Iterative affordance learning with adaptive action generation

Carlos Maestre, Ghanim Mukhtar, Christophe Gonzales, Stephane Doncieux

Iterative affordance learning with adaptive action generation

Carlos Maestre*, Ghanim Mukhtar*, Christophe Gonzales† and Stephane Doncieux*
* UMR 7222, ISIR, Sorbonne Universites, UPMC Univ Paris 06 and CNRS, ISIR, Paris, France
Email: {maestre, mukhtar, doncieux}@isir.upmc.fr
† UMR 7606, LIP6, Sorbonne Universites, UPMC Univ Paris 06 and CNRS, LIP6, Paris, France
Email: christophe.gonzales@lip6.fr

***Abstract*—A robot designer can provide a robot with knowledge to perform tasks on an environment. However, this approach can limit the achievement of future tasks executed by the robot. Providing it with the ability to develop its own skills paves the way for robots that are not limited by design. In this work a task consists in reproducing a given set of effects on an object. A robot must accomplish this task with limited information about the object, learning affordances to reproduce the effects, increasing this information throughout consecutive interactions with the object. We propose a method named Adaptive Affordance Learning (A^2L) which endows a robot with the capacity to learn affordances associated to an object, both adapting the robot's actions to the object position, and increasing the robot's information about the object when needed.**

This paper presents two main contributions: first, an online adaption of the robot actions to interact with the object, decomposing each action into a sequence of movements, adapting each movement, in a close loop, to the object position; and second, to increase the information about the object, we propose an iterative process that alternates between (1) exploration of the environment interacting with the object, (2) affordance acquisition and (3) affordance validation. These contributions are assessed in two experiments where a simulated Baxter robot learns to push a box to different positions on a table.

I. Introduction

Since the initial stages of their development infants interact with the objects around them, adapting their movements to the state of the objects, e.g. position and shape. Those interactions allow them to learn basic actions, e.g. grasping or pushing, and to build their *model of the world* in environments that constantly change, with limited information [1]. For example, pushing a toy and observing the change produced improves the capacity of the infant to interact with the environment and to predict the outcome of his actions. An affordance correlates an action to an object to produce a given effect on it [2]. This basic knowledge is the basis of the development of higher-level behaviours to perform tasks in posterior developmental stages, as planning at the age of 9 months or imitation at the age of 12 months [3].

Robots are expected to perform tasks in similar environments that those where infants develop their skills, relaying on affordance knowledge. In these environments, a robot designer may not know all the tasks that its robot will have to perform. We propose to define an approach in which the robot learns by its own the actions to perform a task adapted to its environment.

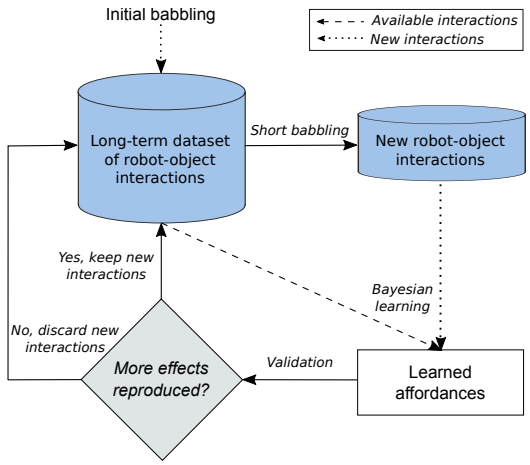

Fig. 1. Iterative process to learn the affordances of an object, enlarging its information about it. A full explanation is available in Section III-C.

In this work a task consists in reproducing a given set of effects on an object. A robot must accomplish this task with limited information about the object, i.e. the changes produced after the robot interacts with it in a certain way. The robot must learn the related affordances through the interactions with the object. We propose two approaches to tackle the execution of a task: (i) to interact with the object, the robot performs an online adaptation of its actions to the object position; and (ii) to increase its information about the object, the robot interacts with the object exploiting the available information about it.

This paper presents two main contributions addressing, respectively, the proposed approaches: first, to generate the online adaption, the robot executes actions composed of a sequence of movements, adapting each movement, in a close loop, to the object position; and second, to increase the information about the object, we propose an iterative process (Fig. 1) that alternates between (1) exploration of the environment, producing contacts between the robot and the object, (2) affordance acquisition and (3) affordance validation. Both contributions are encapsulated by the proposed method named Adaptive Affordance Learning (A^2L). At the beginning of each iteration, the method infers some actions executed by the end-effector of the robot, exploiting the available object information. An action can produce a change in the position

of the object. This information is stored in a dataset. Then, different affordances are learned based on the dataset content. The iterative process entails the validation of the learned affordances at the end of each iteration. Thus, for each effect to reproduce the method infers an action, i.e. a sequence of movements. If at the end of an iteration some effects have not been properly reproduced a new iteration of the process is executed.

The affordances generated by A^2L are intended to serve as basis of more complex behaviours in posterior higher level cognitive processes. These stages are expected to handle abstract concepts with discrete values instead of raw sensorimotor information. Therefore, to facilitate the reuse of the affordances generated, our method transforms the raw perception information of the robot into discrete representations, inferring discrete movements to interact with the object.

Two experiments are executed to asses each of the proposed contributions. In both experiments a simulated Baxter robot tries to push a box in different directions, reproducing a set of available effects. In the first experiment, given a specific interaction of the robot's end-effector with the box to reproduce each effect, the method is able to learn affordances adapting the actions to the object position. In the second experiment these interactions are learned from scratch by the robot interacting with its environment. Also, a simple test shows that the learned affordances can be reused in posterior high-level stages.

II. Related works

In most of the works within the affordance literature [4] an anthropomorphic robot or a robotic arm interacts with one or more objects to reproduce effects on it, learning their affordances. Initially the robot is endowed with a set of actions. Then, three phases are executed sequentially: (i) environment exploration, where the robot interacts with an object to gather information of the effects obtained after executing different actions; (ii) affordance learning, based on different learning methods, exploiting the previous information obtained; and (iii) learning assessment, executing a specific task or playing some imitation games to evaluate the capacity of the robot to reproduce some effects using the learned affordances.

In some works the set of actions is predefined, built-in to generate some effects with the objects in front of it. Fiztpatrick et al. [5] define 4 actions (*pull in, side tap, push away* and *back slap*) to interact with several objects from 4 initial positions. Their approach relies on the correlation between the orientation of an initial position and an object to select the action that it is more likely to make the object roll. Montesano et al. [6] provide *grasp, tap,* and *touch* primitives to play imitation games. In this work, the execution of the primitives depends on some free parameters of the action, as the height of the end effector of a robot related to the object. In other examples, Ugur et al. [7] propose a robot to learn the grasping parts of an object using a predefined *grasp* primitive. In more recent works, Ugur et al. define a set of primitive actions (*side-poke, top-poke, front-poke, stack*) to achieve high-level behaviours, as the learning of paired-object affordances [8], or the generalization in complex manipulation scenarios [9]. In these works the actions are adapted to the features of the objects, e.g. the actions are relative to the object position. However, a robot does not have the capacity to generate by itself another action to reproduce a different effect.

Other works endow the robot with skills to generate its own actions, based on some motion parameters common to all the actions, as the direction or the velocity of the end-effector while executing the actions. These actions are learned before the execution of the three aforementioned phases. Ugur et al. define a parametrized *swipe action* [10], [11] executing a trajectory. Different configurations of the execution parameters produce different primitives, as *push* and *grasp*, based on the effects produced to an object. In another work [12], different primitives are learned based on Dynamic Motion Primitives (DMP), in a pouring task experiment. In this work, affordances relate to the different subparts of an object, i.e, the robot must correlate the grasping action with the handle of an object. This work relies on learning by demonstration, providing a set of pouring examples using a large watering can, and generalizing to other objects. Despite the fact that in these works a robot can generate different actions to reproduce different effects, all the actions share some motion parameters. And thus the effects share some features, limiting the capacity of the robot to accomplish a task, e.g. all the effects are reproduced interacting with an object from the same side, as push to the left, poke to the left, etc.

In contrast to the previous works, A^2L does not create a set of primitives in an initial phase to interact with an environment in posterior sequential phases. In an iterative process, the method simultaneously learns to infer an action adapted to the object position while learning affordances. Instead of relying on some motion parameters, as in previous works, this learning approach endows a robot with the capacity to infer an action based on the physical relation between the robot and its environment, called *relation state*, e.g. pushing an object in a specific direction from different initial positions of the robot's end-effector. The selection of these relation states relates to the complexity of the expected effects reproduced, and not to a set of initial actions. For example, to push an object the relevant relation values are the distance and orientation between the robot's end-effector and the object.

III. Iterative developmental framework

Adaptive Affordance Learning (A^2L) is an iterative method to endow a robot with the capacity to learn affordances associated to an object, both adapting the robot's actions to the object position, and increasing the robot's information about the object when needed. These features allow the robot to reproduce effects on the object.

In this work, an action, a_e, is a trajectory of the robot's end-effector associated to an effect, e. An action is a sequence of movements. A movement, $\Delta_t^{(x)}$, is a displacement of the robot's end-effector between two subsequent instants of time:

$$x_t = \textit{end effector position}$$
$$\Delta x_t = x_t - x_{t-1}$$
$$a_e = \{ \Delta x_t \}$$

Actions are effect-oriented. Therefore, given an effect to reproduce (also called goal) an action generator infers sequentially the next movement to be executed to reproduce the effect until a displacement of the object is observed, or a maximum number of movements are inferred. The inference of a movement relies on the available knowledge of the object affordances, and the states of both the robot and the object at each instant of time.

The action generator and the affordances are learned within the execution of an iterative method to reproduce a set of effects. This method alternates between three steps during each iteration (see Fig. 1): (1) exploration of the environment, producing contacts between the robot and the object (gathering information about these interactions), (2) affordance learning (based on the previous information) and (3) affordance validation (trying to reproduce the available set of effects). The methods stops when all the effects have been reproduced, or a maximum number of iterations has been reached.

The method relies on available knowledge about (i) the robot itself, associated with the concept of *self* and basic motion skills, i.e. its kinematic model; and (ii) knowledge to interpret its context, i.e. the position of a surrounding object.

This section is split up into three parts: first, a description of the initial information available before the execution of A^2L; then, an explanation about how both the adapted actions and object affordances are simultaneously learned using the available information about the object (step 2). Then, the iterative process to increasingly learn to reproduce a set of effects is described (focusing on steps 1 and 3). The second and third parts corresponds, respectively, to the contributions defined in Section I.

A. Initial available information

Before the execution of the method, two sets of information are required: (i) an initial dataset representing some initial information about the object, outcome of the interactions of the robot with it, to bootstrap the learning process and to be increased in posterior iterations; and (ii) a set of effects to be reproduced by the robot, to asses the affordance learning at the end of each iteration.

Initial dataset: The dataset is environment-dependent, and it must be generated by the robot. It is composed of the raw perceptions perceived by the robot during an initial random, goal-free, babbling [13] of its environment (Algo. 1). The perception of the robot is composed of both the position of one of its end-effector's and the object position at an instant of time:

$$y_t = \textit{object position}$$
$$XY_k = \{ (x_0^k, y_0^k), \ldots, (x_{T_i}^k, y_{T_i}^k) \}$$
$$e = \textit{label associated to a specific effect}$$
$$\mathbb{D} = \{ (XY_k, e) \}$$

Algorithm 1 Initial random babbling

nb_int_p: number of current interaction executed
nb_int_{max}: maximum number of interactions
$\mathbb{D}_b$: raw dataset from babbling
y_p: position of the object at the end of iteration p
y_{p-1}: position of the object at the end of iteration p-1

1: nb_int_p is initially set to 1
2: $\mathbb{D}_b$ is initially empty
3: **while** $nb\_int_p \leq nb\_int_{max}$ **do**
4: $\quad XY_p \leftarrow GeneratedRandomTrajectory()$
5: $\quad y_p \leftarrow ExecuteTrajectory(XY_p)$
6: $\quad \Delta f_p \leftarrow y_p - y_{p-1})$
7: $\quad$ **if** $\Delta f_p \neq 0$ **then**
8: $\quad\quad e_p \leftarrow IdentifyEffect(\Delta f_p$
9: $\quad\quad \mathbb{D}_b \leftarrow AddTrajectoryToDataset(XY_p, e_i)$
10: $\quad Increase(nb\_int_p)$

where XY_i represents an interaction between the robot's end-effector and the object when executing a trajectory, and $\mathbb{D}$ represents a dataset composed of interactions. Each time the robot's end-effector reaches a waypoint of the trajectory the raw perception of the robot at that instant of time is stored.

The execution of a trajectory by a robot has a high temporal cost, possibly taking few seconds to complete it. The execution of an unconstrained random babbling by a robot could entail the execution of hundreds or thousands of trajectories, most of them not having any impact on the object. In this work, this issue has been addressed through the definition of a *virtual setup*, where the robot can compute a mathematical approximation of the possible outcome of the execution of a random trajectory. Only those trajectories interacting the object in this virtual space will be executed on the actual robot. An example of a virtual setup is depicted in Figure 2.

Set of effects: The set of effects to reproduce is externally provided:

$$f_t = y_t$$
$$\Delta f_t = y_t - y_{t-1}$$
$$E = \{ (e_1, \widehat{\Delta} f_1), \ldots, (e_N, \widehat{\Delta} f_N) \}$$

where f_t represents the *object feature vector* of the object at an instant of time, Δf_t represents a variation of the object feature vector between two instants of time, E represents the set of effects to reproduce, N is the number of effects, and $\widehat{\Delta} f_i$ represents the change of the object feature vector associated to the effect i, which is also provided to the robot.

B. Affordance learning

Actions produce changes in the feature vector of the object:

$$\Delta f_T^{a_e} = y_T^{a_e} - y_{T-1}^{a_e} \in S_f$$

where S_f represents the set of changes of the object position produced by the robot during the babbling.

An action is a *success*, if after its execution the change produced in the object relates to the desired effect:

$$\exists\, i \in [1,N],\ \exists\, t \in [0,T\text{-}1],\ \Delta f_t^{a_e} \in S_f,\ \Delta f_t \approx \widehat{\Delta} f_i,$$

a *false positive*, if the change relates to another effect:

$$\exists\ \mathrm{i} \in [1,\mathrm{N}],\ \exists\ \mathrm{t} \in [0,\mathrm{T\text{-}1]},\ \Delta f_t^{a_e} \in \mathrm{S_f},\ \Delta f_t \neq \widehat{\Delta} f_\mathrm{i}$$

or a *failure*, if there is no change in the position of the object:

$$\forall\ \mathrm{t} \in [0,\mathrm{T\text{-}1]},\ \nexists\ \Delta f_t^{a_e}$$

These results guide the extension of the information about the object within the iterative process (Section III-C). The dataset consequence of a babbling is composed of actions touching the object, generating both *success* and *false positive*.

At an instant of time, given a dataset of interactions between the robot's end-effector and the object, we have defined a process to learn the object affordances adapting the robot's actions to the object position (step 2 of the method). In this work an affordance is an action generator to reproduce an effect on an object. The generation of an action entails the inference of a sequence of movements of the robot's end-effector, each of them adapted to the object position:

$$\delta_\mathrm{t} \equiv \delta_\mathrm{t}(f_t,\ x_t)$$
$$\Delta \mathrm{x_t} = \phi(\mathrm{e},\ \delta_\mathrm{t})$$

where δ_t represents the physical relation between the robot's end-effector and the object at each instant of time, called *relation state*, and ϕ represents the action generator.

Therefore, an affordance associates a set of movements to an expected effect and the state of an object:

$$\alpha(e, f) = \{\ \Delta \mathrm{x_t}\ \} = \{\ \phi(e,\ \delta_\mathrm{t})\ \},$$

where the correlation between the effect, the object and the action, at each instant of time, relies on the action generator. This correlation has been extended to include the robot state in δ to adapt the robot actions to the object position.

In this work the robot learns to *push* an object to different positions. Hence the relevant relation state consists of the distance and orientation between the robot's end-effector and the object at each instant of time, i.e. the cylindrical coordinates of the vector with initial point the end-effector position, and final point the object position (see Fig. 2.):

$$\delta_\mathrm{t} = \mathrm{cyl}(\mathrm{y_t}\ -\ \mathrm{x_t})$$

From continuous actions to discrete affordances: In order to learn affordances that can be reused in posterior high-level stages, the method transforms the available raw information gathered by the robot into discrete representations, inferring discrete movements to interact with the object. As the effects are already labeled, only the movements of the robot and its relation to the object position must be discretized.

The discretization relies on a *discretization configuration*. This configuration may have a deep impact in the results obtained using the method. A suitable configuration must entail a trade-off between being generic to be suitable for different sets of effects, and specific enough for the current set of effects. In this work this configuration has been empirically designed, being available in Figure 2.

Algorithm 2 Iterative object information acquisition

nb_it_p: number of current iteration
nb_it_{max}: maximum number of iterations
x_I: initial position of the end-effector
ψ_p: score computed in an iteration
ψ_max: maximum score obtained
E: set of available effects
$\mathbb{D}_\mathrm{s}$: raw dataset from short babbling
D_s: discrete dataset from short babbling
D_p^*: current discrete extended dataset
BN_p: BN representing the affordance knowledge

1: nb_it_p is initially set to 1
2: ψ_max is initially set to 0
3: **while** $nb\_it_p \leq nb\_it_{max}$ **do**
4: $\quad \mathbb{D}_\mathrm{s} \leftarrow$ *ExecuteShortBabbling()*
5: $\quad D_\mathrm{s} \leftarrow$ *DiscretizeShortDataset*($\mathbb{D}_s$)
6: $\quad D_\mathrm{p}^* \leftarrow$ *AddToCurrentDiscreteDataset*(D_s)
7: $\quad BN_\mathrm{p} \leftarrow$ *LearnAffordances* (D_p^*)
8: $\quad \psi_\mathrm{p} \leftarrow$ *LearningAssessment*(E, BN_p, x_0)
9: $\quad$ **if** $\psi_\mathrm{p} \geq \psi_{\mathrm{p}\text{-}1}$ **then**
10: $\quad\quad \psi_\mathrm{max} = \psi_\mathrm{p}$
11: $\quad\quad \mathbb{D}_\mathrm{p} = \mathbb{D}_\mathrm{p}^*$
12: $\quad$ *Increase*(nb_it_p)

All movements and the related relation states are discretized. Then, both are stored together with the correlated effect into a discrete dataset:

$$\mathrm{D} = \{\ (\mathrm{e},\ \Delta \mathrm{x_t}\ ,\ \delta_\mathrm{t})\ \}$$

The generator of discrete actions, ϕ, is implemented as a Bayesian Network (BN) [14], similarly to the work of Montesano et al. [6]. The reasons for using BN are twofold: (i) it has a strong inference capability; and (ii) its representation of affordances as probabilistic dependencies allows one to analyze and understand the outcomes of learning.

The affordance learning consists in creating a BN representing the correlations among action, relation state and effect within the discrete dataset, comprising two steps: (i) the generation of a structure representing the causal relations of the components of an affordance, and (ii) the computation of their conditional probability distributions (CPDs). Therefore, the action generator is formalized as:

$$\phi(e, \delta_\mathrm{t}) = \arg\max_{\Delta x_\mathrm{t}} P(\Delta x_\mathrm{t} \mid e, \delta_\mathrm{t})$$

C. Iterative object information acquisition

The process presented in Section III-B allows the learning of the affordances of an object using a dataset representing interactions of the robot's end-effector with the object. However, given a set of effects, some of them could not be reproduced at an instant of time. The learning of new object affordances to reproduce those effects entails the extension of the information the robot has about the object. $\mathrm{A^2L}$ addresses this issue through an iterative process (Algo. 2) (i) increasing the dataset of interactions with new trajectories, at the beginning of each

Algorithm 3 Affordance learning validation

E: set of identified effects
BN_{p}: BN representing the current affordance knowledge
a_{e}: action inferred for effect e
x_0: initial position of the end-effector
x_{pq}: current position of the end-effector
y_{pq}: current position of the object
N: number of available effects
nb_mov_{q}: current number of movement
nb_mov_{max}: maximum number of movements to execute
θ: available discretization configuration
δ_{pq}: relation state based on environment and robot states
$\Delta \mathrm{x}_{\mathrm{pq}}$: discrete movement
virt_contact: indicates a prospective contact of the simulated end-effector with the object (virtual setup)
$\hat{e}$: expected effect
e_i: obtained effect
res_{e_i}: result of an effect
$w_{res_{e_i}}$: weight associated to a result
ψ_{p}: score obtained

function LEARNINGASSESSMENT(E, BN_{p}, x_0)
  ψ_{p} is initially set to 0
  for $\hat{e} \in E$ **do**
    $x_{pq} = x_0$
    nb_mov_{q} is initially set to 0
    $a_{\hat{e}}$ is empty
    while $\neg virt\_contact \cap nb\_mov_{\mathrm{q}} < nb\_mov_{\mathrm{max}}$ **do**
      $y_{pq} \leftarrow$ *GetObjectPosition()*
      $\delta_{\mathrm{pq}} \leftarrow$ *ComputeInteractionFeatures(x_{pq}, y_{pq})*
      $\Delta \mathrm{x}_{\mathrm{pq}} \leftarrow$ *InfereMovement(BN_p)*
      $a_{\hat{e}} \leftarrow$ AddMovementToAction($a_{\hat{e}}$, $\Delta \mathrm{x}_{\mathrm{pq}}$)
      SimulateMovementExecution(x_{pq}, Δx_{pq}
      Update end-effector position, $\mathrm{x}_{\mathrm{pq+1}} = \mathrm{x}_{\mathrm{pq}} + \Delta \mathrm{x}_{\mathrm{pq}}$
      virt_contact $\leftarrow$ *CheckObjectContact()*
      Increase(nb_mov_q)
    if *virt_contact* **then**
      $y_{a_{\hat{e}}} \leftarrow$ *ExecuteInferredAction($a_{\hat{e}}$)*
      $e_{a_{\hat{e}}} \leftarrow$ *IdentifyObtainedEffect($y_{a_{\hat{e}}}$)*
      if $e_{a_{\hat{e}}} = \hat{e}$ **then**:
        $res_{\hat{e}}$ = *success*
      else
        $res_{\hat{e}}$ = *false_positive*
    else
      $res_{\hat{e}}$ = *failure*
    The score is updated, $\psi_{\mathrm{p}} = \psi_{\mathrm{p}} + res_{\hat{e}} * w_{res_{\hat{e}}}$
  return ψ_{p} * N

iteration (step 1 of the method), and (ii) validating them later on concerning the number of effects reproduced (step 3).

To increase the dataset the robot executes a short babbling, i.e. a small number of trajectories, around the object generating interactions with it[1], increasing the available dataset, $\mathbb{D}_{\mathrm{i}}^{*}$. The short babbling consists in the random modification of the position of one or more waypoints of the trajectories within the available dataset, and in their posterior execution to generate new interactions. Those trajectories generating unforeseen effects are stored into the extended dataset, which is discretized, D_{i}^{*}. Then, the object affordances are learned using the extended discrete dataset (Section III-B).

The learning assessment of a extended dataset (Algo. 3) consists in the comparison of the number of effects reproduced before and after the extension. This comparison is based on the computation of a score, ψ_{i}:

$$\psi_{\mathrm{i}} = (\sum_{e \in E} res_{\mathrm{e}} * W_{res_e}) * N$$

where res_e is the result obtained after the execution of an action reproducing an effect, W represents the predefined weight associated to the result, and N the total number of effects.

If the score associated to the extended dataset (current iteration) is lower than the score associated to the dataset before the extension (previous iteration), $\psi_{\text{i-1}}$, then the current score is discarded together to the dataset extension. In contrast, if the current score is bigger or equal than the previous one, the extended dataset is used at the beginning of the next iteration as the available dataset to be increased.

IV. EXPERIMENTS

Two experiments are executed to asses each one of the contributions of this paper (Sections IV-B and IV-C). In both experiments a simulated Baxter robot tries to reproduce the same set of effects, i.e. pushing a box far from it (pushed_far), close to it (pushed_close), to the left (pushed_left), to the right (pushed_right) and their combinations (pushed_far_left, pushed_close_left, pushed_close_right and pushed_far_right). Thus a maximum of 8 effects can be reproduced. Next, it is tested if the actions generated in both experiments can be reused in posterior high-level stages (Section IV-D). The common features shared by the experiments are explained in Section IV-A. This sections finishes with an analysis of the results obtained. The code of the experiments is available online[2].

A. Experimental framework

The scenario simulates a three-dimensional (3-D) Cartesian setup composed by a table and a box. The robot is located in front of the table (Fig. 2). The dimensions of the box are 7 x 8.5 x 8 centimeters (cm) of width, length and height, respectively. The position of the box at the beginning of the experiment is at 65 cm in front of the robot, and 10 cm to the left. The origin of coordinates is located at the base of the robot, and thus the perceptions perceived by the robot are relative to itself. The only perceptual representation of the setup available for the robot is the position of the box. The position of the box is located at the center of the object. The box can change its position during the experiment. However, in order to allow the method to infer actions that the robot can execute, if the box is moved more than 10 centimeters away of its initial position it is automatically relocated around the initial position.

In both experiments the robot moves its left end-effector to interact with the box starting from two initial positions.

[1]Any change in the object position is assumed to be produced by the robot.

[2]https://github.com/robotsthatdream/a2l_exp_baxter_core

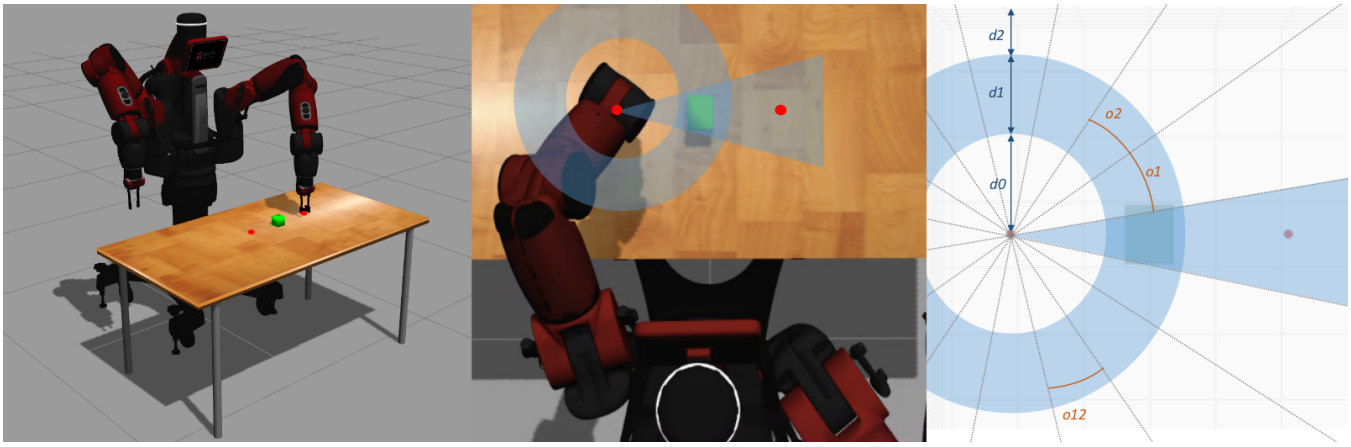

Fig. 2. At the left, a simulated Baxter robot in the setup used for the experiments. This setup is composed of a box on a table in front of the robot. At the center, top view of the same setup. The white circles represent both initial positions of the end-effector from where the actions are inferred and executed. The blue triangle and torus represent, respectively, the *orientation* and *distance* from the end-effector to the box. At the right, the *virtual setup* emulating the previous top view to compute the possible outcome of the execution of an action. In red the initial positions. In this work, the *distance* is discretized in three sections (dark blue text), and the *orientation* in 16 sections (brown text). In this example, the box is at *distance 1* and *orientation 0* of the end-effector.

These initial positions are located at 20 cm to the left of the box position and 20 cm to the right, respectively. Due to the use of a simple random heuristic action generation in Section III-C, the dimensionality of the actions has been constrained to a 2D space, i.e. with a constant Z value. Furthermore, these actions are composed of discrete movements, moving only a fix distance in horizontal (left, right), vertical (up, down) or diagonal (left-up, left-down, right-up and right-down). Examples of movements are shown on Fig. 3.

The displacements of the box are categorized as representing an effect, or discarded otherwise. A displacement is categorized within the individual effects (i.e. pushed far, close, to the left or to the right) if the displacement in one axis is at least three times bigger than in the other axis; and within the diagonal effects if the displacement in one axis is at least bigger than the half of the displacement in the other axis. The robot can execute a maximum of 16 actions (2 initial positions * 8 effects) during each iteration of the experiment.

In this work, the learning of affordances relies on two different methods, with different a priori information about the dependency among the elements involved in an affordance, i.e. a *movement*, the related *effect* and the *relation state*. First, a *hard-coded* method provides the complete structure of the BN, directly connecting the movement to each one of the other elements. Second, *K2* [15] provides an order of these elements to drive the learning of the structure. Once the structure is available, both methods learn the corresponding Conditional Probabilistic Distributions (CPDs). Providing the hard-coded method allows us to build a baseline to measure the learning capabilities of K2. The Bayesian learning is performed using the aGrUM[3] library.

The weights chosen for the learning assessment must reflect that a *successful* result of a trajectory is better than a *false positive*, and much better than a *failure*. Choosing different sets

[3]http://agrum.gitlab.io/

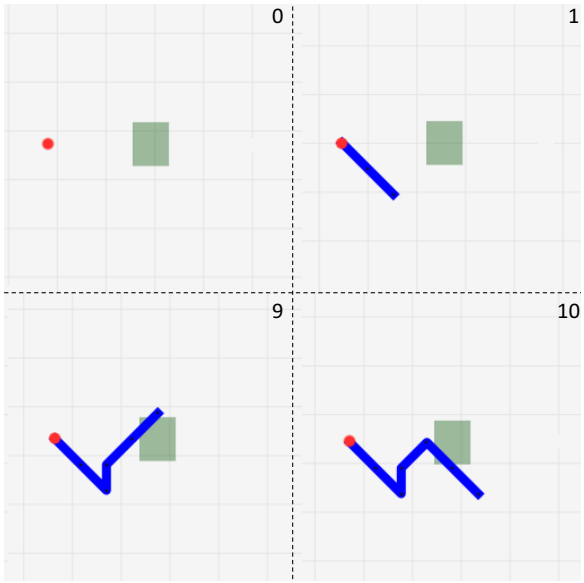

Fig. 3. Evolution of the result produced by the action generator, learned using the hard-coded method, to reproduce the *pushed_close_right* effect from an initial position (red circle). The numbers show the current iteration of the process when the action (blue line) was inferred. The action generator evolves from not inferring any movement (top-left) to infer an action producing the effect (bottom-right), going through intermediary step, as inferring an action not contacting the box (top-right) or reproducing another effect (bottom-left).

of values based on this principle does not have a relevant impact in the behaviour of the method. Based on experience we have selected a value of 4 for $W_{success}$, 1 for $W_{false\_positive}$, and 1 for $W_{failure}$. As the number of effects does not change along the experiment, the computation of the score (Equation III-C) only depends on the results of the trajectories. The value of the score is normalized between 0 and 100. Inferring the 16 successful trajectories aforementioned would produce the highest possible score 64.

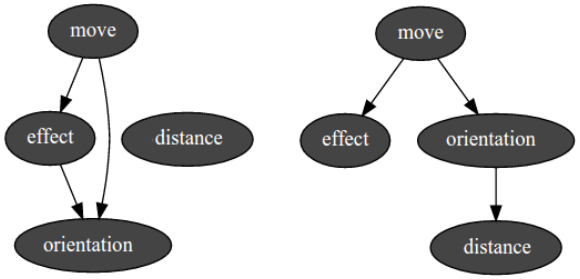

Fig. 4. Results of the BN structure learning of the K2 method: at the left, given the predefined interactions (Section IV-B); at the right using the random interactions obtained at the end of the iterative process (Section IV-C). In both cases the K2 has identified the direct dependency of the movement with the effect and the orientation, but it is missing the direct dependency with the distance.

B. Affordance learning experiment

In the first experiment a dataset of predefined interactions is provided, each one of them reproducing one of the effects (e.g. to reproduce the *pushed_left* effect the robot's end-effector, from each initial position, moves at a position 15 centimeters to the right of the center of the box, and then it moves 20 centimeters to the left). This dataset is used to asses the affordance learning capacity of the process described in Section III-B. It is expected to obtain very high success ratios, and thus very high scores.

C. Iterative object information acquisition experiment

In the second experiment the dataset is generated from scratch. An initial goal-free babbling is executed, before the execution of the method, generating several interactions between the robot and the box, stored into an initial dataset. This dataset is increased and evaluated at each iteration of the iterative process. An online video is available[4].

D. High-level stage example

In this simple test a real Baxter robot reuses the actions generated to reproduce effects, within the previous experiments, to continuously interact with a box. The interaction is performed using both end-effectors of the robot, located randomly at each side of the box, respectively. In each run of the test the robot identifies, for each end-effector, the effect with the highest mean probability of being reproduced. The mean value and the actions are directly computed by the BNs corresponding to each of the experiments. An online video is available[5] showing the results produced by the K2 method used in the first experiment.

E. Results

The results obtained for the experiments are available in the Table I, where *S* stands for trajectories producing successful results, *FP* stands for trajectories producing false-positives, *F* stands for trajectories producing failures, and *Sc* stands for the normalized score, and ΔSc stands for the variation of the score between the initial and the final iteration.

[4]https://youtu.be/5a02TkaaaRk

[5]https://youtu.be/RKJRXmRTHDc

TABLE I
RESULTS OF THE EXPERIMENTS.

		Affordance exp. (predefined actions)	Iterative exp. (random actions)
Hard-coded	S	16	8
	FP	0	2
	F	0	6
	Sc	100	53.1
	ΔSc	-	33.1
K2	S	15	4
	FP	1	12
	F	0	0
	Sc	95.3	43.57
	ΔSc	-	43.57

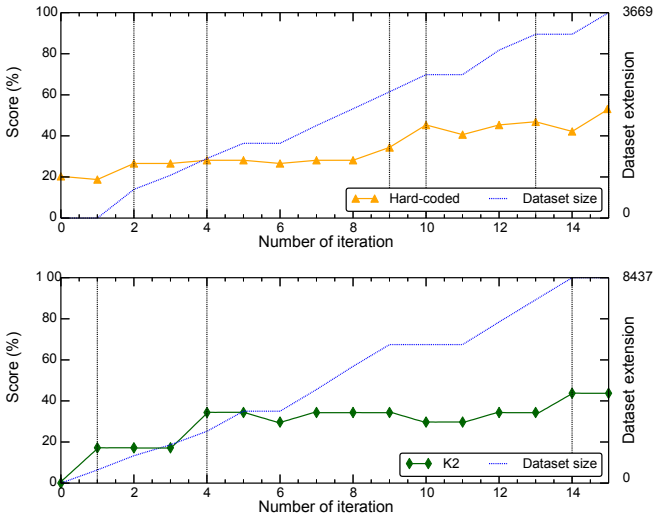

Fig. 5. Results of the *iterative object information acquisition experiment* experiment (Section IV-C) at the end of each iteration. These results are composed of the score obtained (yellow/green lines), and the size of the information related to the interactions between the robot and the box (blue lines). At the top, results obtained using the hand-coded method; at the bottom, using K2. The dotted vertical lines represent iterations where the score improved respect the previous iteration.

Affordance learning experiment (left column): based on the dataset generated using predefined trajectories, the robot reproduces all the effects for the hand-coded method and most of them for K2. These results reflect that, given the right information about the object, i.e. a dataset of interactions to reproduce all the effects, our model is suitable for learning affordances to reproduce a set of the effects adapting to the object position, as described in the first approach defined in Section I.

Iterative object information acquisition experiment (right column): the robot has increased the number of effects reproduced throughout the running of the iterative process with both available learning methods (Fig. 5), confirming the second approach defined in Section I. The hard-coded method reproduces half of the effects at the end of the iterative process, meaning that information about the other half of the effects

is missing. Using the same information, K2 only reproduces 4 effects although it produces a high number of false positive results. This means that the structure of the BN has not identified the right dependency among the movement, effect and the relation state (at the right of Fig. 4). Therefore, more information is needed to both, identify the right structure of the BN and to reproduce the complete set of effects.

High-level stage example: the real Baxter interacts with the box using the affordances generated in both experiments, as it is shown in the related video, until the robot pushes the box out of the table. This result proves that (i) affordances based on discrete representations can be reused in posterior high-level stages; and that (ii) the affordances generated by the simulated Baxter can be directly executed by a real Baxter, if their environments share the same features.

V. Discussion and conclusion

In this paper we have proposed an iterative method, Adaptive Affordance Learning (A^2L), which endows a robot with the capacity to learn affordances associated to an object, both adapting the robot's actions to the object position; and enlarging the robot's information about the object when needed. These features allow the robot to reproduce effects on the object. The performance of our method has been assessed in two experiments where a simulated Baxter robot learns to push a box to different positions on a table.

On the one hand, the action adaptation relies on the decomposition of an action into a sequence of movements, inferring each one of them based on the object position. This approach allows the robot to reproduce an effect given the proper information for it. However, the inferred actions are discrete, based on a predefined discretization. This approach can limit the reuse of the inferred actions to execute some tasks. A possible improvement would be to use the values of a continuous Gaussian function or a force field to compute the relation state, instead of just a simple trajectory.

On the other hand, to increase the information about the object, the method executes an iterative process that alternates between (1) exploration of the environment interacting with the object, (2) affordance acquisition and (3) affordance validation. The affordance learning relies on Bayesian techniques. Two methods, with different a priori knowledge, have been used for the affordance learning. Both methods have generated a progressive increase of the number of learned affordances to reproduce the expected effects. Nevertheless, the number of effects reproduced is low. Experiments with more iterations could increase the object information, and thus reach higher number of effects reproduced. Besides, it could drive the identification of the right structure of the BN.

Although these big datasets could be built, the use of randomly generated actions entails the repetition of previously executed actions, generating a low diversity of actions, and thus of effects. The babbling of the environment could be driven by some goal-oriented intrinsic motivation, as novelty or curiosity, generating action-effect relations related to all the effects. And thus improving the affordance learning.

A^2L needs external information to execute, i.e. the set of effects to reproduce, the discretization configuration, and the relation state must be available. This information could have been computed in an unsupervised fashion within the iterative process.

The use of more realistic scenarios, with daily objects, and different actions, as grasp, would challenge the capabilities of the method. Also, the setup of the experiments could be extended to more than one object, i.e. multi-object affordance learning.

Acknowledgment

This research is sponsored in part by the DREAM project[6]. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 640891. This work was performed within the Labex SMART (ANR-11-LABX-65) supported by French state funds managed by the ANR within the Investissements d'Avenir programme under reference ANR-11-IDEX-0004-02.

References

[1] D. Vernon, *Artificial Cognitive Systems. A Primer*, 2014, vol. 53, no. 9.
[2] J. Piaget and M. Cook, "The origins of intelligence in children." p. 419, 1952.
[3] F. Guerin, N. Krüger, and D. Kraft, "A Survey of the Ontogeny of Tool Use : from Sensorimotor Experience to Planning," *IEEE TAMD*, vol. 5, no. 1, pp. 18 – 45, 2013.
[4] L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater, and J. Santos-Victor, "Affordances in psychology, neuroscience and robotics: a survey," *IEEE TCDS*, no. August, pp. 1–1, 2016.
[5] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action-initial steps towards artificial cognition," in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, 2003, pp. 3140–3145.
[6] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-victor, "Learning Object Affordances: From Sensory–Motor Coordination to Imitation," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 15–26, 2008.
[7] E. Ugur, H. Celikkanat, E. Sahin, Y. Nagai, and E. Oztop, "Learning to grasp with parental scaffolding," in *IEEE-RAS International Conference on Humanoid Robots*, 2011.
[8] S. Szedmak, E. Ugur, and J. Piater, "Knowledge propagation and relation learning for predicting action effects," in *IEEE IROS*, 2014, pp. 623–629.
[9] S. Hangl, E. Ugur, S. Szedmak, and J. Piater, "Hierarchical Haptic Manipulation for Complex Skill Learning," *arXiv preprint arXiv:1603.00794*, 2016.
[10] E. Ugur, E. Sahin, and E. Oztop, "Self-discovery of motor primitives and learning grasp affordances," *IEEE IROS*, 2012.
[11] E. Ugur, Y. Nagai, E. Sahin, and E. Oztop, "Staged development of robot skills: Behavior formation, affordance learning and imitation with motionese," *IEEE TAMD*, 2015.
[12] O. Kroemer, E. Ugur, E. Oztop, and J. Peters, "A Kernel-based approach to Direct Action Perception," *IEEE ICRA*, pp. 2605–2610, 2012.
[13] Y. Demiris and A. Dearden, "From motor babbling to hierarchical learning by imitation: a robot developmental pathway," *International Workshop on Epigenetic Robotics*, pp. 31–37, 2005.
[14] J. Pearl, "Causality: Models, Reasoning, and Inference," *Cambridge University Press*, pp. 1–386, 2000.
[15] G. F. Cooper and E. Herskovits, "A bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, no. 4, pp. 309–347, 1992.

[6] http://www.robotsthatdream.eu/