C-GRAIL: Autonomous Reinforcement Learning of Multiple and Context-Dependent Goals

Vieri Giuliano Santucci, Davide Montella, and Gianluca Baldassarre

***Abstract*—When facing the problem of autonomously learning to achieve multiple goals, researchers typically focus on problems where each goal can be solved using just one policy. However, in environments presenting different contexts, the same goal might need different skills to be solved. These situations pose two challenges: 1) recognize which are the contexts that need different policies to perform the goals and 2) learn the policies to accomplish the same goal in the identified relevant contexts. These two challenges are even harder if faced within an open-ended learning framework where potentially an agent has no information on the environment, possibly not even about the goals it can pursue. We propose a novel robotic architecture, contextual GRAIL (C-GRAIL), that solves these challenges in an integrated fashion. The architecture is able to autonomously detect new relevant contexts and ignore irrelevant ones, on the basis of the decrease of the expected performance for a given goal. Moreover, C-GRAIL can quickly learn the policies for new contexts leveraging on transfer learning techniques. The architecture is tested in a simulated robotic environment involving a robot that autonomously discovers and learns to reach relevant target objects in the presence of multiple obstacles generating several different contexts.**

***Index Terms*—Autonomous robotics, context-dependent goals, developmental robotics, intrinsic motivations (IMs), multitask reinforcement learning (RL).**

I. Introduction

IN RECENT years, the development of autonomous agents able to choose their own tasks solely on the basis of their interaction with the environment and of the motivations generated by this interaction has gained increasing interest within artificial intelligence, robotics, and machine learning. Although autonomy and versatility are pursued via different approaches, such as information theory [1], [2], evolutionary computation [3], deep learning [4], or epigenetic models [5], the field of developmental robotics [6], and in particular, the research on intrinsically motivated open-ended learning [7], [8], is producing a growing number of promising results.

Manuscript received 21 June 2021; revised 10 December 2021; accepted 3 February 2022. Date of publication 16 February 2022; date of current version 13 March 2023. This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme under Grant 713010, and in part by the Project "GOAL-Robots—Goal-Based Open-Ended Autonomous Learning Robots." *(Vieri Giuliano Santucci and Davide Montella contributed equally to this work.) (Corresponding author: Vieri Giuliano Santucci.)*

The authors are with the Istituto di Scienze e Tecnologie della Cognizione, Consiglio Nazionale delle Ricerce, 00185 Rome, Italy (e-mail: vieri.santucci@istc.cnr.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCDS.2022.3152081.

Digital Object Identifier 10.1109/TCDS.2022.3152081

The concept of intrinsic motivations (IMs) is borrowed from the biological [9] and psychological literature [10] describing how novel or unexpected "neutral" stimuli, as well as the perception of control, can drive learning processes in the absence of rewards or assigned goals. In the computational field, IMs have been implemented to foster different autonomous processes, such as state-space exploration [11]–[13], knowledge gathering [14], [15], learning repertoire of skills [16]–[18], affordance exploitation [19], [20], goal selection [21]–[23], and also boosting imitation learning techniques [24], [25].

In a reinforcement learning (RL) perspective, solving a task can be described as the process of an agent learning a policy to maximize a reward function R usually associated to a goal [26], i.e., a target state or effect. When facing the problem of autonomously learning multiple goals, researchers typically consider the case where each goal can be considered a single task [27], [28], thus training one policy for each goal (or one goal-parameterized policy). However, in real-world scenarios, the environment might change over time: as an example, objects might be displaced in different positions, or may only be present at certain times, and their locations might obstruct the achievement of some goals. The different configurations of the environment constitute the different contexts in which the system will have to learn to achieve its goals. In other words, different from the perspective in which each goal corresponds to a task, here for each goal there are as many tasks as there are contexts, to which potentially the system should associate a specific policy. In an application-oriented perspective, we can think of a harvesting robot in agrifood scenarios, where solving a goal (reach-and-pick the same fruit/vegetable) might necessitate different policies given contexts that might not be anticipated at design time. Moreover, "picking up" can be considered as a single, high-level goal that the robot has to solve in different contexts, determined by both the specific field and the specific fruits/vegetables to be collected. Having robots capable of autonomously and dynamically facing these situations might reduce the costs for small producers and improve automation. Similarly, service robots applied in highly unstructured or even unknown environments, such as houses, have to face tasks that might be affected (or not) by many elements that vary not only between different houses but also between different rooms in the same house.

The machine learning literature has proposed RL models to face problems where the transition and/or reward functions change in time and the agent is not informed on the context it is facing. These problems have, in particular, been tackled under

the heading of "nonstationary environments" [29], where the challenge is both to recognize the contexts to associate with different models/policies, and to manage them [30]–[32]. Different from this approach, in this work, we consider the case where a robotic agent can use its sensors to perceive the context it is facing. For example, in the domain we consider here, the agent focuses on objects and obstacles relevant to solve the tasks at hand. This approach is related to the object-action compound framework (OAC [33], see also [34]), stressing the importance of considering object-related contexts for the successful performance of actions. However, even under this assumption, the challenge of managing all possible contexts remains: indeed, a real-world scenario presents a large set of features whose variations might generate a large (possibly infinite) set of different contexts. In the perspective where each context might be considered as a separated task (even if sharing the same final goal), the system should therefore associate to each one a different policy to be trained to reach the goal in that specific configuration of the environment, eventually overloading the agent's learning processes and memory resources. Reducing the learning time has been successfully addressed through transfer learning (TL) techniques [35], also within the RL framework [36]: when learning a new policy, transferring previously acquired knowledge to a similar task can significantly speed up the process. However, this technique does not cope with the problem of managing and limiting the number of contexts (thus policies) in which the system is trying to learn.

Our proposal is to consider that some configurations of the environment may not be relevant for the achievement of a goal, and therefore, they should be grouped together and "solved" using a single strategy: for example, the positioning of some objects far from the target of the robot could be irrelevant for the policy acquired so far, so that the same policy would allow to achieve the same goal in different contexts. In a limited or controlled environment, the contexts could be potentially identified and enumerated through the analysis of the world structure, grouped by their similarity and coded into the artificial system. However, in the perspective of autonomous development and learning in unstructured and eventually unknown scenarios, the artificial agent has to be endowed with a mechanism through which it can autonomously understand which contexts could be "ignored" (and hence, associated to previously existing policies) and which ones require new strategies to achieve the same goals.

To the best of our knowledge, this problem has not been previously faced with intrinsically motivated autonomous learning agents and in the current work, we thus compare various versions of the proposed robotic architecture endowed with different mechanisms (or with combinations of them) to properly analyze their roles. In particular, we present contextual-GRAIL (C-GRAIL), an extension of GRAIL architecture (*goal-discovering robotic architecture for intrinsically motivated Learning* [37]). Similar to its precursor, C-GRAIL is able to autonomously discover, select, and learn to achieve goals on the basis of IMs, but new mechanisms allow C-GRAIL to assign goals different values according to contexts. Moreover, C-GRAIL is equipped with TL capabilities and with a smart context-detector (SCD) mechanism to cope with the problem of large numbers of possibly equivalent contexts. The system is tested in an object-reaching scenario, where multiple obstacles might be positioned around the targets, generating many different contexts. Despite the simplicity of the experiments and the limitations of the specific implementation of C-GRAIL (highlighted in the conclusions), the results show that the approach and the mechanisms introduced in this new architecture constitute a viable proposal for dealing with this type of problem. In this sense, C-GRAIL can be considered as a blueprint architecture and the experiments proposed here constitute a proof of concept of its validity and of the possibility to be implemented in a robotic agent.

The remainder of this article is structured as follows. Section II introduces the general problem of autonomous, open-ended learning of multiple goals (Section II-A), the specific problem of autonomous goal learning under varying contexts (Section II-B), and our proposed solution (Section II-C). Section III describes C-GRAIL general functioning (Section III-A) and its mechanisms and components (Section III-B), while specific implementation details can be found in the Appendix. Section IV describes the experimental scenario and the compared systems, while the results are presented in Section V. Finally, in Section VI, we draw conclusions, highlight limitations, and describe possible future extensions of this work.

II. Problem Description and Proposed Solution

A. Autonomous Learning of Multiple Goals

In classical RL, the objective of the agent is to find an optimal policy π^* that at each time step t, given the current state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$ to maximize the expected returns, defined as the sum of rewards $R = \sum_{t=0}^{T} r_t$ over the maximum time T. Rewards might be distributed over the state space or associated only with the goal g, i.e., a specific target state s^g or, as in [28], a set of states $S^g \in \mathcal{S}$, so that whenever the agent reaches any $s^g \in S^g$, the goal is achieved.

When learning a set of multiple goals $\mathcal{G}$, the objective of the agent is thus to find multiple goal-dependent policies π^g, each one maximizing a goal-dependent reward function R^g. In particular, the overall objective is to find, for each g, an optimal policy such that

$$\forall g \in \mathcal{G}, \quad \pi^g(a_t|s_t) = \underset{\pi^g}{\operatorname{argmax}}\, \mathbb{E}\left[R^g(\pi^g)\right] \tag{1}$$

where each goal-related policy is encapsulated in a separated module ("expert") trained through any RL technique.[1]

Different from the classical multitask RL, where tasks to be learned are scheduled by the human programmer [28], [38], in an autonomous open-ended learning perspective [39], we want the robot to be able to autonomously select its own tasks to maximize the expected competence C over all goals

$$C = \sum_{\mathcal{G}} C^g \tag{2}$$

[1] Alternatively, it could be described as a single parameterized policy [38] defined as $\pi(a_t|s_t, g)$ [28]. The two formalizations are interchangeably with respect to the purposes of this work.

where C^g is a value quantifying the probability that the goal-related policy π^g will bring the agent in one of the goal states $s^g \in S^g$ when executed. In this sense, autonomous open-ended learning is not simply about reward maximization but focuses on developing agents that gain the highest amount of knowledge over an environment (independently from assigned tasks) so that this knowledge might be exploited in the future (e.g., to solve users' requests). This has been often depicted as a two-phase scenario [40], [41]: in a first "intrinsic phase,"[2] the agent is presented with a (potentially extremely large) set of learnable goals $\mathcal{G}$, while in a second "extrinsic phase,"[3] the system will be asked to solve a specific (and previously unknown) subset of goals $G^{\text{test}} \subset \mathcal{G}$. The research on autonomous open-ended learning, as well as the work presented here, focuses on the first phase. In particular, under the assumption that the autonomous exploration-and-learning phase has a limited but unknown time horizon L, the problem consists in appropriately allocating the training time between goals so that C_L (the overall competence of (2) at time L) is maximum.

At each trial, the system focuses on a goal g, training π^g for a certain amount of learning time lt, so that previous competence for g at time t (C^g_t) will be (possibly) improved by a delta

$$\Delta C^g = C^g_{t+lt} - C^g_t. \tag{3}$$

Consequentially, according to (2), also the overall competence C is improved by the same value ΔC^g. The objective of the agent in the intrinsic phase is thus to build a metapolicy Π that, at each *trial*, selects a goal so that after the entire learning period L, the competence over all goals C is maximized

$$\Pi^*(g) = \underset{\Pi}{\text{argmax}}\ \mathbb{E}[C \mid \Pi]. \tag{4}$$

Since the time horizon L is unknown and since it is not possible to analytically determine how C is changing after the selection and training of each goal, a common solution [43] is to use a greedy approach that at each trial maximizes the expected competence improvement ΔC, which is the intrinsic reinforcement used to train the metapolicy Π. With these premises, goal selection can then be modeled as an N-armed bandit problem [26] as done in many architectures for autonomous open-ended learning [23], [37], [44]–[46], where goals $g \in \mathcal{G}$ are the arms of the bandit and ΔC^g are goal-specific returns (depending on the training of the low-level, goal-related policies π^g), so that the value of each goal $Q(g)$ is dependent on the currently expected ΔC^g for practicing/selecting that goal. Note that given the transient nature of ΔC (and in general of IM signals), which decreases while the competence for a goal improves toward its maximum, the goal selection N-armed bandit should be more precisely considered as a *rotting bandit* [47].

Under these assumptions, the general problem of autonomous open-ended learning of multiple skills can thus be seen as a training-time allocation problem on top of a multitask RL problem or, from a hierarchical perspective, as a two-level problem: 1) the high-level problem of selecting a goal g to train to rapidly maximize the overall competence and 2) the low-level problem of training the goal-related policies π^g.

B. Introducing Context Dependency

As described in Section II-A, usually multiple goal learning is addressed assuming that for each goal g, a single goal-dependent policy π^g may be sufficient. Differently, here we are considering the case where the same goal might need different policies given different environmental conditions. This is common in real-world scenarios, where achieving the same goal may require a robot to use different strategies according to the context: for example, certain objects might be obstacles impairing the reaching of certain targets; hence, the agent should use different policies to reach for the same location in different contexts.

If we assume that the agent is able to identify the current context $\phi \in \Phi$ through its sensors, we can divide the set of features $\mathcal{F}$ perceived by the agent in two subsets: 1) "low-level" features f^s, changing at the time-scale of an attempt to achieve a goal (within trials) and describing the input state for the control policy π^g (e.g., the displacement of the joints of a robotic arm) and 2) high-level features f^ϕ changing over a wider time scale (e.g., every n trials) and describing the current context ϕ, where ϕ can be defined as a vector containing a specific combination of the values (that for simplicity can be thought of as binary, e.g., "obstacle 1 is present/not present") of all contextual features.

Each ϕ can thus be seen as a different task where the system, to achieve the same g, might need a different policy π^g_ϕ. Consequently, the maximization of competence C over all goals $\mathcal{G}$ becomes the problem of maximizing C over $\mathcal{G}$ in all contexts Φ. The objective in (4) will thus change to

$$\Pi^*(g|\phi) = \underset{\Pi}{\text{argmax}}\ [C \mid \Pi] \tag{5}$$

so that goal selection can now be modeled as a contextual-bandit problem [26] in which the selection of the goal to train at each trial is dependent on the current context ϕ, as well as intrinsic returns ΔC^g_ϕ and value assignment $Q(g|\phi)$.

This description assumes that each context ϕ is actually affecting goal achievability. However, in real-world scenarios, it is easy to imagine that many of the features describing a context *do not* affect the policy that the system is learning. The fact that it is night or day might have no effect on a reaching task performed indoor, while the on/off state of the light in the room could; similarly, the presence of objects distant from the target to be reached should not be taken into consideration, while objects positioned close to the target might become obstacles. If the robot is supervised by a human designer, the latter could spot the relevant elements of the context, but in an open-ended learning perspective (where eventually the agent is operating in unknown scenarios), the robot should be able to autonomously identify what is relevant for goal achievement. Indeed, associating a different policy π^g_ϕ to any distribution of

[2]Goals are pursued for competence acquisition and not for immediate reward maximization, although there is a continuum between the two phases [42].

[3]Goals are pursued to solve users' requests.

"high-level" features might substantially slow down the learning process and burden the system computational resources. TL has proven to speed up skill acquisition (also in association with IMs [48]), but this might not be sufficient to cope with the "combinatorial explosion" of features, determining possibly infinite contexts and thus, policies. Finding a smart and autonomous way to avoid context/policy proliferation (i.e., a solution to the problem of considering each context as relevant and then having to associate it with a new policy) without impairing the learning of multiple, context-dependent goals is thus a crucial issue for autonomous learning.

C. Proposed Solution

The autonomous learning of multiple context-dependent tasks presents two main challenges: 1) allocating training time over goals taking into account different contexts and 2) avoiding policy proliferation so as not to slow down the learning process. To tackle the first issue, our approach follows the analysis in Section II-B: task selection is treated as an N-armed contextual bandit where the system evaluates each goal on the basis of the competence improvement ΔC^g_ϕ expected in the current context ϕ. At the lower level of policy learning, for each goal and for each context, the system associates a specific policy π^g_ϕ, trained through any learning algorithm maximizing the rewards R^g for achieving the goal state(s) s^g defining g. However, this general schema is modeled according to our solution to problem 2).

To avoid policy proliferation, we introduce a heuristic based on Ockham's razor ("Numquam ponenda est pluralitas sine necessitate" [49]—"Plurality is never to be posited without necessity"), which in this scenario would prescribe the agent not to multiply policies unless strictly necessary. In particular, we propose to work at the level of context detection: even if the robot perceives through its sensors the current context ϕ as it is in the environment, what the system actually takes into consideration is determined by a binary vector *rcf* (*relevant contextual features*) with the same length of ϕ that stores which contextual features have been relevant for goal achievement so far. Moreover, since certain contextual features might be relevant for a goal but irrelevant for others, for each g, the system has a goal-specific rcf^g vector, working as a mask over ϕ. This process will lead the agent to identify a different list of relevant contexts Φ^g for each goal, and for each ϕ^g in Φ^g, it will associate a specific policy $\pi^g_{\phi^g}$.

Given the current context ϕ in the environment, the agent will consider different goal-specific contexts ϕ^g, each one resulting from the following elementwise multiplication:

$$\forall g \in \mathcal{G}, \quad \phi^g = \phi \odot rcf^g \tag{6}$$

so that the contextual bandit modeling goal selection is now a particular instantiation of the classical problem, where the evaluation of each arm/goal of the bandit depends on a context, which is arm specific and determined by (6).

In accordance with Ockham's principle, initially all rcf^g masks contain only *zeros* so that the agent starts to face the environment by actively ignoring the contextual features: in other words, at the beginning, the system selects goals *as if* there was only one "baseline/blank" context ϕ^g_0, thus associating just one policy $\pi^g_{\phi^g_0}$ to each goal. It is only when "something goes wrong" that the system evaluates the possibility that some contextual features may be relevant and should therefore be taken into consideration. To assess this necessity, within a model-free framework, the agent relies on the behavior of its low-level policies π^g. In particular, novelty or surprise, as suggested by the IM framework [50], can be used to signal that an anomaly occurred: similar to [38] (although that was a model-based system), we propose that an "unexpected failure" of $\pi^g_{\phi^g_n}$ can be used as a signal for the identification of new relevant contextual features (see Section III-B). A new context ϕ^g_{n+1} will be then actively perceived by the agent and a new policy $\pi^g_{\phi^g_{n+1}}$ will be associated with it. Moreover, $\pi^g_{\phi^g_{n+1}}$ will possibly use TL drawing knowledge from policies developed for the same goal in previously identified contexts (see Section III-B for an example).

However, if this smart context-detection strategy is applied from the very beginning of the learning process, when the agent is still performing (almost) random behavior, this would still possibly result in an explosion of detected contextual features. For this reason, our proposal is to trigger this mechanism only when the competence $C^g_{\phi^g}$ for g in ϕ^g is higher than a threshold (whose optimal setting is beyond the scope of this work; see the Appendix for the values used in this work). Furthermore, considering all the contextual features of the environment when a failure occurs would not follow Ockham's principle. The system thus considers only those features in ϕ that are "relevant" for the goal at hand. Heuristics and previous knowledge might be used for this purpose: for example, in a manipulation task, they could be those related to an object the robot hits in proximity of the target. If these heuristics are not sufficient, the agent should then consider also the other features (but again, refining these heuristics is not the purpose of this research).

III. C-GRAIL

In this section, we instantiate contextual-GRAIL (C-GRAIL), a system that extends the GRAIL architecture [37] encapsulating the proposals described in Section II-C. GRAIL is able to autonomously: discover goals; select goals according to competence-based IMs (CB-IMs); recognize goal achievement; and select computational resources to learn goal-related skills. Despite its advancements, GRAIL (similar to other systems [23], [44], [45]) is not able to manage context-dependent goal selection, nor to identify relevant contexts and avoid policy proliferation. Instead, the C-GRAIL architecture can cope with multiple-context learning based on a smart context detector (SCD) that identifies and stores new relevant contextual features, thus contexts, when needed, allowing the enrichment of the system behavior. Although autonomous goal discovery is not the focus of this work, C-GRAIL is able to perform it inheriting this capability from GRAIL.

Following the problem analysis and the suggested solution presented in Section II, C-GRAIL can be seen as a two-level architecture (with respect to the tackled problems), where the

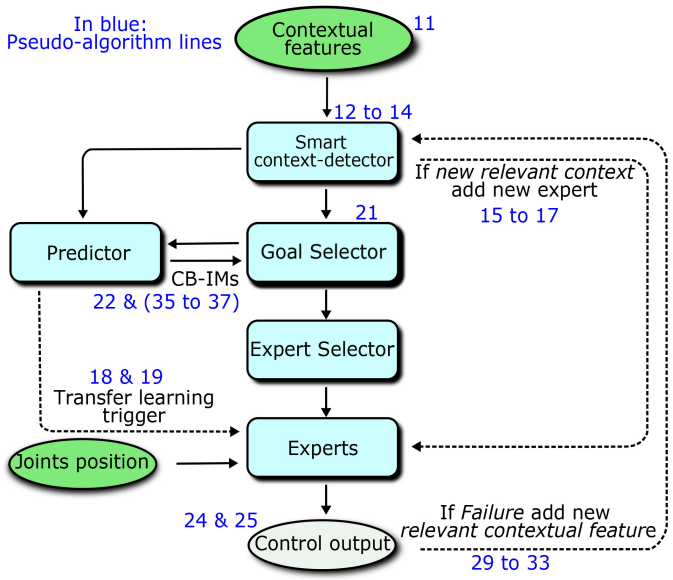

Fig. 1. C-GRAIL with main components and references to the lines in the pseudoalgorithm describing the different processes. Note that experts are goal-and-context specific.

high level learns to properly allocate training time solving a (rotting) contextual bandit that maximizes the CB-IMs provided by the different goals, while the low level learns the goal-related policies maximizing the rewards (self-)provided for achieving the selected goals. IMs are thus used, similar to [43] and [44], to explore the goal space and select the most promising goal (with respect to competence gain) in the current context. C-GRAIL hence learns inverse models to achieve the goals it has selected on the basis of CB-IMs. Furthermore, it is important to emphasize that the problem of learning different skills to reach the same goal in different contexts is addressed here from the perspective of the high-level task of training time allocation: at the low-level of policy learning, we could have implemented different algorithms (or a different architectural structure) without modifying our general solution.

Section III-A presents a formal description of C-GRAIL capturing the main issues tackled in this article. Other functions that are not specifically relevant to this work, such as goal-discovery and expert selection, as well as the specific implementation of the system, are reported in Section III-B and the Appendix.

A. General Functioning

Algorithm 1 shows the general functioning of the C-GRAIL architecture (Fig. 1). We assume the system operates within an episode-based RL framework involving *trials*, each lasting T time steps. We assume the state of the environment s includes two sets of different features: 1) low-level features f^s (such as proprioception) that *might change within the trial* and 2) high-level contextual features f^ϕ (henceforth also denoted as ϕ for simplicity; these involve for example the presence of an obstacle in a certain location) that *might change between the trials* and can be used to identify a context. At the beginning of each *trial*, the system observes s and passes ϕ to a context-matching

Algorithm 1: C-GRAIL

1 Let $known\_goals = \{\}$ be the list of discovered goals
2 **foreach** g ***in*** $known\_goals$ **do**
3 $rcf^g = \mathbf{0}$ `(init rcf masks)`;
4 $\Phi^g = \{\}$ `(identified relevant contexts` ϕ^g`)`;
5 Let $\Delta C^g_{\phi^g}$ be the agent's intrinsic motivation for achieving goal g in context ϕ^g *in* Φ^g;
6 **for** $trial \leftarrow 1$ ***to*** End **do**
7 Set the initial configuration of the environment s_0;
8 Let t be the current time step within the trial;
9 **(Determine current relevant context for each goal; store new contexts and create new policies)**;
10 $I\Phi_{trial} = \{\}$ `(reset identified context list)`;
11 Observe current state of contextual features f^ϕ;
12 **foreach** g ***in*** $known\_goals$ **do**
13 $\phi^g = CM(f^\phi, rcf^g)$;
14 $I\Phi_{trial} \leftarrow I\Phi_{trial} + \{\phi^g\}$;
15 **if** ϕ^g *not in* Φ^g **then**
16 $\Phi^g \leftarrow \Phi^g + \{\phi^g_{n+1}\}$ `(add new context)`;
17 Create new policy $\pi^g_{\phi^g_{n+1}}$;
18 With a certain probability, select a policy $\pi^g_{\phi^g_k}$ among those, if any, with a *competence* $\geq$ *transfer_threshold*;
19 $\pi^g_{\phi^g_{n+1}} \xleftarrow{transfer} \pi^g_{\phi^g_k}$;
20 **(Select a goal)**;
21 $g \sim \Pi(I\Phi_{trial})$ `(based on` $\Delta C^g_{\phi^g}$`)`;
22 prior_prob $\leftarrow \chi(g|\phi^g)$ `(estimate the probability of achieving g given` ϕ^g`)`;
23 **for** *each time step* $t \leftarrow 1$ ***to*** T *within the trial* **do**
24 $a_t \sim \pi^g_{\phi^g}(f^s)$ `(select an action)`;
25 Perform action a_t in the environment;
26 Observe next state s_{t+1};
27 Let $r_t = GM(s_{t+1}, g)$;
28 Update $\pi^g_{\phi^g}$ `(update policy based on` (s_t, a_t, r_t, s_{t+1}) `)`;
29 **if** *failure* & $t < T$ & *prior_prob* > *competence_threshold* **then**
30 Observe features f^ϕ in s_{t+1};
31 Let f^{fail} be failure-related features in f^ϕ;
32 **if** $f^{fail} \neq 1$ *in* rcf^g **then**
33 $f^{fail} \leftarrow 1$ *in* rcf^g
34 **(Update success predictor and intrinsic motivation)**;
35 Update goal-success predictor χ based on GM;
36 posterior_prob $\leftarrow \chi(g|\phi^g)$ `(updated probability of achieving goal)`;
37 $\Delta C^g_{\phi^g} \leftarrow$ posterior_prob − prior_prob;

(CM) function, returning the context ϕ^g, which is relevant for each goal g (in the list of currently *known_goals*) given the current state and the goal-specific *relevant contextual feature* masks rcf^g as described in (6).

If a new relevant context ϕ^g_{n+1} is identified for a goal, it is added to the list of known contexts Φ^g for g and a new policy $\pi^g_{\phi^g_{n+1}}$ is created and associated to it. TL techniques are

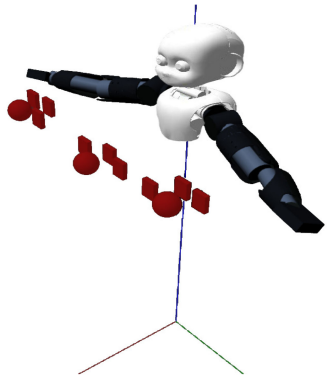

Fig. 2. Robot, of which we used here only the two actuated arms and a camera in the head, the three target spheres, and the nine possible obstacles.

used to speed up the training of the policy: with a certain probability, a policy $\pi^g_{\phi^g_k}$ is randomly selected from those (if any) used to achieve the same goal in different contexts and having a competence higher than a certain *transfer_threshold*. The selected policy is then used to initialize the new policy (see Section III-B for details). The list of *identified contexts* ($I\Phi$) for the current trial, one for each goal, such that

$$I\Phi_{\text{trial}} = \left\{\phi^{g_1} \dots \phi^{g_j} \dots \phi^{g_n}\right\} \tag{7}$$

is passed to the goal selector policy Π determining the goal to pursue in the current trial on the basis of the expected competence improvement $\Delta C^g_{\phi^g}$. We make no assumptions on how to implement Π, but it can be seen as a stochastic policy selecting a goal according to a softmax distribution based on IMs. The competence measure $C^g_{\phi^g}$ is defined as the probability estimate of achieving g given ϕ^g, autonomously assessed through a "predictive function" $\chi(g, \phi^g)$. $\Delta C^g_{\phi^g}$ is computed as the difference between prior (*prior_prob*) and posterior (*posterior_prob*) estimated probabilities of achieving g given the same relevant context ϕ^g.

At each time step t within the trial, on the basis of the low-level features f^s of the current state s_t, an action a_t is selected through the low-level policy $\pi^g_{\phi^g}(f^s)$ bringing the agent in s_{t+1}. The new state is observed and a reward is autonomously calculated using a goal matching (*GM*) function, returning 1 for *success* and 0 otherwise. The policy $\pi^g_{\phi^g}$ is hence updated accordingly. If the trial ends before its time up T with a failure, and the computed *prior_prob* is over a certain *competence_threshold*, the high-level features f^{fail} in f^ϕ related to the condition at hand (see Section III-B) are set to 1 in the rcf^g vector. Finally, the competence predictor χ is updated accordingly to goal achievement [0; 1] and the difference between *prior_prob* and *posterior_prob* is calculated to provide the CB-IM $\Delta C^g_{\phi^g}$ that biases goal selection.

B. Implementation

1) Robot and Sensory Input: C-GRAIL is implemented in a simulated iCub robot through the 3-D physical engine GAZEBO [51] and dedicated YARP plugins [52]. The two arms have 4 degrees of freedom: wrist joints are kept fixed and the hands are substituted by two scoops (see Fig. 2). Collisions are not taken into consideration: a sensor in the center of each scoop detects whether the robot has "touched" an object. Inputs are provided by two sources: 1) the camera of the right eye, kept fixed so that objects are always in the visual field and 2) the arm joints, whose angles determine the robot proprioception.

2) Goal Discovery: As in other versions of GRAIL [37], [53], the system has a mechanism to discover "interesting" states and store them as possible goals: we use a simple, biologically inspired [54] built-in strategy detecting those states resulting from a change in the visual input (in this work, the change of color of a sphere, which "lights up" when touched by the robot). The visual inputs corresponding to two succeeding time steps are subtracted and whenever there is a change (arms movements are excluded, see the Appendix), the resulting image is compared with previously saved ones: if new, it is stored in the goal-representation map (GR-M) and associated with the first available unit of the goal selector (see Section III-B4).

3) Smart Context Detector: This component is introduced in C-GRAIL to avoid policy proliferation. It is composed of two mechanisms: 1) the detection of current relevant contexts and 2) the gathering of *relevant contextual features* (*rcf*). At the beginning of each trial, the SCD receives the set of contextual features ϕ, i.e., the values of those features that are not changing during the trial, which in the presented experimental case (see Section IV) are the occupied/nonoccupied positions of the obstacles. For each goal g, the system has a mask vector rcf^g where the features that have so far been identified as relevant to the achievement of g are set to 1. A *context matching* function *CM* filters the current features ϕ with the rcf^g of each goal, returning a list of goal-specific contexts $\{\phi^{g_1}, \phi^{g_2}, \dots, \phi^{g_n}\}$ for all the currently *known goals*. The list is used by the goal selector to choose the goal to pursue on the basis of context-specific IMs. If a new context ϕ^g_{n+1} is detected for g, it is added to the list of *relevant contexts* Φ^g for that goal, and a new policy $\pi^g_{\phi^g_{n+1}}$ associated to the goal g is added to the repertoire of the system as a new expert (see Sections III-B6 and III-B7).

While training on g, if there is a failure that terminates the trial before its maximum duration T (see Section IV), and if the predicted performance level (see Section III-B9) in the current context ϕ^g is higher then a *competence_threshold* (whose value has been set through experimental heuristics, see the Appendix), the SCD sets to 1 in rcf^g the contextual feature(s) f^ϕ related to the environmental state s_t where the agent is situated when the trial terminates (here represented by the center of the scoop on the actuated arm). Concretely, if the agent bumps into an obstacle, only the features related to that specific obstacle are set to 1, while the others are not taken into consideration since the system, following the law of parsimony, assumes that only the features related to its current end-effector position are involved in the signalled failure.

4) Goal Selector: At each trial, the goal selector determines the goal to pursue on the basis of CB-IMs and the current context. In particular, the goal selector takes as input the list $I\Phi$ of all the goal-specific relevant contexts ϕ^g identified by the SCD component for the current trial, and selects the

goal to pursue according to a *softmax* selection rule based on the current context-specific values of the goals, updated through a standard exponential moving average (EMA) based on goal-and-context specific CB-IMs (Section III-B9). As in other GRAIL versions, at the beginning of the experiment, the goal selector is a "blank vector" whose units are not associated with any specific goal, but during exploration the system is able to autonomously discover new goals and associate them to the units in the goal selector as described in Section III-B2.

5) Expert selector: C-GRAIL can be implemented by associating to each couple $<g, \phi^g>$ a single expert encoding the policy controlling the robot when learning g in ϕ^g. However, we endow the robot with the capability of autonomously selecting different computational resources, one for each arm, to achieve the same goal even in the same context. All the goals (in each context) can be achieved using both arms, but in this way, the system has a further degree of autonomy whose advantages were investigated in [55]. Given the selected goal and the current context for that goal, the selection of the expert is based on an EMA of the rewards provided by the goal-matching (GM, Section III-B8) function for that goal.

6) Experts: Each expert is implemented as a neural network actor–critic model modified to work with continuous states and action spaces [56]. At each time step, it receives as input the low-level features f^s corresponding to the four actuated joints of the related arm (three for the shoulder and one for the elbow). The four output units of the expert's actor encode the displacement of the joints through position control. In particular, the outputs are remapped into a range determining the variation (the delta) in the position of the four joints. At each step, the selected expert is trained through a TD RL algorithm [26] maximizing the rewards generated by the GM function for achieving the currently selected goal.

7) Transfer Learning: After a goal has been selected, the system checks through its competence predictor χ (see Section III-B9) the expected performance for that goal in the current context ϕ_k^g. If the prediction is under a certain *learning threshold*, the system checks if a policy $\pi_{\phi_j^g}^g$ exists for the same goal in a different context ϕ_j^g whose expected performance is higher than a *transfer threshold* and, with a certain *transfer probability*, uses it for TL (if multiple candidates are available, one is randomly picked with uniform probability). The robot action is then controlled by the policy of the selected *source expert* and the output of its actor and critic components is used to train the *target expert* (see the Appendix). Moreover, if the trial ends achieving the goal the parameters of both the policy and the evaluation function of the *source expert* are entirely copied into the *target expert* associated with that goal.

8) Goal Matching: The GM function allows the agent to autonomously check if a pursued goal has been achieved. When a goal is selected, its representation in the GR-M is activated. While operating, if a change is detected in the visual input, GM compares it with the representation of the selected goal: GM thus generates a signal of 1 if there is an overlap of the two images (0 otherwise). In the specific experimental domain presented here, the GM generates a binary signal, but it can be continuous in different domains (as in [57]). The signal is used as a reward to train the experts and the expert selector; moreover, it is used as a teaching input for the predictor whose activity determines the CB-IM signals.

9) Intrinsic Motivations: The CB-IMs signal, modeled as a competence improvement signal [43], is the result of the activity of the competence predictor χ. At the beginning of each trial, χ receives as input the selected goal g and the goal-specific relevant context ϕ^g, and outputs the predicted performance of the agent. At the end of the trial, the prediction is updated according to the GM output, and the competence prediction improvement (CPI), calculated over a fixed time window (see the Appendix), determines the goal context-dependent CB-IM signal biasing goal selection.

IV. Experimental Setup

A. Environment and Task

In addition to the robot, the environment presents three spherical objects representing potential reaching targets (Fig. 2). Moreover, a varying number of rectangular parallelepiped obstacles (with a maximum of 9) are placed close to the targets. For each target, there might be simultaneously three obstacles facing it on its right, left, and in the middle. All the objects are anchored to the world at reaching distance. When touched, targets "switch on" changing their color to green. The task consists in learning to activate the spheres in the shortest amount of time and in different environmental contexts. At the beginning of each *trial*, the number of the obstacles changes, thus configuring a new context (with nine present/absent obstacles, there are 512 possible contexts). The experiment lasts for 50 000 trials, each ending after 800 steps or when any object is touched.

Despite its simplicity, and to some extent because of it, this scenario allows us to focus on the main functions of the proposed system and analyze their contribution to the learning process. Furthermore, this type of task is typical of the IMOL literature [25], [44], [58], as well as our previous work on the GRAIL system [37], [53]: this allows us to both place this study in continuity with previous ones, and facilitate a comparison with other systems by highlighting advances and differences.

B. Compared Systems

We compared *C-GRAIL* to other three systems, each one lacking some of the mechanisms implemented in our architecture and reflecting other systems in the literature.

1) *Bandit:* This system is built similar to other architectures for autonomous multigoal learning [23], [37], [44], where goal selection is treated as a multiarmed bandit and each goal is associated with a single policy. Moreover, no TL is used.
2) *C-Transfer:* This system implements autonomous goal-selection as a contextual bandit similar to [22] and [59], but it is not endowed with the SCD mechanism as C-GRAIL, thus all possible contexts are actually taken into consideration by the agent. The system is equipped with TL to share knowledge between policies achieving the same goal in different contexts.

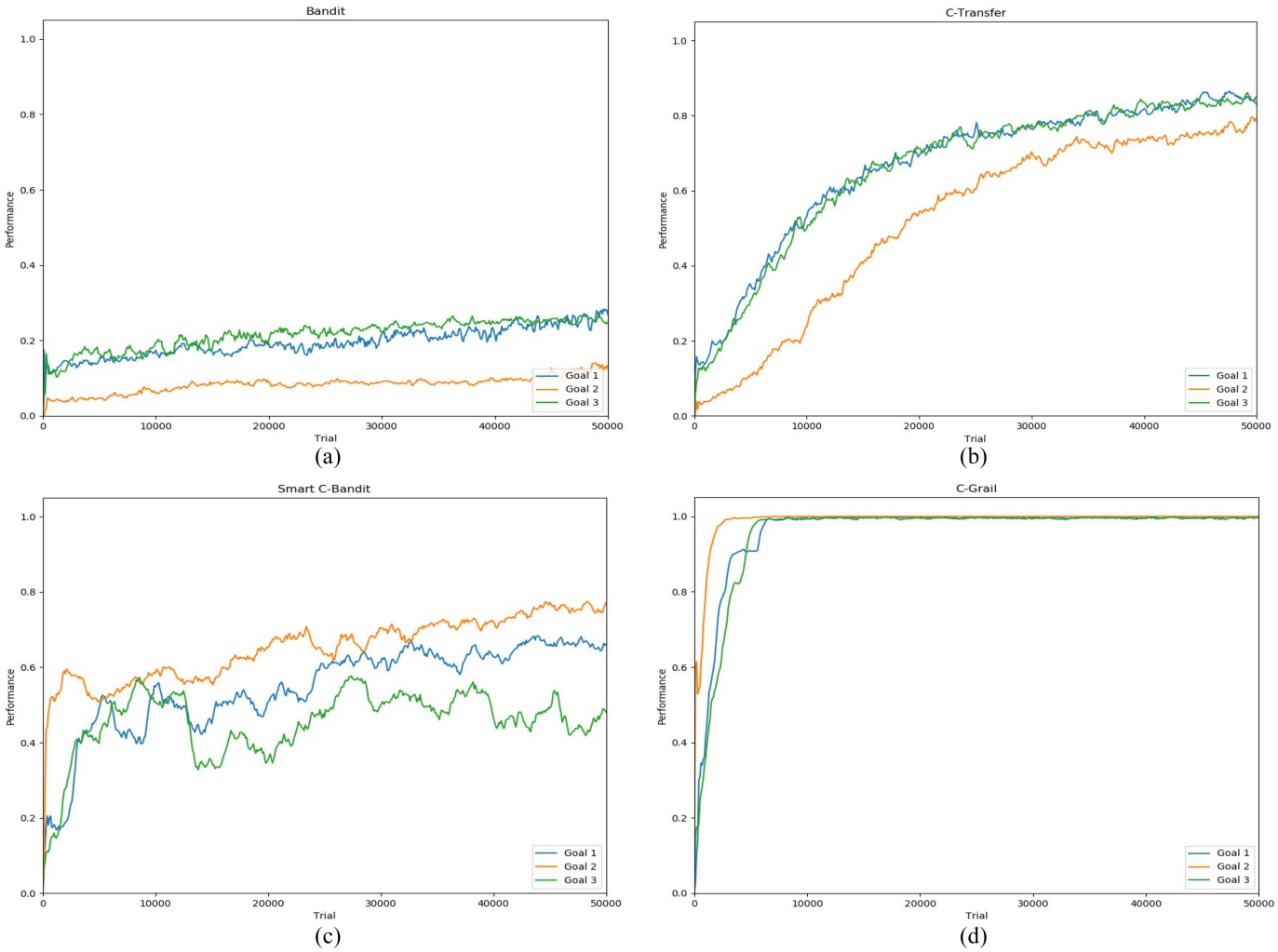

Fig. 3. Average performances (over ten repetitions) of the four tested systems in learning the three tasks. (a) Bandit. (b) C-Transfer. (c) Smart C-Bandit. (d) C-GRAIL.

3) *Smart C-Bandit:* This system implements the SCD mechanism as C-GRAIL, but it lacks TL. The system is thus able to limit policy proliferation, but it has to train each new expert from scratch.

The performance of the systems is calculated during the whole simulation as the average, over the last 30 attempts, of the success in achieving a goal given a context. When general results are presented (as in Fig. 3), the graphs report, for each goal, the average of the averages over all the possible contexts.

V. Results

Fig. 3 shows the average performance (over ten repetitions) of the four systems in learning to reach the three target spheres. As expected, the *Bandit* system cannot achieve a sufficient performance. The agent is not able to distinguish the different contexts, so it cannot select most profitable goals accordingly; moreover, having just one policy per goal, it continues to modify the same expert, thus incurring in catastrophic forgetting.

C-Transfer achieves a good performance over all the goals (∼80%) only at the very end of the experiment, Smart C-Bandit reaches a lower average performance (between ∼80% on goal 2 and ∼45% on goal 3), while C-GRAIL is capable to achieve a 100% performance after only ∼8000 trials. To investigate the differences between these last three systems and understand how the components of C-GRAIL contribute to its performance, we focus on the learning of a single goal (goal 1, i.e., the sphere positioned on the left of the robot) considering the data of a single representative seed for each system.

Fig. 4 shows the performance of C-Transfer on goal 1 as an average over all the contexts, while Fig. 5 shows the learning and performance for that goal with respect to the different contexts depending on the configuration of the obstacles. To make the visualization easier, we only show the contexts determined by the three obstacles close to the sphere associated with goal 1, i.e., 2^3 contexts plus the one without obstacles (identified as 1, 0, 0, 0 in Fig. 5 and Fig. 7). Notice that the total number of possible contexts is given by the combination of the nine obstacles that may be present in the world (512 possible configurations). Since C-Transfer does not have the SCD, the system considers, for each goal, all 512 possible contexts, thus potentially 512 policies to train. However, the

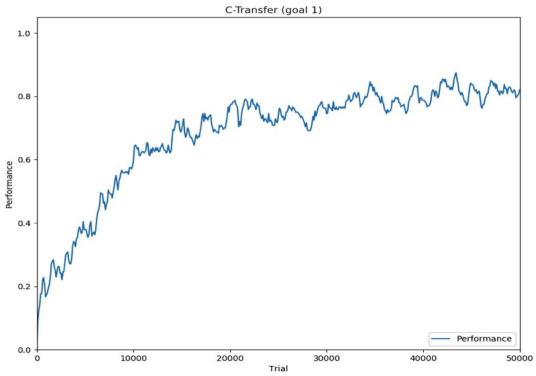

Fig. 4. C-Transfer: performance on goal 1 averaged over all the possible contexts.

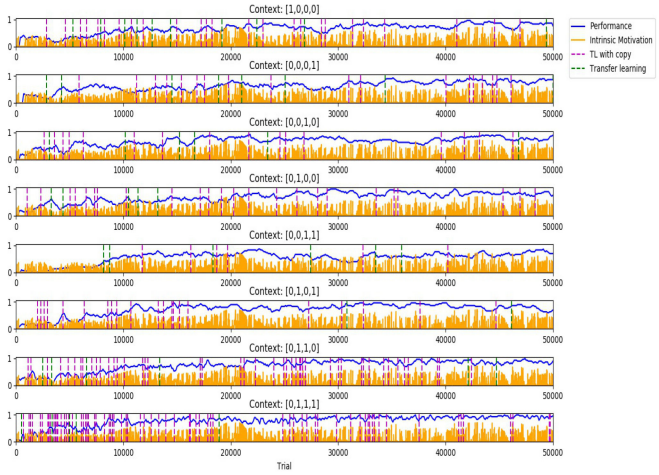

Fig. 5. C-Transfer: analysis of context-related learning for goal 1 (we plot here the contexts determined by three obstacles positioned close to goal 1), with TL between contexts and IM signal.

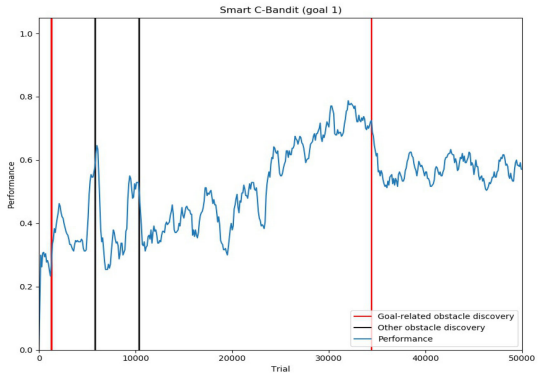

Fig. 6. Smart C-Bandit: performance on goal 1 (averaged on all the possible contexts) and discovery of relevant contexts.

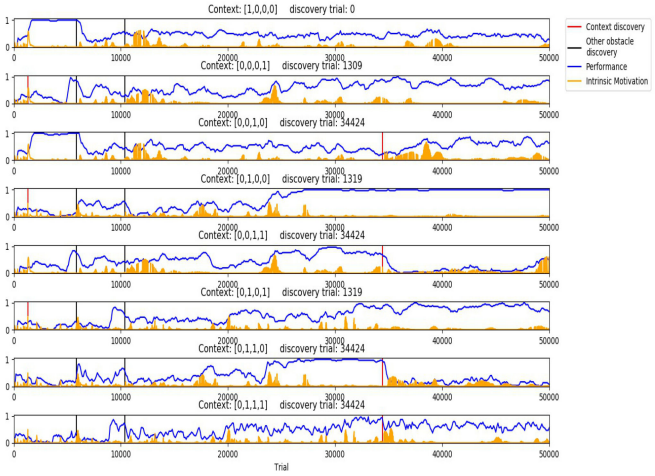

Fig. 7. Smart C-Bandit: analysis of context-related learning for goal 1 (contexts as in Fig. 5), with relevant context discovery and IM signal.

transfer mechanism guarantees the system to share the motor competence acquired in the contexts, both with partial transfer (dotted green lines, Fig. 5) or copying the entire policy from a different context when it can achieve the goal (dotted purple lines, Fig. 5). Although C-Transfer manages to achieve a high performance, the learning process is slowed down by the fact that the system considers all the possible configurations of the environment as relevant. This not only wastes time in transferring skills between irrelevant contexts but also slows down the goal selection process: as shown in Fig. 5, the IM signal (orange lines) is always present even when competence has been properly acquired. This is because the plotted signals incorporate the combination of single goal-related contexts with all the configurations of the obstacles that are close to the other spheres. Since IMs are context related, the system might still have motivations to improve its competence in a context even if it is actually able to achieve the goal: this is clear if we look at the end of the simulation, where even if the performance is near the 80% the magnitude of the IM signal is high, and where the system is still performing TL between contexts (which are those not explicitly reported in the graph).

Fig. 6 shows the performance of the Smart C-Bandit system on goal 1, averaged on all the possible contexts. Thanks to the SCD described in Section III-B3, the system adds new context-related policies only when hitting on obstacles (and only if the current policy has reached a minimum level of performance). In particular, when performing goal 1, Smart C-Bandit "discovers" (or rather "considers") only two obstacles close to the target sphere (dotted red lines, Fig. 6) plus two other obstacles standing close to the other spheres (dotted black lines, Fig. 6), resulting in four goal-related contexts plus all the combinations with the other two discovered obstacles for a total of "only" 16 relevant contexts, way less than 512 that C-Transfer has to consider. This is not only an advantage from a "storaging" perspective but also from that of the learning process, since while C-Transfer has 512 different policies to manage and train, Smart C-Bandit has only 16. However, this advantage is not enough to make Smart C-Bandit perform better than C-Transfer. Indeed, not being able to exploit the power of TL, every time a new relevant context is identified the system has to start training the new policy from scratch, thus losing time in reacquiring previously learned behaviors. This is clear from Fig. 7: in many cases, when the system identifies a new obstacle/context, there is a significant drop in the performance even when the agent is properly accomplishing the task. This limitation does not only lead to a slowness in learning but

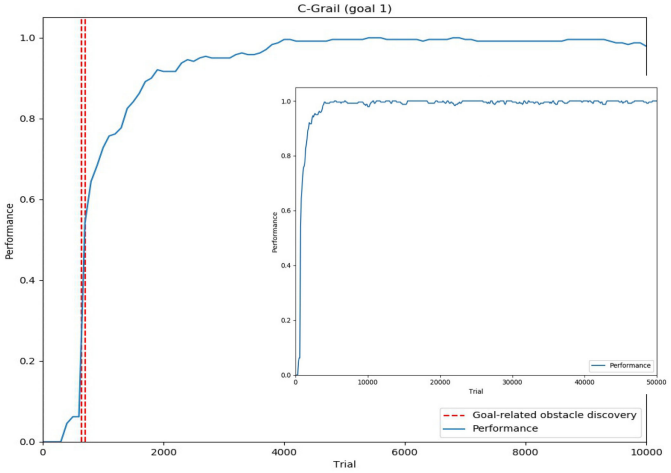

Fig. 8. C-GRAIL performance on goal 1, averaged over all the contexts. Small figure: entire simulation, 50 000 trials. Big figure: zoom on 10 000 trials and timing of the new relevant context identification.

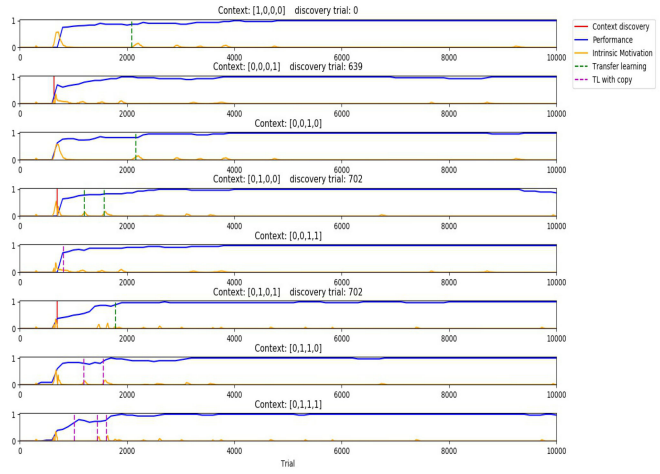

Fig. 9. C-GRAIL: analysis of context-related learning for goal 1 on the first 10 000 trials, with relevant context discovery, TL between relevant contexts, and IM signal.

also increases the possibility of causing a further proliferation of contexts: training new policies from scratch increases the likelihood that initially inefficient movements will lead the system to bump into obstacles that had already been avoided in previously learned trajectories.

If TL alone is not able to reduce learning time due to the large number of contexts generated by only nine obstacles, and if the smart reduction of contexts cannot cope with the problem of having to relearn skills from scratch, integrating the two mechanisms into one architecture results in a profitable solution to the autonomous learning of multiple context-dependent goals. Fig. 8 shows the performance of C-GRAIL on goal 1 averaged over different contexts (for a video, see https://youtu.be/e6Pt4uyH2XQ). The system reaches a 100% performance at trial ∼4000 and identifies two goal-related obstacles such as Smart C-Bandit but no other obstacles; thus, only four different contexts for this goal. Thanks to TL, the agent has no drop in performance when new relevant contexts are identified and a rapid learning of the four experts, and can thus easily cope with all the 512 real contexts in the environment. This guarantees fast learning, reflected by a rapid decrease in the IM signals related to goal 1 (Fig. 9), which allows the system to focus on (and learn) the other goals as reported in Fig. 3(d). Fast learning also helps the system to reduce the number of identified obstacles: the quicker the system learns, the less exploration and therefore, the lower the risk of hitting and discovering other obstacles.

VI. Discussion and Conclusion

In this work, we tackled the problem of the autonomous learning of multiple context-dependent goals. What we proposed is to combine the well-known practice of TL with a mechanism that guarantees the reduction of the number of contexts to handle, focusing only on those that are relevant for the goals. To this purpose, we presented C-GRAIL, an integrated robotic architecture for intrinsically motivated open-ended learning that enhances the previous GRAIL system [37] with mechanisms for autonomous learning of multiple context-dependent goals, smart context detection, and TL.

In particular, the results show that TL alone ensures the rapid sharing of acquired skills, but it cannot be sufficient in environments with multiple, possibly not relevant contexts. On the contrary, a system able to take into account only relevant conditions can significantly reduce the skills to learn. In the presented work, such a mechanism is able to identify eight relevant contexts out of 512, thus also increasing the advantages of TL. The experiments also showed how IMs and "artificial curiosity" can be useful not only for goal management but also to identify relevant contexts. Indeed, the surprise caused by an unexpected failure is a suitable trigger to bring the attention of the system to those environmental features that need to be taken into account to build new context-related policies. Notwithstanding these achievements, we are aware that the current operationalization of C-GRAIL has limitations that should be addressed in the future. This could be facilitated by the fact that C-GRAIL can be considered as a blueprint architecture whose specific implementation can be modified and improved with computationally more effective elements and additional mechanisms.

A relevant limitation is related to the perceptual components of the architecture. In particular, we now arrange the system perception in terms of low-level features used for motor control and high-level features relevant for context detection. Future work might enhance the system by endowing it with the capacity of autonomously extracting relevant high-level features, for example, using autoencoders to extract high-level meaningful features [60], as well as further strategies for input generalization [61].

A problem related to the previous point is that when the agent fails to obtain the expected success for a goal due to the changed context, the identification of new relevant features (obstacles) is facilitated by the fact that the obstacles are discrete, their encoding is given to the agent as a vector of high-level features, and that the robot identifies as relevant those features that are close to the hand when a hit happens. Future work might thus aim at allowing the agent to

autonomously guess which are the features that are actually relevant, for example, by using algorithms such as predictive coding [62] to identify unexpected features on which to possibly focus.

Finally, testing the enhanced features of C-GRAIL discussed above would require the use of scenarios and tasks going beyond those presented here. For example, here we considered only the learning of simple reach-and-touch actions performed on spherical objects; moreover, the environment contained only distinct regular obstacles with the same size and similar locations with respect to the target. A more complex challenge might require the pursuit of more complex goals, such as picking-and-placing of objects in desired locations in a continuous space, and several obstacles with variable size, shape, and spatial arrangement. These scenarios possibly require the execution of other goals (or subgoals) as a precondition for their achievement. M-GRAIL [53], another extension of GRAIL, was conceived exactly for learning interrelated goals and could be hybridized with C-GRAIL. Continuous spaces could be faced through the segmentation of relevant areas based on dissimilarity thresholds, as previously done in [57].

Notwithstanding these possible improvements, C-GRAIL represents an advancement with respect to previous systems for open-ended learning as it integrates for the first time two critical features needed for fully autonomous learning, namely, the capacity to actively consider contextual features only when they are relevant for a specific pursued goal, and the capacity to adapt to new relevant contexts through TL.

APPENDIX

This Appendix provides details not included in Section III-B.

Visual Input and Change Recognition: The visual input is an 80 × 60 binary (black and white) image resulting from the pixel-by-pixel subtraction of two consecutive frames provided by the camera of the robot. If there is at least one pixel activated a change has occurred, the map is normalized (norm equal to 1) and associated to a unit in the goal selector. Arms and scoops are set to blue color: when blue is present in one of the two consecutive images, we exclude change detection in correspondence to such pixels.

Contextual Features and Competence Threshold: Contextual features f^{ϕ} are encoded in a binary vector stating the presence (1) or absence (0) of each of the possible nine obstacles. New contextual features are added to goal-specific relevant contextual features vector rcf^g if a failure occurs and the current competence on goal g is above a *competence_threshold* (0.4).

Goal Selection and Value Assignment: The probability $p(g|\phi^g)$ of g to be selected given the goal-specific context ϕ^g is determined by a softmax selection rule

$$p(g|\phi^g) = \frac{\exp\left(\frac{Q(g|\phi^g)}{\tau}\right)}{\sum_{i=0}^{n}\exp\left(\frac{Q(n|\phi^n)}{\tau}\right)} \tag{8}$$

where $Q(g|\phi^g)$ is the value of g given ϕ^g, n and ϕ^n are all the currently known goals and their respective perceived relevant contexts, and τ is the softmax temperature set to 0.01. *Q-values* are updated through an EMA of the competence improvement intrinsic reinforcements $\Delta C^g_{\phi^g}$, with a smoothing factor γ set to 0.3

$$Q_{t+1}(g|\phi^g) = Q_t\left(g|\phi^g\right) + \gamma\left(\Delta C^g_{\phi^g} - Q_t\left(g|\phi^g\right)\right). \tag{9}$$

Intrinsic Motivations: CB-IM signal ΔC is calculated on the basis of a predictor evaluating the competence of the robot to achieve g given ϕ^g. In particular, $\Delta C^g_{\phi^g}$ at time t is the difference between two averages of competence predictions (CP), each one covering a period PT of 20 attempts to g in ϕ^g

$$\Delta C^g_{\phi^g} = \frac{\sum_{i=t-(PT-1)}^{t}|CP_i|}{PT} - \frac{\sum_{i=t-(2PT-1)}^{t-PT}|CPi|}{PT}. \tag{10}$$

Before covering the entire period ($PTx2$), we divide the current collection in two groups.

Expert Selector: Given a goal and a context, a softmax selection rule (8) with temperature set to 0.05 is used by the Expert Selector to determine the expert controlling the robot, based on the values updated through an EMA [(9), with a smoothing factor set to 0.3] of the matching signal generated for achieving the selected goal (1 for success, 0 otherwise).

Experts: The selected expert receives as input the angles of the four actuated joints of the arm encoded through Gaussian radial basis functions (RBF) with centers on the equally distributed vertexes of a 4-dimensional grid having 5 units per dimension

$$y_i = e^{-\sum_d\left(\frac{(c_d - c_{id})^2}{2\sigma_d^2}\right)} \tag{11}$$

where y_i is the activation of unit i, c_d is the input value of dimension d, c_{id} is the preferred value of unit i with respect to dimension d, and σ_d^2 is the width of the Gaussian along dimension d (widths are parameterized so that when an input is equidistant, along a dimension, to two contiguous units, the activation of both the units is 0.5). The output of the critic component (V) is a linear combination of the weighted sum of the input units

$$V = \sum_{i}^{N} y_i u_i + b_V \tag{12}$$

where u_i is the weight projecting from input unit i and b_V is the bias. The four outputs of the actor are determined by a logistic transfer function

$$o_j = \Phi\left(b_j + \sum_{i}^{N} u_{ji} y_i\right) \qquad \Phi(x) = \frac{1}{1+e^{-x}} \tag{13}$$

where b_j is the bias of output unit j, N is the number of input units, y_i is the activation of input unit i, and u_{ji} is the weight linking unit i to unit j. Motor commands o_j^n are generated by adding noise to the activation of the relative output o_j. Since the desired positions of the joints are modified progressively, the noise (n) added to the output of the actor is generated with a normal Gaussian distribution with average 0 and standard deviation (sd) 2.0, and passed through an EMA with a smoothing factor set to 0.08. Moreover, to manage the exploration/exploitation problem, we implemented an algorithm that lets the system autonomously regulate n. The sd of each expert

is dependent on a "noise-decrease parameter" (d) determined by an EMA (with smoothing factor set to 0.0005) of the success of the expert in achieving the associated goal-context couple (1 for success, 0 otherwise): the higher the competence, the lower the noise. The sd of the selected expert e at trial k (sd_k^e) is updated as follows:

$$sd_k^e = sd(1-d). \tag{14}$$

The actual motor commands are then generated as follows:

$$o_j^n = o_j + n \tag{15}$$

where the resulting commands are limited in [0; 1] and then remapped into a movement range specific for each joint determining the *variation* in their position. Experts are trained through the standard TD reinforcement learning algorithm [26], with a discount factor set to 0.99 and a positive reinforcement of 1 for goal achievement, 0 otherwise. To speed up the exploration process, we introduced negative rewards for hitting obstacles (-1) and when trials end without collisions (-0.5). The critic learning rate is set to 0.02, the one of the actor to 0.4.

Transfer Learning: The *learning threshold* to activate transfer learning is set to 0.4; the *transfer threshold* for the teaching expert is set to 0.7; the *transfer probability* is set to 0.5. During transfer learning, both the "teaching expert" (T_exp) and the "learning expert" (L_exp) receive the current actuated joints as input. The robot is controlled by the actor of T_exp and only the weights of L_exp are updated using the output of the actor and the critic of T_exp as "teaching input."

ACKNOWLEDGMENT

The authors thank Bruno Castro da Silva for his support and contribution in discussing topics relevant to this research.

REFERENCES

[1] A. S. Klyubin, D. Polani, and C. L. Nehaniv, "Keep your options open: An information-based driving principle for sensorimotor systems," *PloS One*, vol. 3, no. 12, 2008, Art. no. e4018.

[2] G. Martius, R. Der, and N. Ay, "Information driven self-organization of complex robotic behaviors," *PloS One*, vol. 8, no. 5, 2013, Art. no. e63400.

[3] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evol. Comput.*, vol. 19, no. 2, pp. 189–223, 2011.

[4] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017, pp. 3389–3396.

[5] J. Lones, M. Lewis, and L. Cañamero, "A hormone-driven epigenetic mechanism for adaptation in autonomous robots," *IEEE Trans. Cogn. Devel. Syst.*, vol. 10, no. 2, pp. 445–454, Jun. 2018.

[6] A. Cangelosi and M. Schlesinger, *Developmental Robotics: From Babies to Robots*. Cambridge, MA, USA: MIT Press, 2015.

[7] G. Baldassarre and M. Mirolli, *Intrinsically Motivated Learning in Natural and Artificial Systems*. Heidelberg, Germany: Springer, 2013.

[8] V. G. Santucci, P.-Y. Oudeyer, A. Barto, and G. Baldassarre, "Intrinsically motivated open-ended learning in autonomous robots," *Front. Neurorobot.*, vol. 13, p. 115, Jan. 2020.

[9] R. W. White, "Motivation reconsidered: The concept of competence," *Psychol. Rev.*, vol. 66, no. 5, p. 297, 1959.

[10] R. M. Ryan and E. L. Deci, "Intrinsic and extrinsic motivations: Classic definitions and new directions," *Contemp. Educ. Psychol.*, vol. 25, no. 1, pp. 54–67, 2000.

[11] M. G. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Proc. NIPS*, 2016, pp. 1479–1487.

[12] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "VIME: Variational information maximizing exploration," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 1117–1125.

[13] G. Schillaci, A. P. Villalpando, V. V. Hafner, P. Hanappe, D. Colliaux, and T. Wintz, "Intrinsic motivation and episodic memories for robot exploration of high-dimensional sensory spaces," *Adapt. Behav.*, vol. 29, no. 6, pp. 549–566, 2021.

[14] J. Schmidhuber, "Formal theory of creativity, fun, and intrinsic motivation (1990–2010)," *IEEE Trans. Auton. Mental Develop.*, vol. 2, no. 3, pp. 230–247, Sep. 2010.

[15] T. Hester and P. Stone, "Intrinsically motivated model learning for developing curious robots," *Artif. Intell.*, vol. 247, pp. 170–186, Jun. 2017.

[16] N. Chentanez, A. G. Barto, and S. P. Singh, "Intrinsically motivated reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 1281–1288.

[17] P.-Y. Oudeyer, A. Baranes, and F. Kaplan, "Intrinsically motivated learning of real-world sensorimotor skills with developmental constraints," in *Intrinsically Motivated Learning in Natural and Artificial Systems*. Heidelberg, Germany: Springer, 2013, pp. 303–365.

[18] J. A. Becerra, A. Romero, F. Bellas, and R. J. Duro, "Motivational engine and long-term memory coupling within a cognitive architecture for lifelong open-ended learning," *Neurocomputing*, vol. 452, pp. 341–354, Sep. 2021.

[19] S. Hart and R. Grupen, "Intrinsically motivated affordance discovery and modeling," in *Intrinsically Motivated Learning in Natural and Artificial Systems*. Heidelberg, Germany: Springer, 2013, pp. 279–300.

[20] A. Manoury, S. M. Nguyen, and C. Buche, "Hierarchical affordance discovery using intrinsic motivation," in *Proc. 7th Int. Conf. Human Agent Interact.*, 2019, pp. 186–193.

[21] K. Merrick, "Value systems for developmental cognitive robotics: A survey," *Cogn. Syst. Res.*, vol. 41, pp. 38–55, Mar. 2017.

[22] S. Forestier, Y. Mollard, and P.-Y. Oudeyer, "Intrinsically motivated goal exploration processes with automatic curriculum learning," 2017, *arXiv:1708.02190*.

[23] S. Blaes, M. V. Pogančić, J. Zhu, and G. Martius, "Control what you can: Intrinsically motivated task-planning agent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 12520–12531.

[24] S. M. Nguyen, A. Baranes, and P.-Y. Oudeyer, "Bootstrapping intrinsically motivated learning with human demonstrations," in *Proc. IEEE Int. Conf. Develop. Learn. (ICDL)*, vol. 2, 2011, pp. 1–8.

[25] N. Duminy, S. M. Nguyen, J. Zhu, D. Duhaut, and J. Kerdreux, "Intrinsically motivated open-ended multi-task learning using transfer learning to discover task hierarchy," *Appl. Sci.*, vol. 11, no. 3, p. 975, 2021.

[26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[27] R. F. Reinhart, "Autonomous exploration of motor skills by skill babbling," *Auton. Robots*, vol. 41, no. 7, pp. 1521–1537, 2017.

[28] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, Jul. 2018, pp. 1514–1523.

[29] S. Padakandla, K. Prabuchandran, and S. Bhatnagar, "Reinforcement learning algorithm for non-stationary environments," *Appl. Intell.*, vol. 50, no. 11, pp. 3590–3606, 2020.

[30] B. C. Da Silva, E. W. Basso, A. L. Bazzan, and P. M. Engel, "Dealing with non-stationary environments using context detection," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 217–224.

[31] A. Nagabandi, C. Finn, and S. Levine, "Deep online learning via meta-learning: Continual adaptation for model-based RL," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–15. [Online]. Available: https://openreview.net/forum?id=HyxAfnA5tm

[32] S. Abdelfattah, K. Merrick, and J. Hu, "Intrinsically motivated hierarchical policy learning in multi-objective Markov decision processes," *IEEE Trans. Cogn. Devel. Syst.*, vol. 13, no. 2, pp. 262–273, Jun. 2021.

[33] C. Geib *et al.*, "Object-action complexes: Grounded abstractions of sensory-motor processes," *Robot. Auton. Syst.*, vol. 59, no. 10, pp. 740–757, 2011.

[34] G. Baldassarre, W. Lord, G. Granato, and V. G. Santucci, "An embodied agent learning affordances with intrinsic motivations and solving extrinsic tasks with attention and one-step planning," *Front. Neurorobot.*, vol. 13, p. 45, Jul. 2019.

[35] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[36] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Dec. 2009.

[37] V. G. Santucci, G. Baldassarre, and M. Mirolli, "GRAIL: A goal-discovering robotic architecture for intrinsically-motivated learning," *IEEE Trans. Cogn. Devel. Syst.*, vol. 8, no. 3, pp. 214–231, Sep. 2016.
[38] B. C. Da Silva, G. Baldassarre, G. Konidaris, and A. Barto, "Learning parameterized motor skills on a humanoid robot," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2014, pp. 5239–5244.
[39] C. Colas, T. Karch, O. Sigaud, and P.-Y. Oudeyer, "Intrinsically motivated goal-conditioned reinforcement learning: A short survey," 2020, *arXiv:2012.09830*.
[40] E. Cartoni, F. Mannella, V. G. Santucci, J. Triesch, E. Rueckert, and G. Baldassarre, "Real-2019: Robot open-ended autonomous learning competition," in *Proc. Competition Demonstration Track*, 2020, pp. 142–152.
[41] C. Colas, P. Fournier, M. Chetouani, O. Sigaud, and P.-Y. Oudeyer, "Curious: Intrinsically motivated modular multi-goal reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1331–1340.
[42] S. Singh, R. L. Lewis, and A. G. Barto, "Where do rewards come from?" in *Proc. Annu. Conf. Cogn. Sci. Soc.*, 2009, pp. 2601–2606.
[43] V. G. Santucci, G. Baldassarre, and M. Mirolli, "Which is the best intrinsic motivation signal for learning multiple skills?" *Front. Neurorobot.*, vol. 7, p. 22, Nov. 2013.
[44] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robot. Auton. Syst.*, vol. 61, no. 1, pp. 49–73, 2013.
[45] A. Graves, M. G. Bellemare, J. Menick, R. Munos, and K. Kavukcuoglu, "Automated curriculum learning for neural networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1311–1320.
[46] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, "Teacher–student curriculum learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3732–3740, Sep. 2020.
[47] N. Levine, K. Crammer, and S. Mannor, "Rotting bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3074–3083.
[48] E. Uchibe, "Cooperative and competitive reinforcement and imitation learning for a mixture of heterogeneous learning modules," *Front. Neurorobot.*, vol. 12, p. 61, Sep. 2018.
[49] W. Of Ockham, *Quaestiones et Decisiones in Quattuor Libros Sententiarum Petri Lombardi: Centilogium Theologicum*. Lyon, France: Johannes Trechsel, 1495.
[50] A. Barto, M. Mirolli, and G. Baldassarre, "Novelty or surprise?" *Front. Psychol.*, vol. 4, p. 907, Dec. 2013.
[51] N. P. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IROS*, vol. 4, 2004, pp. 2149–2154.
[52] E. M. Hoffman *et al.*, "Yarp based plugins for Gazebo simulator," in *Proc. Int. Workshop Model. Simulat. Auton. Syst.*, 2014, pp. 333–346.
[53] V. G. Santucci, G. Baldassarre, and E. Cartoni, "Autonomous reinforcement learning of multiple interrelated tasks," in *Proc. Joint IEEE 9th Int. Conf. Devel. Learn. Epigenetic Robot. (ICDL-EpiRob)*, Aug. 2019, pp. 221–227.
[54] L. Jacquey, G. Baldassarre, V. G. Santucci, and J. K. O'Regan, "Sensorimotor contingencies as a key drive of development: From babies to robots," *Front. Neurorobot.*, vol. 13, p. 98, Dec. 2019.
[55] V. G. Santucci, G. Baldassarre, and M. Mirolli, "Autonomous selection of the 'what' and the 'how' of learning: An intrinsically motivated system tested with a two armed robot," in *Proc. 4th Int. Conf. Develop. Learn. Epigenetic Robot.*, 2014, pp. 434–439.
[56] K. Doya, "Reinforcement learning in continuous time and space," *Neural Comput.*, vol. 12, no. 1, pp. 219–245, Jan. 2000.
[57] K. Seepanomwan, V. G. Santucci, and G. Baldassarre, "Intrinsically motivated discovered outcomes boost user's goals achievement in a humanoid robot," in *Proc. Joint IEEE Int. Conf. Develop. Learn. Epigenetic Robot. (ICDL-EpiRob)*, 2017, pp. 178–183.
[58] A. Romero, F. Bellas, J. A. Becerra, and R. J. Duro, "Motivation as a tool for designing lifelong learning robots," *Integr. Comput. Aided Eng.*, vol. 27, no. 4, pp. 353–372, 2020.
[59] A. Oddi *et al.*, "Integrating open-ended learning in the sense-plan-act robot control paradigm," in *Proc. 24th Eur. Conf. Artif. Intell. (ECAI)*, 2020, pp. 2417–2424.
[60] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, *arXiv:1312.6114*.
[61] C. Wilmot, G. Baldassarre, and J. Triesch, "Learning abstract representations through lossy compression of multi-modal signals," *IEEE Trans. Cogn. Devel. Syst.*, early access, Aug. 30, 2021, doi: 10.1109/TCDS.2021.3108478.
[62] M. Choi, T. Matsumoto, M. Jung, and J. Tani, "Generating goal-directed visuomotor plans based on learning using a predictive coding-type deep visuomotor recurrent neural network model," 2018, *arXiv:1803.02578*.

Vieri Giuliano Santucci received the bachelor's degree in philosophy from the University of Pisa, Pisa, Italy, in 2006, the master's degree in philosophy from the University of Rome "La Sapienza," Rome, Italy, in 2009, and the Ph.D. degree in computer science from the University of Plymouth, Plymouth, U.K., in 2016.

Since 2010, he has been with the Institute of Cognitive Science and Technologies, Italian National Council of Research, Rome, where he collaborated both as a Ph.D. and Post-Doc, and where he has been a Researcher with permanent position since 2018. His research mainly focuses on developmental robotics and machine learning, and in particular on the study of intrinsically motivated open-ended learning systems and motivational signals. He is also interested in topics related to cognitive sciences and computational neuroscience.

Davide Montella received the bachelor's degree in computer science from the University of Rome "La Sapienza," Rome, Italy, in 2019.

He collaborated with the Institute of Cognitive Sciences and Technologies, Italian National Research Council, Rome, from 2019 to 2021, where he contributed to the development of robotic architectures based on intrinsically motivated open-ended learning. He is a Science and Technology Enthusiast and believes that these can be used to improve everyone's quality of life. His life purpose is to develop technologies and contribute to the betterment of the world through them.

Gianluca Baldassarre received the bachelor's and master's degrees in economics in 1998 and a Specialization Course in "Cognitive Psychology and Neural Networks" in 1999 from the Sapienza University of Rome, Rome, Italy, and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2003.

He later joined the Institute of Cognitive Sciences and Technologies (ISTC), Italian National Research Council, Rome, where he was a Postdoctoral Fellow, a Researcher since 2006, the Director of Research since 2021, the Coordinator of the Research Group ISTC-CNR "Laboratory of Embodied Natural and Artificial Intelligence" since 2016 and the EU Projects "IM-CLeVeR—Intrinsically Motivated Cumulative Learning Versatile Robots" from 2009 to 2013, and "GOAL-Robots—Goal-Based Open-Ended Autonomous Learning Robots" from 2016 to 2021, and the President of the "Advanced School in AI" since 2018. His research interests span open-ended learning of sensorimotor skills, driven by extrinsic and intrinsic motivations, in animals, humans, and robots.