DREAM Architecture: a Developmental Approach to Open-Ended Learning in Robotics

Stephane Doncieux, Nicolas Bredeche, Léni Le Goff, Benoît Girard, Alexandre Coninx, Olivier Sigaud, Mehdi Khamassi, Natalia Díaz-Rodríguez, David Filliat, Timothy Hospedales, et al.

DREAM Architecture: a Developmental Approach to Open-Ended Learning in Robotics

Stephane Doncieux[1]*, Nicolas Bredeche[1], Leni K. Le Goff[1], Benoît Girard[1], Alexandre Coninx[1], Olivier Sigaud[1], Mehdi Khamassi[1], Natalia Díaz-Rodríguez[2], David Filliat[2], Timothy Hospedales[3], A.E. Eiben[4], Richard Duro[5]

Abstract

Robots are still limited to controlled conditions, that the robot designer knows with enough details to endow the robot with the appropriate models or behaviors. Learning algorithms add some flexibility with the ability to discover the appropriate behavior given either some demonstrations or a reward to guide its exploration with a reinforcement learning algorithm. Reinforcement learning algorithms rely on the definition of state and action spaces that define reachable behaviors. Their adaptation capability critically depends on the representations of these spaces: small and discrete spaces result in fast learning while large and continuous spaces are challenging and either require a long training period or prevent the robot from converging to an appropriate behavior. Beside the operational cycle of policy execution and the learning cycle, which works at a slower time scale to acquire new policies, we introduce the *redescription cycle*, a third cycle working at an even slower time scale to generate or adapt the required representations to the robot, its environment and the task. We introduce the challenges raised by this cycle and we present DREAM (Deferred Restructuring of Experience in Autonomous Machines), a developmental cognitive architecture to bootstrap this redescription process stage by stage, build new state representations with appropriate motivations, and transfer the acquired knowledge across domains or tasks or even across robots. We describe results obtained so far with this approach and end up with a discussion of the questions it raises in Neuroscience.

Keywords

Open-ended learning – developmental robotics – representational redescription – intrinsic motivations – transfer learning – state representation learning – motor skill acquisition

[1] *Sorbonne Université, CNRS, ISIR, Paris, France*
[2] *U2IS, INRIA Flowers, ENSTA, Institut Polytechnique Paris*
[3] *University of Edinburgh*
[4] *Vrije Universiteit Amsterdam*
[5] *GII, CITIC, Universidade da Coruña*
***Corresponding author**: stephane.doncieux@sorbonne-universite.fr

1. Introduction

What do we miss to build a versatile robot, able to solve tasks that are not pre-programmed, but that could be given on-the-fly and in an unprepared environment? Robots with these capabilities would pave the way to many applications, from service robotics to space exploration. It would also reduce the need for deep analyses of a robot's future environments while designing it.

Floor cleaning robots are the only autonomous robots that have been able to solve real-world problems out of the lab and have proved their efficiency on the market where they have been sold by millions. The variability of our everyday environments, that are unknown to the robot designers, is handled by a carefully tuned behavior-based architecture dedicated to this single, well-defined task [1]. But even in this case, and after years of improvement involving tests and upgrades by skillful engineers, a significant number of users finally stop using them because their behavior is not adapted to their home [2]. Dealing with variable environments is thus a challenge, even in this context, and users call for more adaptivity [2].

The adaptivity we will focus on here is the ability to solve a task when the appropriate behavior is not known beforehand. The goal-driven exploration strategy to learn a policy will be modeled as a reinforcement learning process. Reinforcement learning relies on a Markov Decision Process (MDP) that includes a state space, an action space, a transition function and a reward function [3]. Although widely used in machine learning, reinforcement learning is notoriously hard to apply in robotics as the definition of the relevant MDP critically impacts the learning performance and needs to be adapted to the task [4]. This raises an issue: if the task is not known by the robot designer, it will not be possible to define an appropriate MDP at robot design time. A possibility is to rely on end-to-end learning [5], but even in this case, some careful preparation is required. Besides, it would be interesting to

avoid starting from scratch for each new task and to transfer the previously acquired knowledge to a new context [6, 7, 8, 9, 10].

Reusing past experience is particularly important in robotics where the sampling cost is high: testing a policy may damage the robot if it does not respect safety constraints [11] and even if it is safe, it will increase the wear and tear of the robot. Many different approaches have been proposed to reuse the already acquired experience [12, 13, 14, 15]. Transfer learning is also particularly interesting to learn in simulation before transferring to reality, as it drastically reduces the number of required samples on the real robot [16, 17, 18, 19, 20]. Specific knowledge representations can clearly facilitate the transfer of acquired knowledge [21], thus the choice of an appropriate representation is also important for this question.

Human beings do not use a single representation to solve the problems they are facing. Their ability to build new representations even seems to be a critical factor of their versatility [22, 23, 24, 25]. The features of an MDP representation constrains the kind of learning algorithms that can be used: a small and discrete set of actions and states facilitates an exhaustive exploration to discover the most relevant policy [3], while a large dimension and continuous state and action spaces raise exploration issues that can be solved, for instance, by restraining exploration to the neighborhood of an expert demonstration, if available [26, 27], by endowing the agent with intrinsic motivation mechanisms [28, 29] or by combining fast and slow learning [30, 31, 32, 10].

What if a robot could switch between different representations and build new ones on-the-fly? An adapted representation would allow the robot to (1) understand a task, by identifying the target and associating it with a state space that it can control or learn to control and (2) search for a solution. When the robot knows little about the environment and the task, it could use end-to-end strategies and switch to faster decision or learning processes based on adapted representations. Following the framework introduced in [33], it is assumed here that a single state and action space cannot cover all the tasks the robot may be confronted with. Therefore we go beyond a single task resolution and consider the acquisition of an appropriate representation as a challenge to be explicitly addressed. Besides, we consider the acquisition of new knowledge representations as a challenge per se, that may require specific processes that are not necessarily task-oriented. As human infants, the proposed approach needs to face the challenges of understanding the robot's environment and its own capabilities [34, 35, 36, 37].

In this context, the robot behavior can be described by three loops: (1) Operational, (2) Learning and (3) Redescription (Figure 1). The operational loop corresponds to the execution of a known policy. The learning loop is in charge of acquiring new and appropriate policies from a known representation of states and actions. The redescription loop corresponds to the acquisition of a (more) adapted knowledge representation. This loop is called redescription to emphasize the iterative nature of the process [36]: the knowledge representation acquisition process necessarily starts with a

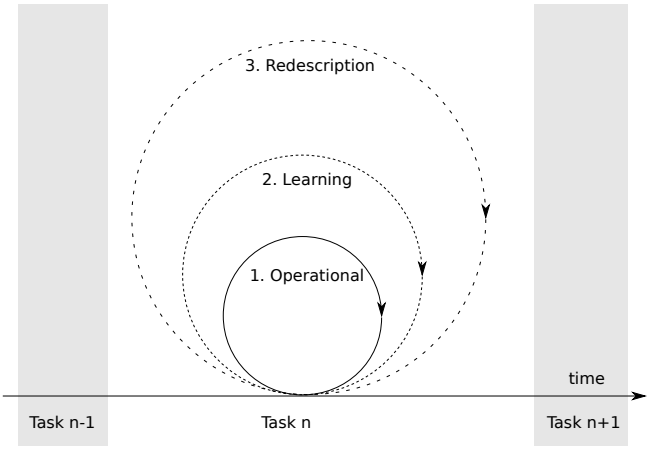


Figure 1. The three required loops for a robot to be endowed with the ability to adapt to new environments: 1. Operational: the robot applies a known policy, i.e. a mapping from states to actions, 2. Learning: the robot acquires a new policy, 3. Redescription: the robot discovers new states and action spaces.

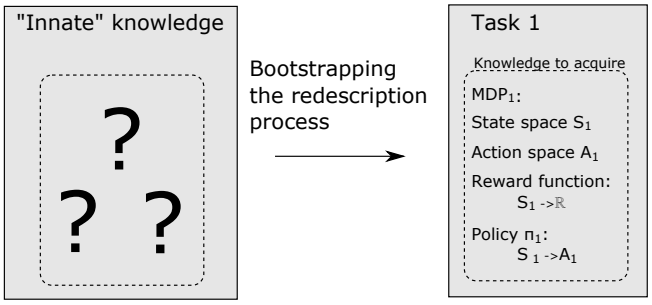


Figure 2. The bootstrap problem: how to generate a first Markov Decision Process when little is known about the task and the domain?

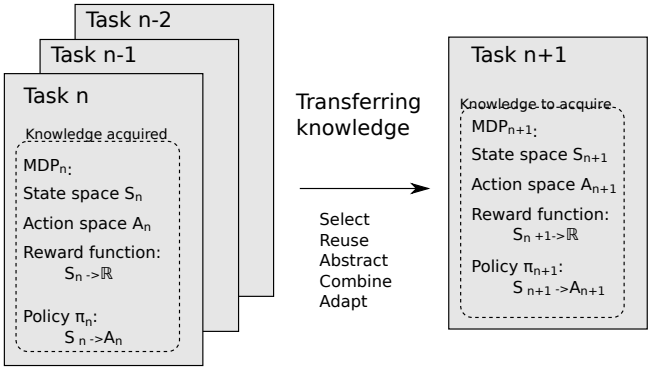


Figure 3. The transfer learning problem: how to build a new MDP from a set of known MDPs?

representation[1] and aims at building a new one. It is thus a transformation more than a creation.

The focus of this work is on the outer redescription loop whose goal can be described as follows: *How to acquire the knowledge required to learn the underlying (state, action) representation of policies able to solve the tasks the robot is facing, when those tasks are not known to the robot designer?* This question raises three different challenges: how to bootstrap the process and build the first state and action spaces when little is known about the task and the domain (Figure 2)? How to consolidate the acquired knowledge to make generated policies more robust? and then how to transfer acquired representations to a new task (Figure 3)?

The article begins with a discussion of the challenges raised by the adaptive capability we are looking for and that we call "open-ended learning", and the representational redescription it implies. The following section presents an overview of DREAM architecture (Deferred Restructuring of Experience in Autonomous Machines), the proposed approach to deal with these challenges developed during the DREAM European project[2]. The next sections introduce in more details how we have dealt with four of the challenges raised by representational redescription: bootstrapping of the process, state representation acquisition, consolidation of acquired knowledge and knowledge transfer. Next, a related work section follows before a discussion mainly oriented towards the link of this work with Neuroscience.

2. Open-ended learning

2.1 A definition of open-ended learning

In this work, open-ended learning is an adaptation ability with two major features:

1. the system can learn a task when both the task and the domain are unknown to the system designer,
2. the system acquires experience along time and can transfer knowledge from a learning session to another.

The proposed definition is related to two concepts: lifelong learning and open-ended evolution.

Lifelong learning [38, 39], also called never ending learning [40] or continual learning [41] consists in going beyond a single learning session and considers that the robot may be faced with different tasks in different environments. The goal is to avoid to start from scratch for each task and exploit acquired knowledge when considering a new task [7, 42] while avoiding catastrophic forgetting. In Thrun and Mitchell's original definition, the state and actions spaces are common between the different tasks and known beforehand. Open-ended learning proposes to go beyond this view, considering that using a single action space and a single state space is a strong limitation to the adaptive ability of the system.

[1] The lowest possible level is the raw sensori-motor flow, which is already a representation.

[2] http://dream.isir.upmc.fr/

Open-ended evolution is a major feature of life [43]. It is described as the ability of nature to continuously generate novel [44, 45] and adapted [43] lifeforms. Open-ended evolution considers large timescales and how new lifeforms can emerge as a result of the dynamics of an evolutionary process. It does not consider a single individual, but a species or even a whole ecosystem. We propose to define open-ended learning similarly, but with a focus on a single individual. It could thus be defined as the *ability to continuously generate novel and adapted behaviors*. *Novel* suggests the ability to explore and find new behaviors while *adapted* suggests that these behaviors fulfill a goal. This association between novelty and adaptation can also be called creativity [46]. Another definition of open-ended learning could thus be the *ability to continuously generate creative behaviors*.

2.2 Goals and challenges

A robot with an open-ended learning ability is expected to solve all the tasks it is facing without the need for its designer to provide it with appropriate state and action spaces [33]. It implies that the system is not built to solve a single task, but needs to solve multiple tasks in a life-long learning scenario. We will thus make the following assumptions: (1) the robot will be confronted to n different tasks, with $n > 1$ and (2) the robot may be confronted several times to the same task. In this context, the representational redescription process aims either at making the robot able to solve a task that was previously unsolvable (or at least unsolved) or at making its resolution more efficient when it encounters it again. The goals of the redescription processes can thus be described as follows:

- Bootstrapping task resolution: solving a previously unsolved task without task-specific state and action spaces;
- Improving over experience: increasing efficiency, speed and accuracy of solving a particular task;
- Generalizing through the transfer of knowledge. Using already acquired representations to get more *robustness* and *abstraction*;
- Changing the learning or decision process: building the representations required by a different, and more efficient learning and decision process in order to move towards zero-shot learning to increase *robustness* and *abstraction*.

In this description, *robustness* is defined as the ability to address the same task, but in a different domain, and *abstraction* as the ability to rely on the knowledge acquired while solving a task to address another one.

These goals raise different challenges for representational redescription processes. Some are shared with learning algorithms challenges. Dealing with sparse rewards is an example: from a learning perspective, the challenge is to find a learning algorithm with an appropriate exploration strategy and from

the representational redescription perspective, the challenge is to find state and action spaces that increase the probability to succeed by discarding irrelevant state dimensions, for instance, or by restraining the actions to those leading to success. The corresponding challenge can thus be faced either by adapting the learning process or by finding an appropriate representation.

Some other challenges are specific to representational redescription. In a reinforcement learning scenario, the state and action spaces are supposed to be well chosen by the system designer. Finding them for a robotics setup is notoriously hard [4] and if they are not well designed, it is expected that the system will not be able to learn an efficient policy and the fault will be on the system designer. In an open-ended learning setup, it cannot be assumed that relevant state and action spaces are initially available and what makes them relevant needs to be defined. A state space is useless if it does not provide the system with the information it needs to decide what action is to be performed. It is also required to interpret an observed reward. In an MDP, the reward function associates a value to a state[3]. It means that the system designer determines what reward value results from the system action, *but also to what state this value is associated*. In a representational redescription loop, *understanding* an observed reward value, i.e. finding the state space that best explains the observation, is a challenge per se.

Finally, [33] have identified eight challenges for representational redescription. They can be split into two groups: those related to solving a single task and those related to solving multiple tasks:

- Single task challenges:
    1. Interpreting observed reward: building (or selecting) a state space that makes observed reward predictable and reachable (with an appropriate policy);
    2. Skill acquisition: building the actions to control the state space;
    3. Simultaneous acquisition of state and action spaces as well as policies;
    4. Dealing with sparse rewards, in particular when bootstrapping the redescription process;
- Multi-task challenges:
    1. Detecting task change;
    2. Ordering knowledge acquisition and task resolution;
    3. Identifying the available knowledge to build a new MDP;
    4. Transferring acquired knowledge.

[3] The reward function can also be defined on different spaces, for instance on a (state, action) tuple.

3. Overview of the proposed approach

We now present the DREAM approach to deal with some of the challenges identified in the previous section. The approach is focused on the acquisition of knowledge through interactions of the robot with its environment and follows a stage-by-stage developmental process, where some stages rely on an evolutionary approach. It is thus an *Evolutionary Developmental Intelligence* approach to Artificial General Intelligence [47]. This section describes its main features and the following sections describe the implementation done so far and the results we have obtained.

3.1 Asymptotically end-to-end

One of the main limitations of robotics that has motivated this work is the lack of flexibility of a system limited by a single predefined representation. On a single task, carefully designing the state space, the action space and choosing an appropriate policy representation may lead to impressive results [4], but changing the task or the domain requires a new design phase and thus reduces the robot adaptivity. To maximize the robot versatility, it should be able to rely on the lowest possible level, for both the sensory and motor information: its learning process should be able to exploit the raw sensorimotor data. These approaches are named end-to-end [5] to highlight this capacity to start from the very first data entering the system and generate the data expected from its motors. Any intermediate representation may make learning and decision easier, but it is defined with some a priori in mind that may fit well to some tasks but not to others.

For the perception part, and with a focus on vision, the a priori may be on the kind of relevant information: is it static information (shape, color) or dynamic information (motion)? Does it involve large areas (walls), or small ones (pens)? Does it have a homogeneous texture, or is it made up with parts having different features? Are there ”objects”? And if it is the case, are they solid or deformable? Many other questions of this kind can be raised that will influence the perception model. And the robot won't be able to deal with a new situation that requires some perceptions that have not been covered by the models implemented in the system.

It is the same for the action part: what is important in the robot motion? Is it a question of position control? Velocity control? Torque control? Is it open-loop, closed-loop? If closed loop, what information needs to be taken into account to adapt robot trajectory? As for the perception part, any choice made at this point will limit the final adaptivity of the robot.

End-to-end approaches require to use a learning method that can deal with high dimensions, both as input and as output. Deep learning is the only approach so far that has been able to deal with end-to-end control [5, 19, 48]. Neural networks can deal with these large spaces but at a condition: a large enough training database must exist. It raises a critical bootstrap challenge: how to collect enough *meaningful* data to train the system? A set of random motions will likely not

be appropriate. An arm robot randomly moving, for instance, will only very rarely interact with objects [49]. There is then little chance that the features extracted by the deep neural networks describe them with enough accuracy, thus impeding the convergence of any learning or decision process on an object interaction task.

Several approaches have been used to generate the required data. Some rely on a simulation [19], but require the 3D structure of the environment. Others rely on demonstrations [48], but providing such demonstrations is not straightforward in an open-ended learning scenario. Other approaches reduce the number of required samples by carefully defining the cost function and adapting it to the task with a fitting phase that require human intervention [50]. All these approaches show that end-to-end learning is possible, but also highlight the challenge of acquiring relevant data in an open-ended learning scenario.

The necessary information, may it be demonstrations, the 3D structure of the environment or dedicated cost functions, could actually be acquired in a preliminary stage. We propose to add some processes that rely on predefined representations in order to bootstrap the system and acquire these data. The difference with other approaches like options [51], is that these representations are not a basis on which the whole system is built, as new and independent representations relying on the raw sensori-motor flow can be acquired. After a while, predefined representations may not be required anymore. This is why we have called this feature *asymptotically end-to-end*: the system starts with predefined representations and once it has acquired enough experience, it can start building, from the raw sensori-motor flow, new representations that future learning and decision processes can rely on.

3.2 Focus on representational redescription

The end-to-end approaches evoked so far rely on a single neural network architecture that goes directly from the raw sensors to the raw effectors without any intermediate step. This is an advantage as it reduces the engineering effort related to the definition of the corresponding architecture and it reduces the biases due to the designer choices.

We have made a different choice and we put the focus on the internal representations that are built by the system. Instead of considering them as an internal and somewhat hidden information, that is a by-product of the end-to-end learning process, we explicitly look for those representations, in the form of MDPs. The goal is to build algorithms that create such representations with the features required by the available learning or decision algorithms (is the representation continuous or discrete, in large or small dimensions, etc.). Aside from enabling the use of existing approaches [52, 53], the goal is also to make the system more transparent: an analysis of these representations directly tells what the robot perceives and what it can or cannot do. Another advantage is that it allows us to decompose the problem and define processes focused on state space acquisition and others on action space acquisition. It also separates the open-ended learning process into two different phases (Figure 4): knowledge acquisition (building new representations, but also acquiring the experience required by this process) and knowledge exploitation (task resolution exploiting the representations found so far).

3.3 Stage by stage modular approach

As long as a single process cannot fulfill all the requirements of open-ended learning, it is interesting to decompose the process and clearly identify the inputs and outputs of each part in order to develop them in parallel. This basic software engineering consideration has lead us to decompose the core of our approach, i.e. representational redescription, as a set of modules, each having a clear input and a clear output. The inputs are the required knowledge for the module to be used and the output is the knowledge it builds. Each module can thus be connected to other modules, through a knowledge producer and customer link, resulting in a graph of dependencies. Under this view, an open-ended learning process can be described as a graph of interconnected modules with the first modules that require limited knowledge about the task and the environment, and the last modules that result in task-specific representations (e.g. an MDP) that a learning or decision process can exploit to solve the current task. A cognitive architecture can then select the modules to activate at a given instant in time given their constraints and what the system currently aims at. The implementation described later is a first proof-of-concept that does not include this latter module selection part.

This approach has another consequence: it makes it possible to rely on any kind of learning algorithm, as long as it is possible to build a module or a chain of modules that builds the required knowledge.

Figure 4 illustrates this modular approach. Figure 5 shows a single module to highlight its features. To be end-to-end, the chain building the MDP requires to have at least one module directly connected to the raw sensor flow and at least one module (that may be the same) directly connected to the motor flow. These modules need to be in charge of building resp. the perception part (state space) and the motor part (action space) for learning and decision processes.

3.4 Alternation between awake and dreaming processes

Most animals alternate between awake and sleeping phases. While the awake phases are clearly important for the animal to survive, the role of the sleeping phases has been revealed only recently, at least concerning its impact on cognition. These phases are notably related to memory reprocessing [54]. They may involve the replay of past events to consolidate learning [55, 56] or the exploration of new problem solving strategies [57, 58, 59]. We think that this distinction is also important in robotics and, as shown in Figure 5, we propose to highlight this module category. These two kinds of modules will not impose the same constraints. Awake modules require interactions with the real world. They need to control the robot and

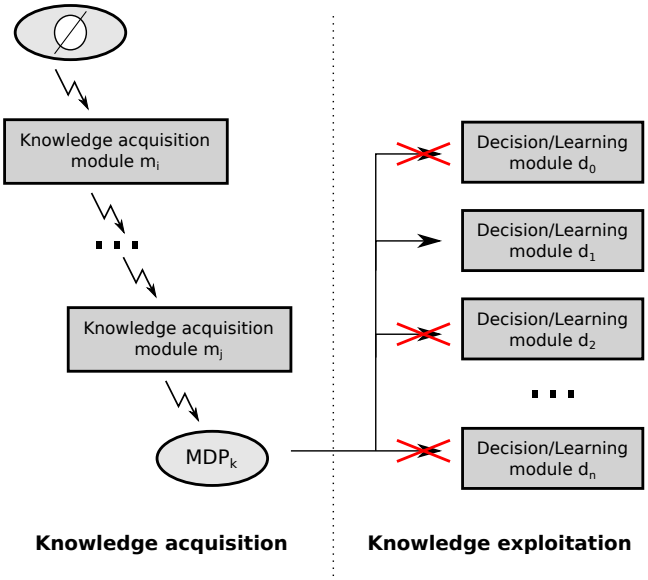


Figure 4. Example of knowledge acquisition chain going from no knowledge about a task k ($\emptyset$) to the design of a dedicated MDP (MDP_k). This MDP is compatible with the decision or learning process d_1, that can exploit it to solve the task.

may produce damages or at least increase the robot's wear and tear. Each interaction has then a high cost. "Dreaming" processes do not create such constraints. They may correspond to data analysis or exploration through the means of a model of the world and their sampling cost is then significantly lower. They should then be preferred.

3.5 Development

The knowledge acquisition chain can be split up into different stages, each stage resulting in an MDP that can, to some extent, solve a task. By reference to biology and child development, we propose to call this approach *development*, as we aim to reproduce some of the functions of human development. It highlights that the proposed open-ended learning process does not homogeneously drive the robot from its naive performance at startup to its acquired expertise after enough learning and practice. As in Piaget's models of development [60], the proposed open-ended learning follows a succession of stages that have their own features and associated modules. Each stage corresponds to one or several knowledge acquisition modules and can be described by a particular goal, stage n being necessary to execute stage $n+1$ (Figure 6). As in the overlapping wave theory [61], stage n is not supposed to be strictly stopped before stage $n+1$ is activated. Stage n can be reactivated after stage $n+1$ has been started. Anyways, this point goes beyond the scope of this article.

It should be noted that we do not aim at building a model of child development, but that our goal is similar for robots to what psychologists and neuroscientists attribute to child development. Put differently, Psychology and Neuroscience may be a source of inspiration, but not a constraint and elements of the current implementation do not systematically have a Neuroscience counterpart. Building such a system can anyway lead to new insights in the neuroscientific study of related processes. This point is further discussed in the Section 9.

The following sections describe the current implementation of the proposed approach. Figure 6 puts each section in the perspective of the whole proposed developmental scheme.

4. Building state representations

State Representation Learning (SRL) is the process of learning, without explicit supervision, a representation extracted from the observations that is adapted to support policy learning for a robot on a particular task or set of tasks. States are the basis of MDPs. They contain the required information to make a decision. Making the right decision to reach a goal implies some exploration of this state space that consequently needs to be low-dimensional for the planning or learning to be efficient. At the same time, the acquisition of a new state space requires to generate observations that cover what the robot may experience while solving a task. The design of a state space is then a chicken-and-egg problem as a policy is required to generate observations to be used later on to generate a new and relevant state space that can be used to learn policies. This problem has been tackled here with random policies on a simple button pushing task and the next section shows how more complex policies could be generated to bootstrap the generation of state spaces for more complex tasks.

Our state-of-the-art survey [53] analyzes existing SRL strategies in robotics control that exploit 4 main learning objectives: reconstructing the observations, learning a forward model, learning an inverse model, or exploiting high-level prior knowledge. Methods were also proposed to exploit several of these objectives simultaneously. Furthermore, we developed and open sourced[4] the S-RL Toolbox [62] containing baseline algorithms, data generating environments, metrics and visualization tools for assessing SRL methods.

We propose a new approach to SRL for goal-based robotics tasks that consists in learning a state representation that is split into several parts where each part optimizes a fraction of the objectives. In order to encode both target and robot positions, auto-encoders, reward and inverse model losses are used:

- *Inverse model:* One important aspect to encode for reinforcement learning is the state of the controlled agent. In the context of goal-based robotics tasks, it could correspond to the position of different parts of the robot, or to the position of the tip of a tool hold by the robot. A simple method consists in using an *inverse dynamics objective*: given the current s_t and next state s_{t+1}, the task is to predict the taken action a_t. The type of dynamics learned is constrained by the network architecture.

[4] https://github.com/araffin/robotics-rl-srl

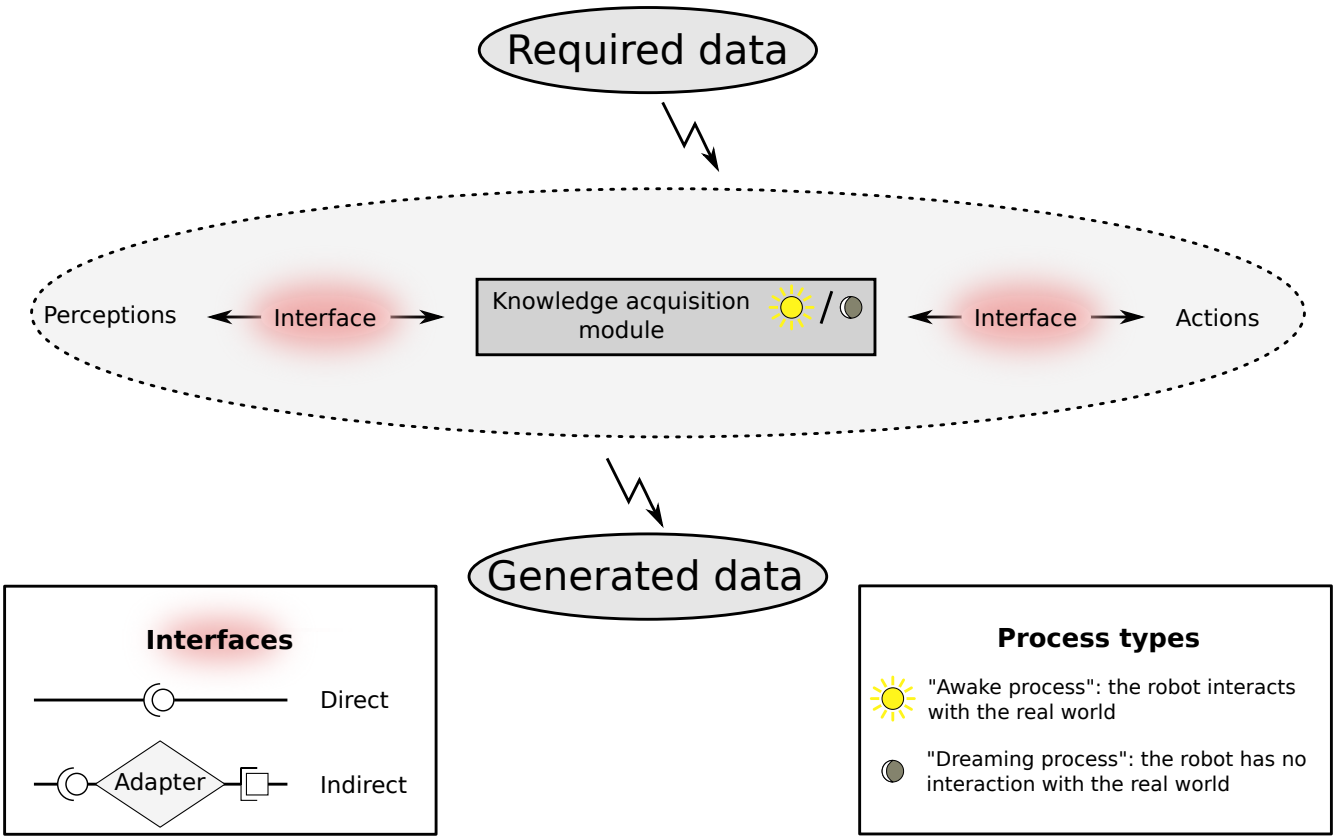


Figure 5. Zoom in on a developmental module. A developmental module has required data and generates data. It can directly use the raw sensorimotor flow or it may require an adapter (a perception model or a motor primitive, for instance). The connections between the module and the perceptions and actions are in both directions as some modules read perception values and generate actions, but others may generate new perceptions (data augmentation methods) or read raw actions (action restructuring modules). Finally, a module may require the robot to perform new interaction with the environment (”awake process”) or not (”dreaming process”). This distinction is important as ”awake modules” create specific constraints (they need access to the robot, they may damage the robot or the environment, and they have a significant cost in terms of time and mechanical wear and tear). Dreaming processes can be executed in parallel. They typically rely on previously acquired sensori-motor data, may it be directly, or through models learned or tuned out of these data.

For instance, using a linear model imposes linear dynamics. The learned state representation encodes only controllable elements of the environment. Here, the robot is part of them. However, the features extracted by an inverse model are not always *sufficient*: in our case, they do not encode the position of the target since the agent cannot act on it.

- *Auto-encoder:* The second important aspect is the goal position. Based on their reconstruction objective, auto-encoders compress all information in their latent space, but they tend to encode only aspects of the environment that are salient in the input [63]. This means they are not task-specific: relevant elements for a task can be ignored and distractors (unnecessary information) can be encoded into the state representation. In our case however, among other information, they will encode the goal position. Therefore, they usually need more dimensions than apparently required to encode a scene (e.g. in our experiments, it requires more than 10 dimensions to encode properly a 2D goal position).
- *Reward prediction:* The objective of a reward prediction module leads to state representations that are specialized in a task, thus improving the representation of the goal position in our case. Note that without the complementary learning objectives, predicting reward would only produce a classifier detecting when the robot is at the goal, thus not providing any particular structure or disentanglement to the state space.

Combining these objectives into a single loss function on the latent space can lead to features that are *sufficient* to solve the task. However, these objectives are partially contradictory and stacking partial state representations will therefore favor *disentanglement* and prevent opposed objectives from cancelling out, thus allowing a more stable optimization and

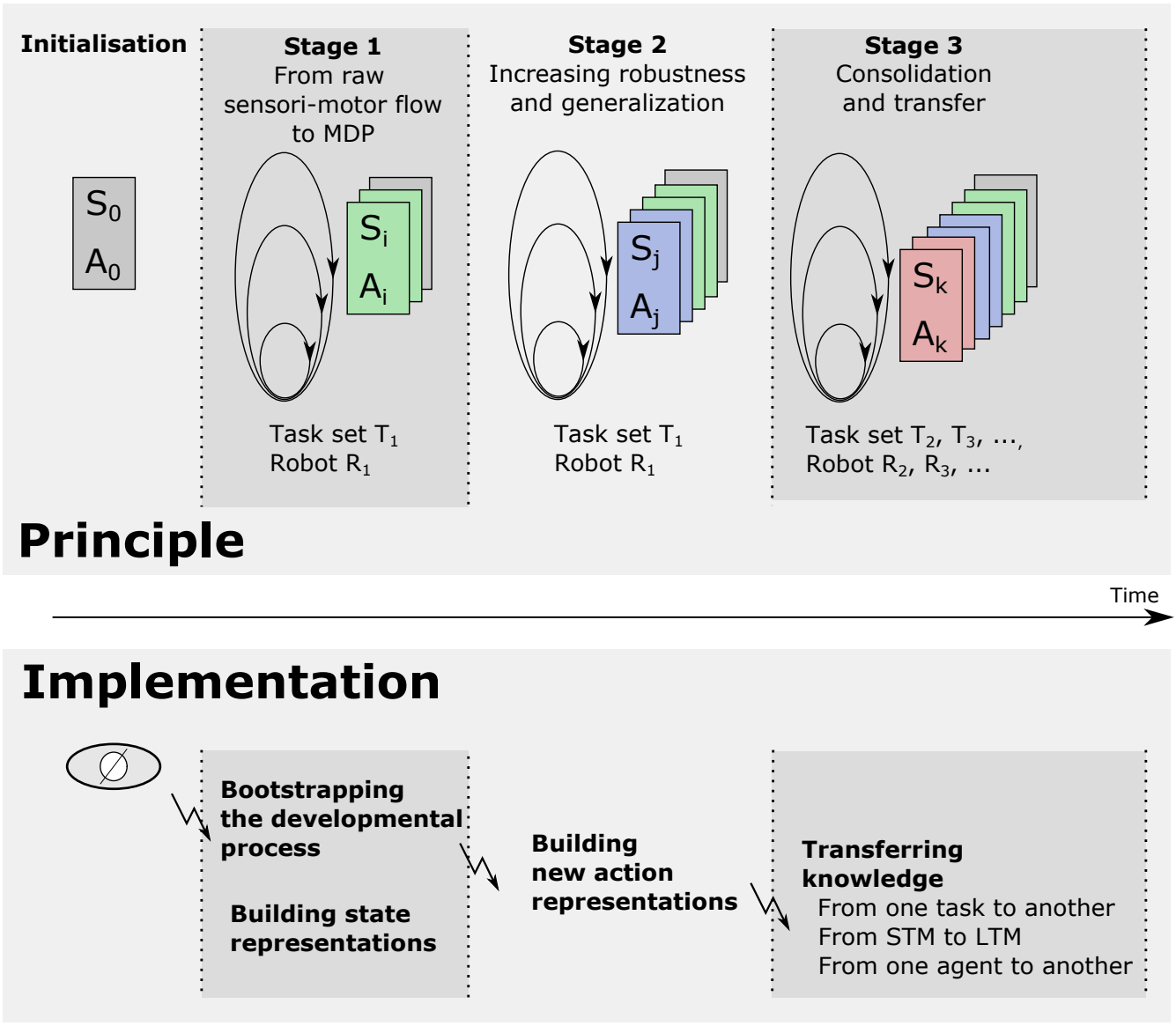


Figure 6. Overview of the DREAM approach. Starting from raw sensorimotor values, stage 1 processes bootstraps the process and builds a first set of representations. Stage 2 processes consolidates these representations on the same set of tasks. Stage 3 processes further restructure representations and acquire the knowledge required to facilitate transfer between tasks, allow knowledge reuse and sharing between robots. S_n is the n-th state space and A_n the n-th action space. The lower part of the figure indicates what has been implemented so far and gives the name of the corresponding sections.

better final performance. Fig. 8 shows our split model where each loss is only applied to part of the state representation (see [64] for a more detailed presentation).

We applied this approach to the environments visualized in Fig. 9, where a simulated arm and a real Baxter robot are in front of a table and image sequences taken from the robot's head camera contain a front view of what the robot is able to see. We consider a "reaching" (*pushing button*) task with a randomly placed button on the table. In this environment RGB images are 224x224 pixels; three rewards are recorded: 0 when the gripper is not touching the button, 1 when touching it, and -1 when the robot gripper is out of the field of view of the frame. The goal of this task is to learn a representation consistent with the actual robot's hand position and button position. Actions are defined by elementary movements of the hand along the X, Y, Z axes in the operational space between timesteps t and $t+1$.

Table 1 shows that our approach outperforms several baselines in terms of correlation of learned states with the Ground Truth Correlation (GTC, see [62]) on the simulated robot, and that using this representation for reinforcement learning using the Proximal Policy Optimization (PPO) algorithm [65] per-

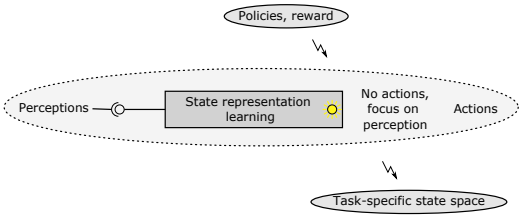


Figure 7. State representation learning module. Starting from a reward that defines a task and policies to observe at least some rewards, it generates a task-specific state space. The approach relies on raw perceptions. Actions are performed by the provided policies and during the validation step, once the state space has been generated. No action is proposed by the state representation module.

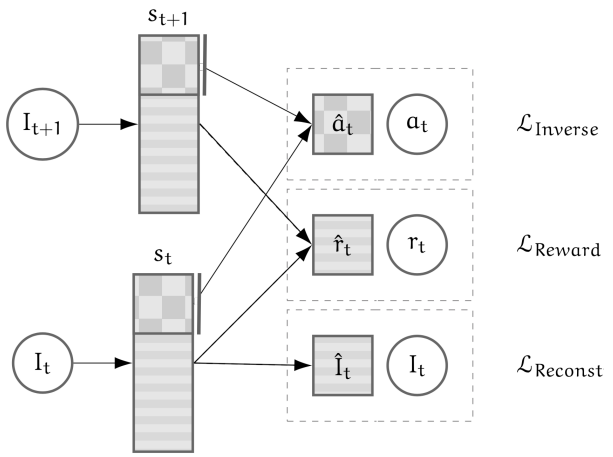


Figure 8. *SRL Splits* model: combines the losses of reconstruction of an image I (auto-encoder) and of reward (r) prediction on one split of the state representation s, and the loss of an inverse dynamics model on the second split of s. Arrows represent model learning and inference, dashed frames represent losses computation, rectangles are state representations, circles are real observed data, and squares are model predictions [64].

forms very close to using the ground truth. Table 2 shows that similar results are obtained for SRL models (GTC) on real robot data.

GTC	x_{rob}	y_{rob}	z_{rob}	x_{targ}	y_{targ}	*Mean*	*Reward*
Ground Truth	1	1	1	1	1	1	4.92 ± 0.10
Supervised	0.57	0.74	1	0.79	0.69	0.76	4.89 ± 0.11
Raw Pixels	NA	NA	NA	NA	NA	NA	4.78 ± 0.15
Rand. Features	0.36	0.54	0.49	0.73	0.83	0.59	2.17 ± 0.44
Auto-Encoder	0.43	0.73	0.67	0.57	0.50	0.58	4.84 ± 0.14
Robotic Priors	0.18	0.03	0.18	0.75	0.42	0.31	2.22 ± 0.43
SRL Splits	0.83	0.87	0.72	0.53	0.63	0.72	4.90 ± 0.14

Table 1. Ground truth correlation and mean reward performance in RL (using PPO) per episode after 3 millions steps, with standard error (SE) for each SRL method in 3D simulated robotic arm with a random target environment.

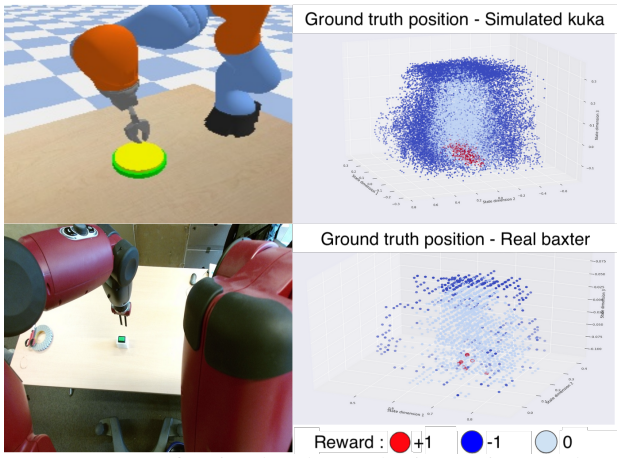


Figure 9. Top Left: Simulated 3D robotic arm in a random target environment. Top Right: Ground truth gripper positions and associated reward (-1: arm out of sight; 1: gripper touching button; 0: otherwise). Bottom Left and Right: similarly for Baxter real robot environment with a fixed target.

GTC	x_{rob}	y_{rob}	z_{rob}	*Mean*
Ground Truth	1	1	1	1
Supervised	0.99	0.99	0.99	0.99
Random Features	0.49	0.51	0.54	0.51
Auto-Encoder	0.63	0.77	0.67	0.69
Priors	0.37	0.25	0.79	0.47
SRL Splits	0.89	0.88	0.69	0.82

Table 2. Ground truth correlation (GTC) for each SRL method in real Baxter robotic arm with a fixed target environment.

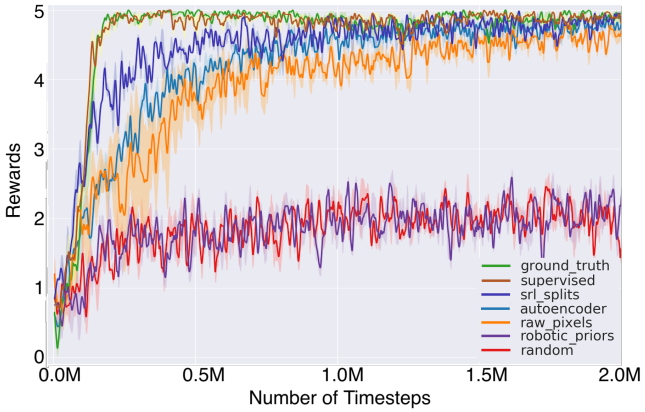


Figure 10. Performance (mean and standard error for 3 runs) for PPO algorithm for different state representations learned in the 3D simulated robotic arm with random target environment.

Learning curves in Figure 10 show that with our proposed approach, RL can use this learned representation to converge faster towards the optimal performance than with unsupervised alternatives, and at a speed close to the one of

the ground truth and the supervised learning cases. The newly defined MDP, based on the learned states, is therefore better adapted to solve this task than the original MDP based on raw images. This new MDP could then be exploited for other tasks that require the same information.

This experiment shows that a task-specific state space can be learned from raw observations. It requires to define a policy or several policies that will generate the data necessary to learn the perception-to-state mapping. This is an important issue as the generated representation critically depends on the observed data. In the experiments reported here, a random policy has been used. Scaling this approach to more complex tasks requires to be able to generate more advanced policies. The next section introduces the proposed approach to bootstrap this knowledge acquisition system. It allows the robot to learn the structure of the environment, i.e. the objects it can interact with and open-loop policies that can be used later on to learn new states with the current representation. This knowledge paves the way to the acquisition of new state representations in which rewards are self-built on the basis of expected effects on identified objects and the policies to generate the required data are the open-loop policies generated by the proposed learning approach.

5. Bootstrapping the developmental process

The bootstrap phase is critical to collect enough data for the later representation redescription modules. It has been split into three modules (Figure 11) that result in a repertoire of actions that can either be used directly to solve simple tasks or as a training set for learning new state spaces (Section 4) or action spaces (Section 6). These modules deal with the following challenges:

- Skill acquisition: building the actions to control state spaces identified so far and pave the way to the acquisition of new and more relevant state spaces;
- Dealing with sparse rewards, in particular when bootstrapping the redescription process;

The bootstrap phase implies learning states and actions. But it relies on predefined representations of these spaces to bootstrap the system and acquire the data required by the redescription processes presented in the other sections.

5.1 Babbling to identify objects

To identify objects, a two-step approach is proposed. In a first step, a segmentation separating the background from the parts with which the robot can interact is learned. Then, from this segmentation, 3D object models are learned. Both steps are based on the interactive perception paradigm [67]. The robot explores an environment by interacting with it in order to collect data and train models on them. This interaction relies on predefined motor primitives. Once the structure of the environment has been identified, new motor primitives are learned that can replace the ones provided at startup, thus implementing the asymptotic end-to-end principle.

This two-step approach relies on minimal environment-specific assumptions. Indeed, the first step builds a simple representation of the environment which does not need a lot of prior knowledge. Then, in the second step, the prior knowledge needed to build object models, like the number of objects or their approximate position, can be easily inferred from the first segmentation.

Relevance Map: A First Segmentation of the Environment. In the first step, the robot builds a perceptual map called *relevance map* by training a classifier with the data collected while the robot interacts with the environment. The relevance map indicates the parts of the visual scene that are relevant for the robot with respect to an action. "Relevant parts" means parts of the environment that have a high probability to produce an expected effect after having applied a given action, for instance moving this part of the environment when touching it through a push primitive.

The exploration is sequential and follows 5 main steps:

- The visual scene is over-segmented using Voxel Cloud Connectivity Segmentation [68]. This method segments a 3D pointcloud into small regions of same size. Then, visual features are extracted from each segment.
- The *relevance map* attributes to each segment a relevance weight computed using the prediction of the classifier trained online.
- Based again on the classifier, a *choice distribution map* is computed which represents the probability of each segment to be chosen as the next interaction target.
- An action primitive is applied with the center of the chosen segment as target.
- Finally, an effect detector is applied to label the visual features of the selected segment. Detected effects are labelled to 1, otherwise a label equal to 0 is attributed.

By following these 5 steps, the robotic system builds a dataset of labeled samples on which a classifier is trained online.

The proposed approach was tested on both a Baxter [69] and a PR2 [66] robots. The experiments were conducted on two set-ups with a push primitive. To detect if an effect occurred, a change detector compares the pointclouds before and after the interaction. If a targeted segment is part of the difference pointcloud, it means that something has moved. In this context, the relevance map represents the areas of the environment that the robot can move. Examples of obtained results are shown in Figures 12 and 13.

Figure 12 shows a sequence of relevance maps at different moments of the exploration. After the first interaction (i.e. only one sample in the dataset), the map is uniform. From the

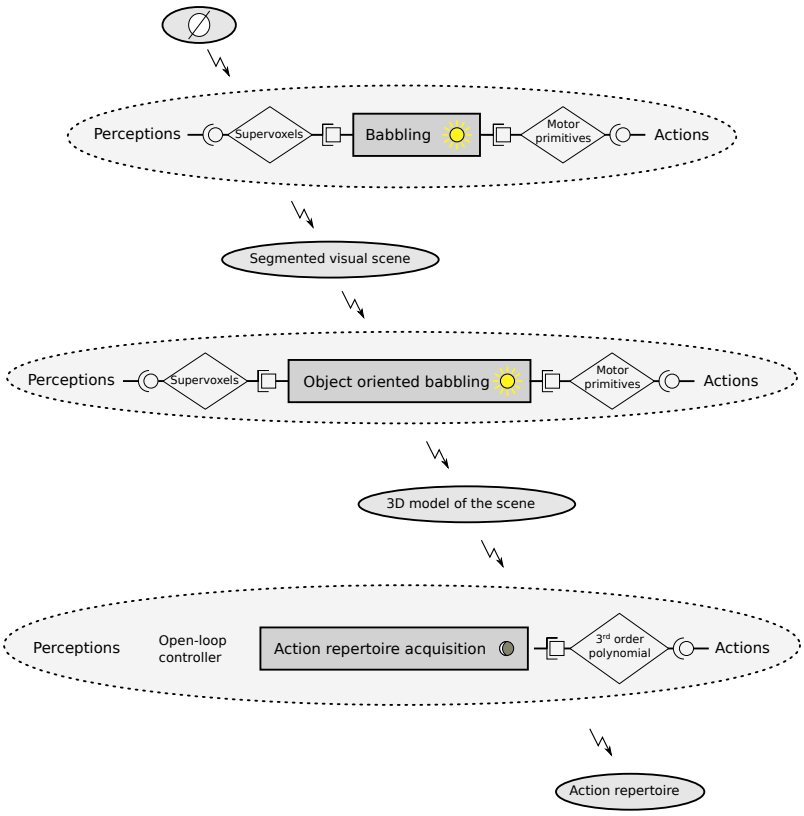
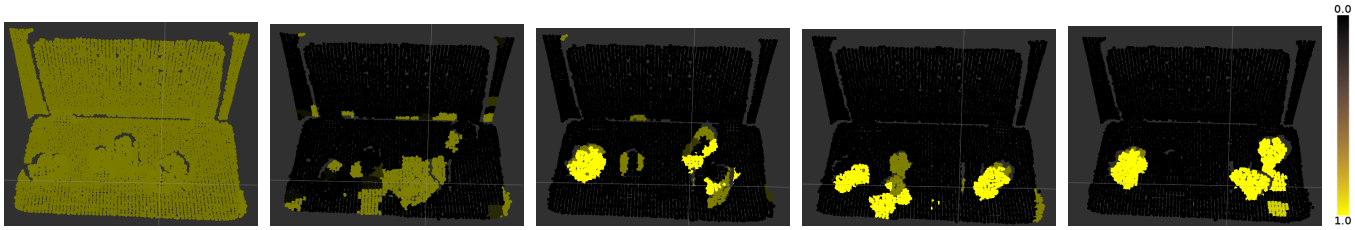


Figure 11. The three modules of the bootstrap phase. The result is a repertoire of actions which can be used either to solve simple problems if the state space is known, or as a training set, or else to collect sensori-motor data.



(a) Relevance map after 1 interaction

(b) Relevance map after 10 interactions

(c) Relevance map after 50 interactions

(d) Relevance map after 100 interactions

(e) Relevance map after 400 interactions

(g) Pointcloud used to generate above image

(h) Pointcloud used to generate above image

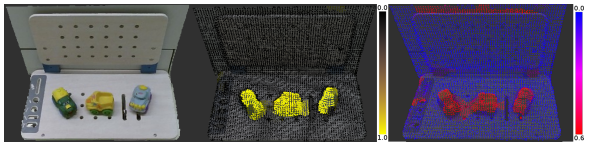
(i) Pointcloud used to generate above image

(j) Pointcloud used to generate above image

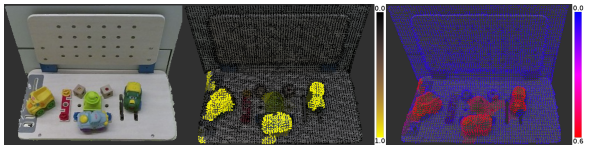
(k) Pointcloud used to generate above image

Figure 12. Sequence of pointclouds representing a relevance map at different points during exploration. These images have been generated after exploration. Adapted from [66].

50^{th} interaction, the map begins to show a meaningful representation. An important feature of the classifier is its ability to give meaningful predictions with few samples. Therefore, the exploration is efficiently directed after a few collected samples. The exploration process is focused on complex areas, i.e. areas which carry a lot of information, as shown on left part



(a) Simplest set-up used for the experiment. A toy workbench with 3 movable cars.



(b) A more complicated set-up. A toy workbench with an object fixed on the horizontal panel and 3 movable cars.

Figure 13. Two examples of relevance map learned during an exploration with a push primitive. From left to right: a colored 3D pointcloud, a relevance map on the left point cloud and the accumulated choice distribution maps during the whole exploration. Adapted from [66].

of Figure 13.

Several relevance maps relative to different action primitives can be learned. This approach was tested with a push primitive, a push-button primitive and a lift primitive [70]. Each of these relevance maps is a representation of the environment depending on the action and on the possible effect considered during exploration. In other words, a relevance map implements an affordance [71]. These relevance maps are finally merged into a new perceptual map, called *affordance map*. An affordance map gives to the robot a rich perception of which action could be applied and where they could be applied.

An example of affordance map is shown in Figure 14. The push-buttons identified in green by our system do not overlap with the pushable and liftable objects identified in red and purple. Thus, the classifier is able to learn different concepts. Also, only small objects are identified as liftable and pushable, and the biggest objects are identified as only pushable.

Object oriented babbling The second step consists in building object models on the basis of the segmented maps acquired during the babbling phases. The description of this module is out of the scope of this article. The module can, for instance, rely on the method proposed by [72]. It aims at providing a 3D model of the environment for learning processes described in the next section.

5.2 Learning to manipulate objects

With the previously described bootstrapping modules, the system can acquire a model of its environment, the various objects it contains and their properties. In order to solve tasks involving those objects, the robotic system must now learn motor skills to manipulate them. This raises the challenge of exploring and mapping the action space of the robot to build motor skills able to engage them. In an open-ended learning context, the objects and environment can vary, and

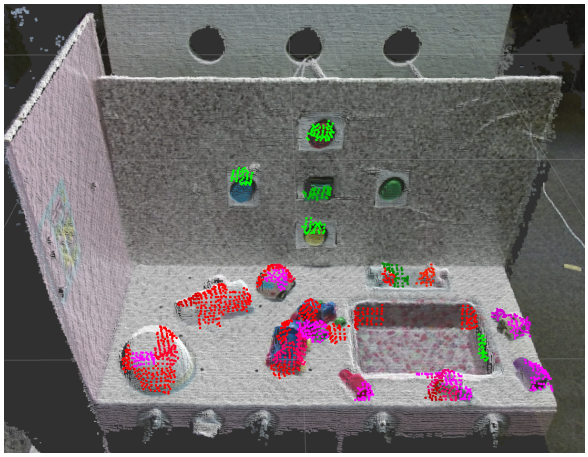


Figure 14. Affordance map of liftable objects (in purple), activable push-buttons (in green) and pushable objects affordances (in red). Only areas classified with a probability of afforded action above 0.5 are represented in the figure. The bottom picture represents the environment from which the affordance map has been extracted. Adapted from [70].

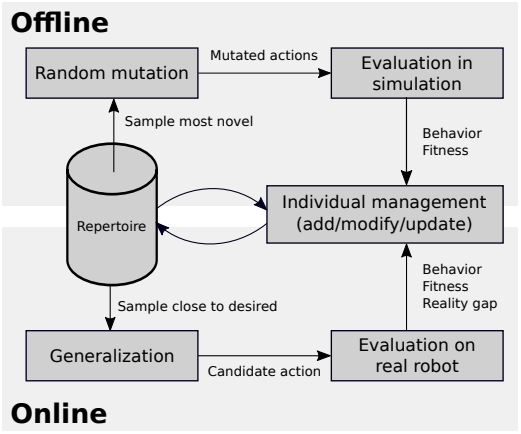


Figure 15. Overview of the object-oriented skills learning module. A quality-diversity algorithm (in blue) is run offline (using a simulation) to build a repertoire of motor skills able to produce diverse behaviors. When the robot must perform a given behavior (in red), it then samples the archive for the skills producing the closest behaviors, and uses them to build a candidate skill expected to reach the target point in the outcome space. If this skill fails due to the reality gap, the error is evaluated and used to adapt the candidate skill, and to update neighboring skills. Adapted from [73].

it is therefore of great importance that the same methods can handle different setups and use no prior knowledge other than

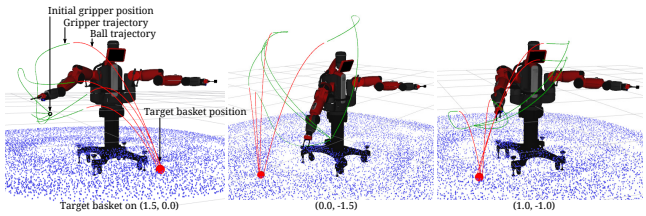
that provided by the bootstrap module. Furthermore, instead of reaching a specific goal for which finding a single policy would be enough, as is typical in a reinforcment learning paradigm, we want the system to be able to tackle different tasks requiring various skills.

A way to address this issue is to define not a goal but a space representing the controllable state of the environment, often called behavior space [44], goal space [74] or outcome space [75], and to learn a wide repertoire of motor skills able to reach many points in this space. As those exploration algorithms tend to be sample-inefficient, they are usually used in simulation, which is possible in the present context considering the knowledge acquired from the bootstrap phase, but introduces a further challenge in the form of the reality gap [76], where policies learnt in simulation must be transferred to the real robot. Our approach [73] uses a Quality-Diversity (QD) algorithm [77, 78] to build such a skill repertoire, and a generalization approach based on a local linear model of the mapping from the action parameter space to the outcome space to adapt those skills to real robot control. This process is summarized in Figure 15 and detailed below. We evaluate this approach for two different problems (throwing a ball at various targets and manipulating a joystick) and show that in both cases, it is able to learn repertoires of diverse skills, to address the reality gap issue, and to generalize to new policies, resulting in efficient control of the outcome space.

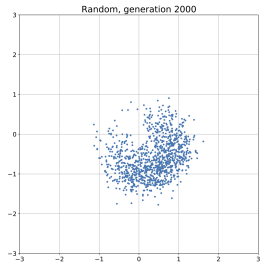
5.2.1 Offline learning of skill repertoires

For both problems, the system was tested on a Baxter robot controlled by simple parameterized motion primitives based on a third order polynomial, whose parameters constituted the action space (see [73] for details). For the ball throwing problem, a ball was initially placed in the robot's gripper and the studied outcome space was the 2D position of the ball when it reached the ground plane (Figure 16a). For the joystick manipulation problem, a joystick was placed on a table in front of the robot and the outcome space was its final pitch and roll (Figure 17a). The quality metric used for ball throwing was torque minimization, and skill robustness to small perturbations for joystick manipulation. Skills were added to the repertoire if they had no close neighbor, or replaced their closest neighbor if they had a higher quality score. Evaluation was done using the DART simulator. Results show that the quality diversity algorithm is able to learn a skill repertoire that densely covers the reachable outcome space for ball throwing (Figure 16c), and another skill repertoire to reach a large and diverse set of final positions for joystick manipulation (Fig. 17c) whereas a random baseline (Figure 16b, 17b) results in much more limited exploration of the outcome space[5].

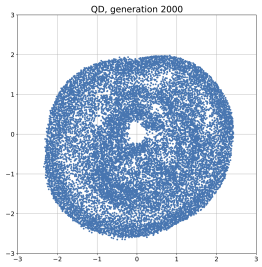
[5]Only one run of the method out of 26 for ball throwing and 12 for joystick manipulation is shown in Figs. 16 and 17; see [73] for full results and analysis.



(a) Example of diverse throwing trajectories



(b) Random baseline



(c) QD search

Figure 16. Skill repertoires built by the random baseline (16b; 1085 ± 26 points) and the QD search (16c; 14473 ± 1619 points) for the ball throwing problem. Each blue point is the contact point of the ball with the ground. QD search was run for 2000 generations. For the random baseline an equal number of actions were uniformly sampled in the action space. Adapted from [73].

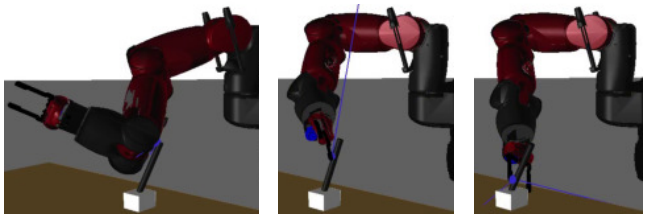
5.2.2 Online generalization and adaptation by local linear Jacobian approximation

Using a skill repertoire generated by the QD algorithm to control the robot in the real world raises two challenges: first, the skills may have different outcomes in reality than in the simulated environment (the reality gap problem); second, despite densely covering the outcome space, the repertoire is still finite, and may not contain the skills to reach some specific points in that space. Our local linear Jacobian approximation method, similar to that of [28], tackles both issues. It proceeds as follows (with $\mathscr{A} = \{(\boldsymbol{\theta}_i, \boldsymbol{b}_i)\}_{i=1,\ldots,N}$ the repertoire containing N action parameters $\boldsymbol{\theta}_i \in \mathscr{G}$ and their outcomes $\boldsymbol{b} \in \mathscr{B}$):

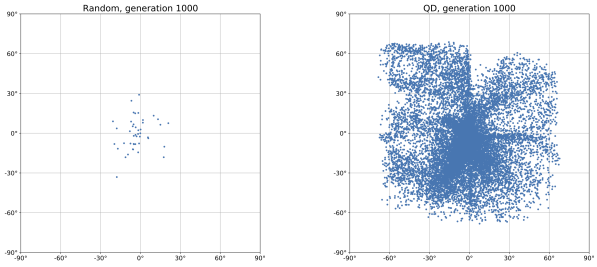
- For an arbitrary target point $\boldsymbol{b}^* \in \mathscr{B}$, find the closest point $\boldsymbol{b}_c$ in the repertoire and its corresponding $\boldsymbol{\theta}_c$.
- Find the K nearest neighbors to $\boldsymbol{\theta}_c$ in the repertoire, and their corresponding outcomes.
- Use those K $(\boldsymbol{\theta}_i, \boldsymbol{b}_i)$ samples to estimate $\boldsymbol{J}_{\boldsymbol{\theta}_c}$ the Jacobian matrix[6] at $\boldsymbol{\theta}_c$ by the least squares method.

This estimation $\widetilde{\boldsymbol{J}}_{\boldsymbol{\theta}_c}$ can be used to define a local linear model between the action parameter space to the outcome space – and a matching local inverse model, by pseudo-inverting the matrix. Although the global mapping of the

[6]the matrix $\boldsymbol{J}_{\boldsymbol{\theta}}$ such as $\Delta\boldsymbol{b} = \boldsymbol{J}_{\boldsymbol{\theta}}\Delta\boldsymbol{\theta}$



(a) Example of diverse joystick manipulation skills. Note that similar joystick positions can be reached by different movements.



(b) Random baseline **(c)** QD search

Figure 17. Skill repertoires built by the random baseline (16b; 34 ± 3 points) and the QD search (16c; $15\,532 \pm 3329$ points). Each blue point is a final joystick position QD search was run for 1000 generations, for the random baseline an equal number of actions were uniformly sampled in the action space. Adapted from [73].

action parameter space to the outcome space is highly non-linear for the considered problem, it is smooth at most points and the skill repertoire is dense enough to define a good linear approximation in most regions. The local linear model can then be used to solve the aforementioned issues:

- **Generalization:** using the local inverse model, compute a candidate action $\widetilde{\boldsymbol{\theta}}^*$ which is expected to reach $\boldsymbol{b}^*$, and try it on the robot;
- **Reality gap crossing:** if the candidate action does not reach $\boldsymbol{b}^*$ accurately enough, record the point $\widetilde{\boldsymbol{b}}^*$ reached and compute the error $\Delta\boldsymbol{b}^* = \widetilde{\boldsymbol{b}}^* - \boldsymbol{b}^*$. The pseudo-inverted Jacobian estimation can then be directly used to compute a correction $\Delta\boldsymbol{\theta}^*$ to the action to apply to $\widetilde{\boldsymbol{\theta}}^*$ to reduce the error. This process can be iterated if needed, until the reality gap has been crossed.

Reality gap crossing was quantitatively evaluated in simulation, with a large simulated reality gap. In both conditions, most actions initially failed due to the reality gap, but could be adaptated by the method in at most 4 iterations of the method in 89% of cases for ball throwing, and 31% of cases for the more difficult joystick manipulation task. Reality gap crossing was also tested on real robot (Fig. 18): over 50 trials on random target positions, only 8 required adaptation, and all 8 succeeded after a single iteration of the method [73].

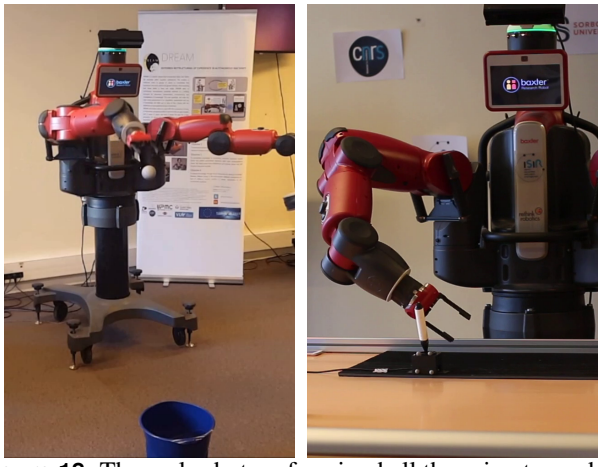


Figure 18. The real robot performing ball throwing towards a basket (left) and joystick manipulation (right) based on the action repertoires built in simulation.

6. Building new action representations

We would like the behavioural capabilities of our robot to be robust to environmental perturbations. Unexpected changes in the environment may require using different actions to achieve the same effect, for instance to reach and grasp an object in changing clutter. A possible approach is to adapt the control policy through, e.g., obstacle localization and explicit re-planning [79]. This approach requires to build a dedicated algorithm in which obstacle representation is given beforehand. This may raise an issue for open-ended learning. A more general purpose and open-ended alternative is to exploit a behavioural repertoire [80] and extract from it an adequate policy [16].

In the bootstrapping section presented earlier, we described learning a repertoire using QD search, exemplified by the Baxter robot throwing a ball. In this case the repertoire spans the space of potential throwing targets. After this bootstrapping phase, one (or a small number of) throwing policies are stored for each potential target. To increase behaviour robustness through diversity, we need multiple diverse throwing movements *for each potential target*. Thus, if a new obstacle appears, diverse behaviours can be tried until one succeeds. A behavior repertoire contains a finite and limited number of policies that can thus adapt only to a certain extent to new situations.

To go beyond this limitation, we present an action re-description process transforming the library-based representation obtained from the bootstrap phase into a new representation that is both more compact and more diverse – through learning a generative adversarial network (GAN) [81] over policies (Figure 19). GANs are neural networks suitable for learning generative models over complex high-dimensional data, typically images. In this case, we train a conditional GAN that accepts a movement (throwing) target as a condition, and generates diverse throwing movements that hit this

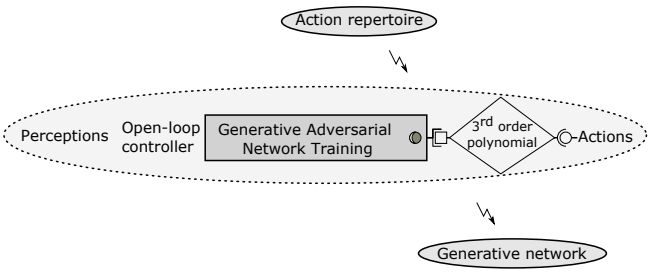


Figure 19. The proposed action redescription phase that builds a policy parameter generative network out of an action repertoire.

target. Now, only the model parameters rather than controller library needs to be stored, and the available diversity is not limited to a fixed length controller library. By sampling the generative model over controllers, an unlimited number of distinct controllers can be obtained. Given a powerful generative model, these need not be simple perturbations of known controllers, but can encode novel solutions to the problem by drawing on diverse aspects of multiple training policies.

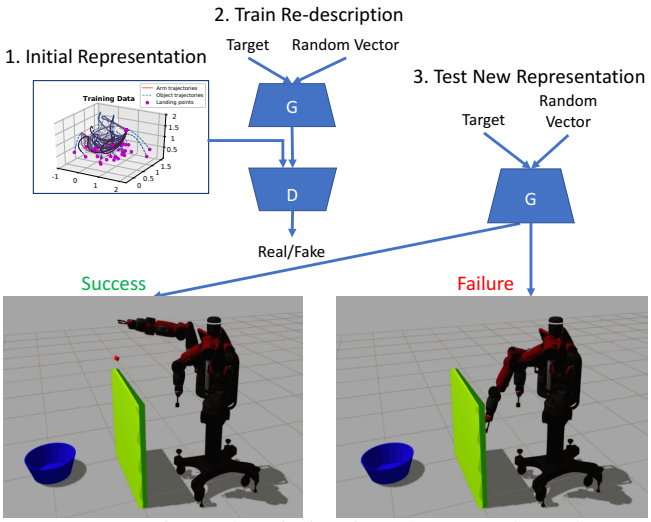


Figure 20. Action redescription for robustness through diversity. 1. The initial representation is the population of controllers from QD search. 2. GAN training produces a generative model over controllers. 3. For robust behaviours, the generative policy network is sampled until one is found that avoids the obstacle.

Figure 20 illustrates our framework, with full details available in [82]. The policy representation here is a 15D vector of parameters defining a low-level open-loop velocity controller for throwing. We start with a set of controllers obtained from QD search. The diversity of this set mainly spans different throwing targets. We then train a target-conditional generative model for controllers by playing a min-max game with a generator and discriminator network. Once the generator network is trained, it maps a target coordinate on the floor, and a random vector to a new controller. Sampling this random vector for a fixed target vector generates diverse ways of throwing to the same target. In the case of an obstacle, controllers can be sampled until one is found where neither the ball nor the arm collide, and the ball hits the target. This is illustrated in Figure 20 where two throwing samples are drawn, and the underhand throw fails while the overhead one succeeds.

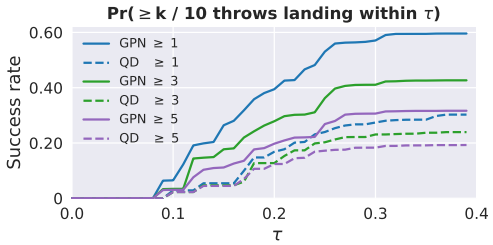


Figure 21. Quantitative evaluation of throwing in the presence of obstacles. Our redescription improves in terms of probability of achieving k hits out of 10 for various target radii τ. Adapted from [82].

For quantitative evaluation, we perform the throwing task averaging over a large number of target positions, randomly placed obstacles, and multiple diverse throwing attempts in each target-obstacle configuration. Our goal is that out of N throwing attempts in each configuration, at least k of them should hit the target. The results in Figure 21 show probability of k hits out of $N = 10$, as a function of how close to the target a ball should land to be considered a hit. We can see that, as expected, the success rate depends on the stringency of the hit criterion. More interestingly, the proposed redescription increases this rate (solid vs dashed lines), at several values of k. Thus, this action redescription succeeds in increasing robustness via increased diversity, while also compressing the prior bootstrap representation.

7. Transferring knowledge

7.1 From one task to another

Learning an individual robot control task from scratch usually requires a large amount of experience and may physically damage the robot. This has motivated a fruitful line of research into transfer learning, which aims to bootstrap the acquisition of novel skills with knowledge transferred from previous acquired skills [6]. For open-ended learning, we would like to transfer knowledge from a lifetime of previous tasks rather than a single source task. One potential way to realize this is to model all tasks as laying on a low dimensional manifold [83]. Based on this assumption, [84] construct a transferrable knowledge base (the manifold) from linear policies through matrix decomposition methods, in the case of supervised learning tasks. [85] extend this approach to non-linear policies, represented by deep neural networks, by stacking policies into 3-way tensors modeled by their low-rank factors under a Tucker decomposition assumption. This manifold-based transfer approach has been shown successful in learning sim-

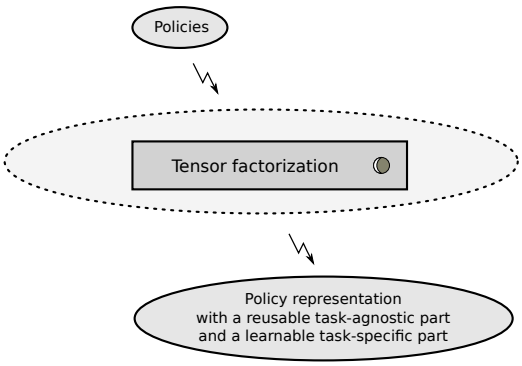


Figure 22. Tensor based knowledge transfer relies on a tensor factorization to perform a policy redescription. It builds, from a set of policies, a new policy representation that includes a reusable part and a learnable part. The use of a task-agnostic, reusable knowledge allows further learning to be faster.

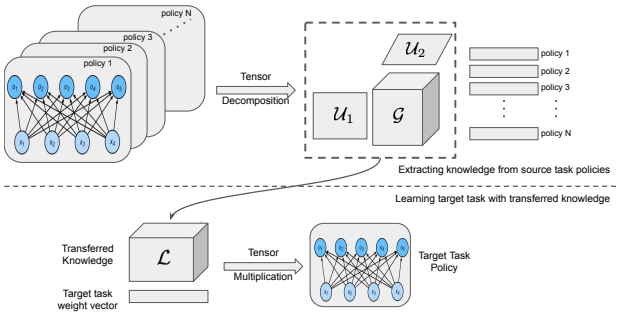
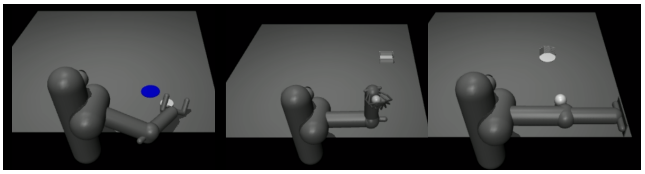


Figure 23. A schematic diagram of tensor based knowledge transfer. Source task policies are stacked and decomposed into a task agnostic knowledge base tensor $\mathscr{L}$ and one task-specific weight vector for each task. Given a target task, the agent learns a target task weight vector and fine-tunes the knowledge base to reconstruct a target task policy.

ple linear robot control tasks, such as cart-pole [86]. Here we further look into more complicated tasks and propose a knowledge transfer approach for learning novel non-linear control tasks.

We consider learning a policy π_{n+1} for task $n+1$ given the policies of the previous n learned tasks (Figure 3) on the basis of an extracted, task-agnostic and thus reusable part and a learnable, task-specific part (Figure 22). We use generic multi-layer perceptron (MLP) networks to model the policy for each task that maps instantaneous proprioceptive state to control torques at each joint. For each network layer, we stack the policies from source tasks into a 3-D tensor. To abstract the previous knowledge, we then factorise the tensor into a task-agnostic knowledge base $\mathscr{L}$ and task-specific task weight vectors with Tucker decomposition. For training the novel $n+1^{th}$ task, the agent alternates between learning the task-specific parameter vector and fine-tuning the task-agnostic tensor. The transfer procedure is illustrated in Figure 23, with full details available in [87].



(a) Pusher **(b)** Thrower **(c)** Striker

Figure 24. Illustrative figures of three robot manipulation tasks used to evaluate tensor-based cross-task knowledge transfer

.

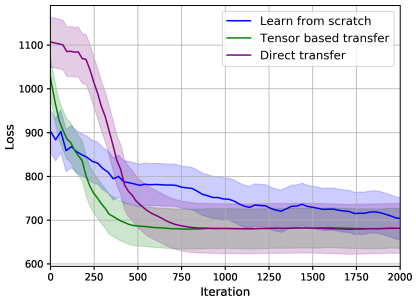


Figure 25. Learning curve for target task (Striker) with knowledge transferred from the source task policies (Thrower and Pusher).

We evaluate our method with three simulated robot manipulation tasks as illustrated in Figure 24: Mujoco's Pusher, Thrower and Striker. We take pusher and thrower as source tasks, and striker as target task. We then evaluate the performance of reinforcement learning of the striker task with CMA-ES, comparing three alternatives for task-transfer: (i) learning from scratch without transfer, (2) directly transferring a randomly sampled source policy (pusher, or thrower) and fine-tuning for the target-task, (3) Our tensor-based transfer method. As shown in Figure 25, our method learns faster compared to both baselines. This is due to the ability to leverage the transferred abstract task agnostic knowledge obtained by re-representing the source policies through tensor factorization, which in this case corresponds to smooth movement primitives.

7.2 From short term memory to long term memory

In the work presented so far, the focus was on state or action representation learning, with the modules necessary to bootstrap it. Once this knowledge has been acquired, the robot has several different MDPs at its disposal, each, with its own state space, action space, reward function and policies. A fundamental question is then to determine which MDP to use and in which context, i.e. an MDP needs to be associated with its context of use. This is the goal of the Long Term Memory.

Long Term Memory is a fundamental part of any cognitive architecture that aims to store and reuse the acquired knowledge. Most traditional cognitive architectures (e.g. ACT-R,

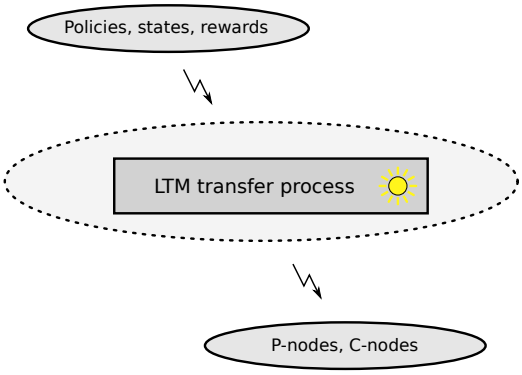


Figure 26. Transfer from short term memory to long term memory. Starting from policies, state spaces and rewards, the transfer process applies the policies and observes their outcome to deduce their context of use in the form of P-Nodes (perception nodes) and C-Nodes (context nodes).

CLARION, SOAR) are based on symbolic representations and, thus, implement relatively straightforward LTM structures where reuse is based on searching for the appropriate labels. When working with subsymbolic structures, such as ANN based representations, labeling is not straightforward and other types of LTM approaches must be used. In this work, we have developed an experience-based associative LTM structure. The basic features of this approach are presented in [88]. The operation of this type of LTM is based on the relationships the system acquires about knowledge nuggets as it interacts with the world. The basic knowledge nuggets that are considered here are the different parts of the MDP, i.e. states, rewards and policies. The idea is that when instances of these knowledge nuggets co-occur and when something relevant was experienced by the system (such as a high reward), they are associated through a new type of knowledge nugget: a Context node (C-node). The definition and operation of these structures have been described in [89]. C-nodes implement conjunctive representations of context and are activated when their associated nuggets are present (or at least most of them). In other words, a C-node implies that sets of the type $\{S_i, R_k, \pi_r\}$ that lead to relevant situations experienced by the system must be identified and stored. This way, when a sub-set of $\{S_i, R_k, \pi_r\}$ is active – S_i allows to predict what happens and a R_k is activated –, the system can infer from its experience that applying policy π_r, should lead to producing the same relevant event. These nodes provide a simple albeit powerful structure to store context related information that allows the system to selectively recall appropriate policies (or other knowledge nuggets) in the presence of known or similar situations.

Most of the knowledge nuggets stored in LTM have not usually been defined in the whole perception or state space. For instance, the accuracy of a reward function cannot be expected to be high far from observed areas of the state space. All knowledge nuggets in LTM, including C-nodes, are reliable only within a particular area of state space. Consequently, they should only be used or activated in this area. To address this issue, it is necessary to introduce the concept of perceptual classes, which are areas of the perceptual space for which knowledge nuggets are valid[7]. Therefore, a perceptual class is a generalization of perceptions into a higher level, discrete, representation linked to a given response of the system. Perceptual classes are represented with a LTM component called perceptual nodes or P-nodes. A P-node is a functional component that is activated when a perceptual state belongs to a given perceptual class.

Different algorithms for the online and offline delimitation of P-nodes have been proposed, both using point-wise distance based representations, that is, heuristic episode clustering approaches [88], and neural network based generalizations. Redescription procedures to go from a more hippocampal-like episode-based representation to a more cortex-like generalized representation in the form of ANNs were studied as reflected in [90]. This process can be carried out in a quasi-online manner with a reasonable quality level and it can also take place during an off-line dreaming-like process leading to much better results. For instance, Figure 27 displays a representation in the form of 2D activation maps of some P-nodes (representing perceptual classes) that were automatically obtained using a Baxter robot that was trying to learn to put objects placed anywhere into baskets, also placed anywhere. The bottom left graph represents one of the P-nodes and maps the angle at which an object is located with respect to the angle of the target basket to put the object. It can be interpreted as object in the wrong side (need to change hands). The bottom central graph maps distance and angle at which an object is located with respect to the robot, and its activation can be interpreted as "the object is reachable" as it provides the reachable area for the robot arm. Finally, the bottom right graph corresponds to non-reachable area or "unreachable object".

Using this type of associative memory, after the system has acquired some experience interacting with an environment or a set of environments, whenever it is faced with a perceptual context, a set of P-nodes becomes active, thus pre-activating all of the knowledge nuggets that might be relevant in that situation in terms of perceptual cues. This provides an opportunistic way of pre-selecting previously learnt knowledge nuggets that might be relevant in order to execute them or to generate new knowledge nuggets for a new environment the robot might be facing.

7.3 From one agent to another

So far, we have limited our scope to individual robot learning. However, due to the variety of situations, exploration can be tedious. As stated in Section 2.2, one multi-task challenge is to transfer acquired knowledge from one robot to another, so as to enable improvement over the experience of others. Indeed, using multiple robots can increase the efficiency with

[7] Perceptual classes can also be used to represent other properties of knowledge nuggets but we focus on this one here.

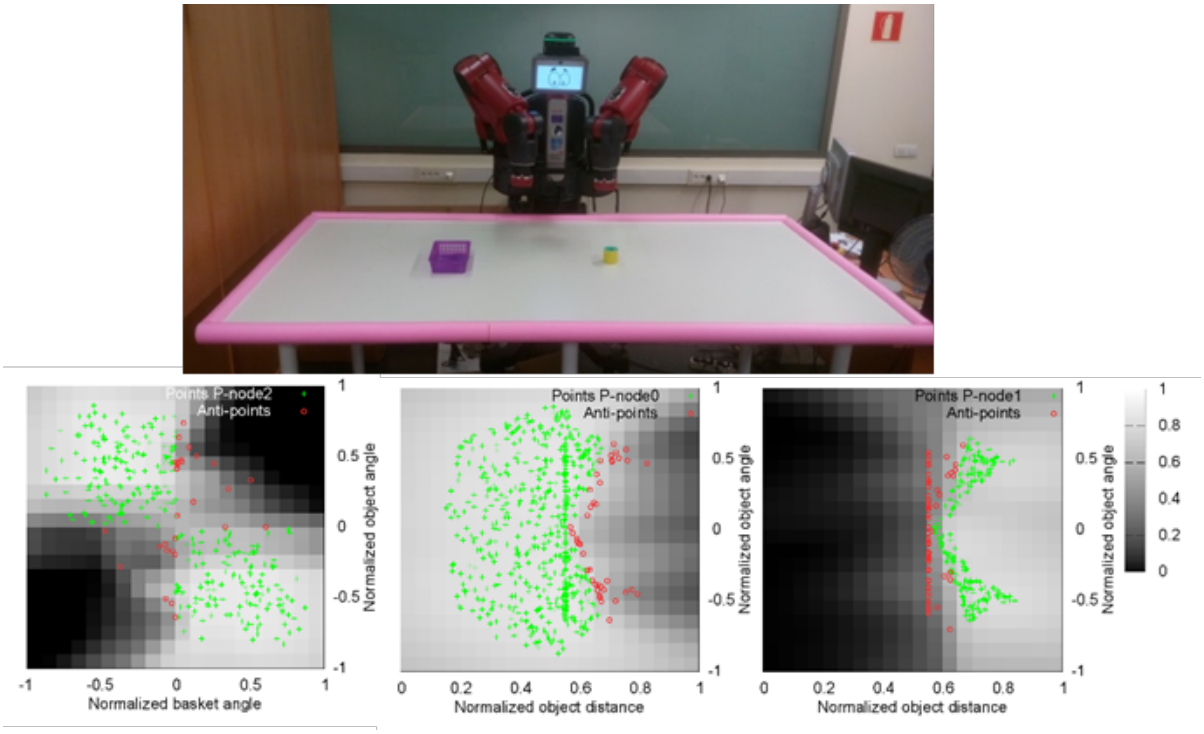


Figure 27. Final activation maps for three P-nodes in an experiment where the robot tries to put randomly placed pucks into randomly placed baskets as shown in the setup on top.

respect to both speed and quality of learning by sharing experience obtained by multiple exploration processes running in parallel. Here we consider robot-to-robot learning where multiple robots share learned skills while completing a task.

Robot-to-robot learning raises its own technical limitations (e.g.: communication bandwidth, network structure) and challenges (what and how much should be transferred, and to whom). Also, the problem of sharing information can be seen as a combination of both an exploration problem and a consensus problem. As described earlier, the exploration problem is addressed through individual learning, which is performed independently from the group. As a consequence, the consensus problem must comply with skills learned by different robots, where skills compete to be transferred to the whole group. Some robots may acquire and share better skills than other robots, and the question is open as to how to select the best skills while maintaining a certain level of diversity resulting in the discovery of even better skills.

We have explored two similar classes of algorithms for information sharing in multi-robot systems: embodied evolution [91] and social learning [92]. While the former emphasizes learning of collective behaviour (i.e. robots interacting with one another), the latter is explicitly concerned with sharing chunks of information that have been acquired by individual robots. However, we have shown that with both families, learning converges towards a homogeneous set of skills shared by the whole group of robots.

The general architecture is illustrated in Figure 28. Each individual robot learns individually and transmits all or part of the description of its skill set to other robots. Information transfer depends on network connectivity: a given robot may broadcast to everyone, or to the subset of reachable robots. As with social learning in nature, selecting which set of skills is to be transferred and accepted depends on a selection process running on each robot. The more exclusive the selection, the faster the convergence, but at the cost of a faster loss of diversity in the collective. We have shown in [93, 94] that selection of incoming skills proportionally to their accounted performances provides an efficient way to maintain diversity of individual learning processes.

Even more importantly, we have also shown that, as expected, social learning provides increased learning speed, but also yields increased performance when compared to individual learning [95]. This is due to the possibility of running several instances of individual learning algorithm with different meta-parameters values, as best values cannot be guessed before run-time. In other words, social learning can efficiently mitigate the negative effect of parameter tuning of the learning process by enabling multiple searches and selecting the best performing one at run-time. In addition, the gain in diversity can actually help to obtain even better results than those that could be obtained by the best robot learner alone.

8. Related work

The DREAM architecture aims at going beyond a single learning and decision process and thus moves towards cognitive

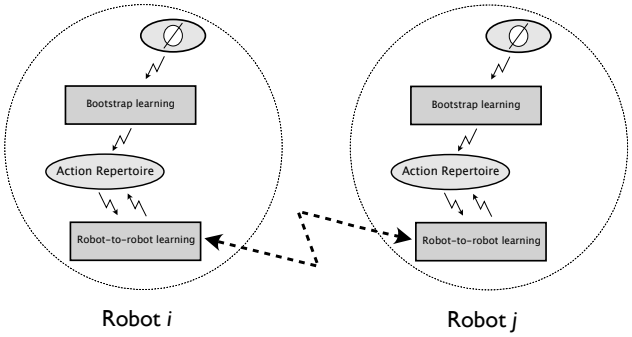


Figure 28. Schematic view of robot-to-robot learning. First, each robot builds its own repertoire of actions by individual learning from experience ("bootstrap learning", see Figure 11 for a detailed view). Then elements from the repertoire of actions can be transferred from one robot to another, and vice versa. Both processes run in parallel.

architectures that are designed to coordinate such processes. In particular, it relies on a hierarchical architecture where higher representational levels are built on top of more elementary ones, and it targets the open-ended learning context, which is closely related to continual learning and developmental processes. In this section we investigate the relationship between our work and all these areas of research.

8.1 Cognitive architectures

Cognitive architectures have been studied for more than 40 years [96]. One of the main goals of cognitive architecture research is to model the Human mind and understand it with a synthetic methodology. It has thus a wider scope than what is proposed here. These architectures are often classified according the kind of representations they can manipulate [97, 96]. They are either:

- *Symbolic:* cognitive architectures, relying on symbols and a dedicated instruction set, as GLAIR [98], EPIC [99] or ICARUS [100]. Knowledge nuggets in this case are generally represented as IF-THEN rules;
- *Emergent:* cognitive architectures, relying on connectionnist approaches, as BECCA [101], MDB [102] or SASE [35]. The knowledge is distributed in neural networks;
- *Hybrid:* cognitive architectures, relying on both, as ACT-R [103] or SOAR [104], that were initially symbolic, but that include non-symbolic representations, at least in their latest versions [104].

Each architecture has it own processes to acquire new knowledge, but the knowledge representation is a core feature that is given beforehand. For practical reasons, it is generally homogeneous [105] and considered as a design choice that the system cannot act on. As such, representational redescription is not addressed in these works. Sometimes, it is even difficult to figure out what type of representation is used, resulting in different classifications of cognitive architectures between different review papers [96].

The DREAM architecture is not a full cognitive architecture. It is a consistent set of modules aimed at providing robotic cognitive architectures with a new ability: the ability to autonomously build new knowledge representations that are adapted to robot's tasks and features. It can be seen as an instance of the *design by use case* approach proposed by [106].

8.2 End-to-end and hierarchical approaches

The ability to deal with multiple representations appears in the machine learning literature under the point of view of hierarchical approaches able to represent available knowledge at multiple levels. It relies on the notion of *options*, that extends the mathematical framework of Markov Decision Processes by adding different levels of temporal abstractions. The Markov Decision Process becomes a semi-Markov Decision Process, with actions that may have different temporal extensions [51]. In this Hierarchical Reinforcement Learning (HRL) framework, options are defined as a triplet $\langle \mathscr{I}, \pi, \beta \rangle$, where $\mathscr{I}$ is the initiation set, i.e. the set of states from which the option can be activated, π is the policy selecting the action to perform and β is the termination condition. With this approach, the combination of different levels of representations is possible. It can accelerate learning [107] or make learning more robust [108].

In the standard option framework, all options are built on top of an initially provided MDP. As a result, the robot controlled by such hierarchical approaches is limited by this initial MDP. Given the difficulty to define an MDP in a robotics context [4], this is a strong limitation to implement versatile robot learning capabilities.

Deep reinforcement learning holds the promise of circumventing this difficulty by making it possible to learn from raw inputs to raw outputs [109]. With such end-to-end approaches, it becomes possible to build policies taking images as inputs and generating raw motor commands, with applications in grasping [5] or self-driving cars [110], for instance. However, these approaches are still limited. For the grasping experiment, full observability is required [5]. The considered tasks also require to move the robot end effector to positions that are known and this knowledge is taken into account in the cost functions used. Current approaches thus need to be completed with modules in charge of preparing the data they need. This is one of the goal of the proposed approach. Besides, these approaches often require very large data sets which makes them generally impractical. The self-driving car application, for instance, relies on 72 hours of driving data [110].

This issue is addressed in a variety of ways, such as learning a lower dimensionality representation [111, 53], or using learned models of the environment [112]. But to keep with the hierarchical learning perspective, standard HRL methods have been extended to the deep learning context with various frame-

works [113, 114, 115, 116, 117]. However, it is still hard to find application of these frameworks to a real robot, except when crossing the reality gap is straightforward [118, 119].

Abstract policies can also be extracted from a set of demonstrations [120] or at least "informative" policies that may result from a former training session [121, 122]. These works assume that such policies are available or that informative policies can be learned. In an open-ended learning scenario, the acquisition of a relevant data set should be included in the learning process itself to make the system more autonomous. Our approach aims at building such policies in realistic setups with sparse reward and is thus complementary with them. Other approaches avoid this issue by adding intrinsic motivations [123, 124, 125]. Again, only few of them have been applied to a real robot [126], and they have not been connected to representational redescription concerns yet, even if some preliminary work comes into this direction [74, 127].

8.3 Continual learning and development

Hierarchical approaches are focused on structuring policy representations to better exploit what has been learned on a new task. Tasks are, in general, implicitly related and in a limited number. When learning over long time periods and many different tasks, new issues arise: training data cannot be completely saved and the number of learned policies increases. The consequence is a risk of *catastrophic forgetting* or the difficulty to identify relevant stored policies. Approaches dealing with these issues have been given different names: lifelong learning [38, 39], never ending learning [40] or continual learning [41]. With the transfer between short term and long term memory, our approach includes an instance of dual-memory learning systems [39]. However, while the focus in these works is in building learning processes that can deal with the continuous flow of data and the different tasks with a single learning process [41], our approach decomposes learning into different processes in charge of bootstrapping the representational redescription process, acquiring skills and consolidating them to make them more robust and transferrable between domains, tasks and robots.

The decomposition of the learning process into different phases is a feature of developmental robotics [34, 35, 37], that draws inpiration from human and animal development. In these approaches, the robot is not ready to solve a task when it is first turned on. It needs to acquire first information about itself and its environment. During this phase, no task is considered. The robot is just exploring, with dedicated intrinsic motivations [128, 129, 130, 131], to identify what is possible and generate data to learn models of the world (including itself) and sensorimotor skills. This is a fundamental difference with continual learning approaches where the robot does nothing else than solving a task from the very first moment it is turned on. The approach we have proposed is focused on the acquisition of adapted representations, it is thus an *early developmental AI* system according to the classification of [132].

Pioneering works on this topic drew inspiration from Piaget's developmental Psychology work and applied it to simplified environments with predefined representations [133, 134]. Later works focus on the question of building appropriate discrete representations from low-level sensorimotor values [135, 136, 137], some even going towards abstract symbolic representations [138, 139]. While these works do consider simulated robots, the authors of [140] propose a method to build abstract representations that has been tested on real robots. It starts from a fixed set of known actions and thus does not address the skill discovery challenge.

To summarize, many different challenges have to be faced when trying to apply machine learning methods to real robots (see Section 2). There are some approaches combining deep HRL and lifelong learning mechanisms [141] but, to the best of our knowledge, the approach introduced here is the first one that considers most of them, lists and tests relevant approaches and describes how to connect and articulate them while applying it to real robots.

9. Impact on Neuroscience

So far, we have shown the importance of representational redescription from a robotics perspective, arguing that it is a critical process for the versatility of robots in realistic open-ended learning scenarios. But representational redescription is above all a key process in the cognitive capabilities of living creatures. In this section, we investigate what Neuroscience can learn from our work.

9.1 Neuroscience and state representation redescription

The hippocampus is well known, even in the machine learning and robotics communities, for hosting neurons whose receptive fields appear to represent locations, the so-called *place cells* [142]. Many models of the hippocampus have been implemented on real robots [143, 144, 145, 146, 147, for example] or have inspired robot navigation algorithms [148, 31, for example], either to test the efficiency of neural theories in realistic settings, or as bioinspired tools for robotic navigation. The resulting compact and efficient representation of spatial states has then often been used to learn navigation behaviors by reinforcement [149, 143, 150, 151, 152, 31, for example]. In particular, this has been done by connecting models of the hippocampus to a model of the basal ganglia (a group of subcortical nuclei known to be involved in action selection) in order to generate goal-directed behaviors. This fits quite well with the traditional machine learning approach where one designs a state representation (here, the spatial position) adapted to the task one wants to solve (here navigation), and then use a reinforcement learning algorithm to learn the optimal policy.

However, recent experimental Neuroscience results shed a new light on what the hippocampus might in fact be doing: rather than only representing places, it may in fact encode the general representation that is the most appropriate to handle

the task at hand, even when the task is not spatial (e.g., categorization of social agents [153, 154]). This representation would correspond to 2D localization when reward delivery is driven by the ability to reach a given location in space, but it could also be the integration of time and/or distance [155], or the position in a sequence [156], if these variables are the essential ones to earn reward. This suggests that some reinforcement signals are used in the hippocampus so as to sort out which of the few dimensions that can be extracted from the sensory data are relevant for the task at hand: encoding durations, distances or sequence order, only if using them helps getting rewards/not using them leads to poorer performance.

These observations fit quite well with the approach advocated here for learning in robotics: the hippocampus may indeed be a central player, when it comes to performing *representational redescription*.

Actually, the algorithms proposed here could be used to derive robotics-informed hypotheses about how the brain might perform representational redescription. For instance, in Section 4, we showed that an efficient State Representation Learning (SRL) algorithm must combine several different objectives into the loss function: reconstructing observations (e.g., learning to encode both target and robot positions), learning forward and inverse models, and predicting reward. In particular, we found that without these complementary learning objectives, predicting reward leads only to a classifier detecting when the robot is at the goal, thus not providing any particular structure or disentanglement to the rest of the state space.

In contrast, recent Neuroscience work attempting to model adaptive state creation during animal reward learning generally focused on too restricted representations [23, 157]. We argue that some knowledge of the SRL work presented in this paper may help them better understand how animals perform representation learning when facing tasks represented with a larger number of dimensions.

9.2 Neuroscience and task-sets coordination

The present work may also have implications for another field of neuroscience research, namely the study of task-set learning and coordination in the prefrontal cortex [158, 24]. Learning different task-sets means learning different sets of action values (e.g., different Q-tables) in parallel and adaptively shifting to the set which seems the most appropriate for the task at hand. This process has also been called " episodic control". In these tasks, Human subjects need to perform various cognitive operations: autonomously detect that the task has changed though these changes are not signalled, store in memory the set of action values associated to the previous task, search in memory whether there already exists a previously learned set of action values that corresponds to the new task, or instead learn the new action values, and so on after each abrupt task change.

Importantly, computational models of task-set learning typically use the same state/action descriptions. The difference between task-sets thus relies in the mapping between states and actions (i.e., a different Q-table per task-set). The focus of these models is on how the prefrontal cortex detects task context changes to decide which memorized (previously learned) task-set is now relevant, or whether the situation is novel and a new task-set should be created and learned. Now, the work presented in the present paper proposes to go beyond this, by adding the possibility of "representational redescription". In other words, different tasks may not only require different action values for the same state/action representation, but also sometimes different state or action representations.

One interesting question is then: are prefrontal cortex mechanisms for state/action representational redescription completely different from those used to change/coordinate task-sets? Interestingly, neural representations of different cognitive tasks can be learned sequentially in a continual learning setting; and it seems that the compositionality they give rise to could be used as part of the redescription or recombination strategies [159]. These novel questions open the road for further research at the crossroads between machine learning, autonomous robotics and computational Neuroscience.

9.3 Neuroscience and action representation redescription

A third important link which can be drawn with neuroscience research is about action representation redescription. More precisely, to our knowledge, most cognitive Neuroscience researches does not address the question of how novel action representations emerge in the nervous system. Instead, most assume that action representations are already in place, and focus on the question of how an agent can learn to select appropriate actions at a given moment. This action selection problem has received a lot of attention since several decades, and appears to involve the basal ganglia, a group of subcortical nuclei involved in the temporal organization of motor decisions [160, 161, 30].

Nevertheless, interestingly, the basal ganglia has been found to also contribute to an action chunking mechanism resulting in the encoding of novel macro-actions constituted of a sequence of existing unitary actions [162, 163, 164]. The rationale is that actions that are often repeated one after another within the same sequence (*e.g.,* grasping a bottle, lifting the bottle, bending the bottle, pouring water into a glass, putting the bottle back on the table) can become a chunked routine that can then be triggered as a unitary habitual behavior when faced with the same context and goal [162, 165, 166, 167]. Mechanistically, it is thought that neural activity related to each individual action, when sequentially activated within short periods of time, can be associated through Hebbian learning within the motor cortex [168, 169], so that after habit acquisition basal ganglia neurons become only active at the beginning of the sequence (putatively to initiate it) and at its end (putatively to indicate that it's finished) [170, 171].

Importantly, the action representation redescription experimental results presented here go further simple action

chunking mechanisms for habit learning, and thus have the potential to raise novel insightful ideas for Neuroscience.

9.4 Neuroscience & sleep-related learning processes

A last piece of results which may have impacts on Neuroscience, is about sleep-related learning processes. As we previously mentioned, within the mammal brain, the hippocampus contains neurons which encode specific locations of the environment, the so-called "place cells" [142]. Strikingly, hippocampal place cells are reactivated during sleep while an animal is immobile and may be "dreaming" about the task that it previously performed [172]. Such a replay not only occurs during sleep, but also during periods of quite wakefulness where the animal seems to be thinking about what it just did during a task [173, 58]. Neuroscientists wonder what might be the role of such a replay phenomenon. One recent hypothesis is that such a replay might be useful to bootstrap reinforcement learning by using an internal model of the task structure to update state-action value functions offline [174, 59], similarly to the Dyna architecture [175].

Nevertheless, applying this type of neuro-inspired replay models to continuous state space navigation, which constitutes a step towards implementing them in real robots, revealed that offline replay of these models is not only useful to bootstrap the state-action value function, but also to learn a stable model of the world [176, 56]. More precisely, when solving a given navigation problem, the states are often encountered one after another, almost always in the same order. Online learning with a neural network is disrupted by such temporal correlations of the samples [177]. Replaying past experiences in a random order was thus necessary to break these repeating temporal correlations, so as to learn a stable and coherent model of the world.

This last example gives us the opportunity to highlight a more general impact that robotics research may have on Neuroscience: because robots have to interact with the real world, testing learning algorithms in robots often leads to different results than perfectly controlled simulation models [178]. Some models that work perfectly well in simulations can lead to disappointing results when tested on a real robot. Alternatively, some models that appear suboptimal compared to other models in simulation may turn out more robust when facing noisy, multidimensional, sometimes unpredictable situations to which a robot is often confronted. Finally, some models that give the best performance in simulation may turn out computationnally too costly to work in real time on a robot. We thus hope that the open-ended learning robotics results presented in this paper, and thoroughly discussed above, can help convince neuroscientists of the interest of paying attention to robotics research, in addition to dematerialized work done in artificial intelligence, to better understand the properties of learning processes that can work in the real-world in a variety of experimental contexts.

10. Conclusion

The open-ended learning capability corresponds to the ability to solve tasks without having been prepared for them. A strong requirement to reach this capability is to give the robot the ability to build, on its own, the representations adapted to these tasks, may it be state or action spaces. We have discussed the challenges it raises and proposed the DREAM architecture, an asymptotically end-to-end, modular, and developmental framework to address them, emphasizing "awake" processes, that require the robot to interact with its environment and have thus a high cost, and "dreaming" processes, that do not. A partial implementation of this framework has been presented together with the results it has generated. Future work will complete this implementation to reach a complete open-ended learning ability.

11. Acknowledgments

This work has been supported by the FET project DREAM[8], that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 640891.

References

[1] Joseph L Jones. Robots at the tipping point. *IEEE Robotics and Automation Magazine*, 13(1):76, 2006.

[2] F. Vaussard, J. Fink, V. Bauwens, P. Rétornaz, D. Hamel, P. Dillenbourg, and F. Mondada. Lessons learned from robotic vacuum cleaners entering the home ecosystem. *Robotics and Autonomous Systems*, 62(3):376 – 391, 2014. Advances in Autonomous Robotics — Selected extended papers of the joint 2012 TAROS Conference and the FIRA RoboWorld Congress, Bristol, UK.

[3] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[4] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[5] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[6] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[7] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[8] http://dream.isir.upmc.fr/

[8] Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

[9] Jie Lu, Vahid Behbood, Peng Hao, Hua Zuo, Shan Xue, and Guangquan Zhang. Transfer learning using computational intelligence: a survey. *Knowledge-Based Systems*, 80:14–23, 2015.

[10] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[11] Javier García and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[12] Fernando Fernández, Javier García, and Manuela Veloso. Probabilistic policy reuse for inter-task transfer learning. *Robotics and Autonomous Systems*, 58(7):866–871, 2010.

[13] Enrique Munoz de Cote, Esteban O Garcia, and Eduardo F Morales. Transfer learning by prototype generation in continuous spaces. *Adaptive Behavior*, 24(6):464–478, 2016.

[14] Trung Thanh Nguyen, Tomi Silander, Zhuoru Li, and Tze-Yun Leong. Scalable transfer learning in heterogeneous, dynamic environments. *Artificial Intelligence*, 247:70–94, 2017.

[15] Mohsen Kaboli, Di Feng, and Gordon Cheng. Active tactile transfer learning for object discrimination in an unstructured environment using multimodal robotic skin. *International Journal of Humanoid Robotics*, 15(01):1850001, 2018.

[16] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503, 2015.

[17] Ignasi Clavera, David Held, and Pieter Abbeel. Policy transfer via modularity and reward guiding. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1537–1544. IEEE, 2017.

[18] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 2169–2176. IEEE, 2017.

[19] Stephen James, Andrew J Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *arXiv preprint arXiv:1707.02267*, 2017.

[20] René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Guanghang Cai, Natalia Díaz-Rodríguez, and David Filliat. Discorl: Continual reinforcement learning via policy distillation, 2019.

[21] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[22] Annette Karmiloff-Smith. Beyond modularity: A developmental approach to cognitive science. *MITPress, Cambridge*, 1992.

[23] A David Redish, Steve Jensen, Adam Johnson, and Zeb Kurth-Nelson. Reconciling reinforcement learning models with behavioral extinction and renewal: implications for addiction, relapse, and problem gambling. *Psychological review*, 114(3):784, 2007.

[24] Anne Collins and Etienne Koechlin. Reasoning, learning, and creativity: frontal lobe function and human decision-making. *PLoS biology*, 10(3):e1001293, 2012.

[25] D Gowanlock R Tervo, Joshua B Tenenbaum, and Samuel J Gershman. Toward the neural implementation of structure learning. *Current opinion in neurobiology*, 37:99–105, 2016.

[26] Pieter Abbeel, Adam Coates, and Andrew Y Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[27] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013.

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

[29] Cédric Colas, Pierre-Yves Oudeyer, Olivier Sigaud, Pierre Fournier, and Mohamed Chetouani. CURIOUS: Intrinsically motivated multi-task, multi-goal reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1331–1340, 2019.

[30] Mehdi Khamassi and Mark D Humphries. Integrating cortico-limbic-basal ganglia architectures for learning model-based and model-free navigation strategies. *Frontiers in behavioral neuroscience*, 6:79, 2012.

[31] K. Caluwaerts, M. Staffa, S. N'Guyen, C. Grand, L. Dollé, A. Favre-Felix, B. Girard, and M. Khamassi. A biologically inspired meta-control navigation system for the psikharpax rat robot. *Bioinspiration & Biomimetics*, 7(2):025009, 2012.

[32] Matthew Botvinick, Sam Ritter, Jane X. Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in Cognitive Sciences*, 23(5):408 – 422, 2019.

[33] Stephane Doncieux, David Filliat, Natalia Díaz-Rodríguez, Timothy Hospedales, Richard Duro, Alexandre Coninx, Diederik M. Roijers, Benoît Girard, Nicolas Perrin, and Olivier Sigaud. Open-ended learning: A conceptual framework based on representational redescription. *Frontiers in Neurorobotics*, 12:59, 2018.

[34] Max Lungarella, Giorgio Metta, Rolf Pfeifer, and Giulio Sandini. Developmental robotics: a survey. *Connection science*, 15(4):151–190, 2003.

[35] Juyang Weng. Developmental robotics: Theory and experiments. *International Journal of Humanoid Robotics*, 1(02):199–236, 2004.

[36] Alexander Stoytchev. Some basic principles of developmental robotics. *IEEE Transactions on Autonomous Mental Development*, 1(2):122–130, 2009.

[37] Angelo Cangelosi and Matthew Schlesinger. *Developmental robotics: From babies to robots*. MIT press, 2015.

[38] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[39] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

[40] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhanava Dalvi, Matt Gardner, Bryan Kisiel, et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018.

[41] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 58:52–68, 2020.

[42] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI Global, 2010.

[43] Mark A. Bedau. The nature of life. In *The philiosophy of artificial life*. Oxford University Press, 1996.

[44] Joel Lehman and Kenneth O. Stanley. Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2):189–223, 2010.

[45] Russell K Standish. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(02):167–175, 2003.

[46] Stéphane Doncieux. Creativity: A driver for research on robotics in open environments. *Intellectica*, 65(1):205–219, 2016.

[47] Kenji Doya and Tadahiro Taniguchi. Toward evolutionary and developmental intelligence. *Current Opinion in Behavioral Sciences*, 29:91–96, 2019.

[48] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. *Proceedings of Robotics: Science and Systems. Pittsburgh, Pennsylvania*, 2018.

[49] Carlos Maestre, Antoine Cully, Christophe Gonzales, and Stephane Doncieux. Bootstrapping interactions with objects from raw sensorimotor data: a novelty search based approach. In *2015 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, pages 7–12. IEEE, 2015.

[50] Sergey Levine, Nolan Wagener, and Pieter Abbeel. Learning contact-rich manipulation skills with guided policy search. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 156–163. IEEE, 2015.

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

[52] Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

[53] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Fançois Goudou, and David Filliat. State Representation Learning for Control: An Overview. *Neural Networks*, 108:379–392, December 2018.

[54] Robert Stickgold, J Allen Hobson, Roar Fosse, and Magdalena Fosse. Sleep, learning, and dreams: off-line memory reprocessing. *Science*, 294(5544):1052–1057, 2001.

[55] Gaetan De Lavilléon, Marie Masako Lacroix, Laure Rondi-Reig, and Karim Benchenane. Explicit memory creation during sleep demonstrates a causal role of place cells in navigation. *Nature neuroscience*, 18(4):493, 2015.

[56] Romain Cazé, Mehdi Khamassi, Lise Aubin, and Benoît Girard. Hippocampal replays under the scrutiny of reinforcement learning models. *Journal of neurophysiology*, 120(6):2877–2896, 2018.

[57] Ullrich Wagner, Steffen Gais, Hilde Haider, Rolf Verleger, and Jan Born. Sleep inspires insight. *Nature*, 427(6972):352, 2004.

[58] Anoopum S Gupta, Matthijs AA van der Meer, David S Touretzky, and A David Redish. Hippocampal replay is not a simple function of experience. *Neuron*, 65(5):695–705, 2010.

[59] Mehdi Khamassi and Benoît Girard. Modeling awake hippocampal reactivations with model-based bidirectional search. *Biological Cybernetics*, pages 1–18, 2020.

[60] Jean Piaget. Part i: Cognitive development in children–piaget development and learning. *Journal of research in science teaching*, 40, 2003.

[61] Robert S Siegler. *Emerging minds: The process of change in children's thinking*. Oxford University Press, 1998.

[62] Antonin Raffin, Ashley Hill, René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, and David Filliat. S-RL Toolbox: Environments, Datasets and Evaluation Metrics for State Representation Learning. In *NIPS 2018 Deep RL workshop*, Montreal, Canada, December 2018.

[63] Timothée Lesort, Mathieu Seurin, Xinrui Li, Natalia Díaz Rodríguez, and David Filliat. Unsupervised state representation learning with robotic priors: a robustness analysis. In *International Joint Conference on Neural Networks*, 2019.

[64] Antonin Raffin, Ashley Hill, Kalifou René Traoré, Timothée Lesort, Natalia Díaz-Rodríguez, and David Filliat. Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics. In *Workshop on "Structure and Priors in Reinforcement Learning" (SPiRL) at ICLR*, 2019.

[65] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[66] Léni K Le Le Goff, Ghanim Mukhtar, Alexandre Coninx, and Stéphane Doncieux. Bootstrapping robotic ecological perception from a limited set of hypotheses through interactive perception. *arXiv preprint arXiv:1901.10968*, 2019.

[67] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

[68] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation - Supervoxels for point clouds. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2027–2034, 2013.

[69] Leni K Le Goff, Ghanim Mukhtar, Pierre-Henri Le Fur, and Stephane Doncieux. Segmenting objects through an autonomous agnostic exploration conducted by a robot. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 284–291. IEEE, 2017.

[70] Leni K Le Goff, Oussama Yaakoubi, Alexandre Coninx, and Stephane Doncieux. Building an affordances map with interactive perception. *arXiv preprint arXiv:1903.04413*, 2019.

[71] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[72] Georg Biegelbauer, Markus Vincze, and Walter Wohlkinger. Model-based 3d object detection. *Machine Vision and Applications*, 21(4):497–516, 2010.

[73] Seungsu Kim, Alexandre Coninx, and Stéphane Doncieux. From exploration to control: learning object manipulation skills through novelty search and local adaptation. *CoRR, abs/1901.00811*, 2019.

[74] Alexandre Péré, Sébastien Forestier, Olivier Sigaud, and Pierre-Yves Oudeyer. Unsupervised Learning of Goal Spaces for Intrinsically Motivated Goal Exploration. mar 2018.

[75] Olivier Sigaud and Freek Stulp. Policy search in continuous action domains: an overview. *Neural Networks*, 113:28–40, 2019.

[76] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proceedings of ECAL 1995*, pages 704–720, 1995.

[77] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40, 2016.

[78] Antoine Cully and Yiannis Demiris. Quality and diversity optimization: A unifying modular framework. *IEEE Transactions on Evolutionary Computation*, 22(2):245–259, 2017.

[79] F. Stulp, E. Oztop, P. Pastor, M. Beetz, and S. Schaal. Compact models of motor primitive variations for predictable reaching and obstacle avoidance. In *IEEE-RAS International Conference on Humanoid Robots*, 2009.

[80] M Duarte, J Gomes, S M Oliveira, and A L Christensen. Evolution of Repertoire-Based Control for Robots With Complex Locomotor Systems. *IEEE Transactions on Evolutionary Computation*, 22:314–328, 2018.

[81] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[82] Marija Jegorova, Stéphane Doncieux, and Timothy Hospedales. Generative adversarial policy networks for behavioural repertoire. In *2019 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2019.

[83] Arvind Agarwal, Samuel Gerber, and Hal Daume. Learning multiple tasks using manifold regularization. In *Advances in neural information processing systems*, pages 46–54, 2010.

[84] Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. *preprint arXiv:1206.6417*, 2012.

[85] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *ICLR*, 2017.

[86] Paul Ruvolo and Eric Eaton. Ella: An efficient lifelong learning algorithm. In *International Conference on Machine Learning*, pages 507–515, 2013.

[87] Chenyang Zhao, Timothy M Hospedales, Freek Stulp, and Olivier Sigaud. Tensor based knowledge transfer across skill categories for robot control. In *IJCAI*, pages 3462–3468, 2017.

[88] Richard J Duro, Jose A Becerra, Juan Monroy, and Francisco Bellas. Perceptual generalization and context in a network memory inspired long-term memory for artificial cognition. *International Journal of Neural Systems*, pages 1–22, 2018.

[89] Richard J Duro, Jose A Becerra, Juan Monroy, and Luis Calvo. Context nodes in the operation of a long term memory structure for an evolutionary cognitive architecture. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1172–1176. ACM, 2017.

[90] Jose A Becerra, Richard J Duro, and Juan Monroy. A redescriptive approach to autonomous perceptual classification in robotic cognitive architectures. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 522–529. IEEE, 2018.

[91] Nicolas Bredeche, Evert Haasdijk, and Abraham Prieto. Embodied evolution in collective robotics: A review. *Frontiers in Robotics and AI*, 5:12, 2018.

[92] Jacqueline Heinerman, Massimiliano Rango, and Agoston Endre Eiben. Evolution, individual learning, and social learning in a swarm of real robots. In *2015 IEEE Symposium Series on Computational Intelligence*, pages 1055–1062. IEEE, 2015.

[93] Jean-Marc Montanier, Simon Carrignon, and Nicolas Bredeche. Behavioral specialization in embodied evolutionary robotics: Why so difficult? *Frontiers in Robotics and AI*, 3:38, 2016.

[94] Nicolas Bredeche, Jean-Marc Montanier, and Simon Carrignon. Benefits of proportionate selection in embodied evolution: a case study with behavioural specialization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1683–1684. ACM, 2017.

[95] Jacqueline Heinerman, Evert Haasdijk, and A. E. Eiben. Importance of parameter settings on the benefits of robot-to-robot learning in evolutionary robotics. *Frontiers in Robotics and AI*, 6:10, 2019.

[96] Iuliia Kotseruba and John K Tsotsos. 40 years of cognitive architectures: core cognitive abilities and practical applications. *Artificial Intelligence Review*, pages 1–78, 2018.

[97] Peijun Ye, Tao Wang, and Fei-Yue Wang. A survey of cognitive architectures in the past 20 years. *IEEE transactions on cybernetics*, 48(12):3280–3290, 2018.

[98] Stuart C Shapiro and Jonathan P Bona. The glair cognitive architecture. *International Journal of Machine Consciousness*, 2(02):307–332, 2010.

[99] David E Kieras. A summary of the epic cognitive architecture. *The Oxford handbook of cognitive science*, 1:24, 2016.

[100] Pat Langley and Dongkyu Choi. A unified cognitive architecture for physical agents. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[101] Brandon Rohrer. Becca: Reintegrating ai for natural world interaction. In *2012 AAAI Spring Symposium Series*, 2012.

[102] Francisco Bellas, Richard J Duro, Andrés Faiña, and Daniel Souto. Multilevel darwinist brain (mdb): Artificial evolution in a cognitive architecture for real robots. *IEEE Transactions on autonomous mental development*, 2(4):340–354, 2010.

[103] John R Anderson, Daniel Bothell, Michael D Byrne, Scott Douglass, Christian Lebiere, and Yulin Qin. An integrated theory of the mind. *Psychological review*, 111(4):1036, 2004.

[104] John E Laird. *The Soar cognitive architecture*. MIT press, 2012.

[105] Antonio Lieto. Representational limits in cognitive architectures. In *EUCognition Meeting (European Society for Cognitive Systems)" Cognitive Robot Architectures"*, volume 1855, pages 16–20. Ceur-ws, 2017.

[106] David Vernon. Two ways (not) to design a cognitive architecture. *Cognitive Robot Architectures*, 42, 2017.

[107] Timothy Mann and Shie Mannor. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *International conference on machine learning*, pages 127–135, 2014.

[108] Daniel J Mankowitz, Timothy A Mann, Pierre-Luc Bacon, Doina Precup, and Shie Mannor. Learning robust options. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[109] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[110] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[111] Wendelin Böhmer, Jost Tobias Springenberg, Joschka Boedecker, Martin Riedmiller, and Klaus Obermayer. Autonomous learning of state representations for control: An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations. *KI-Künstliche Intelligenz*, 29(4):353–362, 2015.

[112] Konstantinos Chatzilygeroudis, Vassilis Vassiliades, Freek Stulp, Sylvain Calinon, and Jean-Baptiste Mouret. A survey on policy search algorithms for learning robot controllers in a handful of trials. *IEEE Transactions on Robotics*, 2019.

[113] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pages 1726–1734, 2017.

[114] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.

[115] Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical actor-critic. *arXiv preprint arXiv:1712.00948*, 2017.

[116] Andrew Levy, Robert Platt, and Kate Saenko. Hierarchical reinforcement learning with hindsight. *arXiv preprint arXiv:1805.08180*, 2018.

[117] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[118] Zhaoyang Yang, Kathryn Merrick, Lianwen Jin, and Hussein A Abbass. Hierarchical deep reinforcement learning for continuous action control. *IEEE transactions on neural networks and learning systems*, 29(11):5174–5184, 2018.

[119] Yilun Chen, Chiyu Dong, Praveen Palanisamy, Priyantha Mudalige, Katharina Muelling, and John M Dolan. Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[120] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[121] Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *preprint arXiv:1703.08294*, 2017.

[122] Sanjay Krishnan, Roy Fox, Ion Stoica, and Ken Goldberg. Ddco: Discovery of deep continuous options for robot learning from demonstrations. *arXiv preprint arXiv:1710.05421*, 2017.

[123] Christopher M Vigorito and Andrew G. Barto. Intrinsically motivated hierarchical skill learning in structured environments. *IEEE Transactions on Autonomous Mental Development*, 2(2):132–143, 2010.

[124] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[125] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[126] Sébastien Forestier, Yoan Mollard, and Pierre-Yves Oudeyer. Intrinsically motivated goal exploration processes with automatic curriculum learning. *arXiv preprint arXiv:1708.02190*, 2017.

[127] Adrien Laversanne-Finot, Alexandre Péré, and Pierre-Yves Oudeyer. Curiosity driven exploration of learned disentangled goal spaces. *arXiv preprint arXiv:1807.01521*, 2018.

[128] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

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

[130] Gianluca Baldassarre and Marco Mirolli. *Intrinsically motivated learning in natural and artificial systems*. Springer, 2013.

[131] Pierre-Yves Oudeyer. Computational theories of curiosity-driven learning. *arXiv preprint arXiv:1802.10546*, 2018.

[132] Frank Guerin. Learning like a baby: a survey of artificial intelligence approaches. *The Knowledge Engineering Review*, 26(2):209–236, 2011.

[133] Gary L Drescher. *Made-up minds: a constructivist approach to artificial intelligence*. MIT press, 1991.

[134] Harold Henry Chaput. *The constructivist learning architecture: A model of cognitive development for robust autonomous robots*. PhD thesis, University of Texas at Austin, 2004.

[135] Jeremy Stober and Benjamin Kuipers. From pixels to policies: A bootstrapping agent. In *2008 7th IEEE International Conference on Development and Learning*, pages 103–108. IEEE, 2008.

[136] Jonathan Mugan and Benjamin Kuipers. Autonomous learning of high-level states and actions in continuous environments. *IEEE Transactions on Autonomous Mental Development*, 4(1):70–86, 2011.

[137] Matthieu Zimmer and Stephane Doncieux. Bootstrapping q-learning for robotics from neuro-evolution results. *IEEE Transactions on Cognitive and Developmental Systems*, 10(1):102–119, 2017.

[138] George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[139] George Konidaris. Constructing abstraction hierarchies using a skill-symbol loop. In *IJCAI: proceedings of the conference*, volume 2016, page 1648. NIH Public Access, 2016.

[140] Emre Ugur and Justus Piater. Bottom-up learning of object categories, action effects and logical rules: From continuous manipulative exploration to symbolic planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2627–2633. IEEE, 2015.

[141] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*, pages 1553–1561, 2017.

[142] John O'Keefe and Jonathan Dostrovsky. The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 1971.

[143] Angelo Arleo and Wulfram Gerstner. Spatial cognition and neuro-mimetic navigation: a model of hippocampal place cell activity. *Biological cybernetics*, 83(3):287–299, 2000.

[144] Jeffrey L Krichmar, Anil K Seth, Douglas A Nitz, Jason G Fleischer, and Gerald M Edelman. Spatial navigation and causal analysis in a brain-based device modeling cortical-hippocampal interactions. *Neuroinformatics*, 3(3):197–221, 2005.

[145] Thomas Strösslin, Denis Sheynikhovich, Ricardo Chavarriaga, and Wulfram Gerstner. Robust self-localisation and navigation based on hippocampal place cells. *Neural networks*, 18(9):1125–1140, 2005.

[146] C Giovannangeli, Ph Gaussier, and JP Banquet. Robustness of visual place cells in dynamic indoor and outdoor environment. *International Journal of Advanced Robotic Systems*, 3(2):19, 2006.

[147] Alejandra Barrera and Alfredo Weitzenfeld. Biologically-inspired robot spatial cognition based on rat neurophysiological studies. *Autonomous Robots*, 25(1-2):147–169, 2008.

[148] Michael Milford and Gordon Wyeth. Persistent navigation and mapping using a biologically inspired slam system. *The International Journal of Robotics Research*, 29(9):1131–1153, 2010.

[149] Alex Guazzelli, Mihail Bota, Fernando J Corbacho, and Michael A Arbib. Affordances. motivations, and the world graph theory. *Adaptive Behavior*, 6(3-4):435–471, 1998.

[150] DJ Foster, RGM Morris, and Peter Dayan. A model of hippocampally dependent navigation, using the temporal difference learning rule. *Hippocampus*, 10(1):1–16, 2000.

[151] Ricardo Chavarriaga, Thomas Strösslin, Denis Sheynikhovich, and Wulfram Gerstner. A computational model of parallel navigation systems in rodents. *Neuroinformatics*, 3(3):223–241, 2005.

[152] Benoît Girard, David Filliat, Jean-Arcady Meyer, Alain Berthoz, and Agnès Guillot. Integration of navigation and action selection functionalities in a computational model of cortico-basal-ganglia–thalamo-cortical loops. *Adaptive Behavior*, 13(2):115–130, 2005.

[153] Rita Morais Tavares, Avi Mendelsohn, Yael Grossman, Christian Hamilton Williams, Matthew Shapiro, Yaacov Trope, and Daniela Schiller. A map for social navigation in the human brain. *Neuron*, 87(1):231–243, 2015.

[154] Seongmin A Park, Douglas S Miller, Hamed Nili, Charan Ranganath, and Erie D Boorman. Map making: Constructing, combining, and navigating abstract cognitive maps. *BioRxiv*, page 810051, 2019.

[155] Benjamin J Kraus, Robert J Robinson II, John A White, Howard Eichenbaum, and Michael E Hasselmo. Hippocampal "time cells": time versus path integration. *Neuron*, 78(6):1090–1101, 2013.

[156] Henrique O Cabral, Martin Vinck, Celine Fouquet, Cyriel MA Pennartz, Laure Rondi-Reig, and Francesco P Battaglia. Oscillatory dynamics and place field maps reflect hippocampal ensemble processing of sequence and place memory under nmda receptor control. *Neuron*, 81(2):402–415, 2014.

[157] Samuel J Gershman, David M Blei, and Yael Niv. Context, learning, and extinction. *Psychological review*, 117(1):197, 2010.

[158] Earl K Miller and Jonathan D Cohen. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1):167–202, 2001.

[159] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

[160] Peter Redgrave, Tony J Prescott, and Kevin Gurney. The basal ganglia: a vertebrate solution to the selection problem? *Neuroscience*, 89(4):1009–1023, 1999.

[161] Mehdi Khamassi, Loïc Lachèze, Benoît Girard, Alain Berthoz, and Agnès Guillot. Actor–critic models of reinforcement learning in the basal ganglia: from natural to artificial rats. *Adaptive Behavior*, 13(2):131–148, 2005.

[162] Ann M Graybiel. The basal ganglia and chunking of action repertoires. *Neurobiology of learning and memory*, 70(1-2):119–136, 1998.

[163] Terra D Barnes, Yasuo Kubota, Dan Hu, Dezhe Z Jin, and Ann M Graybiel. Activity of striatal neurons reflects dynamic encoding and recoding of procedural memories. *Nature*, 437(7062):1158–1161, 2005.

[164] Xin Jin, Fatuel Tecuapetla, and Rui M Costa. Basal ganglia subcircuits distinctively encode the parsing and concatenation of action sequences. *Nature neuroscience*, 17(3):423–430, 2014.

[165] Amir Dezfouli and Bernard W Balleine. Habits, action sequences and reinforcement learning. *European Journal of Neuroscience*, 35(7):1036–1051, 2012.

[166] Kevin J Miller, Amitai Shenhav, and Elliot A Ludvig. Habits without values. *Psychological Review*, 126(2):292, 2019.

[167] Adam Morris and Fiery Cushman. Model-free rl or action sequences? *Frontiers in Psychology*, 10, 2019.

[168] F Gregory Ashby, Benjamin O Turner, and Jon C Horvitz. Cortical and basal ganglia contributions to habit learning and automaticity. *Trends in cognitive sciences*, 14(5):208–215, 2010.

[169] Michael J Frank and Eric D Claus. Anatomy of a decision: striato-orbitofrontal interactions in reinforcement learning, decision making, and reversal. *Psychological review*, 113(2):300, 2006.

[170] Catherine A Thorn, Hisham Atallah, Mark Howe, and Ann M Graybiel. Differential dynamics of activity changes in dorsolateral and dorsomedial striatal loops during learning. *Neuron*, 66(5):781–795, 2010.

[171] Xin Jin and Rui M Costa. Start/stop signals emerge in nigrostriatal circuits during sequence learning. *Nature*, 466(7305):457–462, 2010.

[172] Matthew A Wilson and Bruce L McNaughton. Reactivation of hippocampal ensemble memories during sleep. *Science*, 265(5172):676–679, 1994.

[173] David J Foster and Matthew A Wilson. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature*, 440(7084):680–683, 2006.

[174] Marcelo G Mattar and Nathaniel D Daw. Prioritized memory access explains planning and hippocampal replay. *Nature neuroscience*, 21(11):1609–1617, 2018.

[175] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[176] Lise Aubin, Mehdi Khamassi, and Benoît Girard. Prioritized sweeping neural dynaq with multiple predecessors, and hippocampal replays. In *Conference on Biomimetic and Biohybrid Systems*, pages 16–27. Springer, 2018.

[177] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[178] Mehdi Khamassi, Stéphane Lallée, Pierre Enel, Emmanuel Procyk, and Peter F Dominey. Robot cognitive control with a neurophysiologically inspired reinforcement learning model. *Frontiers in neurorobotics*, 5:1, 2011.