Emergent Proximo-Distal Maturation through Adaptive Exploration

Freek Stulp, Pierre-Yves Oudeyer
Cognitive Robotics, ENSTA-ParisTech, Paris, France
FLOWERS Team, INRIA Bordeaux Sud-Ouest, Talence, France

***Abstract*—Life-long robot learning in the high-dimensional real world requires guided and structured exploration mechanisms. In this developmental context, we investigate here the use of the recently proposed $\text{PI}^2_{\text{CMAES}}$ episodic reinforcement learning algorithm, which is able to learn high-dimensional motor tasks through adaptive control of exploration. By studying $\text{PI}^2_{\text{CMAES}}$ in a reaching task on a simulated arm, we observe two developmental properties. First, we show how $\text{PI}^2_{\text{CMAES}}$ autonomously and continuously tunes the global exploration/exploitation trade-off, allowing it to re-adapt to changing tasks. Second, we show how $\text{PI}^2_{\text{CMAES}}$ spontaneously self-organizes a maturational structure whilst exploring the degrees-of-freedom (DOFs) of the motor space. In particular, it automatically demonstrates the so-called *proximo-distal maturation* observed in humans: after first freezing distal DOFs while exploring predominantly the most proximal DOF, it progressively frees exploration in DOFs along the proximo-distal body axis. These emergent properties suggest the use of $\text{PI}^2_{\text{CMAES}}$ as a general tool for studying reinforcement learning of skills in life-long developmental learning contexts.**

I. Introduction

Building robots capable of life-long learning of skills poses many challenges. For instance, the limited number of physical experiments a robot can make within its life-time severely limits the regions in which it is able to explore its high-dimensional sensorimotor space. Also, a robot should be able to adapt its skills to changing environments or to novel tasks. To achieve efficient life-long learning in such complex spaces, humans benefit from various interacting developmental mechanisms, which generally structure exploration from simple learning situations to more complex ones [4], [10], [22], [7]. Recent research in developmental robotics has proposed several ways to transpose these developmental learning mechanisms to robots. This includes mechanisms of intrinsically motivated active learning, which automatically select training examples or tasks of increasing complexity [16], [2], social learning where a teacher can scaffold a training program of increasing difficulty [11] and also curriculum learning [3], as well as maturation and body growth, where the degrees-of-freedom in the sensori- and motor spaces evolve through phases of freezing and freeing [6], [8], [1].

Mechanisms for structured exploration and data collection strongly interact with the learning algorithms that process data, jointly aiming to optimize the parameters of an action policy to achieve a given task. Due to its generality, temporal abstraction, and ability to learn without models, reinforcement learning (RL) has been proposed as an appealing paradigm for organizing learning and behavior in developmental robotics [18]. Using RL in the context of robotics, and developmental robotics in particular, introduces several challenges, including high-dimensional continuous action spaces, adaptation to changing tasks and environments, and continual reconsideration of the exploration/exploitation trade-off.

Recently, we have proposed to address these challenges by combining state-of-the-art evolutionary optimization strategies with direct reinforcement learning [20], [19]. In particular, we use the "Policy Improvement with Path Integrals" algorithm (PI^2) [21] to enable robust, efficient learning in high-dimensional action spaces, with features from the "Covariance Matrix Adaptation - Evolutionary Strategy" [12] to continuously determine the best exploration/exploitation trade-off over time. Even though CMA-ES and PI^2 are derived from very different principles, combining these algorithms is feasible, as they both iteratively update parameters with *probability-weighted averaging* to reduce future costs.

The research focus of this paper is on using the $\text{PI}^2_{\text{CMAES}}$ algorithm for continual adaptive exploration, and in particular for re-adaptation to changing tasks, as well as self-organization of **structured maturational exploration**. After summarizing the $\text{PI}^2_{\text{CMAES}}$ algorithm (Section II), we therefore focus on using $\text{PI}^2_{\text{CMAES}}$ in the context of developmental robotics by investigating two complementary phenomena through an experiment, described in Section III, where a robot arm with parameterized motor synergies learns to reach a point while minimizing energy.

First, Section IV presents $\text{PI}^2_{\text{CMAES}}$'s ability to automatically adapt to changing tasks by determining the appropriate exploration magnitude *autonomously* – exploration decreases once a task has been learned (exploitation), but increases again automatically if the task or environment changes such that the task must be re-learned. This exploration behavior is not explicitly encoded in the algorithm, but is rather an emergent feature of updating the covariance matrix, which governs exploration, with probability-weighted averaging.

Second, Section V shows how $\text{PI}^2_{\text{CMAES}}$ spontaneously self-organizes a maturational structure while exploring the degrees-of-freedom of the motor space. The algorithm automatically generates the so-called proximo-distal maturation observed

This work is partly supported by the French ANR program MACSi (ANR 2010 BLAN 0216 01).

when infants learn to reach [5], [13]: after a short phase of uniform body babbling, it begins by freezing distal DOFs/joints while predominantly exploring with the most proximal joints, and then progressively frees exploration in joints along an ordered structure following the proximo-distal axis.

II. $\text{PI}^2_{\text{CMAES}}$ Algorithm

The $\text{PI}^2_{\text{CMAES}}$, or "Policy Improvement with Path Integrals and Covariance Matrix Adaptation – Evolutionary Strategy", was recently proposed in [20]. It is a combination of the PI^2 direct reinforcement learning algorithm [21] with the evolutionary optimization strategy CMA-ES. For a discussion of the similarities and differences between these algorithms, and their relation to $\text{PI}^2_{\text{CMAES}}$, we refer to [20]; in this section briefly describe the resulting algorithms.

The goal of $\text{PI}^2_{\text{CMAES}}$ is to optimize a set of policy parameters $\boldsymbol{\theta}$ with respect to a cost function $R(\boldsymbol{\tau}) = \phi_{t_N} + \int_{t_i}^{t_N} r_t$, where $\boldsymbol{\tau}$ is a trajectory that starts at time t_0 in state $\mathbf{x}_{t_0}$ and ends at t_N. ϕ_{t_N} is the terminal reward at t_N, and r_t is the immediate reward at time t. For example, ϕ_{t_N} may penalize the distance to a goal at the *end* of a movement, and r_t may penalize the acceleration at each time step *during* a movement. To optimize $R(\boldsymbol{\tau})$, $\text{PI}^2_{\text{CMAES}}$ uses an iterative approach of exploring and updating in policy parameter space, as listed in Algorithm 1, which we now describe in more detail.

A. Exploration

$\text{PI}^2_{\text{CMAES}}$ first takes K samples $\boldsymbol{\theta}_{k=1\ldots K}$ from a Gaussian distribution (line 9). The vector $\boldsymbol{\theta}$ represents the parameters of a policy, which for instance controls the sequence of desired joint angle of an arm, or desired x-coordinate of an end-effector. Executing the policy with parameters $\boldsymbol{\theta}_k$ yields a trajectory $\boldsymbol{\tau}$ with N time steps (line 10). An entire trajectory is referred to as $\boldsymbol{\tau}$, whereas $\boldsymbol{\tau}_i$ refers to the subtrajectory of $\boldsymbol{\tau}$, starting at t_i, and ending at t_N. In this nomenclature, $\boldsymbol{\tau}$ is therefore just a convenient abbreviation for $\boldsymbol{\tau}_0$. From now on, indices i and j refer to time steps, and k and l refer to trials. A *trial* or *roll-out* is the full trajectory resulting from executing the policy with parameters $\boldsymbol{\theta}_k$. The K executions within one iteration of $\text{PI}^2_{\text{CMAES}}$ are called an *epoch*.

B. Parameter Update for Each Time Step: Probability-weighted Averaging

After exploration, a new parameter vector $\boldsymbol{\theta}^{\text{new}}$ is computed, which is expected to lead to a lower trajectory cost than the current $\boldsymbol{\theta}$. This parameter update is done in two phases: 1) compute different parameters $\boldsymbol{\theta}_i^{\text{new}}$ for each of the N time steps i, using probability-weighted averaging; 2) compile the N updates $\boldsymbol{\theta}_i^{\text{new}}$ into one vector $\boldsymbol{\theta}^{\text{new}}$, using temporal averaging.

Although exploration is performed in the space of $\boldsymbol{\theta}$, costs are incurred by actually executing the policy, and are thus defined in terms of the trajectory $\boldsymbol{\tau}$ that results from this execution. $\text{PI}^2_{\text{CMAES}}$ optimizes the parameters $\boldsymbol{\theta}$ not only for the entire trajectory, but also for all subtrajectories $\boldsymbol{\tau}_{i=1\ldots N}$ of $\boldsymbol{\tau}$. The cost of a subtrajectory $\boldsymbol{\tau}_i$ is computed as the sum over the costs throughout the rest of the trajectory starting at time step i: $S(\boldsymbol{\tau}_i) = \phi t_N + \sum_i^N r_{t_i}$. This is known as the *cost-to-go*, because it represents the accumulated 'cost to go' from i to the end of the trajectory. The underlying principle (the Bellman principle) is that a subtrajectory $\boldsymbol{\tau}_i$ starting at t_i can only be optimal if the subtrajectory $\boldsymbol{\tau}_{i+1}$ is optimal too.

The probability of each trajectory in the epoch $P(\boldsymbol{\tau}_{i,k})$ is then computed by exponentiating the cost-to-go $S(\boldsymbol{\tau}_{i,k})$ of that trajectory at each time step (line 18). For illustration purposes, this transformation from cost to probability is depicted in Fig. 1. Here, we see the $K = 10$ samples in a two-dimensional $\boldsymbol{\theta}$ space. The mapping from cost to probability is visualized in the lower-left graph. High-cost samples are assigned a low probability, and low-cost samples a high probability. This mapping follows directly from the PI^2 derivation, and may be interpreted as preferring trajectories with lower cost to occur with a higher probability. The parameter h determines the exact shape of the mapping from cost to probability, where higher values of h lead to a more greedy search, that places more emphasis on low-cost samples.

The two core steps in $\text{PI}^2_{\text{CMAES}}$ (line 21 and line 22) are then to update the mean and covariance matrix of the sampling distribution by using *probability-weighted averaging*: $\boldsymbol{\theta}^{\text{new}} = \sum P_k \boldsymbol{\theta}_k$. Since low-cost samples have a higher probability, this means they will contribute more to the update. The resulting update is visualized in Fig. 1. As we see, the distribution mean $\boldsymbol{\theta}$ is now closer to the minimum, and the covariance matrix is also 'pointing' more towards to the minimum. Using probability-weighted averaging avoids having to estimate a gradient, which can be difficult for noisy and discontinuous cost functions.

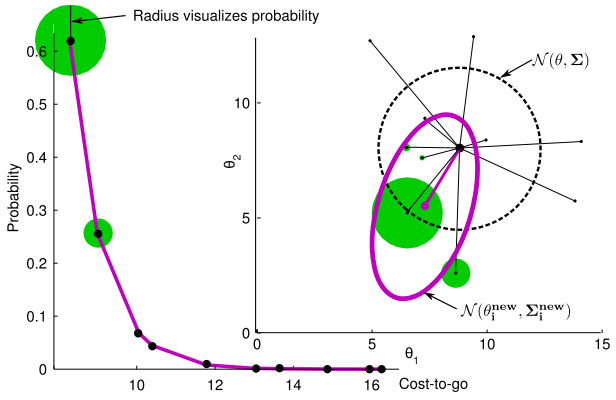

Fig. 1. Visualization of a parameter update at one time step i. The upper right graph shows the 2D parameter space, with the current Gaussian distribution (dashed black), and $K = 10$ random samples taken from it. The lower left graph shows the mapping from cost to probability. The green circles represent the probability for each sample. The new distribution $\boldsymbol{\theta}_i^{\text{new}}, \boldsymbol{\Sigma}^{\text{new}}{}_i$ for time step i (purple ellipse) is acquired through probability-weighted averaging.

C. Parameter Update: Temporal Averaging

In line 21, a different parameter update $\boldsymbol{\theta}_i^{\text{new}}$ is computed for each time step i. If the trajectory has 500 time steps, we therefore perform probability-weighted averaging 500 times.

To acquire a single new parameter vector $\boldsymbol{\theta}^{\text{new}}$, the final step is therefore to average over all time steps (line 27)[1]. This average is weighted such that earlier parameter updates in the trajectory contribute more than later updates, i.e. the weight at time step i is $T_i = (N-1)/\sum_{j=1}^{N}(N-1)$. The intuition is that earlier updates affect a larger time horizon and have more influence on the trajectory cost [21].

input :
- $\boldsymbol{\theta}$; *initial parameter vector*
- $J(\boldsymbol{\tau})$; *cost function*
- $\lambda_{init}, \lambda_{min}, \lambda_{max}$, ; *exploration level (initial,min,max)*
- K ; *number of roll-outs per update*
- h ; *eliteness parameter*

1 $\Sigma = \lambda_{init}\mathbf{I}$
2 **while** *true* **do**
3 *Exploration: sample parameters and execute policies*
4 **foreach** k **in** K **do**
5 $\boldsymbol{\theta}_k \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$
6 $\boldsymbol{\tau}_k = executepolicy(\boldsymbol{\theta}_k)$
7 **end**
8 *Compute parameter update for each time step*
9 **foreach** i **in** N **do**
10 *Evaluation: compute probability for each time step and trial*
11 **foreach** k **in** K **do**
12 $S(\boldsymbol{\tau}_{i,k}) = \sum_{j=i}^{N} J(\boldsymbol{\tau}_{j,k})$
13 $E(\boldsymbol{\tau}_{i,k}) = e^{\left(\frac{-h(S(\boldsymbol{\tau}_{i,k}) - \min(S(\boldsymbol{\tau}_{i,k})))}{\max(S(\boldsymbol{\tau}_{i,k})) - \min(S(\boldsymbol{\tau}_{i,k}))}\right)}$
14 $P\left(\boldsymbol{\tau}_{i,k}\right) = \frac{E(\boldsymbol{\tau}_{i,k})}{\sum_{l=1}^{K} E(\boldsymbol{\tau}_{i,l})}$
15 **end**
16 *Update: Probability-weighted averaging over K trials*
17 $\boldsymbol{\theta}_i^{new} = \sum_{k=1}^{K}\left[P\left(\boldsymbol{\tau}_{i,k}\right)\ \mathbf{M}_{i,k}(\boldsymbol{\theta}_k - \boldsymbol{\theta})\right]$
18 $\Sigma_i^{new} = \sum_{k=1}^{K}\left[P\left(\boldsymbol{\tau}_{i,k}\right)\ (\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^{\intercal}\right]$
19 $\Sigma_i^{new} = \text{evolutionpaths}(\Sigma_i^{new})$
20 $\Sigma_i^{new} = \text{boundcovar}(\Sigma_i^{new})$
21 **end**
22 *Update: Temporal averaging over N time steps*
23 $\boldsymbol{\theta}^{new} = \frac{\sum_{i=0}^{N}(N-i)\boldsymbol{\theta}_i^{new}}{\sum_{l=0}^{N}(N-i)}$
24 $\Sigma^{new} = \frac{\sum_{i=0}^{N}(N-i)\Sigma_i^{new}}{\sum_{l=0}^{N}(N-i)}$
25 **end**

Algorithm 1: The $\text{PI}^2_{\text{CMAES}}$ algorithm.

D. Covariance Matrix Updating: Evolution Paths

Line 23 uses the function 'evolutionpaths' to further update the covariance matrix. This function is very specific to the CMA-ES algorithm, and uses so-called evolution paths to make a more robust update of the covariance matrix $\boldsymbol{\Sigma}_i$. Since this is not part of the core algorithm, we refer to equations (14)-(17) in Hansen et al. [12] or equations (20)-(23) in Stulp et al. [20] for the implementation of evolution paths.

[1]Temporal averaging over covariance matrices is possible, because 1) every positive-semidefinite (PSD) matrix is a covariance matrix and vice versa 2) a weighted averaging over PSD matrices yields a PSD matrix [9].

E. Covariance Matrix Updating: Lower Bounds

In $\text{PI}^2_{\text{CMAES}}$, the initial $\boldsymbol{\Sigma}$ is set to $\boldsymbol{\Sigma} = \lambda^{\text{init}}\mathbf{I}_B$, where B is the dimensionality of the policy parameter vector $\boldsymbol{\theta}$, which corresponds to the number of basis functions (cf. Section III). In $\text{PI}^2_{\text{CMAES}}$, $\boldsymbol{\Sigma}$ is then subsequently adapted over time. A common problem with covariance matrix adaptation is premature convergence, and *"[i]n the practical application, a minimal variance [...] should be ensured"* [12]. This problem is illustrated in Fig. 2, where the solid 'slim' $\boldsymbol{\Sigma}$ is the result of probability weighted averaging.

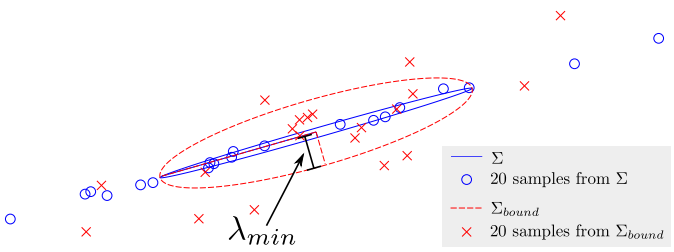

Fig. 2. Enforcing a lower bound on the eigenvalues of $\boldsymbol{\Sigma}$, to avoid exploration degeneracy along one of the axes (line 24 in Alg. 1).

To avoid such degeneracy of $\boldsymbol{\Sigma}$, we compute its eigenvalues, place a lower bound of λ^{min} on the eigenvalues, and reconstruct the bounded covariance matrix from the eigenvectors and the bounded eigenvalues. The bounded $\boldsymbol{\Sigma}$ is also depicted in Fig. 2. This procedure is implemented in the 'boundcovar' function in line 24 in Alg. 1. In our experience, covariance matrix bounding is essential, as the algorithm always prematurely converges without it.

For robotics applications, we also recommend putting an upper bound λ^{max} on the eigenvalues of $\boldsymbol{\Sigma}$, as too much exploration might lead to dangerous behavior on the robot, e.g. reaching joint limits, too high accelerations.

From now on, we will refer to the exploration magnitude of a 1-D policy as the largest eigenvalue λ of the covariance matrix $\boldsymbol{\Sigma}$. Note that the initial covariance matrix $\boldsymbol{\Sigma} = \lambda^{\text{init}}\mathbf{I}_B$ has a 'largest' eigenvalue (they are all the same) of λ^{init}. We use Λ to denote the exploration magnitude of an M-D policy; it is the sum over the exploration magnitudes of the individual dimensions: $\Lambda = \sum_{m=1}^{M}\lambda_m$.

F. Multi-dimensional policies

Algorithm 1 is applied to the parameters of a 1-D policy. Optimizing the parameters of an M-dimensional policy, e.g. 7-D for the 7 joints of an arm, or 3-D for the end-effector position, is done by running the algorithm in parallel for each of the dimensions of the policy, with the same costs but different parameter vectors $\boldsymbol{\theta}_{m=1...M}$ and covariance matrices $\boldsymbol{\Sigma}_{m=1...M}$

After having presented the $\text{PI}^2_{\text{CMAES}}$ algorithm, we now demonstrate its relevance to developmental robotics. We first showing how $\text{PI}^2_{\text{CMAES}}$ is able to adapt to changing tasks, which enables life-long learning (Section IV). Then, we show how $\text{PI}^2_{\text{CMAES}}$ automatically freezes and frees joints in an arm,

thereby sequentially freeing joints in a proximo-distal order (Section V). Before doing so, we introduce the evaluation task in the next section.

III. Evaluation Task

The evaluation task in this paper consists of a kinematically simulated arm with $M = 10$ degrees of freedom. The length of each arm segment is 0.6 times the length of the previous segment, and the total length of the arm is 1. The arm should learn to reach for a specific goal $[0.0\ 0.5]$ with minimal joint angles (expressing a 'comfort' factor), and whilst minimizing acceleration at each time step. Initially, all joint angles are 0, as depicted in Fig. 3, and have a null speed.

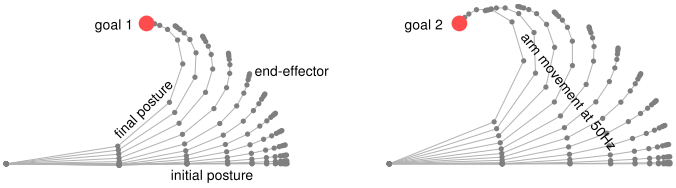

Fig. 3. Visualization of the reaching motion (after learning) for 'goal 1' and 'goal 2'

a) Cost function: The terminal costs of this task are expressed in (1), where $||\mathbf{x}_{t_N} - g||$ represented the distance between the 2-D Cartesian coordinates of the end-effector ($\mathbf{x}_{t_N}$) and the goal $g_1 = [0.0\ 0.5]$ or $g_2 = [0.0\ 0.25]$ at the end of the movement at t_N. The terminal cost also penalizes the joint with the largest angle at $\max(\mathbf{q}_{t_N})$, expressing a comfort effect, with maximum comfort being the initial position. The immediate costs at each time step r_t in (2) penalize joint accelerations. The weighting term $(M + 1 - m)$ penalizes DOFs closer to the origin, the underlying motivation being that wrist movements are less costly than shoulder movements for humans, cf. [21][2].

$$\phi_{t_N} = 10^4 ||\mathbf{x}_{t_N} - g||^2 + \max(\mathbf{q}_{t_N}) \quad \text{Terminal cost} \quad (1)$$

$$r_t = 10^{-5} \frac{\sum_{m=1}^{M} (M + 1 - m)(\ddot{q}_{t,m})^2}{\sum_{m=1}^{M} (M + 1 - m)} \quad \text{Immediate cost} \quad (2)$$

b) Policy representation: The acceleration $\ddot{q}_{m,t}$ of the m^{th} joint at time t is determined as a linear combination of basis functions, where the parameter vector $\boldsymbol{\theta}_m$ represents the weighting of joint m.

$$\ddot{q}_{m,t} = \mathbf{g}_t^\intercal \boldsymbol{\theta}_m \quad \text{Acc. of joint } m \quad (3)$$

$$[\mathbf{g}_t]_b = \frac{\Psi_b(t)}{\sum_{b=1}^{B} \Psi_b(t)} \quad \text{Basis functions} \quad (4)$$

$$\Psi_b(t) = \exp\left(-(t - c_b)^2 / w^2\right) \quad \text{Kernel} \quad (5)$$

[2]This cost term was taken from [21]. In the context of this paper, it cannot be the reason for the proximo-distal maturation we shall see in Section V. Rather than favoring a proximo-distal maturation, this cost term works *against* it, as proximal joints are penalized *more* for the accelerations that arise due to exploration.

The centers $c_{b=1\ldots B}$ of the kernels Ψ are spaced equidistantly in the 0.5s duration of the movement, and all have a width of $w = 0.05s$. Since we do not simulate arm dynamics, the joint velocities and angles are acquired by integrating the accelerations.

c) PI^2_{CMAES} parameterization: For all experiments, we use $\text{PI}^2_{\text{CMAES}}$. Its input parameters are set as follows. The initial parameter vector is $\boldsymbol{\theta} = \mathbf{0}$, which means the arm is completely stretched, and not moving at all over time. The number of trials per update is $K = 10$, and the eliteness parameter is $h = 10$ (the default values suggested by [21]). The initial and minimum exploration magnitude of each joint m is set to $\lambda_m^{\text{init}} = \lambda_m^{\text{min}} = 0.1$, unless stated otherwise.

IV. Re-Adaptation to Changing Tasks

In this first experiment, we evaluate $\text{PI}^2_{\text{CMAES}}$'s capability to adapt to changing tasks, by changing the x-coordinate of the goal for reaching both abruptly and gradually, as illustrated in the top graph of Fig. 4. First the goal is set to 'goal 1' (cf. Fig. 3) and after 150 update to 'goal 2'. Between updates 200 and 250, the x-coordinate of the goal is a sinusoidal, and ends up in 'goal 1' again at update 250.

The middle and bottom graph in Fig. 4 depict the learning curves and total exploration magnitude Λ respectively. The caption of this figure interprets these results,and explains the interaction between changing the task, the automatic adaptation of exploration, and the consequences for learning progress.

Conclusion: $\text{PI}^2_{\text{CMAES}}$ is able to automatically adapt its exploration magnitude to (re)adapt to abruptly or continuously changing tasks. This complements our results on a dynamic task in [19], where a simulated humanoid was to bat a baseball in a specified region, and the position of the baseball was changed abruptly.

V. Emergent Proximo-distal Maturation

In this experiment, our initial aim was to use the exploration magnitude as a measure of competence to drive the release of degrees of freedom over time, thus using competence progress to adaptively maturing the action space such as proposed and experimented in [1]. However, after running some initial experiments we noticed that, without any modification, the $\text{PI}^2_{\text{CMAES}}$ already frees and freezes joints automatically. Therefore maturation appears to be an emergent property of the use of $\text{PI}^2_{\text{CMAES}}$ in such a sensorimotor space, and there was no need to implement a specific scheme to release degrees of freedom. Rather than conducting a novel experiment, we therefore investigate the first 100 updates of the first experiment in Section IV, in particular the exploration magnitudes of the individual joints $\lambda_{m=1\ldots M}$. These are depicted in Fig. 5, and interpreted in its caption.

Fig. 6 plots the movement of the arm during different stages of learning, and visualizes λ_m for each joint as a bar plot. This allows the interpretation of learning in terms of the movement of the arm, and a more direct association between the exploration magnitude of a joint and its position in the arm.

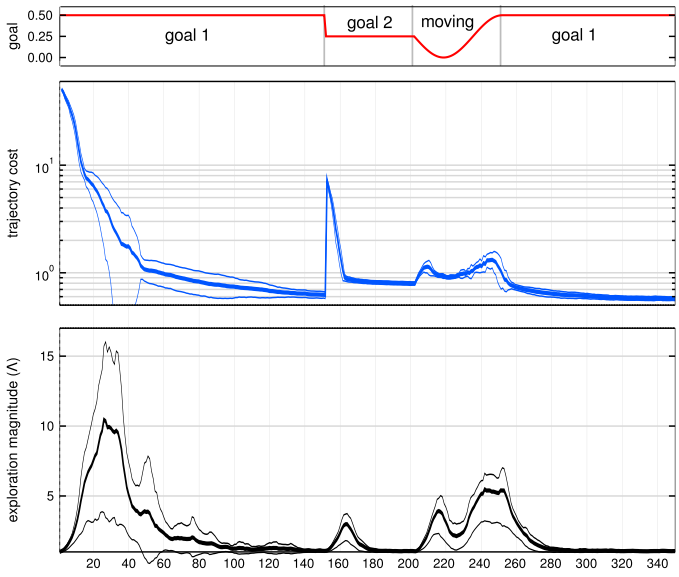

Fig. 4. Top: x-coordinate of the task goal. Center: Learning curve ($\mu \pm \sigma$ over 10 learning sessions). Note the logarithmic y-axis. Bottom: Total exploration magnitudes over all joints Λ ($\mu \pm \sigma$ over 10 learning sessions).
Interpretation: In the first 30 updates, the exploration magnitude goes up, which enables fast learning, and consequently the cost goes down rapidly. Between updates 30-100, the exploration decreases, and after 100 updates it approximately reaches its minimum level of $M \cdot \lambda^{\text{init}} = 10 * 0.1 = 1$. Thus, the task has been learned. When the goal changes abruptly at update 150, exploration goes up again. Note that we do not notify the algorithm that the task has changed; the increasing exploration is an emergent property of using probability-weighted averaging to update the covariance matrix. At update 180, the task has again been learned, and exploration is minimal. Whilst the goal is moving, exploration is constantly on, but when the goal remains still again at update 250, it decreases again.

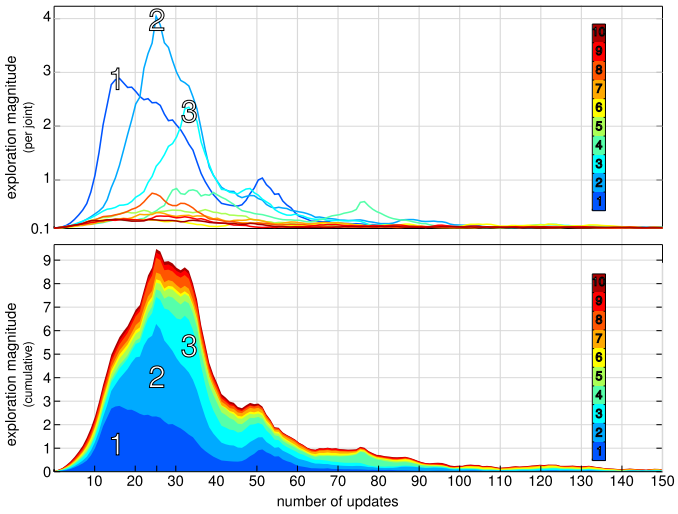

Fig. 5. These plots present a closer look at the exploration magnitudes during the first 100 updates of the experiment depicted in Fig. 4. Top: Exploration magnitudes λ_m for each joint $m = 1 \ldots 10$ separately. These values are averaged over 10 learning sessions, and thus represents consistent, reproducible behavior. Bottom: The total exploration magnitude Λ, split into the individual components λ_m, i.e. the cumulative of the top graph. In this last graph, $\lambda^{\text{min}} = 0.1$ has been subtracted from all λ_m, as we want to emphasize the exploration *above* the baseline, on which $\text{PI}^2_{\text{CMAES}}$ has an influence. The last graph therefore starts at 0.
Interpretation: When inspecting the development of λ_m as learning progresses, we notice the following. The exploration magnitude of the first joint λ_1 increases very quickly , i.e. it is freed. After 18 updates λ_1 peaks, and accounts for more than 50% of the total exploration Λ. Then, the second joint is freed and even overtakes the first joint, peaking at update 26 . Subsequently, joint 3 increases, and peaks at update 34. It thus becomes clear that the first three joints, which have the largest effect on end-effector position, are freed from proximal to more distal ones. At update 50, the goal is reached, and the rest of the learning is concerned with minimizing joint angles and accelerations, which involves all joints. At update 150 the task is learned, and all exploration (beyond λ^{min}) has decayed. Thus, when the task is learned, the exploration in all joints ceases.

Here, we see that the different phases in exploration magnitude tuning in Fig. 5 correspond to different phases of learning the movement.

Conclusion: $\text{PI}^2_{\text{CMAES}}$ freezes and frees joint sequentially, depending on where the robot is on its self-organized developmental trajectory to learn the task. Furthermore, as learning progresses, joints are freed in a proximal-to-distal order, as is observed when infants learn to reach [5], [13]. Rather than having to specify the order of freeing/freezing joints [1], and/or their timing [15], structured maturation is an emergent property of probability-weighted covariance matrix updating in the $\text{PI}^2_{\text{CMAES}}$ algorithm.

VI. Related Work

In this paper, our aim was to demonstrate the potential of the $\text{PI}^2_{\text{CMAES}}$ RL algorithm for developmental learning of motor tasks, through adaptive control of exploration magnitude and self-organization of structured maturation.

Several previous works have also considered mechanisms for progressive release of motor degrees of freedom. Some models have studied the impact of pre-defined and fixed stages of freeing and freezing DOFs [6]. Others have shown how the pace of the sequencing of discrete stages [8] or of the continuous increase of explored values of DOFs along a proximo-distal scheme [1] could be adaptively and non-linearly controlled by learning progress and lead to efficient motor learning in high-dimensional robots. In this article, we have shown that without an explicit mechanism for motor maturation, such efficient maturational schedules, alternating freezing and freeing of DOFs, can be generated by $\text{PI}^2_{\text{CMAES}}$ entirely automatically.

Because our focus has not been on the algorithms on which $\text{PI}^2_{\text{CMAES}}$ is based, we have not been able to do justice to the sound derivations on which these algorithms are based. For a more in-depth discussion of the advantages of using update histories to improve covariance matrix updates, as is done in the CMA-ES is described in [12]. PI^2 is derived from first principles of optimal control, and gets its name from the application of the Feynman-Kac lemma to transform the Hamilton-Jacobi-Bellman equations into a so-called path integral, which can be approximated with Monte Carlo methods. For the full derivation, we refer to [21]. An excellent discussion of the relationship between direct reinforcement learning algorithms and evolution strategies is given by Rückstiess et al. [17], where extensive empirical comparisons between several methods in both fields are made.

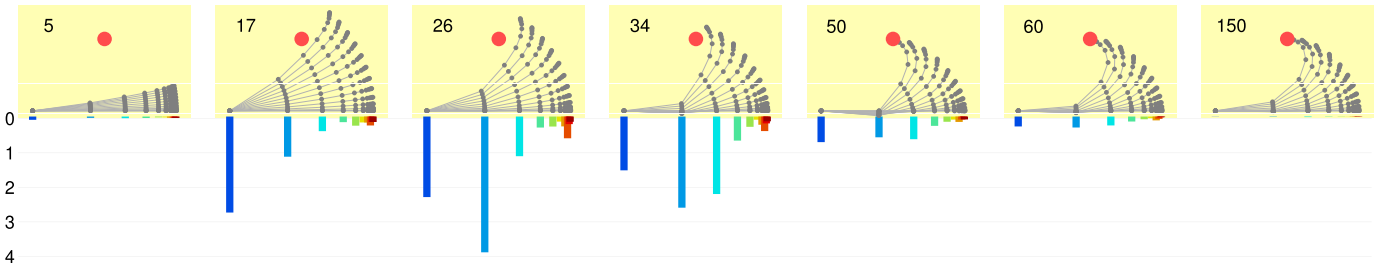

Fig. 6. The arm motion at different stages of learning. The numbers next to the arm indicates the number of updates. The exploration magnitude per joint λ_m is plotted as a bar graph below each arm. **Interpretation:** In the first phase (update 1-17), a simple sweeping movement is made, using only the first 'shoulder' joint. This is the most effective way of quickly decreasing the low cost, which is why the 'shoulder' joint has the most exploration. In the second phase (update 18-50), the arm starts using other joints (2 and 3) to 'reach inward' towards the goal. In phase 3 (after update 51-120), the algorithm learns to minimize the maximum joint angle in the final posture. In the final phase (after update 120), the task has been learned, the joint are frozen (exploration has decayed almost completely), and we hardly see any improvement.

VII. Conclusion

We have shown, in the context of a (changing) reaching task, that $\text{PI}^2_{\text{CMAES}}$ shows useful developmental properties for adaptive exploration and life-long learning of motor skills. First, we demonstrated how it could continually and automatically adapt to abruptly or continuously changing tasks, and without direct external information about these changes. Second, we have shown how the proximo-distal maturation structure well-known in humans [5], [13], and previously demonstrated to be highly useful for robot learning in high-dimensions [1], was here entirely self-organized.

Self-organization of maturational structure can be intuitively understood as a result of the interaction between a sensorimotor space where there is intrinsic asymmetry (e.g. a uniform variance over the control dimensions produces a non-uniform variance over the cost) and a learning algorithm like $\text{PI}^2_{\text{CMAES}}$ capable to automatically leverage this asymmetry. Identifying the detailed and respective roles of body structure, learning algorithm and their coupling for maturational self-organization will thus be a focus of future work.

Future work will also study how *perceptual* maturation [15], [14] could adaptively be driven using $\text{PI}^2_{\text{CMAES}}$, for example using exploration magnitude a measure of competence progress to drive the increase of perceptual capabilities, thus transposing to $\text{PI}^2_{\text{CMAES}}$ the perceptual maturation mechanism proposed in [1].

A third strand of future work will consider how such adaptive exploration and maturational structures can be leveraged in a multi-task setting, where maturational structures generated to learn to reach a given set of goals can be generalized to bootstrap even more efficiently the learning of novel goals.

References

[1] A. Baranes and P-Y. Oudeyer. The interaction of maturational constraints and intrinsic motivations in active motor development. In *IEEE International Conference on Development and Learning*, 2011.

[2] A. Baranes and P-Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 2012.

[3] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *International Conference on Machine Learning, ICML*, 2009.

[4] N. Bernstein. *The Coordination and Regulation of Movements*. Pergamon, 1967.

[5] N. E. Berthier, R.K. Clifton, D.D. McCall, and D.J. Robin. Proximodistal structure of early reaching in human infants. *Exp Brain Res*, 1999.

[6] L. Berthouze and M. Lungarella. Motor skill acquisition under environmental perturbations: On the necessity of alternate freezing and freeing degrees of freedom. *Adaptive Behavior*, 12(1):47–63, 2004.

[7] D.F. Bjorklund. The role of immaturity in human development. *Psychological Bulletin*, 122(2):153–169, September 1997.

[8] Josh C. Bongard. Morphological change in machines accelerates the evolution of robust behavior. *Proceedigns of the National Academy of Sciences of the United States of America (PNAS)*, January 2010.

[9] Jon Dattorro. *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing USA, 2011.

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

[11] J. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48:71–99, 1993.

[12] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[13] J. Konczak, M. Borutta, T Helge, and J. Dichgans. The development of goal-directed reaching in infants: hand trajectory formation and joint torque control. *Experimental Brain Research*, 1995.

[14] M.H. Lee, Qinggang. Meng, and F. Chao. Staged competence learning in developmental robotics. *Adaptive Behavior*, 15(3):241–255, 2007.

[15] Y. Nagai, M. Asada, and K. Hosoda. Learning for joint attention helped by functional development. *Advanced Robotic*, 20(10), 2006.

[16] P-Y. Oudeyer, F. Kaplan, and V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):pp. 265–286, 2007.

[17] Thomas Rückstiess, Frank Sehnke, Tom Schaul, Daan Wierstra, Yi Sun, and Jürgen Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn. Journal of Behavioral Robotics*, 1:14–24, 2010.

[18] Andrew Stout, George D. Konidaris, and Andrew G. Barto. Intrinsically motivated reinforcement learning: A promising framework for developmental robot learning. In *AAAI*, 2005.

[19] Freek Stulp. Adaptive exploration for continual reinforcement learning. In *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[20] Freek Stulp and Olivier Sigaud. Path integral policy improvement with covariance matrix adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012. `http://icml.cc/2012/papers/171.pdf`

[21] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11:3137–3181, 2010.

[22] B. Vereijken, R.E.A. van Emmerik, H.T.A. Whiting, and K.M. Newell. Free(z)ing degrees of freedom in skill acquisition. *Journal of Motor Behavior*, 24:133–142, 1992.