LEARNING A SET OF INTERRELATED TASKS BY USING A SUCCESSION OF MOTOR POLICIES FOR A SOCIALLY GUIDED INTRINSICALLY MOTIVATED LEARNER

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose an active learning algorithmic architecture, capable of organizing its learning process in order to achieve a field of complex tasks by learning sequences of primitive motor policies : Socially Guided Intrinsic Motivation with Procedure Babbling (SGIM-PB). The learner can generalize over its experience to continuously learn new outcomes, by choosing actively what and how to learn guided by empirical measures of its own progress. In this paper, we are considering the learning of a set of interrelated complex outcomes hierarchically organized.

We introduce a new framework called "procedures", which enables the autonomous discovery of how to combine previously learned skills in order to learn increasingly more complex motor policies (combinations of primitive motor policies). Our architecture can actively decide which outcome to focus on and which exploration strategy to apply. Those strategies could be autonomous exploration, or active social guidance, where it relies on the expertise of a human teacher providing demonstrations at the learner's request. We show on a simulated environment that our new architecture is capable of tackling the learning of complex motor policies, to adapt the complexity of its policies to the task at hand. We also show that our "procedures" increases the agent's capability to learn complex tasks.

1 INTRODUCTION

Recently, efforts in the robotic industry and academic field have been made for integrating robots in previously human only environments. In such a context, the ability for service robots to continuously learn new skills, autonomously or guided by their human counterparts, has become necessary. They would be needed to carry out multiple tasks, especially in open environments, which is still an ongoing challenge in robotic learning. Those tasks can be independent and self-contained but they can also be complex and interrelated, needing to combine learned skills from simpler tasks to be tackled efficiently.

The range of tasks those robots need to learn can be wide and even change after the deployment of the robot, we are therefore taking inspiration from the field of developmental psychology to give the robot the ability to learn. Taking a developmental robotic approach (Lungarella et al., 2003), we combine the approaches of active motor skill learning of multiple tasks, interactive learning and strategical learning into a new learning algorithm and we show its capability to learn a mapping between a continuous space of parametrized outcomes (sometimes referred to as tasks) and a space of parametrized motor policies (sometimes referred to as actions).

1.1 ACTIVE MOTOR SKILL LEARNING OF MULTIPLE TASKS

Classical techniques based on Reinforcement Learning (Peters & Schaal, 2008; Stulp & Schaal, 2011) still need an engineer to manually design a reward function for each particular task. Intrinsic motivation, which triggers curiosity in human beings according to developmental psychology (Deci & Ryan, 1985), inspired knowledge-based algorithms (Oudeyer & Kaplan, 2009). The external reward is replaced by a surprise function: the learner is driven by unexpected outcomes (Barto

et al., 2004). However those approaches encounter limitations when the sensorimotor dimensionality increases. Another approach using competence progress measures successfully drove the learner's exploration through goal-babbling (Baranes & Oudeyer, 2010) (Rolf et al., 2010).

However when the dimension of the outcome space increases, these methods become less efficient (Baranes & Oudeyer, 2013) due to the curse of dimensionality, or when the reachable space of the robot is small compared to its environment. To enable robots to learn a wide range of tasks, and even an infinite number of tasks defined in a continuous space, heuristics such as social guidance can help by driving its exploration towards interesting and reachable space fast.

1.2 INTERACTIVE LEARNING

Combining intrinsically motivated learning and imitation Nguyen et al. (2011) has bootstrapped exploration by providing efficient human demonstrations of motor policies and outcomes.

Also, such a learner has been shown to be more efficient if requesting actively a human for help when needed instead of being passive, both from the learner or the teacher perspective (Cakmak et al., 2010). This approach is called interactive learning and it enables a learner to benefit from both local exploration and learning from demonstration. Information could be provided to the robot using external reinforcement signals (Thomaz & Breazeal, 2008), actions (Grollman & Jenkins, 2010), advice operators (Argall et al., 2008), or disambiguation among actions (Chernova & Veloso, 2009). Another advantage of introducing imitation learning techniques is to include non-robotic experts in the learning process (Chernova & Veloso, 2009). One of the key element of these hybrid approaches is to choose when to request human information or learn in autonomy such as to diminish the teacher's attendance.

1.3 STRATEGIC LEARNING

Approaches enabling the learner to choose either what to learn (which outcome to focus on) or how to learn (which strategy to use such as imitation) are called strategic learning (Lopes & Oudeyer, 2012). They aim at giving an autonomous learner the capability to self-organize its learning process.

Some work has been done to enable a learner to choose on which task space to focus. The SAGG-RIAC algorithm (Baranes & Oudeyer, 2010), by self-generating goal outcomes fulfills this objective. Other approaches focused on giving the learner the ability to change its strategy (Baram et al., 2004) and showed it could be more efficient than each strategy taken alone.

Fewer studies have been made to enable a learner to choose both its strategy and its target outcome. The problem was introduced and studied in Lopes & Oudeyer (2012), and was implemented for an infinite number of outcomes and policies in continuous spaces by the SGIM-ACTS algorithm (Nguyen & Oudeyer, 2012). This algorithm is capable of organizing its learning process, by choosing actively both which strategy to use and which outcome to focus on. It relies on the empirical evaluation of its learning progress. It could choose among autonomous exploration driven by intrinsic motivation and low-level imitation of one of the available human teachers to learn more efficiently. It showed its potential to learn on a real high dimensional robot a set of hierarchically organized tasks (Duminy et al., 2016), which is why we consider it to learn complex motor policies.

1.4 LEARNING COMPLEX MOTOR POLICIES

In this article, we tackle the learning of complex motor policies, which we define as combinations of simpler policies. We describe it more concretely as a sequence of primitive policies.

A first approach to learning complex motor policies is to use via-points (Stulp & Schaal, 2011). Via-points enable the definition of complex motor policies. Those via-points are in the robot motor policy space. When increasing the size of the complex policy (by chaining more primitive actions together), we can tackle more complex tasks. However, this would increase the difficulty for the learner to tackle simpler tasks which would be reachable using less complex policies. Enabling the learner to decide autonomously the complexity of the policy necessary to solve a task would allow the approach to be adaptive, and suitable to a greater number of problems. Options (Sutton et al., 1999) are a different way to tackle this problem by offering temporally abstract actions to the learner. However each option is built to reach one particular task and they have only been tested for discrete

tasks and actions, in which a small number of options were used, whereas our new proposed learner is to be able to create an unlimited number of complex policies.

As we aim at learning a hierarchical set of interrelated complex tasks, our algorithm could also benefit from this task hierarchy (as Forestier & Oudeyer (2016) did for learning tool use with simple primitive policies only), and try to reuse previously acquired skills to build more complex ones. Barto et al. (2013) showed that building complex actions made of lower-level actions according to the task hierarchy can bootstrap exploration by reaching interesting outcomes more rapidly.

In this paper, we would like to enable a robot learner to achieve a wide range of tasks that can be inter-related and complex. We allow the robot to use sequences of actions of undetermined length to achieve these tasks. We propose an algorithm for the robot to learn which sequence of actions to use to achieve any task in the infinite field of tasks. The learning algorithm has to face the problem of unlearnability of infinite task and policy spaces, and the curse of dimensionality of sequences of high-dimensionality policy spaces. Thus, we extend SGIM-ACTS so as to learn complex motor policies of unlimited size. We develop a new framework called "procedures" (see Section 2.2) which proposes to combine known policies according to their outcome. Combining these, we developed a new algorithm called Socially Guided Intrinsic Motivation with Procedure Babbling (SGIM-PB) capable of taking task hierarchy into account to learn a set of complex interrelated tasks using adapted complex policies. We will describe an experiment, on which we have tested our algorithm, and we will present and analyze the results.

2 Our approach

Inspired by developmental psychology, we combine interactive learning and autonomous exploration in a strategic learner, which learning process is driven by intrinsic motivation. This learner also takes task hierarchy into account to reuse its previously learned skills while adapting the complexity of its policy to the complexity of the task at hand.

In this section, we formalize our learning problem and explain the principles of SGIM-PB.

2.1 Problem formalization

In our approach, an agent can perform motions through the use of policies π_θ, parametrized by $\theta \in \Pi$, and those policies have an effect on the environment, which we call the outcome $\omega \in \Omega$. The agent is then to learn the mapping between Π and Ω: it learns to predict the outcome ω of each policy π_θ (the forward model M), but more importantly, it learns which policy to choose for reaching any particular outcome (an inverse model L). The outcomes ω can be of different dimensionality and thus be split in subspaces $\Omega_i \subset \Omega$. The policies do not only consist of one primitive but of a succession of primitives (each encoded by the parameters in Π) that are executed sequentially by the agent. Hence, policies also are of different dimensionality and are split in policy spaces $\Pi_i \subset \Pi$ (where i corresponds to the number of primitives used in each policy of Π_i). Complex policies are represented by concatenating the parameters of each of its primitive policies in execution order.

We take the trial and error approach, and suppose that Ω is a metric space, meaning the learner has a means of evaluating a distance between two outcomes $d(\omega_1, \omega_2)$.

2.2 Procedures

As this algorithm tackles the learning of complex hierarchically organized tasks, exploring and exploiting this hierarchy could ease the learning of the more complex tasks. We define procedures as a way to encourage the robot to reuse previously learned skills, and chain them to build more complex ones. More formally, a procedure is built by choosing previously known outcomes $(t_1, t_2, ..., t_n \in \Omega)$ and is noted $t_1 \boxplus t_1 \boxplus ... \boxplus t_n$.

Executing a procedure $t_1 \boxplus t_1 \boxplus ... \boxplus t_n$ means building the complex policy π_θ corresponding to the succession of policies $\pi_{\theta_i}, i \in [\![1, n]\!]$ (potentially complex as well) and execute it (where the π_{θ_i} reach best the t_i $\forall i \in [\![1, n]\!]$ respectively). As the subtasks t_i are generally unknown from the learner, the procedure is updated before execution (see Algo. 1) to subtasks t'_i which are the closest known by the learner according to its current skill set (by executing respectively $\pi_{\theta'_1}$ to $\pi_{\theta'_n}$). When

the agent selects a procedure to be executed, this latter is only a way to build the complex policy which will actually be executed. So the agent does not check if the subtasks are actually reached when executing a procedure.

Algorithm 1 Procedure modification before execution

Input: $(t_1, ..., t_n) \in \Omega^n$
Input: inverse model L
  for $i \in [\![1, n]\!]$ **do**
    $t'_i \leftarrow$Nearest-Neighbour(t_i) // *get the nearest outcome known from* t_i
    $\pi_{\theta'_i} \leftarrow L(t'_i)$ // *get the known complex policy that reached* t'_i
  end for
  return $\pi_\theta = \pi_{\theta'_1} ... \pi_{\theta'_n}$

2.3 Socially Guided Intrinsic Motivation with Procedure Babbling

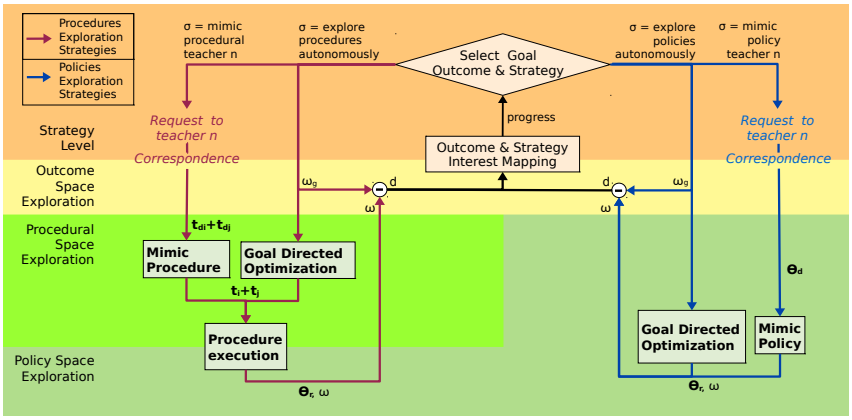

Figure 1: SGIM-PB architecture

The SGIM-PB algorithm (see Algo. 2, Fig. 1) learns by episodes, where an outcome $\omega_g \in \Omega$ to target and an exploration strategy σ have been selected. It is an extension of SGIM-ACTS (Nguyen & Oudeyer, 2012), which can perform complex motor policies and size 2 procedures (sequences of 2 subtasks only). It uses the same interest model and memory based inverse model.

2.3.1 Episodes using exploration strategies

In an episode under the policy space exploration strategy, the learner tries to optimize the policy π_θ to produce ω_g by choosing between random exploration of policies and local optimization, following the SAGG-RIAC algorithm (Baranes & Oudeyer, 2010) (Goal-Directed Policy Optimization(ω_g)). Local optimization uses local linear regression. This is a slightly modified version of the SGIM-ACTS autonomous exploration strategy.

In an episode under the procedural space exploration strategy, the learner builds a size 2 procedure $t_i \boxplus t_j$ such as to reproduce the goal outcome ω_g the best (Goal-Directed Procedure Optimization(ω_g)). It chooses either random exploration of procedures (which builds procedures by generating two subtasks at random) when the goal outcome is far from any previously reached one, or local procedure optimization, which optimizes a procedure using local linear regression. The procedure built is then modified and executed, following Algo. 1.

In an episode under the mimicry of one policy teacher strategy, the learner requests a demonstration ξ_d from the chosen teacher. ξ_d is selected by the teacher as the closest from the goal outcome ω_g in its demonstration repertoire. The learner then repeats the demonstrated policy (Mimic Policy(ξ_d)). It is a strategy directly also available in the SGIM-ACTS algorithm.

In an episode under the mimicry of one procedural teacher strategy, the learner requests a procedural demonstration of size 2 $t_{di} \boxplus t_{dj}$ which is built by the chosen teacher according to a preset function which depends on the target outcome ω_g. Then the learner tries to reproduce the demonstrated procedure by refining and executing it, following Algo. 1 (Mimic Procedure($t_{di} \boxplus t_{dj}$)).

In both autonomous exploration strategies, the learner uses a method, Goal-Directed Optimization, to optimize its input parameters (procedure for the procedure exploration and policy for the policy exploration) to reach ω_g best. This generic method works similarly in both cases, by creating random inputs, if the goal outcome ω_g is far from any previously reached one, or local optimization which uses linear regression.

2.3.2 INTEREST MAPPING

After each episode, the learner stores the policies and modified procedures executed along with their reached outcomes in its episodic memory. It computes its *competence* in reaching the goal outcome ω_g by computing the distance $d(\omega_r, \omega_g))$ with the outcome ω_r it actually reached. Then it updates its interest model by computing the interest $interest(\omega, \sigma)$ of the goal outcome and each outcome reached (including the outcome spaces reached but not targeted): *interest*$(\omega, \sigma) = p(\omega)/K(\sigma)$, where $K(\sigma)$ is the cost of the strategy used and the progress $p(\omega)$ is the derivate of the competence.

The learning agent then uses these interest measures to partition the outcome space Ω into regions of high and low interest. For each strategey σ, the outcomes reached and the goal are added to their partition region. Over a fixed number of measures of interest in the region, it is then partitioned into 2 subregions so as maximise the difference in interest between the 2 subregions. The method used is detailed in Nguyen & Oudeyer (2014). Thus, the learning agent discovers by itself how to organise its learning process and partition its task space into unreachable regions, easy regions and difficult regions, based on empirical measures of competence and interest.

Algorithm 2 SGIM-PB

Input: the different strategies $\sigma_1, ..., \sigma_n$
Initialization: partition of outcome spaces $R \leftarrow \bigsqcup_i \{\Omega_i\}$
Initialization: episodic memory *Memo* $\leftarrow \varnothing$
  loop
    $\omega_g, \sigma \leftarrow$ Select Goal Outcome and Strategy(R)
    if σ = Mimic policy teacher i strategy **then**
      $(\xi_d, \omega_d) \leftarrow$ ask and observe demonstrated policy to teacher i
      Memo $\leftarrow$ Mimic Policy(ξ_d)
    else if σ = Mimic procedural teacher i strategy **then**
      $(t_{di} \boxplus t_{dj}, \omega_d) \leftarrow$ ask and observe demonstrated procedure to teacher i
      Memo $\leftarrow$ Mimic Procedure($t_{di} \boxplus t_{dj}$)
    else if σ = Autonomous exploration of procedures strategy **then**
      Memo $\leftarrow$ Goal-Directed Procedure Optimization(ω_g)
    else
      σ = Autonomous exploration of policies strategy
      Memo $\leftarrow$ Goal-Directed Policy Optimization(ω_g)
    end if
    Update L^{-1} with collected data *Memo*
    $R \leftarrow$ Update Outcome and Strategy Interest Mapping(R,*Memo*,ω_g)
  end loop

The choice of strategy and goal outcome is based on the empirical progress measured in each region R_n of the outcome space Ω. ω_g, σ are chosen stochastically (with respectively probabilities p_1, p_2, p_3), by one of the sampling modes:

- mode 1: choose σ and $\omega_g \in \Omega$ at random;
- mode 2: choose an outcome region R_n and a strategy σ with a probability proportional to its interest value. Then generate $\omega_g \in R_n$ at random;
- mode 3: choose σ and R_n like in mode 2, but generate a goal $\omega_g \in R_n$ close to the outcome with the highest measure of progress.

When the learner computes nearest neighbours to select policies or procedures to optimize (when choosing local optimization in any of both autonomous exploration strategies and when refining procedures), it actually uses a performance metric (1) which takes into account the complexity of the policy chosen:

$$perf = d(\omega, \omega_g)\gamma^n \tag{1}$$

where $d(\omega, \omega_g)$ is the normalized Euclidean distance between the target outcome ω_g and the outcome ω reached by the policy, γ is a constant and n is equal to the size of the policy (the number of primitives chained).

In this section, we have formalized the problem of learning an inverse model between an infinite space of outcomes and an infinite space of policies. We have introduced the framework of procedures to allow the learning agent to learn sequences of primitive policies as task compositions. We have then proposed SGIM-PB as a learning algorithm that leverages goal-babbling for autonomous exploration, sequences to learn complex policies, and social guidance to bootstrap the learning. SGIM-PB learns to reach an ensemble of outcomes, by mapping them with policies, but also with subgoal outcomes.

The formalization and algorithmic architecture proposed are general and can apply to a high number of problems. The requirements for an experimental setup are:

- to define the primitive policies of the robot
- to define the different outcomes the user is interested in by defining the variables from the sensors needed and a rough range of their values (we do not need a precise estimation as the algorithm is robust to overestimations of these ranges, see Nguyen & Oudeyer (2014)).
- a measure for the robot to assess its own performance such as a distance, as in all intrinsic motivation based algorithms. This measure is used as an internal reward function. Contrarily to classical reinforcement learning problems, this reward function is not fine tuned to the specific goal at hand, but is a generic function for all the goals in the outcome space.
- the environment and robot can reset to an initial state, as in most reinforcement learning algorithms.

3 EXPERIMENT

In this study, we designed an experiment with a simulated robotic arm, which can move in its environment and interact with objects in it. It can learn an infinite number of tasks, organized as 6 hierarchically organized types of tasks. The robot is capable of performing complex policies of unrestricted size (i.e. consisting of any number of primitives), with primitive policies highly redundant and of high dimensionality.

3.1 SIMULATION SETUP

The Fig. 2 shows environmental setup (contained in a cube delimited by $(x, y, z) \in [-1; 1]^3$). The learning agent is a planar robotic arm of 3 joints with the base centred on the horizontal plane, able to rotate freely around the vertical axis (each link has a length of 0.33) and change its vertical position. The robot can grab objects in this environment, by hovering its arm tip (blue in the Fig. 2) close to them, which position is noted (x_0, y_0, z_0). The robot can interact with:

- Floor (below $z = 0.0$): limits the motions of the robot, slightly elastic which enable the robot to go down to $z = -0.2$ by forcing on it;
- Pen: can be moved around and draw on the floor, broken if forcing too much on the floor (when $z <= -0.3$);
- Joystick 1 (the left one on the figure): can be moved inside a cube-shaped area (automatically released otherwise, position normalized for this area), its x-axis position control a video-game character x position on the screen when grabbed by the robot;

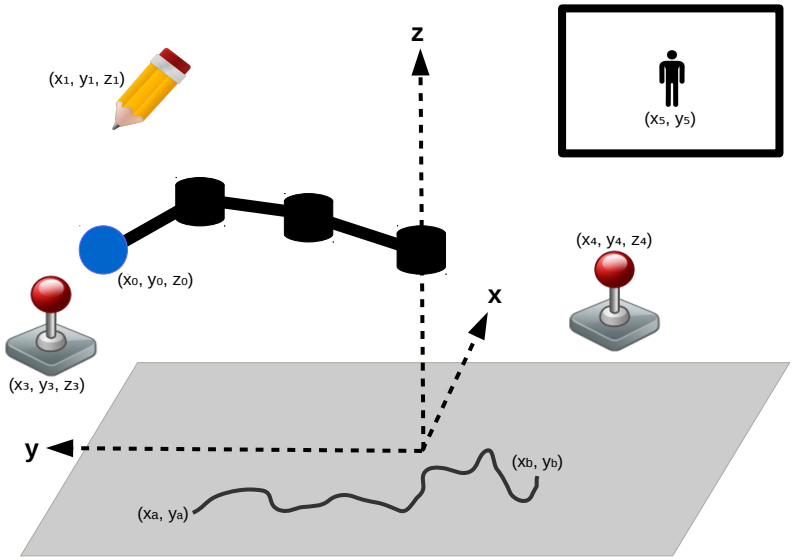

Figure 2: Experimental setup: a robotic arm, can interact with the different objects in its environment (a pen and two joysticks). Both joysticks enable to control a video-game character (represented in top-right corner). A grey floor limits its motions and can be drawn upon using the pen (a possible drawing is represented).

- Joystick 2 (the right one on the figure): can be moved inside a cube-shaped area (automatically released otherwise, position normalized for this area), its y-axis position control a video-game character y position on the screen when grabbed by the robot;
- Video-game character: can be moved on the screen by using the two joysticks, its position is refreshed only at the end of a primitive policy execution for the manipulated joystick.

The robot grabber can only handle one object. When it touches a second object, it breaks, releasing both objects.

The robot always starts from the same position before executing a policy, and primitives are executed sequentially without getting back to this initial position. Whole complex policies are recorded with their outcomes, but each step of the complex policy execution is recorded.

3.2 Experiment variables

3.2.1 Policy spaces

The motions of each of the three joints of the robot are encoded using a one-dimensional Dynamic Movement Primitive (Pastor et al., 2009), defined by the system:

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0)f(s) \tag{2}$$
$$\tau \dot{x} = v \tag{3}$$
$$\tau \dot{s} = -\alpha s \tag{4}$$

where x and v are the position and velocity of the system; s is the phase of the motion; x_0 and g are the starting and end position of the motion; τ is a factor used to temporally scale the system (set to fix the length of a primitive execution); K and D are the spring constant and damping term fixed for

the whole experiment; α is also a constant fixed for the experiment; and f is a non-linear term used to shape the trajectory called the forcing term. This forcing term is defined as:

$$f(s) = \frac{\sum_i \omega_i \psi_i(s) s}{\sum_i \psi_i(s)} \tag{5}$$

where $\psi_i(s) = \exp(-h_i(s - c_i)^2)$ with centers c_i and widths h_i fixed for all primitives. There are 3 weights ω_i per DMP.

The weights of the forcing term and the end positions are the only parameters of the DMP used by the robot. The starting position of a primitive is set by either the initial position of the robot (if it is starting a new complex policy) or the end position of the preceding primitive. The robot can also set its position on the vertical axis z for every primitive. Therefore a primitive policy π_θ is parametrized by:

$$\theta = (a_0, a_1, a_2, z) \tag{6}$$

where $a_i = (\omega_0^{(i)}, \omega_1^{(i)}, \omega_2^{(i)}, g^{(i)})$ corresponds to the DMP parameters of the joint i, ordered from base to tip, and z is the fixed vertical position. When combining two or more primitive policies $(\pi_{\theta_0}, \pi_{\theta_1}, ...)$, in a complex policies π_θ, the parameters $(\theta_0, \theta_1, ...)$ are simply concatenated together from the first primitive to the last.

3.2.2 OUTCOME SUBSPACES

The outcome subspaces the robot learns to reach are hierarchically organized and defined as:

- Ω_0: the position (x_0, y_0, z_0) of the end effector of the robot in Cartesian coordinates at the end of a policy execution;
- Ω_1: the position (x_1, y_1, z_1) of the pen at the end of a policy execution if the pen is grabbed by the robot;
- Ω_2: the first (x_a, y_a) and last (x_b, y_b) points of the last drawn continuous line on the floor if the pen is functional (x_a, y_a, x_b, y_b);
- Ω_3: the position (x_3, y_3, z_3) of the first joystick at the end of a policy execution if it is grabbed by the robot;
- Ω_4: the position (x_4, y_4, z_4) of the second joystick at the end of a policy execution if it is grabbed by the robot;
- Ω_5: the position (x_5, y_5) of the video-game character at the end of a policy execution if moved.

3.3 THE TEACHERS

To help the SGIM-PB learner, procedural teachers were available so as to provide procedures for every outcome subspaces but Ω_0. Each teacher was only giving procedures useful for its own outcome space, and was aware of the task representation. They all had a cost of 5. The rules used to provide procedures are the following:

- ProceduralTeacher1 (Ω_1): $t_1 \boxplus t_0$ with $t_1 \in \Omega_1$ corresponding to the pen initial position and $t_0 \in \Omega_0$ corresponding to the desired final pen position;
- ProceduralTeacher2 (Ω_2): $t_1 \boxplus t_0$ with $t_1 \in \Omega_1$ corresponding to the point on the $z = 1.0$ plane above the first point of the desired drawing, and $t_0 \in \Omega_0$ corresponding to the desired final drawing point;
- ProceduralTeacher3 (Ω_3): $t_3 \boxplus t_0$ with $t_3 \in \Omega_3$ and $t_3 = (0, 0, 0)$, $t_0 \in \Omega_0$ corresponding to the end effector position corresponding to the desired final position of the first joystick;
- ProceduralTeacher4 (Ω_4): $t_4 \boxplus t_0$ with $t_4 \in \Omega_4$ and $t_4 = (0, 0, 0)$, $t_0 \in \Omega_0$ corresponding to the end effector position corresponding to the desired final position of the second joystick;

- ProceduralTeacher5 (Ω_5): $t_3 \boxplus t_4$ with $t_3 \in \Omega_3$ and $t_3 = (x, 0, 0)$ with x corresponding to the desired x-position of the video-game character, $t_4 \in \Omega_4$ and $t_4 = (0, y, 0)$ with y corresponding to the desired y-position of the video-game character.

We also added policy teachers corresponding to the same outcome spaces to bootstrap the robot early learning process. The strategy attached to each teacher has a cost of 10. Each teacher was capable to provide demonstrations (as policies executable by the robot) linearly distributed in its outcome space:

- MimicryTeacher1 (Ω_1): 15 demonstrations;
- MimicryTeacher2 (Ω_2): 25 demonstrations;
- MimicryTeacher3 (Ω_3): 18 demonstrations;
- MimicryTeacher4 (Ω_4): 18 demonstrations;
- MimicryTeacher5 (Ω_5): 9 demonstrations;

3.4 Evaluation method

To evaluate our algorithm, we created a benchmark dataset for each outcome space Ω_i, linearly distributed across the outcome space dimensions, for a total of 27,600 points. The evaluation consists in computing the normalized Euclidean distance between each of the benchmark outcome and their nearest neighbour in the learner dataset. Then we compute the mean distance to benchmark for each outcome space. The global evaluation is the mean evaluation for the 6 outcome spaces. This process is then repeated across the learning process at predefined and regularly distributed timestamps.

Then to asses our algorithm efficiency, we compare its results with 3 other algorithms:

- SAGG-RIAC: performs autonomous exploration of the policy space Π guided by intrinsic motivation;
- SGIM-ACTS: interactive learner driven by intrinsic motivation. Choosing between autonomous exploration of the policy space Π and mimicry of one of the available policy teachers;
- IM-PB: performs both autonomous exploration of the procedural space and the policy space, guided by intrinsic motivation;
- SGIM-PB: interactive learner driven by intrinsic motivation. Choosing between autonomous exploration strategies (either of the policy space or the procedural space) and mimicry of one of the available teachers (either policy or procedural teachers).

Each algorithm was run 5 times on this setup. Each run, we let the algorithm performs 25,000 iterations (complex policies executions). The value of γ for this experiment is 1.2. The probabilities to choose either of the sampling mode of SGIM-PB are $p_1 = 0.15$, $p_2 = 0.65$, $p_3 = 0.2$. The code run for this experiment can be found here.

4 Results

The Fig. 3 shows the global evaluation of all the tested algorithms, which corresponds to the mean error made by each algorithm to reproduce the benchmarks with respect to the number of complete complex policies tried. The algorithms capable of performing procedures (IM-PB and SGIM-PB) have errors that drop to levels lower than the their non-procedure equivalents (respectively SAGG-RIAC and SGIM-ACTS), moreover since the beginning of the learning process (shown on Fig. 3). It seems that the procedures bootstrap the exploration, enabling the learner to progress further. Indeed, the autonomous learner SAGG-RIAC has significantly better performance when it can use procedures and is thus upgraded to IM-PB.

We can also see that the SGIM-PB algorithm has a very quick improvement in global evaluation owing to the bootstrapping effect of the different teachers. It goes lower to the final evaluation of SAGG-RIAC (0.17) after only 500 iterations. This bootstrapping effect comes from the mimicry teachers, as it is also observed for SGIM-ACTS which shares the same mimicry teachers.

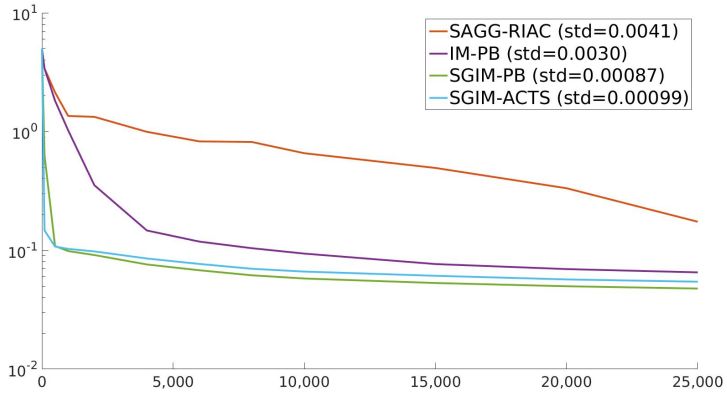

Figure 3: Evaluation of all algorithms (final standard deviation shown in caption)

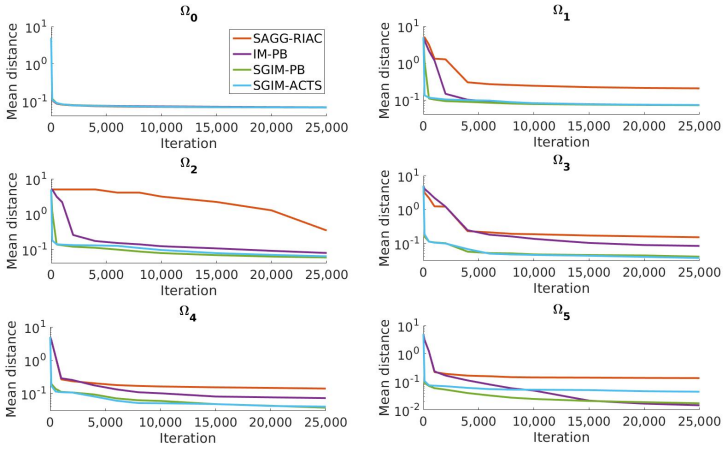

Figure 4: Evaluation of all algorithms per outcome space (for Ω_0, all evaluations are superposed)

If we look at the evaluation on each individual outcome space (Fig. 4), we can see that the learners with demonstrations (SGIM-PB and SGIM-ACTS) outperforms the other algorithms, except for the outcome space Ω_5 where IM-PB is better, due to the fact that IM-PB practiced much more on this outcome space (500 iterations where the goal was in Ω_5 against 160 for SGIM-PB) on this outcome space. SGIM-PB and SGIM-ACTS are much better than the other algorithms on the two joysticks outcome spaces (Ω_3 and Ω_4). This is not surprising given the fact that those outcome spaces require precise policies. Indeed, if the end-effector gets out of the area where it can control the joystick, the latter is released, thus potentially ruining the attempt. So on these outcome spaces working directly on carefully crafted policies can alleviate this problem, while using procedures might be tricky, as the outcomes used don't take into account the motion trajectory but merely its final state. SGIM-PB was provided with such policies by the policy teachers. Also if we compare the results of the autonomous learner without procedures (SAGG-RIAC) with the one with procedures (IM-PB), we can see that it learn less on any outcome space but Ω_0 (which was the only outcome space reachable using only single primitive policies and that could not benefit from using the task hierarchy to be learned) and especially for Ω_1, Ω_2 and Ω_5 which were the most hierarchical in this setup. More generally, it seems than on this highly hierarchical Ω_5, the learners with procedures were better. So the procedures helped when learning any potentially hierarchical task in this experiment.

We further analyzed the results of our SGIM-PB learner. We looked in its learning process to see which pairs of teachers and target outcomes it has chosen (Fig. 5). It was capable to request demonstrations from the relevant teachers depending on the task at hand, except for the outcome space Ω_0 which had no human teachers and therefore could not find a better teacher to help it.

Indeed, for the outcome space Ω_2, the procedural teacher (ProceduralTeacher2) specially built for this outcome space was greatly chosen.

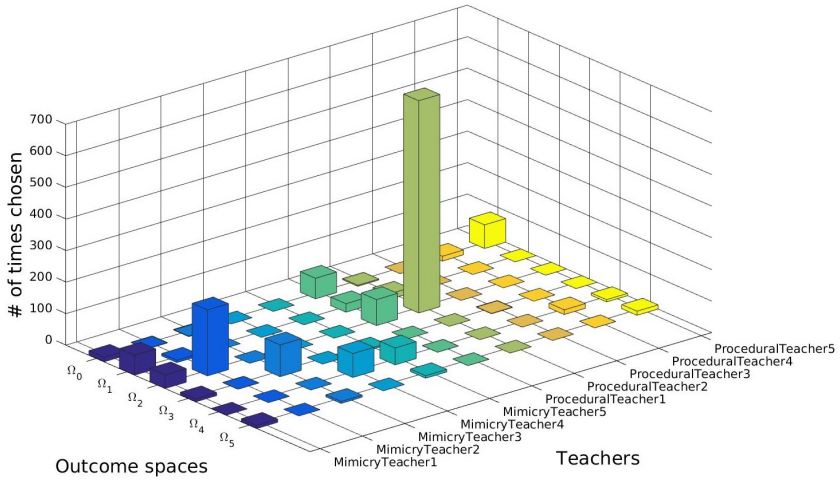

Figure 5: Choices of teachers and target outcomes of the SGIM-PB learner

We wanted to see if our SGIM-PB learner adapts the complexity of its policies to the working task. We draw 1,000,000 goal outcomes for each of the Ω_0, Ω_1 and Ω_2 subspaces (chosen because they are increasingly complex) and we let the learner choose the known policy that would reach the closest outcome. Fig. 6 shows the results of this analysis.

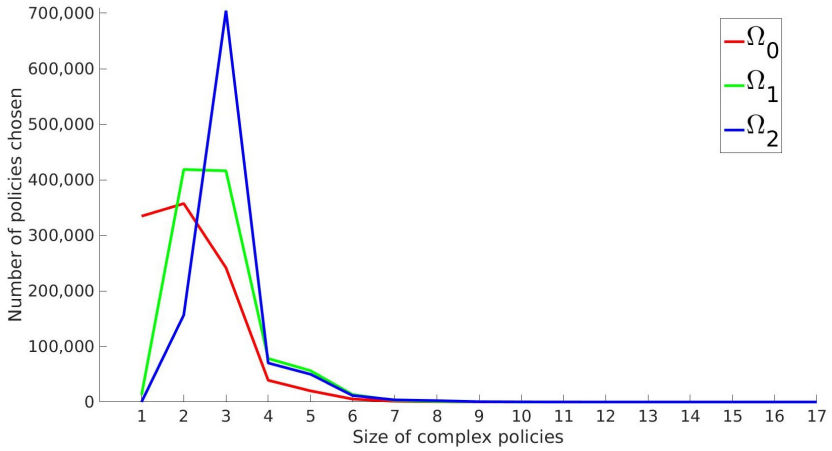

Figure 6: Number of policies selected per policy size for three increasingly more complex outcome spaces by the SGIM-PB learner

As we can see on those three interrelated outcome subspaces (Fig. 6), the learner is capable to adapt the complexity of its policies to the outcome at hand. It chooses longer policies for the Ω_1 subspace (policies of size 2 and 3 while using mostly policies of size 1 and 2 for Ω_0) and even longer for the Ω_2 subspace (using far more policies of size 3 than for the others). It shows that our learner is capable to correctly limit the complexity of its policies instead of being stuck into always trying longer and longer policies.

5 Conclusion and Future Work

With this experiment, we show the capability of SGIM-PB to tackle the learning of a set of multiple interrelated complex tasks. It successfully discovers the hierarchy between tasks and uses complex

motor policies to learn a wider range of tasks. It is capable to correctly choose the most adapted teachers to the target outcome when available. Though it is not limited in the size of policies it could execute, the learner shows it could adapt the complexity of its policies to the task at hand.

The procedures greatly improved the learning capability of autonomous learners, as shows the difference between IM-PB and SAGG-RIAC . Our SGIM-PB shows it is capable to use procedures to discover the task hierarchy and exploit the inverse model of previously learned skills. More importantly, it shows it can successfully combine the ability of SGIM-ACTS to progress quickly in the beginning (owing to the mimicry teachers) and the ability of IM-PB to progress further on highly hierarchical tasks (owing to the procedure framework).

In this article, we aimed to enable a robot to learn sequences of actions of undetermined length to achieve a field of outcomes. To tackle this high-dimensionality learning between a continuous high-dimensional space of outcomes and a continuous infinite dimensionality space of sequences of actions , we used techniques that have proven efficient in previous studies: goal-babbling, social guidance and strategic learning based on intrinsic motivation. We extended them with the procedures framework and proposed SGIM-PB algorithm, allowing the robot to babble in the procedure space and to imitate procedural teachers. We showed that SGIM-PB can discover the hierarchy between tasks, learn to reach complex tasks while adapting the complexity of the policy. The study shows that :

- procedures allow the learner to learn complex tasks, and adapt the length of sequences of actions to the complexity of the task
- social guidance bootstraps the learning owing to demonstrations of primitive policy in the beginning, and then to demonstrations of procedures to learn how to compose tasks into sequences of actions
- intrinsic motivation can be used as a common criteria for active learning for the robot to choose both its exploration strategy, its goal outcomes and the goal-oriented procedures.

However a precise analysis of the impact of each of the different strategies used by our learning algorithm could give us more insight in the roles of the teachers and procedures framework. Also, we aim to illustrate the potency of our SGIM-PB learner on a real-world application. We are currently designing such an experiment with a real robotic platform.

Besides, the procedures are defined as combinations of any number of subtasks but are used in the illustration experiment as only combinations of two subtasks. It could be a next step to see if the learning algorithm can handle the curse of dimensionality of a larger procedure space, and explore combinations of any number of subtasks. Moreover, the algorithm can be extended to allow the robot learner to decide on how to execute a procedure. In the current version, we have proposed the "refinement process" to infer the best policy. We could make this refinement process more recursive, by allowing the algorithm to select, not only policies, but also lower-level procedures as one of the policy components.

REFERENCES

Brenna D. Argall, B. Browning, and Manuela Veloso. Learning robot motion control with demonstration and advice-operators. In *In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 399–404. IEEE, September 2008.

Y. Baram, R. El-Yaniv, and K. Luz. Online choice of active learning algorithms. *The Journal of Machine Learning Research,*, 5:255–291, 2004.

Adrien Baranes and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration for active motor learning in robots: A case study. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 1766–1773. IEEE, 2010.

Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

Andrew G. Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proc. 3rd Int. Conf. Development Learn*, pp. 112–119, 2004.

Andrew G Barto, George Konidaris, and Christopher Vigorito. Behavioral hierarchy: exploration and representation. In *Computational and Robotic Models of the Hierarchical Organization of Behavior*, pp. 13–46. Springer, 2013.

Maya Cakmak, C. Chao, and Andrea L. Thomaz. Designing interactions for robot active learners. *Autonomous Mental Development, IEEE Transactions on*, 2(2):108–118, 2010.

Sonia Chernova and Manuela Veloso. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research*, 34(1):1, 2009.

E.L. Deci and Richard M. Ryan. *Intrinsic Motivation and self-determination in human behavior*. Plenum Press, New York, 1985.

N. Duminy, S. M. Nguyen, and D. Duhaut. Strategic and interactive learning of a hierarchical set of tasks by the Poppy humanoid robot. In *2016 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pp. 204–209, September 2016.

Sébastien Forestier and Pierre-Yves Oudeyer. Curiosity-driven development of tool use precursors: a computational model. In *38th Annual Conference of the Cognitive Science Society (CogSci 2016)*, pp. 1859–1864, 2016.

Daniel H Grollman and Odest Chadwicke Jenkins. Incremental learning of subtasks from unsegmented demonstration. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 261–266. IEEE, 2010.

Manuel Lopes and Pierre-Yves Oudeyer. The strategic student approach for life-long exploration and learning. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pp. 1–8. IEEE, 2012.

Max Lungarella, G. Metta, Rolf Pfeifer, and i G. Sandin. Developmental robotics: a survey. *Connection Science*, 15(4):151–190, 2003.

Sao Mai Nguyen and Pierre-Yves Oudeyer. Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner. *Paladyn Journal of Behavioural Robotics*, 3(3): 136–146, 2012.

Sao Mai Nguyen and Pierre-Yves Oudeyer. Socially guided intrinsic motivation for robot learning of motor skills. *Autonomous Robots*, 36(3):273–294, 2014. doi: 10.1007/s10514-013-9339-y.

Sao Mai Nguyen, Adrien Baranes, and Pierre-Yves Oudeyer. Bootstrapping intrinsically motivated learning with human demonstrations. In *IEEE International Conference on Development and Learning*, volume 2, pp. 1–8. IEEE, 2011.

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:1:6, 2009.

Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 763–768. IEEE, 2009.

Jan Peters and Stefan Schaal. Natural actor critic. *Neurocomputing*, (7-9):1180–1190, 2008.

M. Rolf, J. Steil, and M. Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Trans. Autonomous Mental Development*, 2(3):216–229, 09/2010 2010.

Freek Stulp and Stefan Schaal. Hierarchical reinforcement learning with movement primitives. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pp. 231–238. IEEE, 2011.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Andrea L. Thomaz and Cynthia Breazeal. Experiments in socially guided exploration: Lessons learned in building robots that learn with and without human teachers. *Connection Science*, 20 Special Issue on Social Learning in Embodied Agents(2,3):91–110, 2008.