Autonomous Reuse of Motor Exploration Trajectories

Fabien Benureau, Pierre-Yves Oudeyer

Autonomous Reuse of Motor Exploration Trajectories

Fabien Benureau
Flowers Team, Inria/ENSTA-Paritech
Bordeaux University
fabien.benureau@inria.fr

Pierre-Yves Oudeyer
Flowers Team, Inria/ENSTA-Paritech
pierre-yves.oudeyer@inria.fr

***Abstract*— We present an algorithm for transferring exploration strategies between tasks that share a common motor space in the context of lifelong autonomous learning in robotics. The algorithm does not transfer observations, or make assumptions about how the learning is conducted. Instead, only selected motor commands are transferred between tasks, chosen autonomously according to an empirical measure of learning progress. We show that on a wide variety of variations from a source task, such as changing the object the robot is interacting with or altering the morphology of the robot, this simple and flexible transfer method increases early performance significantly in the new task. We also provide examples of situations where the transfer is not helpful.**

I. MOTIVATION

Lifelong learning is a desirable skill of developmental robotics [10], [20]. Over the life of a robot learner, many different situations are encountered; those situations share common properties, the most prevalent of them being the robot to which they happen, whose morphology and dynamics, while not necessarily constant, remains significantly similar from one situation to the other. Moreover, after training on a particular task, the learning data contains information not only on how to achieve the task, but also on how to learn it. And because learning for robots implies interaction with the world, it is usually particularly costly in time and resources; in a practical setting, a robot learner might not be able to afford the time to near the asymptotic performance at a particular task. Thus, any way to efficiently improve the early performance from past experience is hugely beneficial. Under those assumptions — that a robot's learning experiences share underlying common characteristics, that knowledge on how to learn is present in the learning data, and that learning resources are inherently limited — learning how to learn, autonomously, and transferring this knowledge between tasks is a fundamental part of any lifelong robot learner.

In this article, we present a method for reusing motor *exploration trajectories*. When learning a task, a learner progressively refines its actions to produce more accurate results, as observations accumulate. The *exploration trajectory* is the chronology of those observations, each observation being composed of an action and its observed effect. A *motor exploration trajectory*, then, is the chronology of the actions alone, stripped of their corresponding effect. An *exploration strategy* is the algorithm the learner employs to create its motor exploration trajectory.

Work on intrinsic motivation [7], [9], [11] and on goal babbling [8] in the context of active learning showed that efficient exploration strategies can dramatically increase learning performance. We make the hypothesis that the exploration trajectories of those strategies possess structure that can be transferred and reused across similar tasks.

Specifically, in this paper, we are interested in improving the performance of motor babbling. Random motor babbling is notoriously inefficient in highly-redundant, high-dimension spaces typical of sensorimotor spaces. Goal babbling [7], [8] has been demonstrated to be superior in many scenarios. However, motor babbling is often the only way to start learning an unknown task, and keep discovering unknown areas of the learning space. Providing ways to improve the efficiency of motor babbling could dramatically increase early task performance, decreasing the time before a piece of knowledge can be useful, and provide better observations for later stage exploration, impacting middle and long term performance.

In this paper, we consider environments where a robot is equipped with a motor primitive [21] controlled by continuous parameters, and a sensory primitive, providing a high-level representation of the environment as a vector of continuous variables. We define a *task* over a specific environment as the mapping between the space of the motor parameter and the space of sensory variables. We restrict ourselves to episodic tasks over one time-step. Examples of such tasks include inverse kinematics, interaction with fixed objects, repeated situations such as learning to throw or hit a ball, etc. It is important to note that the environment here includes the morphology of the robot, which can change from one task to the other, which is an important point in a developmental perspective.

A. Previous Work

Transferring knowledge from one task to another has been formalized in the context of *transfer learning* [14], [15]. A *source task* is learned, and knowledge extracted from this task is transferred to a *target task*, where it is leveraged to improve learning performance. Figure 1 introduces the different effect transfer can have on performance. Many different approaches exist to transfer learning.

One can transfer the training data from the source task to the target task, eventually applying relevant transformations to it [13]. In the context of reinforcement learning, starting point methods [18] set the final Q-table of a source task as the initial one of the target task, and usually provide a jumpstart for the performance. However, when those tasks don't match perfectly, an expert is needed to map the first task to the second task.

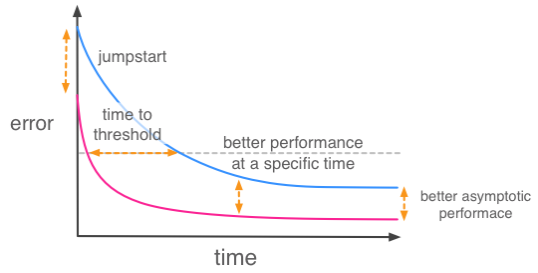


Fig. 1. Different ways transfer can impact learning performance. The blue and the pink curves are the average error without and with transfer, respectively. In the context of this paper, we are mainly interested in the performance difference at a specific time.

Another approach modifies the representation of the target task by leveraging source task knowledge, either reducing the dimensionality of the state or action space by discovering latent space parametrizations [16], or expanding it by adding new state variables in the target space [17].

One can also leverage its model of the world, if the environment is shared between task, to bias the learning of a control function, as in [19].

Our work share some resemblance to the imitation learning approach of [12], where source task policies are reused to direct exploration in the target task. In [12], transferred policies must, in the MDP formalism, share the same transition functions. Our algorithm does not have such a constraint, and tasks are defined as different precisely in the sense that the same actions will lead to different effects in each task; in fact, the space of sensory features itself can differ.

B. Contribution

We contribute a transfer learning algorithm based on exploration trajectories and intrinsic motivation over continuous, high-dimension sensorimotor spaces. Its main distinctive properties are :

- No training data is shared between tasks. This diminishes the risk of negative transfer. However, no jumpstart can be expected.
- Contrary to many transfer learning algorithms, we do not make specific assumptions about the machine learning algorithm employed to learn the source or the target task, or even if they are the same.
- The knowledge transferred depends only on the source task. As such, once computed, it does not need to be adaptated to the target task it is applied to.
- The only constraint for the transfer is that the motor space should be the same between tasks. The sensory space, its features and the environment can be arbitrarily different.
- In this paper, we are not concerned with multitask learning, where multiple tasks are learned, and transfer can happen from any task to any other. We only consider one source and one target task, happenning sequentially.
- Our method is autonomous, requiring no expert or human intervention.

Moreover, to allow easy reproduction of the results and examination of any and all experimental methodology, we also contribute the entire source code (algorithm and simulation environment) used to obtain every results in this article[1].

II. PROBLEM

In this section, we formally define the problem.

Henceforth, a *task* is defined as a set (M, S, f, n).

- M is the motor space, that is, the set of legal commands the robot can execute. It is a bounded hyperrectangle of $\mathbb{R}^{d_M}$, with d_M the dimension of the motor space. Here we typically consider that motor primitives are dynamical systems parametrized by a continuous vector of real values.
- S is the sensory space, an arbitrary, bounded, subset of $\mathbb{R}^{d_S}$, with d_S the dimension of the sensory space. *Effects* and *goals* (desired effects) are elements of S.
- f is a function from M into S, returning the response of the environment of a given motor command.
- n is the maximum number of samples of f allowed. In this paper, it represents the number of trials the robot is allowed to execute to construct an inverse model of the environment, that is, computing a function $g_t : S \mapsto M$ with $\int_S \|f(g_t(\mathbf{y})) - \mathbf{y}\| \mathrm{d}\mathbf{y}$ as small as possible.

In the remaining of the article, $\mathbf{x}$ will be used for motor commands, that is, elements of M, and $\mathbf{y}$ for effects, that is, elements of S. An observation is a pair $(\mathbf{x}, \mathbf{y})$ with $f(\mathbf{x}) = \mathbf{y}$.

In order to test the learning performance on a task $A = (M, S, f, n)$, we need to approximate $\int_S \|f(g_t(\mathbf{y})) - \mathbf{y}\| \mathrm{d}\mathbf{y}$. For that, we define test cases as a set E of points uniformly distributed in S. The uniform distributed in S, rather than in M, is not trivial; we are interested in testing the skill of the robots to act in its environment, rather than predict it. Each test case is a vector of the sensory space that the robot has to produce or approximate by executing an appropriate motor command. A *test case* is the same as a *goal* or an *effect*, although we restrict the use of the latter to sensory vectors actually produced by the environment. The performance is the negative of the average error, and we define the average error over E at time t as :

$$e_A(t) = \frac{\sum_{\mathbf{y}_i \in E} \|f(g_t(\mathbf{y}_i)) - \mathbf{y}_i\|}{|E|}$$

with $|E|$ the cardinal of E. Note that we consider here multiple test cases uniformly distributed over the sensory space, which is not typically completely reachable; some test cases may be impossible to reproduce exactly. As a consequence, $e_A(t)$ is not expected to reach zero. Obviously, the learner does not have access to the test cases it will be evaluated on.

In this paper, we are interested at the improvement in learning performance that can be achieved if information is transferred from a *source task* (M_1, S_1, f_1, n_1) to a *target task* (M_2, S_2, f_2, n_2), with the only condition that $M_1 =$

[1]The code is available at fabien.benureau.com

M_2 : the motor space is the same, but sensory features S_1, S_2 and the environment f_1, f_2 can differ.

The *exploration trajectory* ξ_A of a task $A = (M, S, f, n)$ is defined as the chronological sequence of n observations $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{0 \leq i < n}$ with $f(\mathbf{x}_i) = \mathbf{y}_i$ acquired during exploration. While we do not transfer observations pairs between tasks, the effects produced by the motor commands are used to modulate how the motor commands are transfered from the source task to the target task.

To test the performance of our algorithm, we define $e_B(t, A, \xi_A)$, the average error at time t, on a task B, given that a transfer occurred from a source task A involving the training data ξ_A. Then, we can compute :

$$\frac{e_B(t) - e_B(t, A, \xi_A)}{e_B(t)}$$

which represents the percentage of the reduction of error at time t due to the transfer.

Over the course of a training, we can similarly define :

$$\tau_{t_f} = \frac{\sum_{0 \leq t < t_f} e_B(t)dt - e_B(t, A, \xi_A)}{\sum_{0 \leq t < t_f} e_B(t)\mathrm{d}t} \qquad (1)$$

τ_{n_B}, in particular, characterizes the improvement in terms of performance that the learner will experience during the whole learning phase, expressed as a percentage. This is particularly interesting in a practical developmental setting, where a robot would typically need to apply its knowledge in between and during learning phases.

III. METHOD

Our method is organized around three algorithms. The first describes the learning and exploration of the source task. The second is applied at the end of the learning of the source task, and produces the data to be transferred to the target task. The third controls how the transferred data impacts the exploration algorithm in the target task. They are respectively designated as Algorithm 1, 2 and 3.

A. Exploration and Learning for Source Tasks

Given a task (S, M, f, n), we train a predictor to compute a model of the environment, and use a constrained optimization routine on the predictor to compute the inverse model. At each step, we choose the motor command to execute using a combination of motor and goal babbling.

*1) **Forward Model**:*

To approximate the function f from training data, we employ Locally Weighted Linear Regression (LWLR) [1][2], a lazy machine learning algorithm.

Given a set of observations $D = \{(\mathbf{x}_k, \mathbf{y}_k)\}$ where for each k, $f(\mathbf{x}_k) = \mathbf{y}_k$, and a query vector $\mathbf{x}_q$, for which we which to predict the effect, we compute, for each point $\mathbf{x}_k$, the euclidean distance to $\mathbf{x}_q$ and derive a guaussian weight w_k :

$$w_k = e^{\frac{-\|\mathbf{x}_k - \mathbf{x}_q\|^2}{\sigma^2}}$$

We consider the matrices X with $X_{k,i} = (\mathbf{x}_k)_i$, Y with $Y_{k,i} = (\mathbf{y}_k)_i$, and $W = \mathrm{diag}(w_0, w_1, ..., w_n)$, and compute :

$$\beta = ((WX)^T WX)^+ ((WX)^T WY)$$

where $(WX)^T WX$ is a positive definite symmetric matrix, and $((WX)^T WX)^+$ its Moore-Penrose inverse [3].

Then:

$$\mathbf{y}_e = \beta^T \mathbf{x}_q$$

$\mathbf{y}_e$ is the LWLR estimate of $\mathbf{x}_q$, given the observed data D. We define the function PREDICTLWLR$(\mathbf{x}_q, D)$ that compute $\mathbf{y}_e$ for any $\mathbf{x}_q \in M$ given D.

In our implementation, σ, which control the locality of the regression, is dynamically computed. With d_M as the dimension of the motor space, we define a constant $N = 2d_M + 1$, and compute σ as the average distance of the N closest points of the query vector $\mathbf{x}_q$. Additionally, all other points of D besides the N closest neighbors are given a null weight.

*2) **Inverse Model**:*

Given a query point $\mathbf{y}_q \in S$, we want to compute an estimate of $\mathbf{x}_e \in M$ so that $\|f(\mathbf{x}_e) - \mathbf{y}_q\|$ is minimal.

Since M is a hyperrectangle of $\mathbb{R}^{d_M}$, we use L-BFGS-B [4][5], a quasi-Newton method for bound-constrained optimization, to minimize the the error function :

$$\mathrm{ERR}_{\mathbf{y}_q}(\mathbf{x}) = \|\mathrm{PREDICTLWLR}(\mathbf{x}, D) - \mathbf{y}_q\|$$

The optimization process is initialized with the motor command corresponding to the closest neighbor of $\mathbf{y}_q$ in the training data.

Algorithm 1: EXPLORE(A, K_{boot}, p_{goal})

Data:
- $A = (S, M, f, n)$, source task.
- K_{boot}, duration of pure motor bootstrapping.
- p_{goal}, ratio of goal babbling.

Result: $\xi_A = \{\mathbf{x}_i, \mathbf{y}_i\}_{0 \leq i \leq n} \in (S \times M)^n$, exploration trajectory.

$\xi_A \leftarrow []$
for t **from** 0 **to** n **do**
  if $t \leq K_{boot}$ **or** RANDOM() $\geq p_{goal}$ **then**
    choose $\mathbf{x}_t$ randomly in M
    $\mathbf{y}_t \leftarrow f(\mathbf{x}_t)$ // execute the command
    append $(\mathbf{x}_t, \mathbf{y}_t)$ to ξ_A
  else
    choose a goal $\mathbf{g}_t$ randomly in S
    minimize $||\mathbf{g}_t - \mathrm{PREDICTLWLR}(\mathbf{x}_t, \xi_A)||$ using L-BFGS-B
    $\mathbf{y}_t \leftarrow f(\mathbf{x}_t)$ // execute the command
    append $(\mathbf{x}_t, \mathbf{y}_t)$ to ξ_A

*3) **Exploration**:*

For each trial, our algorithm chooses a motor command to sample f. Usually with robots, the motor space is too large to be sampled exhaustively. In our experiments the number of allowed samples is small in comparison with what would be needed to exhaustively sample the motor space to a useful precision. The works of [7] and [8] has shown that goal babbling is an effective method in these situations.

For each sampling of f, the exploration algorithm can decide to either do some random motor babbling — pick a random point in the hyperrectangle M —, or do some random goal babbling, i.e. pick a random point in the bounded sensory space S as a goal for the inverse model.

The exploration algorithm of the source task proceeds in two phases. The first phase is of pure, random, motor babbling, and last an arbitrary number of K_{boot} samples. The second phase features mixed motor and goal babbling; the algorithm chooses, for each sample, to do random goal babbling with probability p_{goal}, with $0 < p_{\text{goal}} < 1$, and random motor babbling otherwise (see Algorithm 1). The reason for the existence of a pure motor babbling phase is that goal babbling relies on the inverse model, which needs data to derive useful motor commands. In fact, one of the consequence of goal babbling is that the motor command chosen depends on the training data, whereas with random motor babbling it does not. Goal babbling eventually leads to creating heterogeneous exploration of the motor space, where some regions are well sampled and other completely unexplored, since the uniformity of the exploration is enforced on the sensory space.

B. Transfer Algorithm

The main idea behind our transfer algorithm is to identify the motor commands that help the robot to learn the model of the environment faster. The hope is that a portion of these commands are beneficial to the current learning task not only because they are adapted to it, but also because they fit the intrinsic morphological and dynamics properties of the robot. As such, in other situations, they might prove to have better learning value than random commands, if those underlying intrinsic properties are present and shared across situations.

Another way to put it is to say that faced with an unknown situation, the robot will try motor commands that proved valuable in the past, rather than random ones.

1) Formalization: Given a trajectory exploration ξ, we divide the sensory space into regions. In this article, we create regions in S using a simple grid. For each region, we consider the $\mathbf{y}_i$ belonging to that region that were the result of an episode of goal babbling. Those $\mathbf{y}_i$ have an associated goal $\mathbf{g}_i$ that was chosen during learning. We call the negative of the euclidean distance between $\mathbf{y}_i$ and $\mathbf{g}_i$ the *competence* of $\mathbf{y}_i$. For each region, we compute the history of the competence of the $\mathbf{y}_i$ belonging to the region. In region were learning happened, the competence typically exhibits an increasing trend, faster when learning was easier.

We define the *interest* of a region by the derivative of the competence over time. Due to the presence of outliers in the data, we compute the interest using a *Siegel estimator* [6], a robust linear regression method, rather than using a least-square regression method. Mathematically, given a region R, we define :

$$\text{INTEREST}(R) = \text{SEIGEL}(\{||\mathbf{g}_i - \mathbf{y}_i||\}_{i \in R})$$

For each observation $(\mathbf{x}_i, \mathbf{y}_i)$ of D, we associate a weight w_i to the command $\mathbf{x}_i$ corresponding to the interest of the region $\mathbf{y}_i$ is in (note that those weights are not related to the weights of the LWLR regression routines). We thus define the set $\{(\mathbf{x}_i, w_i)_{0 \leq i \leq n}\}$. This is the information that will be transferred between tasks.

Algorithm 2 summarizes the computational steps.

Algorithm 2: TRANSFER(D, G)

Data:
- $\xi = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{0 \leq i \leq n}$, exploration trajectory.
- G, set of goals used during learning.

Result: $W = \{(\mathbf{x}_i, w_i)\}_{0 \leq i \leq n}$, weighted commands.

```
Divide S into a set of regions R
for R ∈ R do
    history = []
    for (x_i, y_i) ∈ ξ, with y_i ∈ R do
        if y_i has an associated goal g_i ∈ G then
            append ||y_i − g_i|| to history
    INTEREST(R) = SEIGEL(history)
W = []
for (x_i, y_i) ∈ D do
    R = region where y_i belongs
    append (x_i, INTEREST(R)) to W
```

C. Exploration in Target Tasks

The transfer algorithm provides motor commands and how high the learning rate was in the region of the effects they produced. We use this information to modify our motor babbling behavior : with a probability p_{transfer}, instead of random motor babbling, the algorithm will draw without replacement from the transferred set of commands, with a probability proportional to the commands' weights. We describe the modified algorithm in Algorithm 3.

IV. EXPERIMENTS

We conducted two experiments, the first to measure the increase in learning performance using the transfer learning on an interaction task, and the second on a situation, the learning of inverse kinematic, where the transfer didn't affect learning significantly.

A. Simulation Settings

All our experiments were done in simulation. We considered a idealized interaction task; a 6-DOF robot arm is inside a room, with an object. The position of the object or of the walls of the room is not known by the learner. The arm is controlled in velocity, and each joint is equipped with a PID

Algorithm 3: TRANSFEREXPLORATION(B, W, K_{boot}, p_{goal}, p_{transfer})

Data:
- $B = (S, M, f, n)$, target task.
- $W = \{(\mathbf{x}_i, w_i)\}_{0 \leq i \leq n}$, weighted commands.
- K_{boot}, duration of pure motor bootstrapping.
- p_{goal}, ratio of goal babbling.
- p_{transfer}, ratio of transfer motor babbling.

Result: $\xi_B = \{\mathbf{x}_i, \mathbf{y}_i\}_{0 \leq i \leq n} \in (S \times M)^n$, exploration trajectory.

$\xi_B \leftarrow []$
for t **from** 0 **to** n **do**
  if **then**
    if RANDOM() $\geq p_{transfer}$ **and** W *is not empty* **then**
      draw $\mathbf{x}_t$ from W without replacement, proportionally to its weight w_t.
    else
      choose $\mathbf{x}_t$ randomly in M
    execute the command; $\mathbf{y}_t \leftarrow f(\mathbf{x}_t)$
    append $(\mathbf{x}_t, \mathbf{y}_t)$ to ξ_B
  else
    GOALBABBLING(B, ξ_B) (see Algorithm 1)

controller on the angular position, with a frequency of 60Hz. The robot prepares an command by selecting a starting pose (6 angles), a target pose (6 angles) and a global maximum rotational velocity for the joints. Each angle can have values between 2.0 and -2.0 radians, and the velocity limit cannot exceed 2.0 rad/s. The arm operate on the horizontal plane (see figure 2); no gravity force applies. Every body in the simulation is simulated using the physic engine jBox2D[2], and possesses properties such as mass, friction, restitution (how much kinetic energy is conserved by the body when colliding with another), linear and angular dampening, the latter two being strictly positive and used to simulate friction with the floor. The 13 parameters of a motor command are thus parameters of a dynamical system whose execution is constrained by collisions (toy, walls, and the arm with itself) and inertia (the joints do not have infinite torque).

An command is executed in the following manner (see figure 2):

1) the robot is placed in the starting pose defined in the command. Note that if the starting pose implies that some links of the robot be placed inside walls, they are effectively placed that way, and the affected joints are immobilized, greatly reducing the movement possibility of the robot over the episode.
2) the object is positioned in the environment, always at the same place.
3) the robot reaches for the target position for 12 seconds.

The environment response is the final spatial position of the object after 12 seconds (the object might still be in motion at that time) and if a collision between the object and the arm occurred during execution (0.0 or 1.0); as such, S is a subset of $\mathbb{R}^3$. This constitute one episode, i.e., one sampling of the environment.

[2]code.google.com/p/jbox2d/

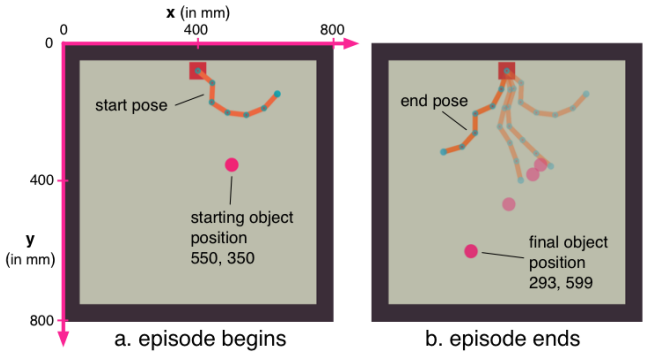


Fig. 2. An example of the execution of a motor command. The arm swings from the init position (1.17, -1.26, 1.33, 0.19, 0.82, 0.75) (in radians) to the target position (-0.31, -0.20, -0.68, 1.16, -0.77, -0.54) at a maximum velocity of 1.0 rad/s, hitting the ball in the process. The motor command is the concatenation of the init pose, the target pose and the velocity, and the sensory response is $(x_e, y_e, 1.0) = (293.2, 599.7, 1.0)$ (in mm).

In each environment, we generate a grid of test cases over the entire space minus the walls.

B. Configuration

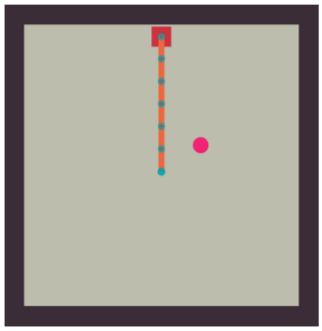
We consider one source task (figure 3.a) and 7 variations (figures 3.b-h).

The room in our simulation is a square of 800 mm by 800 mm, with walls of 50 mm width. The origin is in the upper left corner, with axis oriented as in figure 2. The effective attainable space for the position of the center of a body is thus $[50, 750] \times [50, 750]$, and less as the size of the body increases. The arm is composed of 6 links, each of length 50 mm. The mass of each links is equal to its area (density $= 1$), and the same is true for the ball. In figure 3, the arm is portrayed in position where all joints are in position zero. The ball is placed at the position (500, 350).

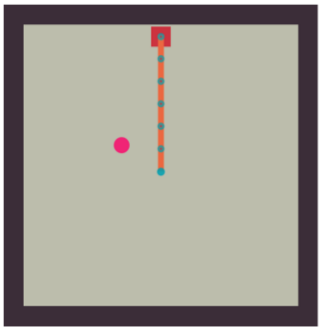
C. First Experiment

In this experiment, we set K_{boot} to 1000, p_{goal} to 0.7 and p_{transfer} to 0.8, which means that after a period of 200 steps of motor babbling, the exploration strategy does 30% of the time motor babbling, and 70% of the time goal babbling. In the case of a target task, when motor babbling is selected (either during in 1000 first steps or after), 80% of the time, transferred motor commands will be chosen, and 20% of the time a random motor command will be generated. Over 10000 steps, approximately 30% of the motor commands come from transfer.

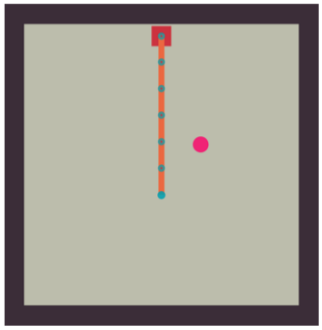
In this experiment, we consider the tasks described previously, and impose a limit of 10000 trials on training. For each configuration, we consider the case of learning without transfer, and with transfer from the source task.



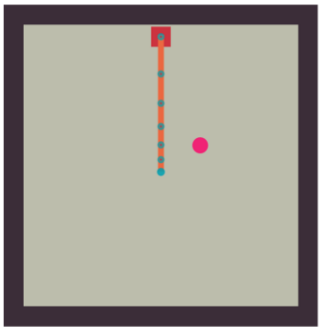
a. Source task, the ball is placed at position 500, 350.



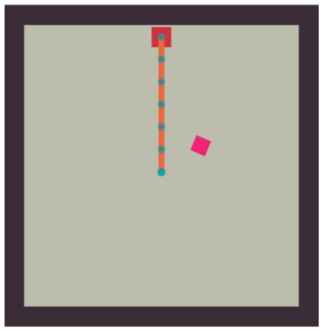
b. "Symmetric" task, the ball is placed at position 300, 350.



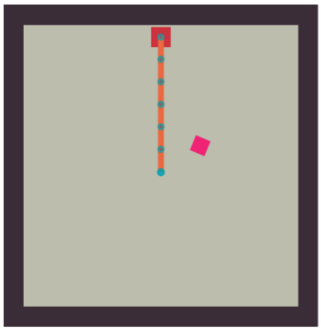
c. "Bigger" task, the arm has links of 60mm, rather than 50mm, for a total length of 360mm.



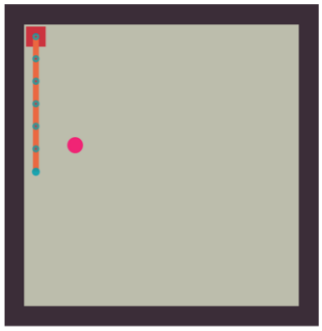
d. "Distal" task, where each link is shorter than the previous one by a factor of 1.3, for the same total length as the source task.



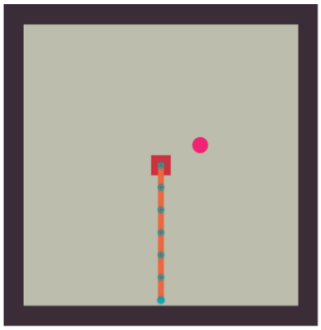
e. "Heavy Cube" task, the cube has a density of 2, and a restitution of 0.35. The interaction is more complex and requires more energy.



f. "Light Cube" task, the cube has a density of 0.5, and a restitution of 1.0. The interaction is more chaotic, as the cube bounces on more walls.



g. "Corner" task, the base and ball is moved to a corner. Many orders will have starting positions in the wall, and the arm will be stuck there.



h. "Center" task, only the base is moved to the center. Orders that previously collided with the wall will have to be used for interacting with the ball.

Fig. 3. Different tasks considered for the first experiment.

D. Second Experiment

In the second experiment, we use the same source setup, and configuration (3.d). The sensory response from the environment for both is not the final position of the toy (although it is still present), but the final position of the tip of the arm, as in a classical inverse kinematic setup.

All and every aspects of the experiments, from the motor representation, to the set of tasks, to the parameters K_{boot}, p_{goal}, and p_{transfer} were fixed before the results were produced.

V. RESULTS

Each experiement was repeated 20 times. We computed and plotted the average over the repetitions of the average error, with and without transfer from the source task. The average error was computed using a set of 400 test cases, using a grid layout covering the area inside the walls. We also computed τ_{2000} and τ_{10000} from (1) to measure the early and overall performance increase due to the transfer.

A. First Experiment

In figure 4, we see that overall, the transfer is significantly beneficial for early performance, although it does not impact long-term performance. Figure 4.a depicts the performance reduction of the source task with itself, which is used as a control case rather than the illustration of a useful one. The error reduction, which is more than half, is concentrated before the 2000th time step, after which, the non-transfer learning performance catches up. This pattern is repeated to varying degrees across all variations, except in the ”corner” task.

The corner task, figure 4.e, sees no effect from the transfer during the first 1500 timesteps, and then is affected negatively by it, staying constantly at a level of error of rougly 12mm more than the task without transfer. In the corner task, many of the motor command that would have connected with the ball have intial positions that start the arm in the wall, where it stays stuck. As such, many of the transfered command have low learning value in regards to the tests that measure performance. Still, more work is needed to understand this result.

Overall, the results are encouraging, because they show that reusing the right commands from a previous task, in situations where the morphology of the robot or the physical property of the objects it is interaction with has changed, a robot can autonomously boostrap its learning, and reach precision levels far quicker than without transfer.

B. Second Experiment

The results in figure 5 paint a different picture than the results of the first experiment. The transfer has no discernable effect on the performance, even when it comes from the same task. One possible explanation is that, when the sensory output is the position of the tip of the arm, a far greater ratio of the possible motor commands provides beneficial learning information, whereas when the object is tracked, most motor commands miss the ball and thus provide little

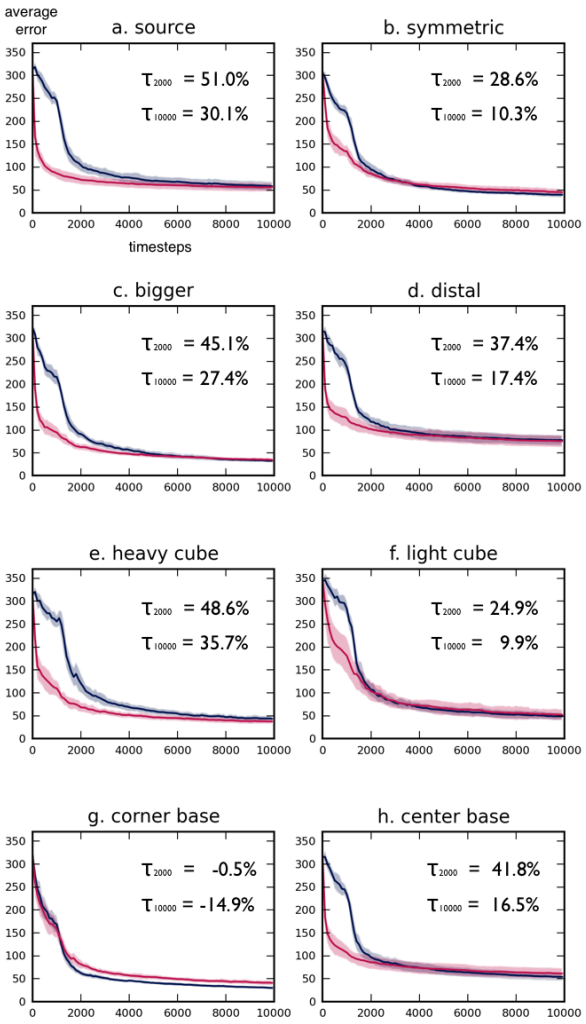


Fig. 4. Results of the first experiment. For each graph, in abscissae the timesteps, and in ordinate the repetitions' average of the average error to the to the testcase set (in mm). The blue curve is for the case without transfer, while the pink is for transfer from the source task. The shaded area around each curve is the standart deviation over the repetitions.

learning information. That makes efficient motor babbling less important. This experiments helps starting to characterize in which situations the transfer algorithm is useful, and in which it is not.

VI. DISCUSSION

A. Limitations

As it stands, our paper presents several limitations :

- While the results are conclusive, they do not show if they could work just as well with a simpler criteria than intrinsic motivation. Some aspects of the results raise more questions than they provide answers; specific analysis are needed to better understand them.
- The motor commands are grouped and discriminated in the sensory space. Yet, commands whose effects belong to the same sensory region might have very different learning benefits. Our method does not distinguish between those.
- The results are demonstrated on noiseless, non-stochastic simulations in 2D worlds.
- The effectiveness of the transfer could be heavily tied to the motor command representation used in this article. While the representation was fixed before we obtained the first results (therefore, no "optimization" of the motor representation occurred to produce better experimental results), we can't rule it out.
- All the same, the effectiveness of the transfer could be heavily tied to the specifics of the learning algorithm used in this article (optimization of a LWLR predictor using L-BFGS-B). More experiments are needed to access the robustness of our methods to different learning algorithms.
- In the experiments we arbitrarily fixed the parameters K_{boot}, p_{goal}, and p_{transfer} (before the results were produced). An empirical analysis of the influence of those parameters is needed to better understand the dynamic of the transfer mechanism.

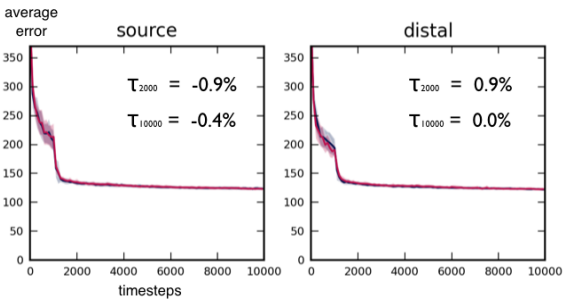


Fig. 5. Results of the second experiment. The axis and curves characteristics and colors are the same as figure 4.

B. Perspectives

Each of the limitations highlighted previously calls for further work, as we view none of them as irremediable. Specifically, we plan to work on :

- Experiments with real robots, in the context of affordance learning in particular.
- Search for values of the parameters or new methods that minimize the amount or frequency of transfer while still retaining good performance. In particular, as motor babbling biggest effect is in the beginning of learning, a mechanism that decreases the amount of transfer as the learning progresses could exhibit comparable performances while increasing the time at which the no-transfer performance catches up, and possibly mitigating the negative transfer issues exposed in the experiments.
- New criteria to further discriminate the learning benefits of different motor commands, even when their effect belong to the same sensory regions.

ACKNOWLEDGEMENTS

I am particularly grateful to Paul Fudal for his technical assistance in the setup of the simulations.

This work was partially financed by the ANR MACSi and the ERC Starting Grant EXPLORERS 240 007. Computing hours for running simulations were graciously provided by the MCIA Avakas cluster.

REFERENCES

LEARNING ALGORITHMS AND METHODS

[1] W.S. Cleveland, S.J. Devlin, "Locally-Weighted Regression: An Approach to Regression Analysis by Local Fitting". *Journal of the American Statistical Association* 83(403) (1988) 596610.

[2] C. G. Atkeson, A. W. Moore, S. Schaal, "Locally Weighted Learning", *Artificial Intelligence Review*, 11(1) (1997) 11-73, 10.1023/A:1006559212014

[3] R. Penrose. "A generalized inverse for matrices", *Proceeding of Cambridge Philosophical Society* 51 (1955) 406-413.

[4] R. H. Byrd, P. Lu and J. Nocedal, "A Limited Memory Algorithm for Bound Constrained Optimization", *SIAM Journal on Scien. and Stat. Computing* 16(5) (1995) 1190-1208.

[5] C. Zhu, R. H. Byrd and J. Nocedal, "L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization", *ACM Transactions on Mathematical Software*, 23(4) (1997) 550-560.

[6] A. F. Siegel, "Robust regression using repeated medians", *Biometrika* 69(1) (1982) 242244.

EXPLORATION ALGORITHMS

[7] A. Baranes, P-Y. Oudeyer, "Active Learning of Inverse Models with Intrinsically Motivated Goal Exploration in Robots", *Robotics and Autonomous Systems*, (2012).

[8] M. Rolf, "Goal Babbling for an Efficient Bootstrapping of Inverse Models in High Dimensions", PhD Thesis *Bielefeld University* (2012)

[9] M. Lopes, T. Lang, M. Toussaint, P-Y. Oudeyer, "Exploration in model-based reinforcement learning by empirically estimating learning progress.", *Neural Information Processing System (NIPS)* , (2012).

[10] M. Lopes, P-Y. Oudeyer, "The strategic student approach for life-long exploration and learning." *Development and Learning and Epigenetic Robotics (ICDL)*, (2012).

[11] J. Schmidhuber. "Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990-2010)". *IEEE Transactions on Autonomous Mental Development*, 2(3) (2010) 230-247

TRANSFER ALGORITHMS

[12] F. Fernández, M. Veloso. "Probabilistic policy reuse in a reinforcement learning agent." In *Proceeding of the fifth conference on Autonomous Agents and Multiagent Systems*, ACM, (2006) 720-727.

[13] M. E. Taylor, N. K. Jong, P. Stone, "Transferring instances for model-based reinforcement learning." *Machine Learning and Knowledge Discovery in Databases.* Springer Berlin Heidelberg, (2008) 488-505.

[14] M. E. Taylor, P. Stone, "Transfer learning for reinforcement learning domains: A survey." *The Journal of Machine Learning Research*, 10, (2009) 1633-1685

[15] L. Torrey, J. Shavlik, "Transfer Learning", *Handbook of Research on Machine Learning Applications* (2009).

[16] F. Doshi-Velez, G. D. Konidaris, "Transfer Learning by Discovering Latent Task Parametrizations." In the *NIPS 2012 Workshop on Bayesian Nonparametric Models for Reliable Planning And Decision-Making Under Uncertainty*, (2012).

[17] M. G. Madden, T. Howley, "Transfer of experience between reinforcement learning environments with progressive difficulty." *Artificial Intelligence Review* 21.3-4 (2004) 375-398.

[18] S. Barrett, M. Taylor, P. Stone. "Transfer learning for reinforcement learning on a physical robot." In *The Ninth International Conference on Autonomous Agents and Multiagent Systems - Adaptive Learning Agents Workshop*, number May, 2010.

[19] S. Thrun and T. Mitchell. "Lifelong Robot Learning". *Robotics and autonomous systems*, (March 1993), 1995.

OTHER

[20] D. L. Silver, Y. Qiang, L. Lianghao, "Lifelong Machine Learning Systems: Beyond Learning Algorithms." *2013 AAAI Spring Symposium Series.*, 2013.

[21] J. Konczak, "On the notion of motor primitives in humans and robots", *Lund University Cognitive Studies* (2005)