Hierarchical Affordance Discovery using Intrinsic Motivation

Alexandre Manoury, Sao Mai Nguyen, Cédric Buche

Hierarchical Affordance Discovery using Intrinsic Motivation

Alexandre Manoury
alexandre.manoury@imt-atlantique.fr
IMT Atlantique
Brest, France

Sao Mai Nguyen
mai.nguyen@imt-atlantique.fr
IMT Atlantique
Brest, France

Cédric Buche
buche@enib.fr
ENIB
Brest, France

ABSTRACT

To be capable of life-long learning in a real-life environment, robots have to tackle multiple challenges. Being able to relate physical properties they may observe in their environment to possible interactions they may have is one of them. This skill, named affordance learning, is strongly related to embodiment and is mastered through each person's development: each individual learns affordances differently through their own interactions with their surroundings. Current methods for affordance learning usually use either fixed actions to learn these affordances or focus on static setups involving a robotic arm to be operated.

In this article, we propose an algorithm using intrinsic motivation to guide the learning of affordances for a mobile robot. This algorithm is capable to autonomously discover, learn and adapt interrelated affordances without pre-programmed actions. Once learned, these affordances may be used by the algorithm to plan sequences of actions in order to perform tasks of various difficulties. We then present one experiment and analyse our system before comparing it with other approaches from reinforcement learning and affordance learning.

KEYWORDS

Intrinsic motivation; Incremental learning; Affordances

ACM Reference Format:
Alexandre Manoury, Sao Mai Nguyen, and Cédric Buche. 2019. Hierarchical Affordance Discovery using Intrinsic Motivation. In *Proceedings of the 7th International Conference on Human-Agent Interaction (HAI '19), October 6–10, 2019, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3349537.3351898

1 INTRODUCTION

Continuous adaptation to the environment constitutes a key feature of the human learning process. It enables humans to learn to interact with newly discovered objects, either by reusing and adapting previously acquired knowledge or by building new skills more adapted to the situation at hand. This competence, named life-long learning, is one of the central challenges for service robots to act in our every day environment, towards socially assistive robotics and human agent interaction.

To tackle this, we adopt the approach of developmental robotics. Indeed, studying how infants learn and adapt constitutes an essential example of life-long learning. It highlights multiple mechanisms involved in this learning. Among them, we decide to focus on two in particular: the way infants relate what they see to how they may interact with surrounding objects; and how they explore and interact with their environment while building new skills.

The first one, coined since 1979 by Gibson as the concept of affordance [22], describes the strong relationship between visual cues and possible interactions. Contrarily to classical computer vision, central to the notion of affordance are the concepts of embodiment and of motor capabilities [9]. For instance, adults and infants do not see the same affordances for the same objects because they do not have the same body, the same way humans do not perceive the same affordances as robots. Such affordances evolve all along the life of a person, directly through its interactions with its surroundings.

The latter, the capacity that infants have to autonomously explore their environment, may be one of the answers of how affordances are learned. Indeed, infants use their curiosity to drive their exploration and build new skills through it [8]. This capacity, described as intrinsic motivation in psychology [13], provides a powerful mechanism to learn motor skills such as affordances.

In this paper, we focus on combining those two aspects of the human learning process: affordances and intrinsic motivation, by proposing a robotic framework to learn affordances thanks to active learning. We apply it to the case of a mobile robot. Moreover, as we aim at complex affordances that might require not only primitive actions but a succession of actions to be combined, we use planning to chain actions in order to perform task of various difficulty.

2 RELATED WORK

In this section we review the work related to two aspects of our approach: affordances learning and active learning algorithms, especially those using intrinsic motivation. We also present reinforcement learning methods we compare with later in the article.

2.1 Affordances learning

Many approaches to affordance learning has been developed: the traversability affordance for instance, has been studied in different works [3, 21]. Likewise, the grasp affordance is a recurrent topic and various approaches exist to learn it such as learning based on visual descriptors or raw image input [10, 16]. However such methods are not easily generalised and tend to focus on one, or a

The research work presented is partially supported by the European Regional Fund (FEDER) via the VITAAL Contrat Plan Etat Region.

HAI '19, October 6–10, 2019, Kyoto, Japan

ACM ISBN 978-1-4503-6922-0/19/10…$15.00
https://doi.org/10.1145/3349537.3351898

fixed number of specific affordances, with no mechanism adapting it to new or more complex affordances.

More general approaches, not focusing only on one affordance, have also been proposed. Such approaches build and update a list of affordances through the robot interactions with its environment. In [11], Bayesian Networks are used to learn dependencies between actions, effects, and visual properties; a strongly spread definition of affordances in robotics. Likewise, [18] also uses a Bayesian approach but coupled with a fixed and finite pre-programmed action set to learn affordances. In our case, we aim to continual learning of multiple affordances through the interaction with its environment. Thus, the robot builds itself sensory motor skills using a wide variety of actions. The robot can use actions of unbounded length and duration, in a continuous action space. In [5], Ugur et al. propose a developmental approach of affordance learning. The robot learns by stages: simple affordances first and more complex later. But they limit their approach to simple affordances with no multi-object interaction possible. In all of those methods above, the approach is limited to a single scene and to a single object interaction at a time, guiding the robot exploration without letting it choose autonomously. Furthermore, the considered setups usually focus on fixed robot with an end-point manipulator, while in our case we decide to consider a mobile robot, using its mobility to explore its surroundings by itself and choosing which object to interact with.

2.2 Active motor learning

Central to Gibson's theory is the notion that the motor capabilities of the agent dramatically influence perception. Therefore affordance is closely linked to motor learning. In recent years, multiple approaches have emerged for active motor learning. For instance, several methods exist to learn forward and inverse models, map motor policies to sensorimotor outcomes; as formalised by [7, 23]. However, as the dimensionality of the spaces considered increase, the learner faces the curse of dimensionality [2] and no comprehensive method is possible.

Developmental methods, inspired by how infants explore and learn, have also been proposed to tackle this issue [17]. Indeed, curiosity has been identified as a key mechanism for exploration [13]. Other methods, even further inspired by human psychology has been developed [1], using goal-babbling mechanism to generate goals and drive their exploration.

More recently, methods using intrinsic motivation to build a hierarchy of interrelated skills has been proposed. Firstly by using a pre-programmed and static hierarchy [6], then by learning it through exploration for robotic arms [4] or mobile robots [12]. The latter provides an algorithm, CHIME, that uses intrinsic motivation to guide its exploration. It is capable of adaptive addition and modification of a skill hierarchy based on its interactions. It may also plan of sequence of actions to perform complex tasks.

2.3 Reinforcement learning

In other domains, such as reinforcement learning, numerous methods have emerged to tackle solving daily tasks, using either classical approaches, such as Q-Learning or more recent neural network based ones: DQN[15] or Actor Critic algorithm[14]. But such methods differ from our approach as they are designed to tackle specific tasks, defined by a reward function that need to be provided to the learner. For a more general approach of reinforcement learning methods, Universal Value Function Approximators [19] have been proposed: the task goal is this time learned as an environmental state instead of being fixed. This lets the learned value function to be more general and applicable to various goals. Closer to our approach, the CURIOUS algorithm has been proposed, combining deep reinforcement learning and intrinsic motivation to generate goals to explore.

We decide to base our proposition on the CHIME algorithm and adapt their learning algorithm to the affordance learning problem by using sensorimotor features. Whereas in [12], CHIME could not generalise its skills to new objects, our algorithm is capable to generalise to new objects, as its learning is based on sensory features.

3 PROPOSITION

Before describing our method, we first present the experiment and formalise the learning problem.

3.1 Setup

The experimental setup used in this article is presented in Figure 1.

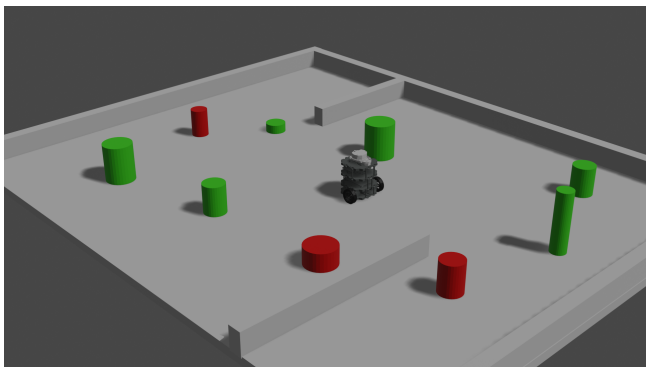

Figure 1: Experimental setup used: at the center is a mobile robot, the green objects represent movable entities, at the opposite of the red ones. The room is closed.

A mobile robot, equipped with two controllable wheels is placed in a rectangular room. It is surrounded by multiple objects and possesses a LIDAR sensor, helping it to detect obstacles. It can navigate between them or learn to push them. The robot starts with a limited prior knowledge:

- how many effectors it possesses, in this case its 2 wheels,
- a method to perform actions on its effectors. In our case, the method lists indexes of the effectors to activate and the action parameters $p \in [-1, 1]^2$ representing the intensity of the current to apply to each wheel.
- a list of the objects present in the room. We consider 9 cylinders in our setup, of various sizes and colors. The green objects are pushables whereas the red ones are not.
- a method to observe the properties of each of the objects (including the robot itself). The properties used in our experiment are: position, shape, radius, height, color, and the

position relative to the robot. A pushable object requires more torque to be moved as its radius and height increase.

With this knowledge, the robot has to autonomously explore its environment and learn how to perform various tasks: moving itself, placing an object somewhere, pushing an object using another one. It also has to learn affordances corresponding to such tasks and to be able to recognise (or estimate) objects on which an affordance may apply or not. E.g. the pushability affordance cannot be applied to red objects, as those are fixed.

The robot explores the room in episodes of arbitrary length of 100 actions. At the beginning of each episode, the position and properties of the objects are initialised at random values and the robot always starts at the center of the room.

In real life operation, the robot can extract this information and properties from its sensors. It our case, the real life system can use an RGB-D sensor to segmentate objects and an variational autoencoder to extract properties from each of them. All the acquired data are used to fill in an environmental map, used in turn to provide data to the learning algorithm described in this article. To keep this system simple and avoid multiplying the source of errors, this article focuses only on the learning algorithm, with direct access to high-level data.

3.2 Problem formalization

Let us consider a robot interacting with its non-rewarding environment by performing sequences of motions of unbounded length in order to induce changes in its surroundings.

Each one of these motions is a primitive action described by a parametrised function with N parameters : $a \in \mathcal{A} \subset \mathbb{R}^N$. Each primitive action a corresponds to a command that may be sent to one or several actuators of the robot.

Our robot can perform sequences of primitive actions. Such sequence may be of any length $n \in \mathbb{N}$, and described by n successive primitive actions: $a = [a_1, \ldots, a_n] \in \mathcal{A}^n$. Thus the primitive action space exploitable by the robot is a continuous space of infinite dimensionality $\mathcal{A}^{\mathbb{N}} \subset \mathbb{R}^{\mathbb{N}}$.

Each of the actions performed by the robot may have consequences on the environment, observable by the robot. We call such consequences *observations* and note them $\omega \in \Omega \subset \mathbb{R}^M$.

Each subspace of Ω is related to a given property of an object $o \in O$ present in the environment (e.g. the position of an object). We consider this relation known to the robot, as such knowledge is required to build an affordance. This is a weak assumption as such information may be extracted from visual segmentation or by exploiting data from a semantic map. In our case, such data are directly given by the simulator itself.

3.3 Formalization of our approach

To learn how to interact with its environment, the robot learns models of relations between primitive actions $a \in \mathcal{A}$ and outcomes $\omega \in \Omega$ (relative observations before and after executing an action) obtained after performing this action within a given context $\tilde{\omega} \in \Omega$ (absolute state before executing the action, for more convenience we indicate with ˜ the context spaces to differentiate them from outcome spaces).

For convenience, we define the controllable ensemble $C = \mathcal{A} \cup \Omega_{controllable}$, regrouping both primitive actions $\in \mathcal{A}$ and observables that may be controlled ($\in \Omega_{controllable}$), i.e. that a model may be used to find one or a sequence of primitive actions to be performed in order to induce a value for the given observable. $\Omega_{controllable}$ is a subset of Ω and this set changes dynamically as the robot discovers new control models.

More generally, the robot may learn models between controllables $c \in C$ (not only primitive actions) and relative observations within a given context. Indeed, our robot may learn how to reach a goal observation value by first inducing a change in another observable of the environment. E.g. pushing an object can be performed after reaching the object.

To formalise affordances we use two elements. First we note an Affordance model $A(C_i, \Omega_j, \tilde{\Omega_k})$, where $C_i \subset C$ is the input space of the affordance, $\Omega_j \subset \Omega$ is the output space and $\tilde{\Omega_k} \subset \Omega$ is its context space. And, secondly, to visually identify this affordance, we associate A with a visual predictor p_A. It learns on Ω and indicates whether A may be applied to an object o in the scene or not, accordingly to its visual or physical properties. Moreover, to be able to learn how to use the affordance to complete tasks, A possesses a forward model M_A and an inverse model L_A. Both models learn the relationship between C_i and Ω_j knowing a context $\tilde{\Omega_k}$. The forward model is used to predict the observable consequences ω of a controllable c_i in a given context $\tilde{\omega}$. Conversely, the inverse model is used to estimate a controllable c_i to be performed in a given context $\tilde{\omega}$ to induce a goal observable state ω as a result of c_i. These models are trained on the data acquired by the robot all along its life and recorded in its dataset. Let us note $\mathcal{D}$ this dataset.

Each affordance A can be seen as a basic skill, letting the robot perform a given simple task, e.g. reaching a position, placing an object somewhere.

Let us note $\mathcal{H}$ the ensemble of the affordances used by our robot. As our robot aimed to be adaptive, $\mathcal{H}$ varies along time.

4 ALGORITHM

For our robot to learn to associate a sequence of primitive actions $[a_1, ..., a_n]$ to desired consequences on multiple objects in its environment, our robot needs to learn which consequences ω can be observed and learn the control actions to realise these consequences. For this learning problem, we propose an algorithm in this section. We first introduce its global architecture before detailing its key processes: how intrinsic motivation drives the exploration, how actions are executed and finally how affordance models are built and updated.

4.1 Global architecture

Our algorithm is based on the CHIME algorithm [12]. Both are iterative and active learning algorithm that learn by episodes, but unlike CHIME our algorithm is designed to consider visual properties during its learning process.

The global layout of the algorithm architecture is presented in Figure 2 and the corresponding pseudo code can be seen on Algorithm 1. At the beginning of the learning, the dataset $\mathcal{D}$ and the affordance hierarchy $\mathcal{H}$ are both empty: the robot autonomously collects data and creates affordances.

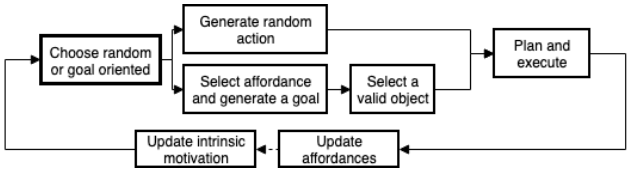

Figure 2: Abstract layout of a learning episode, beginning is on the left on the bold node.

At each episode, the robot explores its environment by performing actions, observes the context and the outcomes obtained and processes the acquired data. One episode is composed of multiple iterations, and at each iteration one primitive action is performed.

Starting an episode, the robot decides either to explore a random action (l. 10), or to use goal babbling to generate a goal to attain during the episode (l. 5). This decision is stochastic, based on a parameter σ, and it also depends on whether interesting goals may be generated or not. E.g. at the first episode, no data has been acquired yet and thus only a random action may be performed.

When choosing a random action, the robot generates a random controllable to be tested $c \in C$ among all the controllable spaces (including the primitive actions). If required, this controllable is then converted to an executable primitive action, as only primitive actions may be performed by the robot effectors. This process is described in Section 4.3.

When choosing to generate a goal, an affordance A and a goal ω_g are selected, based on an interest metric detailed later in section 4.2. The robot next decides on which object this goal will be tested. Currently it just selects the closest object considered as valid by the affordance visual classifier p_A. The robot then uses its inverse models and its planning system to infer a sequence of controllables $c \in C^n$ to be performed in order to reach ω_g. Once again, these controllables are broken down into executable primitive actions, if required, using the same process as previously.

In both cases, the robot generates a sequence of primitive actions $a = [a_1, \ldots, a_n] \in \mathcal{A}^n$ of length n. This corresponds to a random action or to a sequence of actions designed to reach a generated goal ω_g. These actions are then executed by the robot (l. 17): for each sub primitive action a_i, the absolute value of each observable space is first recorded (corresponding to the context of the subaction), a_i is then performed and the difference for each observable space (compared to before the execution) is retrieved.

After finishing an episode, the robot obtains a list of $(a^i, \omega_1^i, \ldots, \omega_k^i, \tilde{\omega}_1^i, \ldots, \tilde{\omega}_k^i)$ for each iteration. Where i corresponds to the iteration index and k to the number of subspaces of Ω. These data are then stored in $\mathcal{D}$ (l. 25). It is also processed and used to improve existing affordances (l. 24), decide whether creating a new affordance is necessary or not, and update the intrinsic motivation system. These different processes are described in the following sub sections.

4.2 Intrinsic motivation

This algorithm uses intrinsic motivation to guide its exploration. It is based on the CHIME algorithm [12], itself inspired by the SAGG-RIAC algorithm [1].

Algorithm 1 Algorithm layout

```
1:  i = 0
2:  loop
3:      D_episodic = ∅
4:      if H ≠ ∅ and Random() ≤ σ then
5:          A = AffordanceSelection(H)
6:          ω = GoalSelection(A)
7:          ω_g = ObjectSelection(A, ω)
8:          c = Plan(ω_g)
9:      else
10:         C_i = RandomControllableSpace(C)
11:         c_r = RandomValue(C_i)
12:         c = [c_r]
13:     a = TransformToPrimitive(c)
14:     for a_k ∈ a do
15:         ω_before = GetObservations(Ω)
16:         a_i = [c_k] if c_k ∈ A else TransformToPrimitive(c_k)
17:         Execute(a_i)
18:         ω_after = GetObservations(Ω)
19:         ω_i = ω_after − ω_before
20:         ω̃_i = ω_before
21:         D_episode ← (a_i, ω_i, ω̃_i)
22:         i += 1
23:     UpdateInterestMaps(D, D_episodic)
24:     UpdateAffordances(D, D_episodic)
25:     D ← D_episodic
```

For each affordance $A(C_A, \Omega_A, \tilde{\Omega_A})$, the system creates an interest map: a partition of Ω_A that is constructed incrementally based on progress measures as described in [1]. The goal of this process is to divide Ω_A into regions and attribute a value of interest to each region. This interest corresponds to a monitoring of how much exploring this region may improve the robot knowledge in the future.

This measure is linked to a notion of *competence*. In our case, we define the competence of an affordance A near a goal $\omega \in \Omega_A$ as $mean(\omega_e - \omega_r)$ for the k last outcomes near ω. ω_e corresponds to an outcome goal estimated by the algorithm for a given controllable c and ω_r to the effective outcome reached during the exploration.

The derivative of this *competence* is used to define a *learning progress*: how much an affordance model has been improved. And the *interest* value of a region then corresponds to the mean of the last *n learning progresses* in this region.

More details about this process and the region splitting mechanism may be found in [12] and in [1].

4.3 Action and controllable execution

To perform sequence of controllables c, our algorithm uses the same system as CHIME. For each element c_i of c:

- if the sub-controllable to be performed c_i is a primitive action, it is directly sent to the effectors and executed without any pre-processing
- in the other case, if c_i is not a primitive action, it corresponds to an observable the robot wants to induce within its environment, i.e. $c_i \notin \mathcal{A}$ but $c_i \in \Omega_{controllable}$. Then it cannot be directly executed by the effectors of the robot and it needs

to be broken down into primitive actions beforehand. An affordance $A(C_A, \Omega_A, \tilde{\Omega_A})$ is then selected (with $c_i \in \Omega_A$) and its inverse model is applied onto c_i in order to obtain a lower level controllable $b_i \in C_A$. If c_i is difficult to reach using only one lower level controllable, a planning phase is used to build a sequence of element of C_A in order to reach c_i when executed. Once again for each element of this newly created sequence, if it is not primitive the same mechanism is applied recursively on it until having only primitive actions.

At the end of this mechanism, we obtain a list $b \in \mathcal{A}^{\mathbb{N}}$ composed only of primitive actions that can be executed directly.

Additional information can be found in [12].

4.4 Affordance addition and update

The CHIME algorithm has been designed to autonomously learn model of data. We diverge from it to autonomously learn affordances instead. In this section we present how affordances are added to $\mathcal{H}$ and updated.

At each episode, the robot has to decide multiple elements: whether a new affordance must be added or if existing affordances are enough; how to train the visual classifiers of affordances and if affordances need to be updated.

To answer those questions, the robot follows the procedure presented on Algorithm 2.

At the end of each episode, subspaces of Ω for which non null relative outcomes ω has been observed are listed. Then the robot randomly picks a space among this list and verifies if it matches an existing affordance. A space matches an affordance if adding the data from this space to the training set of the affordance does not reduce its competence. If not matching, it tries to add context spaces to the affordance or then tries to create a new affordance. The predictor p_A is afterwards trained on the acquired data (positive or negative).

The predictor p_A used in our system is a binary neural network composed of 3 fully connected layers using as input all the properties of the object o currently considered. It is trained using action replay on balanced data (objects on which the A is applicable and the others).

5 PERFORMANCE OF OUR ALGORITHM

We used our experimental setup to perform three series of tests:

- firstly, evaluating our system itself, which affordances are created, and when;
- then, comparing the task performance of our system compared to CHIME;
- finally, comparing our system to other reinforcement learning approaches.

5.1 Evaluation method

To measure the performance of our (or other) algorithm at completing tasks, we define an evaluation metric as follows: for each task, we pre-define a list of points (robot position or object position) to be reached. Then, during evaluation, the system attempts to reach each point in the simulator and 1− the mean error at reaching those points is defined as the evaluation of this task.

Algorithm 2 Autonomous affordances adaptation

Input: a the actions performed during the episode, ω the observations at the beginning of each iteration of the episode.

```
Spaces = SelectSpaces(Ω, ω)
repeat k times
    S = PickSpace(Spaces)
    for A ∈ H do
        matched = False
        if Matches(A, a, ω_S) then
            matched = True
            Add (a, ω_S) to the model training dataset of A
            TrainVisualClassifier(A, ω_S, True)
        else
            repeat k' times
                S'_context = PickSpace(Ω)
                NewA = Copy(A)
                ContextSpace_NewA = ContextSpace_NewA ∪ S'_context
                if Competence(NewA) ≥ τ_modification then
                    A ← NewA
                    matched = True
                    break
    p_A ← TrainVisualClassifier(A, ω_S, matched)
    if matched then
        Add a, ω_S to the model training dataset of A
    else
        NewA = Affordance(a, S, ∅)
        if Competence(NewA) ≥ τ_creation then
            H ← NewA
            p_NewA ← TrainVisualClassifier(NewA, ω_S, True)
```

5.2 Affordances learning

In our first test, we let the robot explore the environment presented in 3.1. This environment is simulated in *python* using a 2D physics engine named *pymunk*.

We perform 10 runs, letting the robot autonomous during 4000 iterations and we report the mean results.

At the end of its exploration, we observe the affordances created and their evaluation, as presented in Figure 3. The robot has successfully discovered multiple affordances, we count 12 at the end for the majority of runs. Among them, 3 where expected:

- A_1: moving the robot itself,
- A_2: pushing an object by moving the robot and
- A_3: pushing an object using another object.

The other affordances discovered are unintended, but still valid: they correspond to unexpected correlations the robot has found between various spaces. In our analysis we focus on the first 3 affordances mentioned above.

Even in this simple environment, the algorithm has managed to create a hierarchy of interrelated skills: A_2 depending on A_1 to be completed, itself depending on A_0.

More than just the final number of affordances, it is interesting to observe the creations, deletions and updates of affordances all along the exploration.

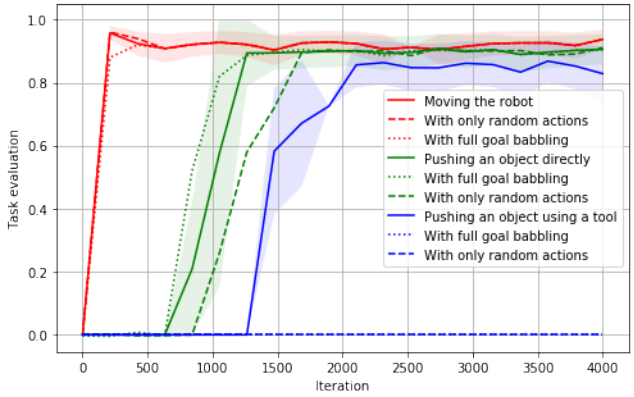

Figure 3: Evaluation during the training for three affordances: moving the robot itself A_0, pushing an object A_1 and pushing an object using another one A_2. Evaluation is done every 200 iterations between 200 and 2000. Thus, affordance A_1 is created between 600 and 800. This also shows mean evaluation value when using only random actions or only goal babbling when possible (standard deviation for those is not displayed for clarity). For A_3, the goal babbling or random only versions does not manage to create the affordance.

Concerning the affordance A_0, we can see in Figure 4 (top) that the affordance is created since iteration t=25. At the moment of the discovery of this affordance, the model created by the robot does not take as input any context space. As no walls or obstacles have been encountered yet the robot thinks the movement of the robot only depends on its wheels speed. At iteration t=150 the affordance is updated and the relative position between an object and the robot is added as context space. A wrong assumption but coherent with the data acquired so far. Then quickly, at iteration 175, this context space is replaced with the robot LIDAR space and kept as such until the end. No physical properties is used here as a context space, this is due to the fact that the robot is the only object using this affordance.

The results of A_1 in Figure 4 (bottom) show that this affordance is created much later than A_0. This is explained by the fact that the robot has first to collide with an object to discover how to push objects directly with its body. The first occurrence of such collision was around t=500 iterations on average. Here again, the context space of the affordance has evolved during the exploration and has finally converged to the relative position between the object and the robot. At the difference of A_0, 2 physical properties are here added as context spaces of this affordance: the radius and the height of the object at hand. As the pushability of each object depends on these two physical properties it is normal to see them appear here, and this confirm that our algorithm has well captured the dependency to such properties.

Once A_1 has been created, its visual classifier p_{A1} is also created and trained to identify to which object A_1 may be applied or not. At the end of the 4000 iterations, we use p_{A1} to check its prediction for each object in the room including the robot itself: it is positive for all the green objects and negative for the robot. This is expected as the robot cannot push itself neither it can push fixed red objects. Hence, our algorithm has successfully managed to construct both a model affordance and the corresponding visual classifier.

For A_0, A_1 and A_2, the affordances are created directly as soon as collected data permit it. This behaviour is desired and due to a low affordance creation threshold $\tau_{addition}$. This favours exploration of newly discovered spaces and regions: indeed, with a low threshold value, affordances are easily created and a goal may be generated to explore them. If the exploration then points out that it is a false positive, that affordance is destroyed. On the contrary if the exploration confirms it as a valid affordance, active learning continues to gradually collect new data to increase the robot's competence for this affordance.

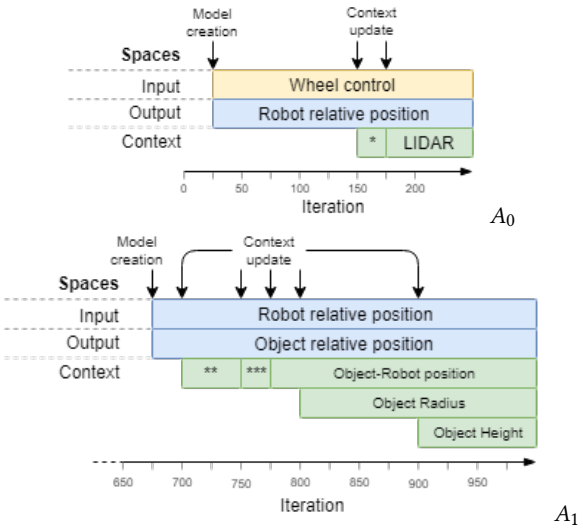

Figure 4: Temporal evolution of affordances A_0 (top) and A_1 (bottom) during the learning process. Please note that the iteration axis is not the same for A_0 and A_1. Colors are not related to the competence graph: yellow spaces are part of $\mathcal{A}$, blue and green ones of Ω: blue ones are using relative data while green ones absolute data.

*** : relative position between an object and the robot**
**** : robot absolute position**
***** : LIDAR data**

5.3 Random and Goal Babbling impact

To further analyse our algorithm we decided to test two extreme situations: one with only random action exploration; and another one using only goal babbling whenever possible.

The first case favours novelty and discovery: the rate of affordance addition is high, but the exploration and the mastering of the already discovered affordance is delayed. In Figure 3 we can see that the competence curve for A_1 requires more time to converge than in the previous test.

At the opposite, using only goal babbling whenever available, the number of affordances discovered is greatly reduced, and focused at the beginning of the exploration. In this configuration, A_2 is discovered later compared to the previous configuration.

6 COMPARISON WITH OTHER APPROACHES

We compare our approach to baselines belonging to two different families: firstly to reinforcement learning algorithms on similar setups. Secondly, we compare it to affordance learning algorithms. But to our knowledge such methods do not focus on mobile robot and are thus evaluated on experimental setups significantly different from ours.

6.1 Reinforcement learning

As we want to compare our algorithm to existing ones on the same setup, we choose to use classical reinforcement learning algorithms such as Q-Learning, DQN (Deep Q-Network) and Actor Critic in our experimental setup. As they are not designed for multi-task learning and require an extrinsic reward, some setup modifications have been made to enable these algorithms to learn in our setup: we limited the experiment to one object at a time (except for A_2) and added a reward function to provide a feedback. Unlike our method, where the exploration is self-guided, the desired behaviours or tasks to be completed with these algorithms must be explicited through the reward function. We test these algorithms on 3 increasingly difficult tasks: moving the robot, pushing an object directly and then by using another object as a tool. To match the general aspect of our algorithm, we use Universal Value Function Approximators [19] for these three algorithms in order to learn how to reach various goals. We use 2 different kinds of reward function for each setup:

Version	Reaching goal	Pushing/going in the right direction	Else
Non-guided (sparse reward)	+1000	-5	
Guided	+1000	max +20	-5

Table 1: Reward functions used by the comparative setup

In addition to these algorithms, we also compare ours to CURIOUS, a reinforcement learning algorithm using intrinsic motivation for exploration. We base its reward on the non-guided version.

When required, Ω has been discretised uniformly. Actions have also been discretised into 4 when needed: forward, backward, turning left, turning right. When reaching the zone or after 1000 iterations the episode ends, the setup is reset and the robot is randomly placed inside the room.

We perform 10 runs over 50000 iterations for each task, reward function and algorithm and report the result in Figure 5.

For the first task (top), we can see that all the algorithms succeed in 10000 to 20000 iterations. With our algorithm, moving the robot is mastered as soon as 250 iterations. The difference mainly comes from the use of planning in our case. It lets the robot reach distant spots even with such a few exploration done.

For the second task (middle), only the guided version are successful, requiring between 12500 and 26000 iterations to be learned. The non-guided versions fail because of the combinatorial explosion of all the states involved and the difficulty to reach the final goal. In our case this task is learned within 1000 iterations.

For the most complex task (bottom), only CURIOUS and our algorithm manage to succeed, CURIOUS reaches a competence of 0.96 using 32000 iterations. The other algorithms, even using the guided rewards, fail due to the complexity of the task at hand. Our algorithm only requires 2100 iterations to reach the final level of competence of CURIOUS.

For all the examples above, as we use UVFA, the Q-Learning is highly dependent to the number of goal it has to explore (as each goal corresponds to a different state). Thus, this adds another prior (in addition to the reward function) that is not required with our algorithm.

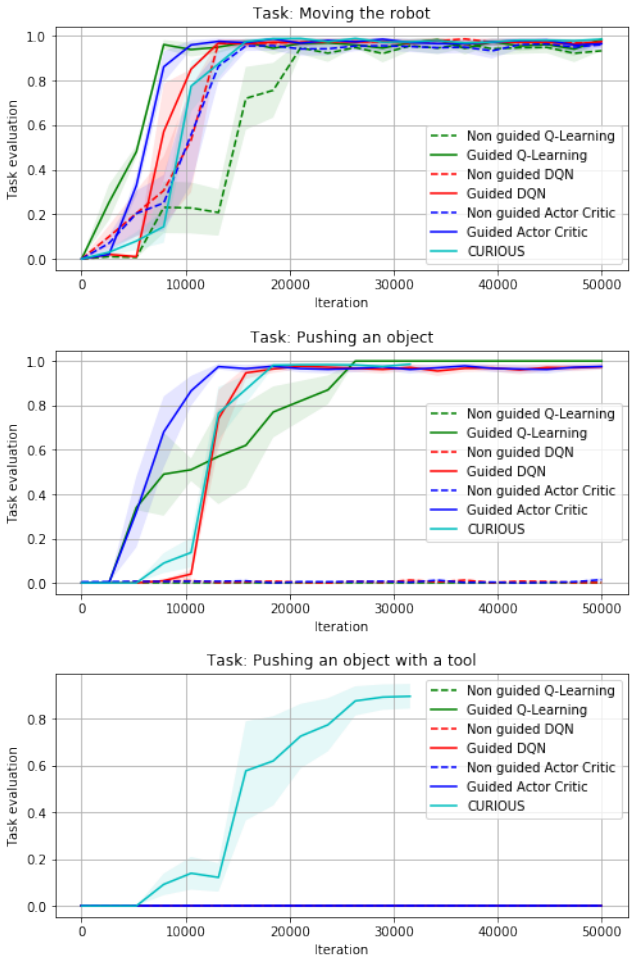

Figure 5: Evaluation of Q-Learning, DQN, Actor Crictic and CURIOUS applied to three tasks: moving the robot, pushing an object and pushing it using another object as a tool. The standard deviation is displayed in transparent.

6.2 Affordance learning

As the majority of works in affordance learning uses robot arms to manipulate objects and not mobile robot, experimental setups are difficult to compare. Thus, we decide to provide a qualitative comparison between our approach and existing affordance learning ones. We analyse the learning process reported for the subsequent affordances in these different setups.

In [11], the system extracts pre-programmed controllables from the considered objects, like in our algorithm, then discretises them and clusterises them. It then builds a dependency graph that encompasses visual controllables, performed action and the action context. In our case, the information contained in this graph are all included in our models and visual classifiers. Thus, our system is capable to build the same affordances. Conversely, the pre-programmed actions in [11] are in our case autonomously learned by the robot, requiring less prior information and adding more flexibility. In both cases, the temporal aspect of sequences of actions is not learned, but in our algorithm, the planning layer automatically creates successive sequences, based on the models learned.

On the contrary, in [20], the system builds a hierarchy of affordances like in our proposition. This time intrinsic motivation is used to select which action to execute within a finite set of pre-defined low level actions. Whereas in our system, the robot manages to learn primitive actions in a continuous space, and is capable to use sequences of actions by chaining primitive actions.

7 CONCLUSION

For affordances learning, we have presented an algorithm combining the affordances concept and intrinsic motivation exploration. It allows a robot to autonomously discover unknown affordances and learn actions to exploit them. The learning is based on active learning to collect data through new interactions with the environment, guided by the heuristics of intrinsic motivation; Once learned, these affordance control models are used to plan complex tasks with known or unknown objects, by using their physical properties to decide whether or not a learned affordance may be applied.

Our main contribution in this article is to propose a learning algorithm for multiple objects based on physical properties so as to generalise to new objects. We have shown that it can discover in a developmental manner non-predefined affordances from the easiest to the most complex ones, and can use unbounded sequences of learned actions to complete complex tasks. We have compared our algorithm to others to outline two main properties : the hierarchical and developmental learning process, as well as the capacity to use sequences of actions to adapt to the complexity of the task at hand.

This algorithm broadly relies on the concept of embodiment and is strongly inspired by human development from this point of view; for both the affordance aspect and the intrinsic motivation one.

In future works we want to deepen the comparison with existing methods by considering similar setups, and thus applying our algorithm onto robotic arms. Also, we aim for a more complete system, including a mechanism for visual feature extraction in order to provide inputs for our algorithm.

REFERENCES

[1] Adrien Baranes and Pierre-yves Oudeyer. 2009. R-IAC: Robust intrinsically motivated exploration and active learning. *IEEE Transactions on Autonomous Mental Development* 1, 3 (2009), 155–169.

[2] Richard Bellman. 1957. *Dynamic programming*.

[3] Dongshin Kim, Jie Sun, Sang Min Oh, J. M. Rehg, and A. F. Bobick. 2006. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* 518–525. https://doi.org/10.1109/ROBOT.2006.1641763

[4] Nicolas Duminy, Sao Mai Nguyen, and Dominique Duhaut. 2019. Learning a Set of Interrelated Tasks by Using a Succession of Motor Policies for a Socially Guided Intrinsically Motivated Learner. *Frontiers in Neurorobotics* 12 (2019), 87. https://doi.org/10.3389/fnbot.2018.00087

[5] Erol Sahin Emre Ugur Yukie Nagai, Erhan Oztop, Emre Ugur, Yukie Nagai, Erol Sahin, and Erhan Oztop. 2015. Staged Development of Robot Skills: Behavior Formation, Affordance Learning and Imitation with Motionese. *IEEE Transactions on Autonomous Mental Development* 7, 2 (2015), 119–139. https://doi.org/10.1109/TAMD.2015.2426192

[6] Sébastien Forestier and Oudeyer Pierre-Yves. 2016. Overlapping Waves in Tool Use Development: a Curiosity-Driven Computational Model. *IEEE International Conference Developmental Learning and Epigenetic Robotics* (2016), 1859–1864.

[7] B.A. Francis and W.M. Wonham. 1976. The internal model principle of control theory. *Automatica* 12, 5 (1976), 457 – 465. https://doi.org/10.1016/0005-1098(76)90006-6

[8] Jacqueline Gottlieb, Pierre Yves Oudeyer, Manuel Lopes, and Adrien Baranes. 2013. Information-seeking, curiosity, and attention: Computational and neural mechanisms. *Trends in Cognitive Sciences* 17, 11 (2013), 585–593. https://doi.org/10.1016/j.tics.2013.09.001 arXiv:NIHMS150003

[9] Lorenzo Jamone, Emre Ugur, Angelo Cangelosi, Luciano Fadiga, Alexandre Bernardino, Justus Piater, and Jose Santos-Victor. 2016. Affordances in psychology, neuroscience and robotics: a survey. *IEEE Transactions on Cognitive and Developmental Systems* January (2016), 1–1. https://doi.org/10.1109/TCDS.2016.2594134

[10] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. 2018. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research* 37, 4-5 (2018), 421–436. https://doi.org/10.1177/0278364917710318 arXiv:https://doi.org/10.1177/0278364917710318

[11] Alexandre Bernardino Luis Montesano Manuel Lopes, Jose Santos-Victor, L. Montesano, M. Lopes, a. Bernardino, and Jose Santos-Victor. 2008. Learning Object Affordances: From Sensory–Motor Coordination to Imitation. *Ninth International Conference on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems* 24, 1 (2008), 15–26. https://doi.org/10.1109/TRO.2007.914848

[12] A. Manoury, S. M. Nguyen, and C. Buche. 2019. CHIME: An Adaptive Hierarchical Representation for Continuous Intrinsically Motivated Exploration. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*. 167–170. https://doi.org/10.1109/IRC.2019.00032

[13] Karen A. Miller, Edward L. Deci, and Richard M. Ryan. 1988. Intrinsic Motivation and Self-Determination in Human Behavior. *Contemporary Sociology* 17, 2 (1988), 253. arXiv:arXiv:1011.1669v3 http://www.jstor.org/stable/2070638?origin=crossref

[14] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. *CoRR* abs/1602.01783 (2016). arXiv:1602.01783 http://arxiv.org/abs/1602.01783

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (Feb. 2015), 529–533. http://dx.doi.org/10.1038/nature14236

[16] L. Montesano and M. Lopes. 2009. Learning grasping affordances from local visual descriptors. In *2009 IEEE 8th International Conference on Development and Learning*. 1–6. https://doi.org/10.1109/DEVLRN.2009.5175529

[17] Pierre-Yves Oudeyer, Frederic Kaplan, and Verena Hafner. 2007. Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Transactions on Evolutionary Computation* 11, 2 (2007), 265–286. https://doi.org/10.1109/TEVC.2006.890271

[18] Pierre Luce-Vayrac R. Omar Chavez-Garcia, Raja Chatila, R. Omar Chavez-Garcia, Pierre Luce-Vayrac, and Raja Chatila. 2016. Discovering Affordances Through Perception and Manipulation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vol. 2016-Novem. 3959–3964. https://doi.org/10.1109/IROS.2016.7759583

[19] Tom Schaul, Dan Horgan, Karol Gregor, and David Silver. 2015. Universal Value Function Approximators. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15)*. JMLR.org, 1312–1320. http://dl.acm.org/citation.cfm?id=3045118.3045258

[20] Emre Ugur and Justus Piater. 2016. Emergent structuring of interdependent affordance learning tasks using intrinsic motivation and empirical feature selection. (2016), 1–13.

[21] Emre Uğur and Erol Şahin. 2010. Traversability: A case study for learning and perceiving affordances in robots. *Adaptive Behavior* 18, 3 (2010), 258–284. https://doi.org/10.1177/1059712310370625

[22] Bruce A. Whitehead. 1981. James J. Gibson: The ecological approach to visual perception. Boston: Houghton Mifflin, 1979, 332 pp. *Behavioral Science* 26, 3 (1981), 308–309. https://doi.org/10.1002/bs.3830260313

[23] D.M. Wolpert and M. Kawato. 1998. Multiple paired forward and inverse models for motor control. *Neural Networks* 11, 7 (1998), 1317 – 1329. https://doi.org/10.1016/S0893-6080(98)00066-5