `tape_on_top`, `logo_on_top`, and probably others have to be carefully selected to accurately describe every possible state of the application.

As already anticipated, this work attempts to investigate whether the action model can be learnt without the need for a pre-compiled domain knowledge. In fact, the need for a pre-compiled set of atoms or predicates would require either a complete and application-agnostic vocabulary or an application-specific Domain Knowledge to be provided along with the application. From the one hand, it is hard to envisage an omni-comprehensive vocabulary of predicates. On the other hand, the need for the design of an application-specific Domain Knowledge is somehow voiding the ease of use of collaborative robotics which is specifically meant to facilitate the deployment of robots in Small and Medium-sized Enterprises (SMEs) possibly lacking of skilled engineering personnel.

The method developed to tackle this problem is based on the observation that all the relevant information to describe states, e.g. the ones in Figs. 2(a) and 2(b) in the example, are typically obtained from images (see again Fig. 1), and therefore images themselves are the most complete source of information to be used. Therefore, assuming one or more cameras to be available to detect changes in the working environment, one can simply describe the states before and after the demonstration in terms of one or more images taken from the same perspective.

With reference again to the example in Fig. 2, assuming the robot equipped with an eye-in-hand camera, the two images encoding the states before and immediately after the demonstration would be the first two from the left in Fig. 3. Assuming the images are taken from a fully informative perspective, there is no need to further encode the state information as the two images already account for all the needed details to represent the (tilting) action. What changes between the situations immediately before and after the demonstrated action can be obtained by comparing the two images and segmenting the difference image, rather than semantically describe the two situations and analyse the changing predicates, as the commonly adopted in the literature (refer again to Fig. 1). The formal reason behind this conjecture stands in the dynamics of Eq. (1). By rearranging the equation, one can easily notice that $s' \setminus s = \mathcal{A}$ while $s \setminus s' = \mathcal{D}$. Hence, what differs between s and s' can be used to characterise the action in terms of positive and negative effects, $\mathcal{A}$ and $\mathcal{D}$, respectively. As s and s' are no longer represented as sets of atoms (grounded predicates), the key idea is to use visual features to describe both $\mathcal{A}$ and $\mathcal{D}$.The image in Fig. 3(c) embeds all the relevant changes between the situation before and after the demonstration, i.e. Figs. 3(a) and 3(b), respectively. In particular, white pixels identify regions on the image where some modification has happened, while black pixels stand for unmodified regions.

Simply connected regions of white pixels can be segmented from the difference image to form binary masks. In the example of Fig. 3(c) four different binary masks can be generated. The superposition of each binary mask to each of the two taken images forms a set of masked images that will be used to derive a model of the action. With reference again to the example, four out of the eight possible situations are the ones reported in Fig. 4.

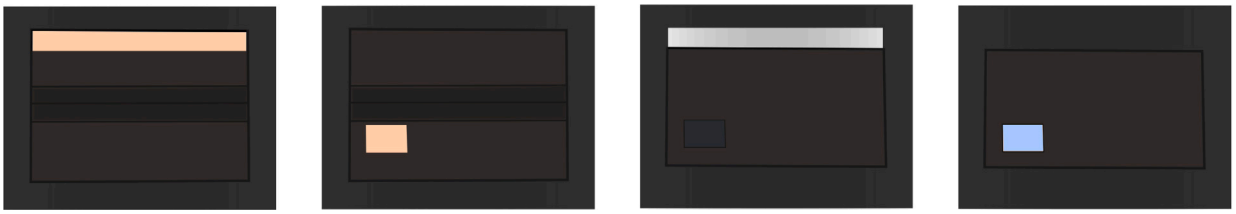
The calculation of the change mask, see Fig. 3(c), from two images, Figs. 3(a) and 3(b), deserves particular attention. In principle, a pixel-wise absolute difference can be computed and thresholded to form a binary image. Unfortunately, minor differences in illuminations, small shifts in the positioning of the camera with respect to the work-objects might substantially increase the number of false positives. To mitigate these possible drawbacks, the two images are first blurred by a Gaussian filter (with kernel $k \times k$ and variance σ^2). Then, after having computed the pixel-wise absolute difference, the resulting image is first eroded and then dilated in order to eliminate small differences at pixel level. Thresholding is applied to binarise the image and contour extraction is performed for the segmentation. The pseudo-code in Algorithm 1 summarises the method to extract the change masks which turned out to be slightly more robust than the one presented in [43].

Algorithm 1 Pseudo-code to compute the change masks between two images $\mathcal{I}_{pre}, \mathcal{I}_{post}$

```
function CHANGE_MASKS(I_pre, I_post)
    I_pre,filt ← gaussian_blur(I_pre, k × k, σ²)
    I_post,filt ← gaussian_blur(I_post, k × k, σ²)
    Δ ← |I_pre,filt − I_post,filt|
    Δ ← erode(Δ, e)
    Δ ← dilate(Δ, d)
    B ← thresholding(Δ, t)
    C ← external_contours(B)
    M ← {}
    for each c ∈ C do
        if area(c) > A_min then
            M.append(fill_with_ones(c, rows(I_pre), cols(I_pre)))
        end if
    end for
    return M, sum(Δ)/sum(B) < ε
end function
```

After the demonstration, for each connected region $M_i \in \mathcal{M}$ in the change mask, predicates in the form $\langle M, \mathcal{I} \rangle$ are automatically generated corresponding to $\langle M_i, \mathcal{I}_{pre} \rangle$ and $\langle M_i, \mathcal{I}_{post} \rangle$. Predicates $\langle M_i, \mathcal{I}_{pre} \rangle$ are representative of the visual features of the workspace that are no longer present after the execution of the demonstrated action (i.e. pre-conditions, $\mathcal{P}$, and negative effects, $\mathcal{D}$). In turn, predicates $\langle M_i, \mathcal{I}_{post} \rangle$ represent the features of the workspace that are present after the demonstration but were not present before (positive effects, $\mathcal{A}$). In the examples of Fig. 4, the image in Fig. 4(a) (`predicate_000`) corresponds to predicate `box_present` of a hand-crafted domain vocabulary, while similarly the image in Fig. 4(d) (`predicate_006`) stands for the predicate `logo_on_top`, etc. In summary, the action demonstrated in Fig. 2 will be eventually represented by the following schema:

```
(:action action_000
 :precondition (
```



(a) predicate_000 (b) predicate_002 (c) predicate_004 (d) predicate_006

Fig. 4. Four out of the eight possible combinations of binary masks and images taken before and after the demonstration.

```
  and (predicate_000) (predicate_001)
  (predicate_002) (predicate_003)
 )
 :effect (
  and (predicate_004) (predicate_005)
  (predicate_006) (predicate_007)
    (not (predicate_000)) (not (predicate_001))
    (not (predicate_002))
    (not (predicate_003))
 )
)
```

In order to avoid the proliferation of predicates in case of multiple demonstrations, Algorithm 2 is introduced. Given two predicates $p_1 = \langle M_1, \mathcal{I}_1 \rangle$ and $p_2 = \langle M_2, \mathcal{I}_2 \rangle$ returns whether the two can be considered similar or not. First, the two masks are compared in terms of Intersection over Union (IoU), then if they are substantially overlapping, the contents of the images are compared. If this check returns a similarity flag the two predicates p_1 and p_2 are considered similar to each other. In case predicate p_1 is already present in the vocabulary, predicate p_2 is not included, being considered a potential duplicate.

Algorithm 2 Pseudo-code to evaluate whether two predicates p_1 and p_2 are similar

```
function ARE_SIMILAR(p1, p2)
    IoU ← countNonZero(p1.M ∧ p2.M)/countNonZero(p1.M ∨ p2.M)
    if IoU > 0.95 then
        Δ ← |p1.M ∧ p1.I − p2.M ∧ p2.I|
        Δ ← erode(Δ, e)
        Δ ← dilate(Δ, d)
        B ← thresholding(Δ, t)
        return sum(Δ)/sum(B) < ϵ
    else
        return false
    end if
end function
```

3.2. Execution phase

At the end of the teaching phase, a vocabulary $\mathbb{P}$ of predicates as well as actions making use of these predicates have been automatically populated. Predicates $p = \langle M, \mathcal{I} \rangle \in \mathbb{P}$ are 0-ary (no variables are concerned) symbols containing visual features in the form of an image $\mathcal{I}$ and a binary mask M. As already introduced in Section 2, actions contain atomic formulas and are represented as $a = \langle \mathcal{P}, \mathcal{A}, \mathcal{D} \rangle$. By construction, the elements of $\mathcal{D}$ are those in $\mathcal{P}$ negated.

In order to complete (and solve) a planning problem, the initial and goal states s_{init} and s_{goal} as well as a planning method are needed. Consistent with the proposed method to learn action models, and similarly with the approach in [22], states are extracted from images. Hence, an image $\mathcal{I}_{init}$ taken from the current arrangement of the workspace can be used to encode the initial state s_{init} in terms of the available predicates (masked images). To this end, for every predicate $p = \langle M, \mathcal{I} \rangle \in \mathbb{P}$, Algorithm 1 is run using $p.\mathcal{I} \wedge p.M$ and $\mathcal{I}_{init} \wedge p.M$ as inputs with $\text{sum}(\Delta)/\text{sum}(B) < \epsilon$ being the returned Boolean output to ground the logical value of the given predicate p. In other words, the mask $p.M$ associated to predicate p is applied to the acquired image $\mathcal{I}_{init}$ and the result $\mathcal{I}_{init} \wedge p.M$ is compared to the image $p.\mathcal{I} \wedge p.M$ identifying the symbol p. If the sum of non-zero pixels $\text{sum}(\Delta)$ normalised by the count of non-zero pixels $\text{sum}(B)$ is higher than threshold ϵ, predicate p is grounded true, false otherwise.

Similarly, the goal state s_{goal} can be acquired either by human demonstration, or be selected within a database of previously acquired configurations of the workspace.

Given the vocabulary $\mathbb{P}$ of predicates forming the automatically derived domain knowledge, and since every predicate takes no variables as arguments, the initial, the goal, and any intermediate state can be represented as a Boolean string of length equal to the cardinality $|\mathbb{P}|$ of $\mathbb{P}$. Similarly, the possibility to execute a demonstrated action $a = \langle \mathcal{P}, \mathcal{A}, \mathcal{D} \rangle$ from the current state s, as well as the calculation of the corresponding outcome s', can be managed with basic Boolean algebra, provided that the sets $\mathcal{P}, \mathcal{A}$ and $\mathcal{D}$ are also represented in terms of Boolean strings of length $|\mathbb{P}|$. As a consequence, any algorithm for graph search can be adopted as candidate planner. In this paper, we adopt a slightly modified version of the Breadth-First Search (BFS), [44, Chapter 3.3]. Such a modification only entails the termination condition that consists in a weaker check in the termination condition. The search terminates if all predicates whose logical value in s_{goal} is true are also true in the considered state s, i.e. $s_{goal} \Rightarrow s$ (in turn, predicates whose logical value in s_{goal} is false are considered as "*don't care*").

4. Verification experiments

The verification scenario entails a Universal Robots UR5e robot equipped with an eye-in-hand Luxonis OAK-D Pro camera and a Robotiq Hand-E electrical gripper, see Fig. 5. All the algorithms have been coded in Python and run on an external PC interfaced with the robot using its Remote Operation interface.[2] The robot is adopted for the assembly of electrical buttons. The case has to be collected from a gravity feeder and then accommodated into the red fixture in front of the robot. Red and green buttons (of four different kinds: plain, off, on, and start) are available on corresponding feeders and have to be positioned in one or the other hole of the case, depending on the production requirements.

Experiments have been divided into four different scenarios of increasing complexity. Every scenario started from scratch, i.e. without accounting for previous demonstrations.

In the first scenario, the pick and place operation of the case has been demonstrated and the corresponding execution has been tested 10 times.

[2] https://www.universal-robots.com/articles/ur/interface-communication/remote-operation-of-robots/

Fig. 5. Setup for the experimental validation. The case has to be picked from the bottom of the gravity feeder (right) and positioned on the red fixture. Four different buttons are arranged into four gravity feeders (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Summary of the experimental outcomes: demonstrated skills, corresponding number of predicates, number of states reachable from the initial condition (the one reported in Fig. 5), and the success rates in the execution (* this state has never been seen by the system during the demonstrations).

Demonstrations	Actions	Predicates, $\lvert\mathbb{P}\rvert$	States	Goal	Success
pick/place of the case	1	2	1		10 out of 10
pick/place of the case pick/insert OFF BUTTON in left hole	2	4	2		9 out of 9
pick/place case pick/insert OFF BUTTON in left hole pick/insert ON BUTTON in right hole	3	6	4	  	9 out of 9* 10 out of 10 10 out of 10
pick/place case pick/insert ON BUTTON in right hole pick/insert OFF BUTTON in left hole (OFF BUTTON manually removed) pick/insert START BUTTON in left hole	4	8	6	   	9 out of 9 9 out of 9 9 out of 9 9 out of 9*

In the second scenario, the demonstration given in the first scenario is repeated and a new demonstration has been added, consisting in the pick and insertion of the OFF BUTTON (red) into the left hole of the case. The overall assembly task (pick and place of the case and insertion of the OFF BUTTON) has been tested 9 times.

In the third scenario, the pick and insertion of the ON BUTTON has been demonstrated and the two demonstrations already given in the previous scenario are demonstrated again. A total number of 29 executions have been tested for 3 different goals (ON BUTTON in the right hole only, OFF BUTTON in the left hole only, and both buttons in the corresponding holes). Notice that the goals to be reached are given to the planner in a cyclic manner, and not as repetitions of the same task in a batch.

Finally, after repeating the three demonstrations from the previous scenario, the OFF BUTTON was manually removed from its designated spot to reset the system, allowing a new demonstration to be performed. Then, the pick and insertion of the START BUTTON in the left hole has been demonstrated. Correspondingly, 36 executions for 4 different goal situations (9 times each) have been tested.[3]

For each scenario, the number of demonstrations/actions, the number of predicates available within the system after each set of demonstrations as well as the goals are summarised in Table 1, together with the success rate of the execution.

Overall, the method successfully achieves the required goals in all the tested scenarios (84 times out of 84 run). From Table 1, a linear dependency between the number of actions and the number of predicates can be appreciated. Hence, the method seems to scale linearly as the complexity of the application increases.

[3] A video recording of the demonstrations of the fourth and last scenario, as well as of the first three test executions is available at https://youtu.be/blCm3HGACoM.

Fig. 6. The eight predicates available after the demonstrations given in the last scenario (from `predicate_000`, left, until `predicate_007`, right), with reference to the last rows of Table 1.

Notice that the first goal in the third scenario (left spot empty and ON BUTTON in the right hole) as well as the last goal in the fourth scenario (OFF BUTTON in the left hole and right spot empty) require the robot to reach a state that has never been seen during the demonstrations. The success rate of these operations gives evidence of the generalisability of the proposed approach.

As one can notice from Fig. 6, which reports the eight predicates available after the demonstrations given in the last scenario, `predicate_004` (fifth from left) and `predicate_006` are visually very similar to each other. From the PDDL of actions reported below, one can further notice that these two predicates are always paired either in preconditions or effects.

```
; pick/place of the case
(:action action_000
 :precondition (predicate_000)
 :effect (
  and (predicate_001) (predicate_002)
  (predicate_004) (predicate_006)
    (not (predicate_000))
  )
)

; pick/insert ON button in right hole
(:action action_001
 :precondition (and (predicate_001)
 (predicate_002))
 :effect (and (predicate_003) (not (predicate_001))
 (not (predicate_002)))
)

; pick/insert OFF button in left hole
(:action action_002
 :precondition (and (predicate_004)
 (predicate_006))
 :effect (and (predicate_005) (not (predicate_004))
 (not (predicate_006)))
)

; pick/insert of START button in left hole
(:action action_003
 :precondition (and (predicate_004)
 (predicate_006))
 :effect (and (predicate_007) (not (predicate_004))
 (not (predicate_006)))
)
```

Therefore, one can surely assert that `predicate_004` $\Leftrightarrow$ `predicate_006`, hence either one predicate or the other can be safely removed. As a consequence, the method to avoid the proliferation of redundant predicates, i.e. Algorithm 2, has further margins for improvement.

Fig. 7 finally reports the graph representing all states that are reachable from the initial condition (the one shown in Fig. 5) in the fourth and last scenario. As one can see, the set of predicates, actions, and states is complete and consistent, though not minimal (because of the already mentioned redundancy of predicates).

5. Features and current limitations

The method presented in this work allows quick configuration and deployment of collaborative robotics applications using Symbolic Programming by Demonstration. End-users are just responsible for teaching the robot elementary actions to be suitably concatenated to perform more complex tasks. The system itself will be responsible for automatically extracting semantic features to derive preconditions and effects from. Domain Knowledge is no longer supplied through code, whether at the symbolic level (PDDL predicates) or the sub-symbolic level (grounding routines and motion-level instructions for the autonomous execution). Instead, all information is derived from visual data collected during the demonstration phase. Although demonstrations can be given in any order, it remains the demonstrator's responsibility to segment them correctly. In Fratini et al. [26], a way to automatically segment demonstrations based on "minimal semantic variations" has been proposed and can be surely adapted to the setting presented in this work in terms of "minimal visual variations", provided that the scene is observed by an off-hand camera as in Fratini and coworkers. Issues related to the correct segmentation of demonstrations fall however outside the scope of this paper.

Since experts' demonstrations are expected to be far from being perfect, repeated or redundant demonstrations, hence repeated or redundant planning operators causing a worthless increase branching factor, could be easily pruned with a simple post-processing of the planning domain.

Knowledge is automatically encoded into 0-ary predicates which dramatically simplify grounding and planning. On the other hand, scalability might be somehow sacrificed. The same action but applied in different positions would require brand new demonstrations. This limitation could be probably overcome by limiting visual features to a region of interest along with the use of template matching and/or 3D homography to translate the waypoints. This would however require to have a precisely calibrated stereo camera, or a 2.5D vision system, which are assumed here. Surely, a trade-off between the reduced number of demonstrations and the additional hardware-related cost has to be carefully analysed depending on the variability of the specific application. As a matter of fact, the method has returned slightly good scalability properties since the number of predicates $|\mathbb{P}|$ has been empirically shown to be linear with the number of demonstrations, in contrast with the number of reachable states which is equal to $2^{|\mathbb{P}|}$, at least in principle. Hence the addition of new demonstrations, and consequently of new predicates, would not substantially complicate the approach.

6. Conclusions

A Symbolic Programming by Demonstration approach has been introduced in this paper to efficiently configure and operate a collaborative robotics workstation. Trajectories and waypoints are recorded through the commonly used lead-through programming method, while the conditions to check before the execution of a motion and its impact on the environment are automatically derived from the comparison of images taken before and after the demonstration. The presented approach only relies on visual feedback and does not require a pre-compiled domain knowledge to encode the semantic characterisation of the states before and after each demonstration.

The approach has been validated within an industrially-relevant use-case, consisting in a collaborative robotics assembly application.

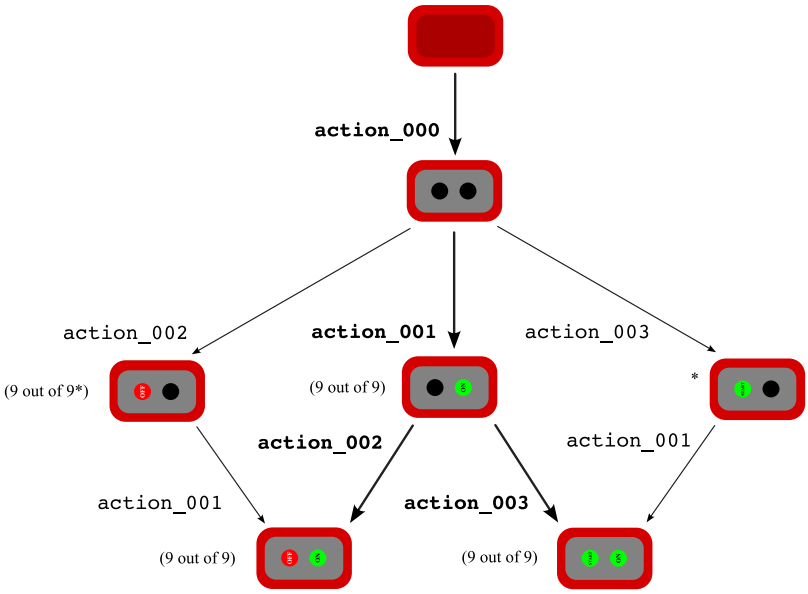


Fig. 7. Reachability graph of the fourth and last scenario (* state that has never been seen by the system during the demonstrations, demonstrations are highlighted in **bold**).

Experimental results confirms the robustness of the method as well as its capability to generalise in situations that were never explicitly demonstrated.

CRediT authorship contribution statement

Andrea Maria Zanchettin: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

I hereby declare NO Conflict of Interest.

Acknowledgements

This study was carried out within the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from Next-Generation EU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10-2022, PE00000004). CUP MICS D43C22003120001.

Data availability

No data was used for the research described in the article.

References

[1] A. Arora, H. Fiorino, D. Pellier, M. Métivier, S. Pesty, A review of learning planning action models, Knowl. Eng. Rev. 33 (2018) e20.
[2] Q. Yang, K. Wu, Y. Jiang, Learning action models from plan examples using weighted MAX-SAT, Artificial Intelligence 171 (2–3) (2007) 107–143.
[3] E. Amir, A. Chang, Learning partially observable deterministic action models, J. Artificial Intelligence Res. 33 (2008) 349–402.
[4] D. Aineto, S.J. Celorrio, E. Onaindia, Learning action models with minimal observability, Artificial Intelligence 275 (2019) 104–137.
[5] C. Rodrigues, P. Gérard, C. Rouveirol, Incremental learning of relational action models in noisy environments, in: Inductive Logic Programming: 20th International Conference, ILP 2010, Florence, Italy, June 27-30, 2010. Revised Papers 20, Springer, 2011, pp. 206–213.
[6] K. Mourão, L. Zettlemoyer, R.P. Petrick, M. Steedman, Learning STRIPS operators from noisy and incomplete observations, in: Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, 2012, pp. 614–623.
[7] J.A. Segura-Muros, R. Pérez, J. Fernández-Olivares, Learning numerical action models from noisy and partially observable states by means of inductive rule learning techniques, KEPS 2018 46 (2018).
[8] L. Lamanna, A. Saetti, L. Serafini, A. Gerevini, P. Traverso, et al., Online learning of action models for PDDL planning., in: IJCAI, 2021, pp. 4112–4118.
[9] A. Billard, S. Calinon, R. Dillmann, S. Schaal, Survey: Robot programming by demonstration, Springer Handb. Robot. (2008) 1371–1394.
[10] A.M. Zanchettin, Symbolic representation of what robots are taught in one demonstration, Robot. Auton. Syst. 166 (2023) 104452.
[11] M. Diehl, C. Paxton, K. Ramirez-Amaro, Automated generation of robotic planning domains from observations, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2021.
[12] N. Lucci, G.F. Preziosa, A.M. Zanchettin, Learning human actions semantics in virtual reality for a better human-robot collaboration, in: 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), IEEE, 2022, pp. 785–791.
[13] M. Aein, E. Aksoy, M. Tamosiunaite, J. Papon, A. Ude, F. Wörgötter, Toward a library of manipulation actions based on semantic object-action relations, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2013, pp. 4555–4562.
[14] F. Steinmetz, V. Nitsch, F. Stulp, Intuitive task-level programming by demonstration through semantic skill recognition, IEEE Robot. Autom. Lett. 4 (4) (2019) 3742–3749.
[15] S. Berretti, A. Del Bimbo, Modeling spatial relationships between 3d objects, in: 18th International Conference on Pattern Recognition, ICPR'06, 1, IEEE, 2006, pp. 119–122.
[16] E.E. Aksoy, A. Abramov, F. Wörgötter, B. Dellen, Categorizing object-action relations from semantic scene graphs, in: 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010, pp. 398–405.
[17] F. Ziaeetabar, E.E. Aksoy, F. Wörgötter, M. Tamosiunaite, Semantic analysis of manipulation actions using spatial relations, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 4612–4619.
[18] M. Ghallab, D. Nau, P. Traverso, Automated Planning: Theory and Practice, Elsevier, 2004.
[19] S. Ekvall, D. Kragic, Learning task models from multiple human demonstrations, in: ROMAN 2006-the 15th IEEE International Symposium on Robot and Human Interactive Communication, IEEE, 2006, pp. 358–363.
[20] N. Abdo, H. Kretzschmar, L. Spinello, C. Stachniss, Learning manipulation actions from a few demonstrations, in: 2013 IEEE International Conference on Robotics and Automation, IEEE, 2013, pp. 1268–1275.
[21] Y.S. Liang, D. Pellier, H. Fiorino, S. Pesty, Iropro: An interactive robot programming framework, Int. J. Soc. Robot. (2021) 1–15.
[22] Z. Zeng, Z. Zhou, Z. Sui, O.C. Jenkins, Semantic robot programming for goal-directed manipulation in cluttered scenes, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 7462–7469.
[23] J. Mao, T. Lozano-Pérez, J.B. Tenenbaum, L.P. Kaelbling, Learning reusable manipulation strategies, in: Conference on Robot Learning, PMLR, 2023, pp. 1467–1483.
[24] O. Gustavsson, M. Iovino, J. Styrud, C. Smith, Combining context awareness and planning to learn behavior trees from demonstration, in: 2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), IEEE, 2022, pp. 1153–1160.

[25] A. Jain, P. Long, V. Villani, J.D. Kelleher, M.C. Leva, Cobt: Collaborative programming of behaviour trees from one demonstration for robot manipulation, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2024.
[26] L. Fratini, N. Lucci, M. Malavenda, A.M. Zanchettin, P. Rocco, Semantic behaviour tree learning through kinesthetic demonstrations for position-force controlled robotic applications, in: Proceedings of the IEEE International Conference of Automation Science and Engineering, CASE, 2024.
[27] G. Konidaris, L.P. Kaelbling, T. Lozano-Perez, From skills to symbols: Learning symbolic representations for abstract high-level planning, J. Artificial Intelligence Res. 61 (2018) 215–289.
[28] Y. Xie, M. Li, S. Yu, M. Littman, Learning generalizable behavior via visual rewrite rules, 2021, arXiv preprint arXiv:2112.05218.
[29] A. Repenning, Bending the rules: steps toward semantically enriched graphical rewrite rules, in: Proceedings of Symposium on Visual Languages, IEEE, 1995, pp. 226–233.
[30] A. Ma, G. Chi, S. Ivaldi, L. Chen, Learning high-level robotic manipulation actions with visual predictive model, Complex & Intell. Syst. 10 (1) (2024) 811–823.
[31] R. Strudel, A. Pashevich, I. Kalevatykh, I. Laptev, J. Sivic, C. Schmid, Learning to combine primitive skills: A step towards versatile robotic manipulation, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2020, pp. 4637–4643.
[32] M. Bagatella, M. Olšák, M. Rolínek, G. Martius, Planning from pixels in environments with combinatorially hard search spaces, Adv. Neural Inf. Process. Syst. 34 (2021) 24707–24718.
[33] M. Asai, A. Fukunaga, Classical planning in deep latent space: Bridging the subsymbolic-symbolic boundary, in: Proceedings of the Aaai Conference on Artificial Intelligence, 32, (1) 2018.
[34] A. Ahmetoglu, M.Y. Seker, J. Piater, E. Oztop, E. Ugur, Deepsym: Deep symbol generation and rule learning for planning from unsupervised robot interaction, J. Artificial Intelligence Res. 75 (2022) 709–745.
[35] Y. Huang, C. Agia, J. Wu, T. Hermans, J. Bohg, Points2Plans: From point clouds to long-horizon plans with composable relational dynamics, 2024, arXiv preprint arXiv:2408.14769.
[36] A. Athalye, N. Kumar, T. Silver, Y. Liang, T. Lozano-Pérez, L.P. Kaelbling, Predicate invention from pixels via pretrained vision-language models, 2024, arXiv preprint arXiv:2501.00296.
[37] Y. Liang, N. Kumar, H. Tang, A. Weller, J.B. Tenenbaum, T. Silver, J.F. Henriques, K. Ellis, Visualpredicator: Learning abstract world models with neuro-symbolic predicates for robot planning, 2024, arXiv preprint arXiv:2410.23156.
[38] M. Han, Y. Zhu, S.-C. Zhu, Y.N. Wu, Y. Zhu, Interpret: Interactive predicate learning from language feedback for generalizable task planning, 2024, arXiv preprint arXiv:2405.19758.
[39] L. Bascetta, G. Ferretti, G. Magnani, P. Rocco, Walk-through programming for robotic manipulators based on admittance control, Robotica 31 (7) (2013) 1143–1153.
[40] M. Ragaglia, A.M. Zanchettin, L. Bascetta, P. Rocco, Accurate sensorless lead-through programming for lightweight robots in structured environments, Robot. Comput.-Integr. Manuf. 39 (2016) 9–21.
[41] F. Ferraguti, C.T. Landi, C. Secchi, C. Fantuzzi, M. Nolli, M. Pesamosca, Walk-through programming for industrial applications, Procedia Manuf. 11 (2017) 31–38.
[42] M. Ghallab, A. Howe, C. Knoblock, I.D. McDermott, A. Ram, M. Veloso, D. Weld, D.W. SRI, A. Barrett, D. Christianson, et al., PDDL: The Planning Domain Definition Language, Tech. rep., 1998.
[43] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[44] S.J. Russell, P. Norvig, Artificial Intelligence: a Modern Approach, Pearson, 2016.

Andrea M. Zanchettin was born in Cremona (Italy) in 1983. He received his Ph.D. in Information Technology, with honour, from Politecnico di Milano in 2012.

From January 2012 until February 2014 he has been a temporary research assistant at the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB). From March 2014 until September 2019 he has been a fixed-term assistant professor at DEIB where he is now a tenured Associate Professor.

His research interests are about mechatronic systems, automatic control, and intelligent human–robot interaction. Andrea Zanchettin has been member of the IEEE Robotics and Automation Society since 2009. Since 2017, he has been co-founder and co-chair of the IEEE RAS Technical Committee on Collaborative Automation for Flexible Manufacturing (CAFM). Dr. Zanchettin has been chair of the Italian Chapter of the IEEE RAS (I- RAS) from 2019 until 2023, as well as vice president for industrial activities of the Italian Institute of Robotics and Intelligent Machines (I-RIM). He is also co-founder and member of the Board of Directors of Smart Robots, a spin-off company of Politecnico di Milano.